Stony Brook University

Computer Aided Diagnosis for Virtual Endoscopy

A Dissertation Presented

by

Wei Hong

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

August 2007

Stony Brook University

The Graduate School

Wei Hong

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Arie Kaufman - Dissertation Advisor
Distinguished Professor, Department of Computer Science

Klaus Mueller - Chairperson of Defense
Associate Professor, Department of Computer Science

Xianfeng Gu - Committee Member
Assistant Professor, Department of Computer Science

Wenli Cai - External Committee Member
Assistant Professor, Department of Radiology
Massachusetts General Hospital and Harvard Medical School

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Computer Aided Diagnosis for Virtual Endoscopy

by

Wei Hong

Doctor of Philosophy

in

Computer Science

Stony Brook University

2007

Thousands of endoscopic procedures are performed each year. They are invasive and often uncomfortable for patients. They sometimes have serious side effects such as perforation, infection, and hemorrhage. Virtual endoscopic visualization avoids the risks associated with real endoscopy, and when used prior to performing an actual endoscopic procedure for therapeutics can minimize procedural difficulties and decrease the rate of morbidity. Additionally, there are many body regions inaccessible to or complicated with real endoscopy but can be explored with virtual endoscopy. In this dissertation, novel algorithms are proposed in segmentation and digital cleansing, volume rendering, surface flattening, and *computer-aided detection* (CAD) to improve and enhance virtual endoscopy applications.

Effective colonoscopic screening for polyps with optical or virtual means requires adequate visualization of the entire colon surface. We have investigated the colon surface visibility coverage using a simulation method to estimate the percentage of the colon surface is missed in the *optical colonoscopy* (OC) and *virtual colonoscopy* (VC). Our simulation study reveals that about 23% of the colon surface is missed in the standard OC examination and about 9% of the colon surface is missed in the VC examination when navigating in both directions.

We have adopted a partial volume model in the segmentation and digital cleansing to handle the partial volume effect. Our algorithm is demonstrated with contrast-enhanced CT colon data sets. The topological noise is automatically removed from the segmentation result by a 3D region growing based algorithm using the concept of *simple point*. Furthermore, the topologically simple colon surface is extracted with a dual contouring method for virtual colon flattening.

Most common methods in virtual endoscopy simulate the behavior of a real endoscope.

Simulating a real endoscopy is not the most efficient technique in many endoscopy procedures. A real endoscopy is restricted due to physical limitations that a virtual endoscopy does not have. We present a conformal colon flattening technique which virtually unfolds the colon, allowing physicians to inspect its surface and detect polyps on a single 2D image.

Direct volume rendering (DVR) can provide high-quality virtual endoscopic views for virtual endoscopy applications. However, DVR of contemporary clinical data sets in real-time at a high resolution is still a challenge. We present a GPU-based object-order ray-casting algorithm to render large volumetric data sets on the GPU. We also exploit the cooperation and trade-off between the GPU and the CPU to obtain further acceleration. Although our ray-casting approach is of general applicability, we have specifically applied it to our VC system.

We further present a novel pipeline for CAD of colonic polyps by integrating texture and shape analysis with volume rendering and conformal colon flattening. Using our automatic method, the 3D polyp detection problem is converted to a 2D image segmentation problem. The polyps are detected by a clustering method on the 2D flattened colon image. The false positives (FPs) are further reduced by analyzing the volumetric shape features. Our system detects 100% of the adenomatous polyps, and yields a low FP rate. The results are easily integrated into a VC system, which allows physicians to perform their diagnoses more accurately and efficiently. Since the suspicious areas are clearly identified to the physician, the physician needs only traverse the colon in one direction, without fear of missing a polyp.

All presented techniques have been tested with a number of data sets to show their feasibility. In this dissertation, we focus on CT colon data sets although our techniques could be used with a variety of other human organs, such as blood vessels and bladder.

To My Wife, Yani Liu
My Parents, Xingnan Hong and Lianzhen Ding
with My Love!

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Endoscopy is a minimally invasive diagnostic medical procedure used to assess the interior surfaces of hollow organs and perform therapeutic procedures by inserting a tiny tube into the body, often, but not necessarily, through a natural body opening. The instrument may have a rigid or a flexible tube and not only provide an image, for visual inspection and photography, but also enable taking biopsies and retrieving of foreign objects. By changing the position of the endoscope, the operator is able to see lesions and other surface conditions of an organ while controlling the viewing position and angle of the probe. During this interactive exploration, the endoscopist has full control of the navigation within the hollow organ. Endoscopy procedures are increasing in medical importance because they have less deleterious effects on the patient. These procedures have been used in gastroenterology, surgery, neurosurgery, interventional radiology and many other fields.

Many endoscopic procedures are relatively painless and, at worst, associated with mild discomfort although patients are sedated for most procedures. Complications are rare (only 5% of all operations) but can include perforation of the organ under inspection with the endoscope or biopsy instrument. If that occurs open surgery may be required to repair the injury. Furthermore, endoscopes display only the inner surface of hollow organs and yield no information about the anatomy within or beyond the wall. This limitation prevents the evaluation of the transmural extent of tumors and limits the ability to localize the lesion relative to surrounding anatomic structures.

In contrast, virtual endoscopy [9] is a convenient alternative. It is based on a 3D scan of the respective body region, such as *computed tomography* (CT) scans, and *magnetic resonance imaging* (MRI) scans of the abdominal area, the heart, the head, the lungs or rotational angiography of blood vessels in various body parts. Based on the resulting volumetric data, the organs of interest are visualized and inspected from interior viewpoints.

In particular, virtual endoscopy can provide information which is unavailable in optical endoscopy due to its limited flexibility and field of view.

With continuing advances in software and hardware, virtual endoscopy offers the promise of quicker and cheaper methods of evaluation. In certain clinical situations, virtual endoscopy may enhance diagnosis, preoperative planning, operative technique, and post-operative follow-up. Although not yet in routine use, the techniques have been found useful in specific scenarios, such as virtual colonoscopy [48, 49] and virtual bronchoscopy [121]. Some hospitals no longer consider virtual colonoscopy a research protocol and are offering it as a screening tool despite its limitations.

Virtual endoscopy has also been used to evaluate the bladder, kidneys, small intestine, stomach, larynx, nasolacrimal ducts and paranasal sinuses. Therefore, this method could potentially provide a means for the screening and surveillance of bladder tumors, which tend to recur. Virtual endoscopy can also be used to simulate endoscopic surgery before the actual performance, thus helping the surgeon plan the operative approach. In summary, virtual endoscopy is a nascent technique with multiple potential applications that could have a significant impact on common clinical issues, especially colorectal cancer screening. Improved screening could detect certain cancers at an early, curable stage and could prevent the development of cancer.

In this dissertation, novel algorithms are presented in segmentation and digital cleansing, conformal virtual flattening, GPU-based volumetric ray-casting, and computer-aided polyp detection techniques to support and enhance virtual endoscopy applications, mainly for diagnosis purposes. We concentrate on the virtual colonoscopy system that focuses on the examination of the colon although our techniques are general and could be used with a variety of human organs.

1.2 Background

1.2.1 Virtual Colonoscopy

The most promising clinical use of endoluminal imaging is in examination of the colon. Colorectal cancer is the second leading cause of cancer related deaths in the United States. Colorectal cancer accounts for approximately $945,000$ new cases and $500,000$ deaths worldwide each year [116].

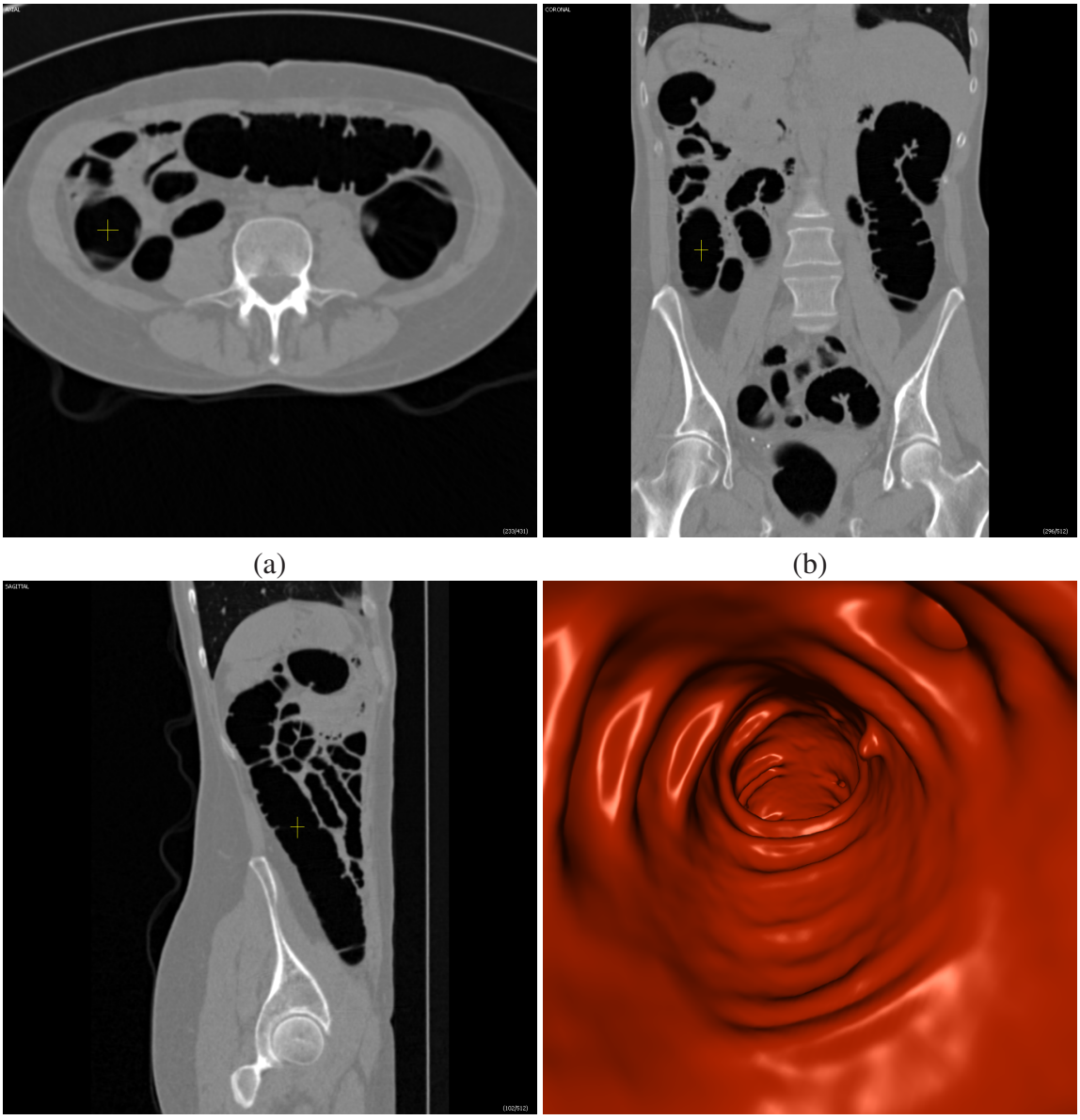
Most colorectal cancers begin as a polyp, which is a small, harmless growth in the wall of the colon. As a polyp gets larger, it can develop into a cancer that grows and spreads. Early detection of colon cancer is the key to a good prognosis. The five-year survival rate for colon cancer is nearly 90% for localized disease versus about 6% for distant metastases. It can take from 10 to 15 years for an adenomatous polyp to become an invasive cancer [94]. Thus, there is a considerable time for detection and clinical intervention if the proper screening methods are used. Studies of fecal occult blood testing, flexible sigmoidoscopy, and colonoscopy have shown that screening for colorectal cancer in high-risk countries can decrease mortality by 50%. Evidence-based guidelines recommend the screening of adults

who are at average risk for colorectal cancer, since the detection and removal of adenomas has been shown to substantially reduce the incidence of cancer and cancer-related mortality. Therefore, many have advocated screening programs to detect polyps with a diameter of less than one centimeter [87]. In *optical colonoscopy* (OC), a thin flexible fiber optic endoscope is inserted into the patient's rectum to inspect the entire colon for potential polyps. Although colonoscopy detects nearly 90% of colorectal caners, it is invasive, uncomfortable, and not without risks. Most people do not follow this advice because of the discomfort and inconvenience of the conventional colonoscopy.

To encourage people to participate in screening programs, *virtual colonoscopy* (VC) [49, 60, 93, 108], also known as *computed tomographic colonography* (CTC), has been proposed and developed to detect colorectal polyps using CT images of a patient's abdomen and a virtual fly-through visualization system that allows physicians to navigate within a 3D model of the colon searching for polyps, the precursors of cancer. In the fly-though navigation of VC, a virtual camera with a specific field of view moves along a special planned path inside the colon to render its internal view. A typical 3D endoscopic view of VC showing a polyp in VC is displayed in Figure 1.1(d). However, this polyp is hard to be identified from the 2D mutually perpendicular slice views (Figure 1.1(a)-(c)), which demonstrates that VC is much more efficient than 2D review. VC is minimally invasive, fast, inexpensive, and has been successfully demonstrated to be more convenient and efficient than the traditional OC. Moreover, VC doesn't require a rigorous bowel cleansing preparation. The patient needs only to have a modified diet with an oral contrast agent, for example barium. The tagged material is enhanced in the CT scan, allowing it to be identified. However, care must be taken during electronic cleansing to restore the CT density values where the partial volume effect occurs. With the recent introduction of multi-detector CT capable of generating 16 images in $0.5s$, CT processing has become remarkably fast and the polyp detection sensitivity has been enhanced. Pickhardt et al. [108] have demonstrated that the performance of a VC compares favorably with that of a traditional OC. The radiation exposure incurred during a virtual colonoscopy examination is currently equivalent to that of two plain abdominal films and will probably decrease with continued software and hardware developments [65]. Advantages and disadvantages of VC are compared and listed in Table 1.1.

Table 1.1: Advantages and Disadvantages of Virtual Colonoscopy.

	Advantages	**Disadvantages**
1	Noninvasive	Cost (no reimbursement code)
2	No sedation required	Radiation exposure
3	Localize polyps and lesions precisely	Cannot take biopsy specimens
4	Less technically demanding	Retained feces can be misinterpreted
5	Sensitivity equal to that of conventional OC for lesions $> 10mm$ in diameter	Cannot show texture and color details of mucosa



(a) (b)

(c) (d)

Figure 1.1: Virtual Colonoscopy: The 2D mutually perpendicular slice views are oriented (a)axial, (b)coronal, and (c)sagittal; (d)A polyp is shown in a typical 3D endoscopic view.

In fly-through navigation, it is crucial to generate an optimal camera path for efficient clinical examination. Automatic path planning is needed because manual planning is difficult and time-consuming due to the complex shape of the human colon. For complete and accurate diagnosis, a planned path should not produce significant blind areas on the colon surface. Previous investigations of automatic path generation can be categorized into approaches based on topological thinning and minimum cost spanning tree. Topological

thinning algorithms can be used to eliminate the outermost layer of a segmented colon successively with only the centerline voxels remaining. Hong et al. [48] have used the peel onion technique to generate the centerline of colon. Paik et al. [98] have proposed the thinning based on Euclidean distance mapping. By iteratively correcting a path toward the medial axis, the necessity of evaluating simple point criteria during morphological thinning is eliminated. Sadleir et al. [112] have optimized 3D topological thinning and reduced the computational burden associated with the thinning process by referring to lookup tables.

With a 3D distance map generated in the preprocessing step, the minimum cost spanning tree can be built using the shortest path algorithm [22]. The combination of maximum length and minimum cost trees enable it to find the optimal centerline. Hong et al. [49] have used the difference between the global maximum distance value and the corresponding value of the distance from the colon surface to calculate the single source shortest path from the user specified start point to the end point. Bitter et al. [12] have introduced a penalty cost for generating the minimum cost path by adding more edges and vertices to incorporate penalties for coming close to the object boundary. Wan et al. [126] have used exact Euclidian distance from each voxel inside the colon to the nearest colon boundary to extract the colon centerline and its associated branches.

In other approaches, Kang et al. [63] have proposed a method to determine camera positions and their view directions to minimize the blind areas during navigation. Kwon et al. [75] have used image-based information generated in the rendering time to determine the camera positions and directions. This method does not require pre-processing or extra storage, but it is highly likely to converge to local minima in complex regions. Level set methods [20, 46] have also been used to extract the centerline of the colon. However, it is computationally expensive to calculate the level set propagation.

All techniques that examine the colon require a clean lumen, eliminating residual materials that may be falsely interpreted as colonic masses. Prior to any of these examinations, patients undergo a bowl cleansing preparation which includes either washing the colon with a large amount of liquids or administering medications and enemas to induce bowel movements [48]. This bowel preparation is often more unpleasant than the examination itself. An alternative method of cleansing the colon would be very attractive. In VC, contrast solutions can be ingested to enhance the image intensities of the stool and fluid. By applying image segmentation algorithms, these colonic materials can be virtually removed from the images without the patient undergoing physical bowel washing [16, 78].

1.2.2 Computer-Aided Detection

VC has shown promising results for colorectal cancer screening. However, physicians have encountered several obstacles in the clinical practicality of VC. For a physician to make an accurate determination about the existence of polyps utilizing a VC system, he/she needs to spend a substantial amount of time carefully inspecting the entire colon wall during navigation. Even with a careful inspection, it is easy for a physician to miss polyps that might be hidden around sharp bends and within deep folds of the colon walls. This

problem can be minimized only through forward and reverse viewing of both supine and prone images. This process does not, however, ensure full coverage and is extremely time-consuming [10]. Therefore, the diagnostic performance of VC is indeterminate, being susceptible to human error [31, 59]. The learning curve for the accurate interpretation of CT colonographic scans can be one of the causes for variable sensitivity among reviewers [102, 111]. Moreover, the absence of visual cues that normally exist with conventional OC also make image interpretation tedious and susceptible to error.

To overcome these difficulties, *computer-aided detection* (CAD) schemes have been developed for the automatic detection of colonic polyps. The second opinion offered by a CAD scheme has the potential to reduce the interpretation time and to enhance diagnostic accuracy. Overall interpretation time can be reduced if physicians focus on the small number of regions indicated as suspicious by a CAD scheme. Thus, physicians can quickly inspect a large portion of the colon that is likely to be normal. In addition, CAD has the potential to reduce physicians' perceptual errors, thereby improving the accuracy of VC.

In order for the CAD system to be useful, it must be able to identify 100% of colonic polyps. In addition, it must be able to substantially reduce the number of false positives (FPs). Without the reduction of FPs, there would be too many areas to be inspected in a limited amount of time. An efficient CAD system would thus be used to make physicians more effective and make their diagnoses more accurate. CAD of colonic polyps is particularly challenging because the colon is highly deformable and colon surfaces have a number of polyp mimics. In the past several years there have been several prototype CAD schemes reported in the literature with variable success in polyp detection.

Shape and texture features are the two major characteristic features that have been used to differentiate polyps from normal soft tissues. Vining et al. [124] have utilized the measure of abnormal colon wall thickness to detect polyp suspects. Summers et al. [122] have employed local variations in curvature of the surface of the colonic wall to detect abnormal shapes. Then, the candidates are filtered by the restrictions of mean curvature, dimensionless ratio sphericity, and minimum polyp size to reduce FPs. Paik et al. [100] have observed that normals on the colon surface usually intersect with normals on neighboring surfaces, as polyps have 3D shape features changing rapidly in many directions. Based on this observation, they have introduced a method to detect polyps by the number of intersecting normal vectors of a patch. Yoshida et al. [137, 136] and Näppi et al. [96] have further characterized the curvature measures by shape index and curvedness to differentiate polyps from haustral folds and the colon wall. The volumetric features gradient concentration (GC) and directional gradient concentration (DGC) are used for reducing FPs.

Based on the assumption that polyps are composed of small, approximately spherical patches, Tomasi and Göktürk [123] have designed a method of locally fitting a sphere to the isosurface of each voxel on the colon wall. Groups of voxels having many neighboring spheres are considered as polyp candidates. Kiss et al. [69] have utilized normal and sphere fitting as the references to extract some geometric features on the polyp surfaces. Wang et al. [130] have introduced a new shape description global curvature for polyp detection. All these shape based methods are sensitive to the irregularity of the colon wall and therefore

share a relatively high FP rate, which is undesirable.

Göktürk et al. [36] have presented a statistical approach that uses support vector machines to distinguish the differentiating characteristics of polyps and healthy tissue, and use this information for the classification of the new cases. Acar et al. [1] have proposed a CAD scheme using an edge-displacement field to analyze and improve the polyp detection. Näppi et al. [97] have employed a conditional morphological dilation strategy to extract the suspected regions. The FPs are reduced by further analyzing three shape features from these suspicious regions. Yao et al. [135] have explored image segmentation methods in CAD to reduce the FPs. This method is based on a combination of knowledge-guided intensity adjustment, fuzzy c-mean clustering, and deformable models. The volumetric features computed from the segmentation can further reduce FPs.

1.2.3 Virtual Dissection

Virtual dissection, also called virtual flattening, is an innovative technique whereby the 3D model of the human organ is virtually unrolled, sliced open, and displayed as a flat surface, similar to a physical pathologic specimen. This technique has the potential to reduce the evaluation time by providing a more rapid 3D image assessment than a 3D fly-through navigation. A disadvantage of virtual dissection is the potential for distortion of lesions and normal anatomy.

The application of virtual dissection technique for colon data sets may ultimately improve the accuracy by reducing blind spots presented with 3D endoluminal displays and by reducing reader fatigue. Numerous algorithms have been investigated to alleviate the distortion that inevitably results from the virtual straightening and flattening of curved colonic sections.

The straightforward method [129] starts with uniformly re-sampling the colonic central path. At each sampling point, a cross section orthogonal to the path is computed. The central path is straightened and the cross sections are unfolded and re-mapped into a new 3D volume. The iso-surface is then extracted and rendered. In this method, nearby cross sections may overlap at high curvature regions. As a consequence, a polyp might appear twice or be missed completely in the flattened image. Balogh et al. [6] have presented an iterative method to correct cross sections, using two consecutive ones at a time. Their method was tested both on artificial and cadaveric phantoms.

Wang et al. [18, 127, 128] have utilized electrical field lines generated by a local charged path to generate curved cross sections instead of planar sections, which is called soft straightening. If the complete path is charged, then the cross sections tend to diverge, avoiding overlaps. However, due to the expensive computation of the global charge, the authors only locally charge the path, which cannot guarantee that the curved cross sections do not intersect each other any more. Zhang et al. [141, 142] have developed a fast algorithm for soft straightening of the colon that greatly accelerates the unraveling process based on the interpolation of representative electronic force lines. They also suppress the geometric distortion associated with soft straightening of the colon by moderately adjusting curved

cross sections, which is equivalent to appropriately modify the underlying electronic field.

Fletcher et al. [32] have proposed a method to use isometric volume rendering, which did not require a colon centerline trace. They have concluded that this method, compared with standard 2D axial and 3D endoluminal reviews, has the potential for improved detection of lesions with specific morphologic characteristics. However, the navigation of the reconstructed images is reportedly time consuming. Paik et al. [99] have used cartographic projections to project the entire solid angle of the camera. This approach samples the solid angle of the camera, and maps it onto a cylinder which is finally mapped to the image. However, this method causes distortions in shape.

Bartrolí et al. [8] have proposed a method to move a camera along the central path of the colon. For each camera position a small cylinder tangent to the path is defined. Rays starting at the cylinder axis and being orthogonal to the cylinder surface are traced. The cylinder is then opened and mapped to a 2D image. The result is a video where each frame shows the projection of a small part of the inner surface of the colon onto the cylinder. This avoids the appearance of double polyps since intersections can only appear between different frames. However, this approach does not provide a complete overview of the colon. They have presented a new two step technique to deal with double appearance of the polyps and the nonuniform sampling problems [7]. First, curved rays are cast along the negative gradient of the distance map from the central path of the colon, which returns the distances between the camera and the intersection points on the colon surface. Then, the height field is unfolded and the nonlinear 2D scaling is applied to achieve area preservation. However, it is important to this method that the central path is smooth and has as many linear segments as possible.

Haker et al. [43] have proposed a method based on the discretization of the Laplace-Beltrami operator to flatten the colon surface onto the plane in a manner which preserves angles. The flattened colon surface is colored according to its mean curvature. A morphological method is used to remove minute handles resulting from the segmentation algorithm, because their algorithm requires the input surface to be a topologically open-ended cylinder. However, the color-coded mean curvature of the extracted surface is not efficient for polyp identification, and it requires a highly accurate and smooth surface mesh to achieve a good mean-curvature calculation.

Hoppe et al. [58] have compared virtual colon dissection with axial interpretation and conventional colonoscopy. They concluded that virtual colon dissection may facilitate detection of colonic polyps in isolated cases, its detection rate is not superior to axial interpretation and conventional colonoscopy, which is mainly attributable to failed rendering of insufficiently distended colonic segments or regions with residual feces. With further improvement of path-finding, and image segmentation, however, virtual colon dissection has the potential to be useful interpretation tool for CT colonography.

Recently, Johnson et al. [61] evaluated virtual dissection in a phantom and in a small patient sample with endoscopic correlation. All 144 polyps in the phantom and all 20 clinically proved polyps in the patients were visible in retrospect. These results indicate a high potential for improved polyp detection with this technique.

1.2.4 Direct Volume Rendering

The term volume rendering [23, 26, 82, 85] describes a set of techniques for rendering three-dimensional, that is, volumetric data. The two major approaches to volume rendering are rendering an isosurface corresponding a given iso-value [86, 114], and direct volume rendering (DVR). A direct volume rendered image with a human abdomen data set is shown in Figure 1.2. DVR methods create images of an entire volumetric data set, without concentrating on, or even explicitly extracting surfaces corresponding to certain features of interest. DVR requires an optical model [90] for describing how the volume emits, reflects, scatters, or occludes light. In general, DVR maps the scalar field constituting the volume to certain optical properties such as color and opacity, and integrates them along viewing rays cast into the volume. The corresponding integral is known as the *volume rendering integral*. For real-time volume rendering, the emission-absorption optical model is usually used, in which a volume is viewed as being comprised of particles at a certain density that are only able to emit and absorb light. In this case, the scalar data constituting the volume denotes the density of these particles. Mapping density values to optical properties is achieved through a *transfer function* [67, 104], which determines how different structures embedded in the volume appear in the final image. That is, transfer functions perform two tasks of identifying different objects via *classification* [85], and subsequently assigning *optical properties* to these objects.

Transfer functions for classification and mapping of volume densities to optical properties are an extremely important part of DVR. Objects in a volume are usually identified using a pure opacity transfer function with additional optical properties specified separately. However, the most common type of transfer function is simply a 1D lookup table in the domain of volume densities that stores RGBA values for colors and opacities. Separable multi-dimensional transfer functions have also been in use for a long time, incorporating gradient magnitude as the second dimension in addition to volume density [82]. Recently, more general multi-dimensional transfer functions [67] have become a very important tool for distinguishing different objects contained in a volume. They can be used in interactive volume rendering on graphics hardware [70]. Transfer functions in the domain of principal curvature magnitudes have also proven to be very powerful for highlighting and identifying different shape structures [47, 113] and non-photorealistic volume rendering [68]. Pre-integrated volume rendering [27] substitutes a 1D transfer function lookup table by a 2D pre-integrated table, and decouples the frequencies contained in the scalar volume from the frequencies in the transfer function. It thus allows to achieve high-quality results even with low sampling rates.

Illumination and shading refer to well-known techniques in conventional computer graphics to greatly enhance the appearance of a geometric model that is being rendered. Shading tries to model effects like shadows, light scattering, and absorption that occur in the real world when light falls on an object. Traditional local illumination models for surface lighting can be easily adapted to volumetric representations. Local illumination models use the notion of a normal vector, which describes the local orientation of a surface.

Such an illumination model calculates the reflection of light as a function of this normal, the viewing direction, the angle of incidence, and a couple of material properties. Almost any surface illumination model can be used in DVR by substituting the surface normal by the normalized gradient vector of the volume.

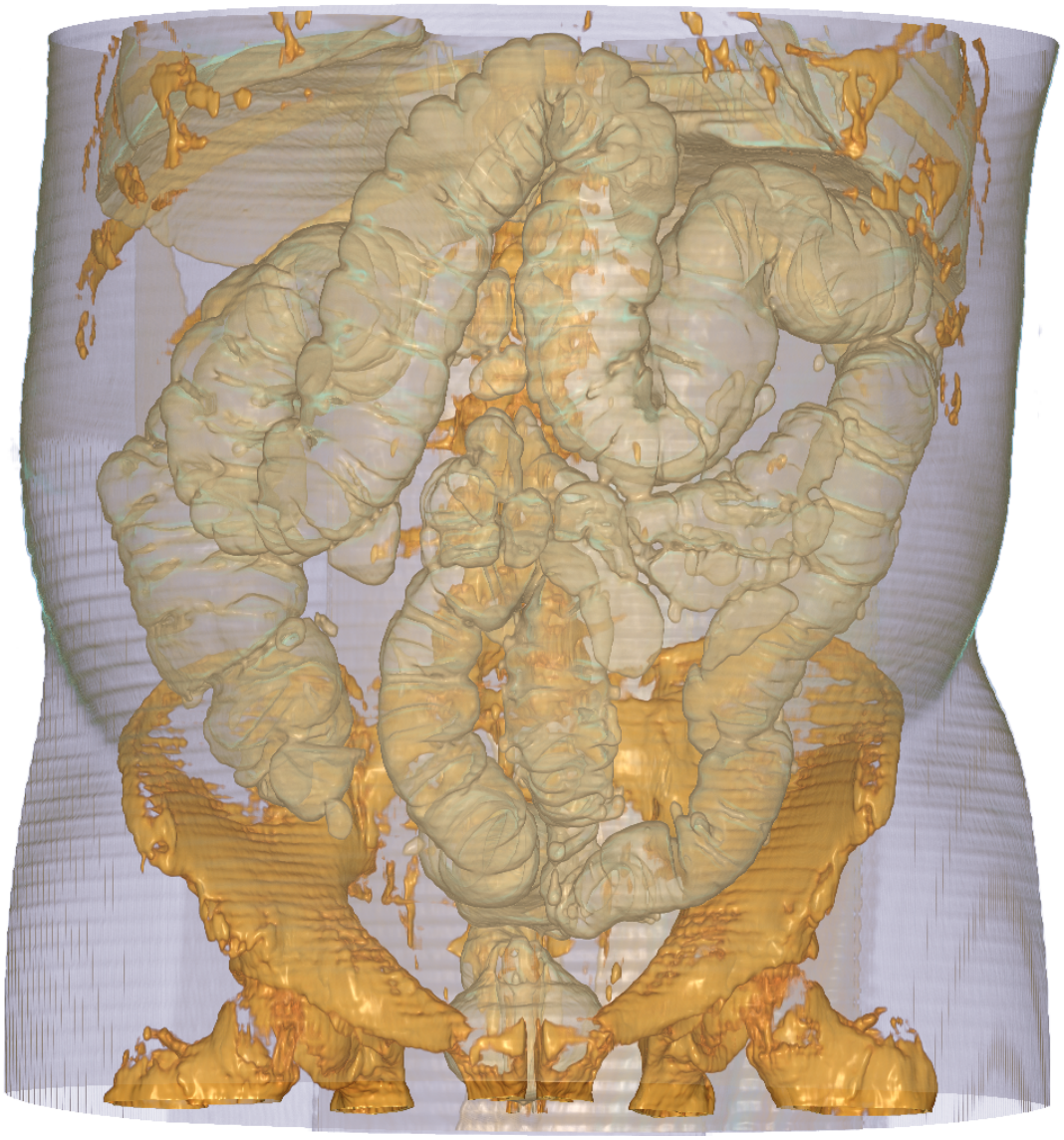


Figure 1.2: Direct volume rendering of a human abdomen data set with a semi-transparent transfer function.

DVR can provide high-quality endoscopic views for virtual endoscopy applications. Algorithms for DVR generally fall into two categories: image-order algorithms (for example, ray-casting [83]) and object-order algorithms (for example, splatting [133] or shear-warp [76]). The ray-casting algorithm can produce high quality images, and can achieve

an interactive rendering speed using graphics hardware. Unfortunately, a major drawback of current consumer graphics hardware is the limited amount of on-board memory, which imposes limits on the size of volumetric data sets that we can render at adequate update rates.

VC fly-through navigation requires the mimicked endoscopic view to be rendered in real-time. However, the typical size of a contemporary clinical 16bit CT data set is about 512 MB. It is still a challenge to render such a large volumetric data set in real-time at a high resolution, even with the help of the contemporary graphics hardware. Real-time rendering of large data sets equal to or larger than 512 MB using the image-order algorithm is currently infeasible unless super-computers [71, 101] or PC clusters [95, 118] are used. Parker et al. [101] have shown that it is feasible to perform interactive iso-surface rendering of the full resolution Visible Woman data set with brute-force ray tracing on an SGI Reality Monster. They achieved up to 20 fps when utilizing 128 processors. Kniss et al. [71] have presented a hybrid volume renderer, which can render a full resolution time-varying data set, such as the Raleigh-Taylor fluid flow data set, at nearly 5 fps on a 128-CPU, 16-pipe SGI Origin 2000 with IR-2 graphics hardware. Muraki et al. [95] have proposed a scalable PC cluster system designed specially for simultaneous volumetric computation and visualization, using compositing hardware devices and the latest PC graphics accelerators. Strengert et al. [118] have described a system for the texture-based direct volume rendering of large data sets on a PC cluster equipped with GPUs. Hierarchical wavelet compression is applied to increase the effective size of volumes that can be handled. However, these large scale solutions do not fit in the needs and capacities in an ordinary medical environment.

Various approaches have been developed to deal with large data sets on PCs. The volume is usually subdivided into blocks, which are usually organized using an octree structure [79]. Data compression is another natural approach to deal with large data sets. Guthe et al. [41, 42] have presented a method for rendering large data sets at interactive frame rates on standard PC hardware. The volumetric data set is converted into a compressed hierarchical wavelet representation in a preprocessing step. During rendering, the wavelet representation is decompressed on-the-fly and rendered using texture mapping hardware. The level of detail used for rendering is adapted to the local frequency spectrum of the data set and its position relative to the viewer. However, the wavelet compression method degrades the performance and quality of the rendering results. Hong et al. [52] have proposed a feature preserved volume simplification method, which produces results visually similar to the original data set.

Ghosh et al. [34] have utilized a multi-board scheme for rendering large volumetric data sets interactively. They implemented an image-partitioned rendering method by loading the entire volume on all available boards, but restricting the range of the image and depth buffers to be filled by each board. They achieved 24 FPS for rendering the Visible Male on one PC with four VolumePro 1000 boards [103]. The limitation of this method is that the size of the volumetric data set that can be rendered is limited by the memory size of the single board.

1.3 Contributions

We contribute to the visualization and CAD research by presenting a novel pipeline for CAD of colonic polyps which integrates texture and shape analysis with volume rendering and conformal colon flattening. Using our automatic method, the 3D polyp detection problem is converted into a 2D pattern recognition problem. By conformal virtual colon flattening the entire inner surface of the colon is displayed as a single 2D image, which is efficient for polyp detection. The final detection results are stored in the 2D image, which can be easily incorporated into any VC system to highlight the polyp locations. Our VC system uses a novel GPU-based volumetric ray-casting algorithm to generate the VC endoscopic view, which is designed to fulfill the requirements of virtual endoscopy applications. In summary, our contributions include:

- Utilizing a simulation method to estimate the percentage of the colon surface missed in the OC and VC examinations. Our simulation study reveals that the standard OC and VC examinations lack adequate visualization of the entire colon surface.
- Utilizing a statistical method for colon segmentation to handle the partial volume effect. The entire colon can be automatically segmented and cleansed. The topological noise is removed by a 3D region growing algorithm based on the concept of *simple point*, allowing a topologically simple colon surface to be extracted.
- Presenting a GPU-based object-order ray-casting algorithm for the rendering of large volumetric data sets on commodity computers. We also exploit the cooperation and trade-off between the GPU and the CPU to obtain further acceleration. The perspective endoscopic view for virtual endoscopy applications can be rendered in real-time at a very high resolution. The efficiency of our algorithm is demonstrated using contemporary clinical 16bit CT data sets and the *Visible Human* data sets.
- Proposing a general method for conformal surface flattening by computing the conformal structure of the surface based on the Riemann surface theory. The global distortion from the 3D surface to the parametric rectangle is minimized, which is measured by the harmonic energy. It is angle preserving, so the shape of colonic polyps is preserved on the flattened colon image.
- Integrating texture and shape analysis with volume rendering and conformal colon flattening for CAD of colonic polyps. Our system is 100% sensitive to polyps with a very low FP rate. The detection results can be used to highlight the polyp locations in the VC system, helping physicians detect polyps faster and with higher accuracy.

In summary, we provide a GPU-based ray-casting algorithm to support real-time virtual endoscopy applications using large volumetric data sets. We propose a conformal colon flattening algorithm to display the entire 3D colon surface as a single 2D image. We

present a novel pipeline for computer-aided colonic polyp detection. We integrate conformal colon flattening and CAD results with our VC system to improve the performance and efficiency of VC. We demonstrate that computer-aided diagnosis in combination with virtual endoscopy is a promising alternative to the conventional endoscopy.

1.4 Outline

The dissertation is organized as follows. The percentage of the covered colon surface during a routine optical colonoscopy is investigated by a simulation method in Chapter 2 [57]. Chapter 3 introduces segmentation and digital cleansing algorithms to handle the partial volume effect and topological noise [54, 56]. In Chapter 4, we discuss the theory and algorithm to flatten the 3D colon surface with conformal mapping and minimizing the global distortion [50, 51]. Chapter 5 describes how to accelerate volume rendering for virtual endoscopy applications using very large volumetric data sets [53, 55]. Furthermore, the pipeline of our computer-aided detection for colonic polyps using conformal colon flattening and volume rendering is presented in Chapter 6 [54, 56]. Finally, concluding remarks are drawn and ongoing research work is summarized in Chapter 7.

Chapter 2

Colonoscopy Simulation

2.1 Introduction

OC is widely accepted as the gold standard for colonic polyp screening, which requires adequate visualization of the entire colonic surface. It frequently happens that sizable areas behind haustral folds are not inspected, although physicians sometimes depress the haustral fold to look behind. Moreover, the hepatic and splenic flexures and other sharp bends are blind spots. Recently, prospective back-to-back or "tandem" colonoscopy studies have reported miss rates for 10 *mm* adenomas ranging from 0% to 6% [111]. VC has proven to be a useful tool for colonic polyp screening. Researchers at Stanford have shown that if all surfaces of the colon lumen have been seen, 3D endoscopic navigation has a higher sensitivity of polyp detection than viewing only 2D axial images [10, 105]. However, VC shares the same problem with OC examination. In the VC examination, the radiologist needs to navigate in both antegrade (from rectum to cecum) and retrograde (from cecum to rectum) directions to improve the degree of surface coverage of the inspection. However, the regions behind haustral folds and around sharp bends still may be missed, unless the physician specifically moves the virtual camera to obtain better views of the mucosa.

In this chapter, we investigate the colon surface visibility coverage using a simulation method to estimate the percentage of the colon surface is missed in the OC and VC examinations. Previously Mori et al. [92] have presented a method using different colors to mark the regions that had been observed versus the unobserved regions during a VC endoscopic examination. This method worked for both surface rendering and volume rendering. Mori et al. have presented two reasons why a given area could be considered unobserved. First, the region does not ever appear on the display, which was termed as a physical factor, and second, it was unobserved due to human error or carelessness, which was termed as a psychological factor. They did not consider performing difficult eye-tracking when calculating visualized surfaces. Kreeger et al. [73] have developed a technique that also marks visualized surfaces during volume rendering endoscopic navigation. The unobserved areas are automatically detected and sorted for quick examination. However, they simulated the OC using a pinhole camera model, while a colonoscope uses a fisheye lens. In this dissertation,

we focus on using a fisheye camera model to simulate the OC examination and comparing the simulation results with the VC examination. We also consider that in OC the camera is not on the colon centerline but rather a hugging corner path. However, we do not consider the psychological factors and the eye tracking problem.

2.2 Fisheye Camera Calibration

Before we simulate the OC, we first need to calibrate the colonoscope. The colonoscope has a fisheye lens. In the computer graphics and computer vision communities, lenses are simplified as the pinhole (linear) model. It depicts the relationship between 3D points and their 2D projections. The perspective projection of a pinhole camera can be described by the following formula:

$$r = f \tan(\theta) \tag{2.1}$$

where θ is the angle between the optical axis and the incoming ray, r is the distance between the image point and the principal point, and f is the focal length.

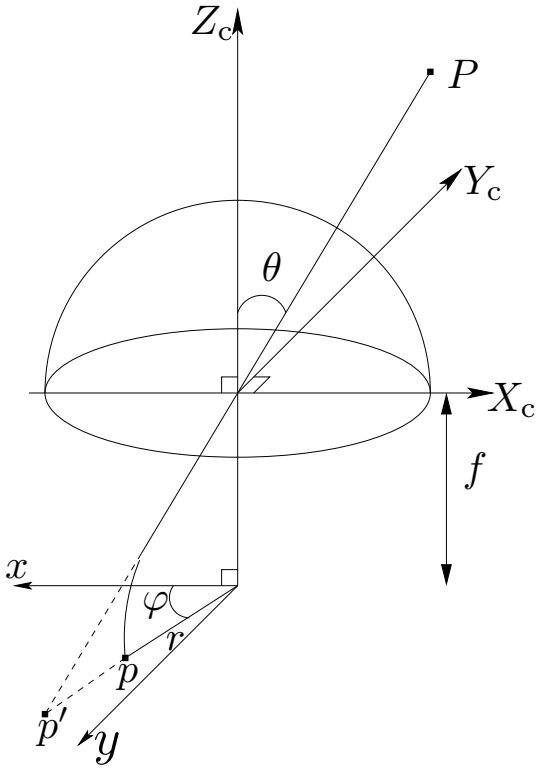


Figure 2.1: Fisheye camera model.

However, the pinhole model is insufficient in the presence of various distortions. Basically, a camera lens could have several types of distortions, including radial distortions and tangential distortions. In most cases, the most salient problem comes from the radial distortion (i.e., the displacement of a pixel is dependent on its distance to the center of the image). For a colonoscopy, radial distortions become more serious because it is a wide-angle lens. To calibrate it, we used the algorithm proposed by Kannala and Brandt [64]. The fisheye camera model is shown in Figure 2.1. Fisheye lenses are usually designed to obey one of the following models:

- Stereographic projection:
$$r = 2f \tan(\theta/2) \tag{2.2}$$
- Equidistance projection:
$$r = f\theta \tag{2.3}$$
- Equisolid angle projection:
$$r = 2f \sin(\theta/2) \tag{2.4}$$
- Orthogonal projection:
$$r = 2f \sin(\theta) \tag{2.5}$$

However, these models could be represented by a generic model using the Taylor series:

$$r(\theta) = k_1\theta + k_2\theta^3 + k_3\theta^5 + k_4\theta^7 + \dots \tag{2.6}$$

To further simplify the equation, the basic idea is to model the radial displacement (distortion) about the image center using a polynomial function: $r(\theta) = k_1\theta + k_2\theta^3$. This approximately captures the possible radial distortions of the colonoscope.

First, we need to understand the imaging (projection) process from a 3D point to its 2D projection. During this projection process, we first transform the camera coordinates of the point into the image pixel coordinates. Then, the normalized image coordinates are computed as:

$$\begin{cases} x = r(\theta)\cos(\varphi) \\ y = r(\theta)\sin(\varphi) \end{cases} \tag{2.7}$$

where (θ, φ) is the polar representation for the 2D projection (x, y). In the end, we apply an affine transformation to compute the actual pixel location:

$$\begin{cases} u = m_u x + u_0 \\ v = m_v y + v_0 \end{cases} \tag{2.8}$$

where (u_0, v_0) is the principal point (image center) , m_u and m_v give the number of pixels per unit distance in the horizontal and vertical directions, respectively. The goal of calibration is to estimate the exact value of the six parameters: $(k_1, k_2, m_u, m_v, u_0, v_0)$.

We have used a planar board with a specific target pattern to collect the data for calibration. Here, each black dot represents one 3D point. Since the points are arranged in a specific way, we know their 3D locations up to an unknown Euclidean transformation. Then, in the image space, we have clusters of pixels representing the projection of these 3D points. The projection center in sub-pixel accuracy is computed as the center of each cluster. Figure 2.2 depicts a single frame of our specific target pattern captured by the fisheye endoscope.

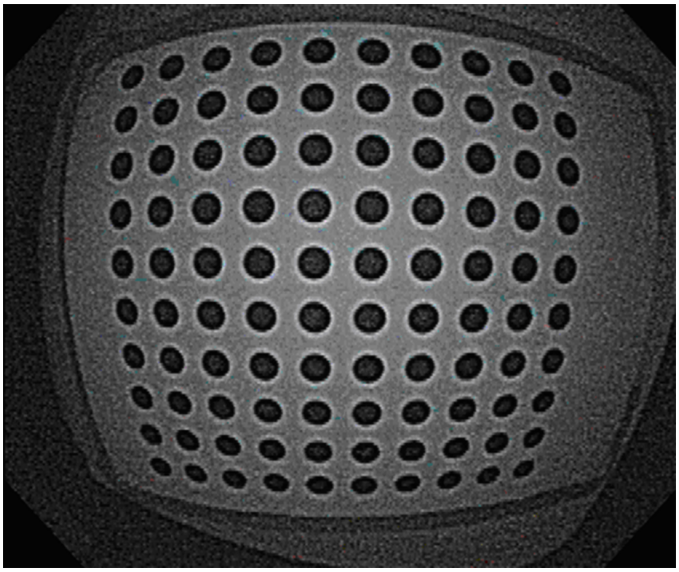


Figure 2.2: A frame of the calibration pattern captured by the colonoscope.

Then, a four-step procedure is conducted for the calibration using M feature points obtained in N views. Here internal parameters are the ones intrinsic to the endoscope. They are not altered by the position and orientation of the camera. And the position and orientation of the endoscope are referred as the external parameters. The four steps are described as follows:

1. Initialization of internal parameters;
2. Refinement of internal parameters;
3. Initialization of external parameters;
4. Minimization of projection error.

For the first step, the initial values for k_1 and k_2 are obtained by fitting to the desired projection with the nominal focal length and field of view provided by the manufacturer. The initial values of m_u, m_v, u_0, and v_0 are estimated by fitting a field of view of the fisheye lens to an ellipse. Because the target pattern is a planar surface, the feature points under different views are related by a homography matrix, which are used to refine the

estimation of these internal parameters. In addition, using the homography but with internal parameters fixed, we could estimate the external parameters. In the end, all parameters are fed into an energy function, which computes the sum of projection errors. The result is obtained by minimizing the energy using the Levenberg-Marquardt algorithm.

2.3 Optical Colonoscopy Simulation

We employ virtual models of the colon obtained by VC CT scans in order to simulate the OC examination. Before the simulation, the virtual colon is electronically cleansed and segmented from the CT scans, and the colon surface is extracted using a dual contouring method [139] and stored using a triangle mesh. In previous OC simulation methods, cameras are usually placed on the colon centerline, while a real colonoscope tends to hug the corner, especially at the sharp bends. Moreover, from the layout of the simulated colonoscope distal end (shown in Figure 2.3), it is noted that the center of the fisheye lens cannot touch the colon surface and is not at the center of the distal end. During the OC examination, endoscopists typically keep the objective lens away from the colon wall. We use the average between 10 *mm* and 2.8 *mm* as the radius of the simulated distal end.

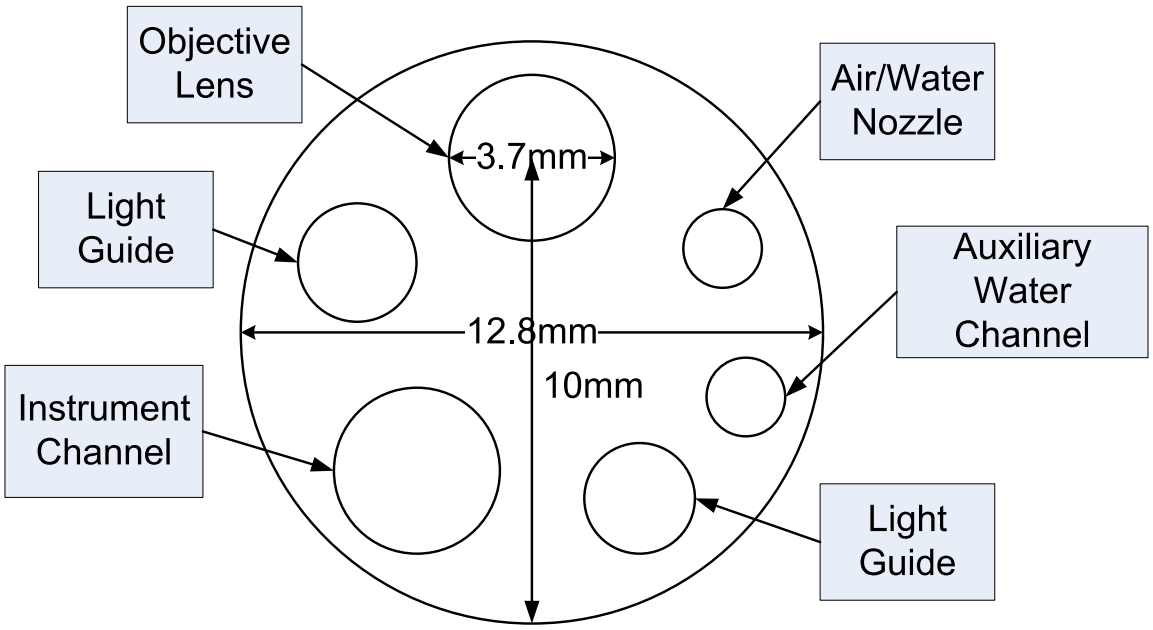


Figure 2.3: Layout of the simulated colonoscope distal end.

We use an efficient distance mapping algorithm [126] to compute the hugging corner shortest path. It treats the volume as a graph, and every voxel is a node. An edge is added between every two 26-connected voxels. Then, the hugging corner shortest path can be computed using the Dijkstra shortest path between the two colon ends (rectum and

cecum). In order to keep the hugging corner shortest path 6.4 *mm* away from the colon wall, we computed the distance from boundary for every voxel using the segmented colon. Voxels with the distance less than 6.4 *mm* are not considered in our algorithm to account for objective lens moving away from the colon wall to simulate the OC examination path, as shown in Figure 2.4.

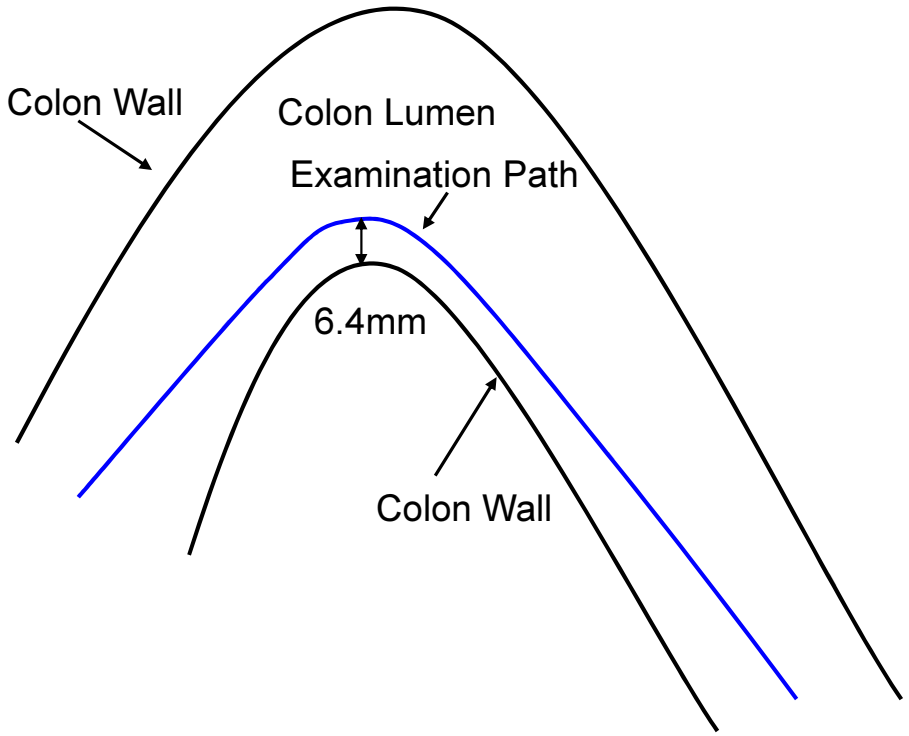


Figure 2.4: The OC examination path hugs the corner at a sharp turn.

After the hugging corner shortest path is obtained, a large number of fisheye cameras are evenly placed along the path in retrograde direction. At each camera position, the ray casting algorithm is used to generate the endoscopic view. The ray direction of each pixel can be determined using the calibrated lens parameters. Since a fisheye camera model is used in our simulation, Mori et al.'s method to detect unobserved regions cannot be used in this situation. Instead, a brute-force ray-casting algorithm is used to determine whether a triangle is observed or not. Because all triangles are very small (triangle area $< 1\ mm^2$), we define that a triangle is observed if it is intersected by at least one ray, which means that it is at least partially displayed on the screen. It is noted that the viewport of the colonoscope is octagonal, because the shape of the objective lens is an ellipse. Therefore, in our simulation the image plane is elliptical. Moreover, we do not consider the unobserved regions caused by psychological factors during fly-through navigation. After all of the cameras are processed, the unobserved triangles are counted. Consequently, the percentage

of covered colon surface can be estimated. Surface coverage at this point may serve as an estimation of readily visualized mucosa at a standard OC examination.

2.4 Virtual Colonoscopy Simulation

The covered colon surface for the VC examination can be computed in a similar way, using the colon centerline and a pinhole camera model. An advantage of using a pinhole camera model is that we can use graphics hardware to accelerate the simulation process. We use OpenGL extension occlusion query to test whether a triangle is observed from a specified camera position. This extension can return the number of samples that pass the depth and stencil tests. Thus, for each triangle we can use occlusion query to obtain how many pixels can be observed at a camera position. At each camera position, the algorithm is described as follows:

1. We first disable the depth test, and render all triangles. For each triangle, we use occlusion query to obtain the number (N_{total}) of pixels that can be observed from the current camera position. In fact, the observed pixels constitute the triangle on the image plane. If this number is zero, the triangle is outside the view frustum and should be clipped, which is unobserved from the current camera position.

2. We then enable the depth test, and render all observed triangles to obtain the depth buffer.

3. For each observed triangle, we perform occlusion query again to obtain the number ($N_{observed}$) of pixels that can be observed in the final image.

   (a) If $N_{observed}$ is zero, the triangle cannot be observed from the current camera position.

   (b) If $N_{observed}$ is equal to N_{total}, the triangle is fully visible.

   (c) Otherwise, the triangle is partially visible.

For the VC, the computation is also performed for both antegrade and retrograde directions. It is noted that the field of view for VC is 90 degree which is very different from the real colonoscope. We use the same depth of field parameters of a real colonoscope for both OC and VC simulations.

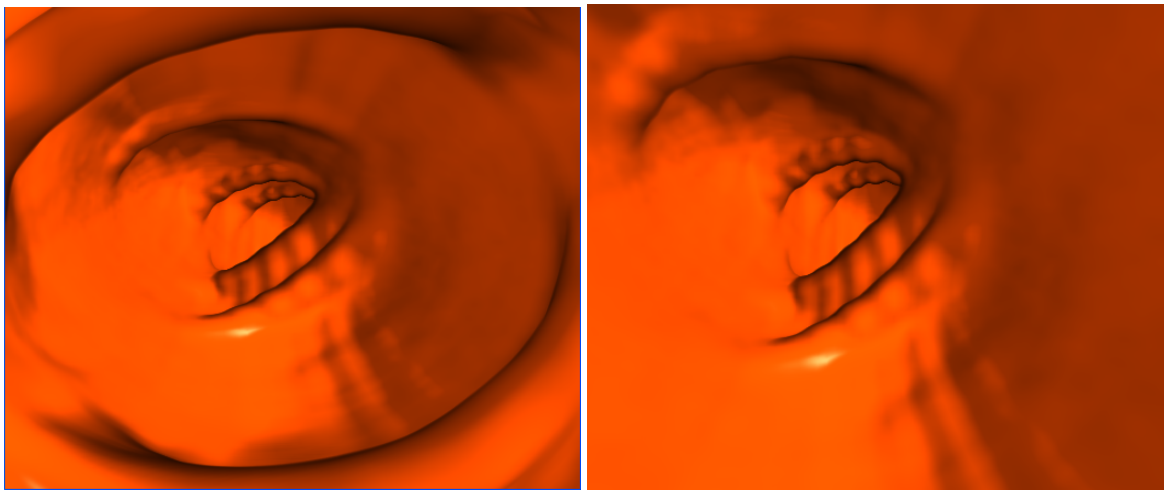
2.5 Results

We used the above approach to calibrate an Olympus colonoscope (Model: CF-Q160L) with a field of view of 140 degrees using a checker board image. The specification of the calibrated colonoscope is shown in Table 2.2. The image acquired is about 525×440 pixels. The visible region is enclosed by an ellipse, centered at $(267.05, 215.01)$ and with

the radius along the X/Y direction as $309.08/294.92$. The calibration result for the six parameters is $(13.5747, -2.2982, 25.5903, 25.6000, 274.2596, 220.6445)$. An endoscopic view for the simulated OC using the six parameters is shown in Figure 2.5 (a), and the corresponding endoscopic view for VC using perspective projection is shown in Figure 2.5 (b). We can see that more information can be obtained when using a fisheye camera.

Table 2.1: Specification of the calibrated colonoscope.

Field of View	140 $degree$
Depth of Field	3 to 100 *mm*
Distal End Outer Diameter	12.8 *mm*
Minimum Visible Distance	5 *mm* from distal end
Objective Lens Inner Diameter	3.7 *mm*
Angulation Range	up/down 180 $degree$, right/left 160 $degree$



(a) (b)

Figure 2.5: (a) The simulated endoscopic view for OC and (b) the corresponding view for VC.

We used 52 CT data sets from the National Institute of Health (NIH) and 46 CT data sets from Stony Brook University Hospital (SBUH) to estimate the average colon surface coverage for both standard OC and VC examinations. The extracted colon triangle meshes usually consists of about one million triangles. The simulation results are shown in Table 2.2:

Simulated OC covered an average of about 77% of the colon surface, primarily missing the backsides of haustral folds and around sharp bends, which is below the average estimation of 90% coverage from 12 experienced endoscopists. In Figure 2.6(a), the covered colon surface obtained after retrograde navigation from rectum to cecum is painted in

Table 2.2: Average percentage covered in simulated OC and VC.

	Number of Cameras	Average Coverage
Simulated OC	1000	76.81
	2000	77.02
VC in retrograde direction	1000	76.94
	2000	77.37
VC in both directions	1000	91.05
	2000	91.10

green. From this figure, we can clearly see that the hepatic and splenic flexures are blind spots, and many regions behind haustral folds are uncovered in the simulated OC.

The VC fly-through navigation in retrograde direction covered an average of about 77% of the colon surface. The combined retrograde and antegrade VC fly-though covered an average of about 91% of the colon surface. From Table 2.2, it is noted that we obtained similar results when a different number of cameras is used in the simulation. In Figure 2.6(b), the covered colon surface obtained after fly-through navigation in both directions is painted in green. We can observe that many regions behind haustral folds and around sharp bends are still missed.

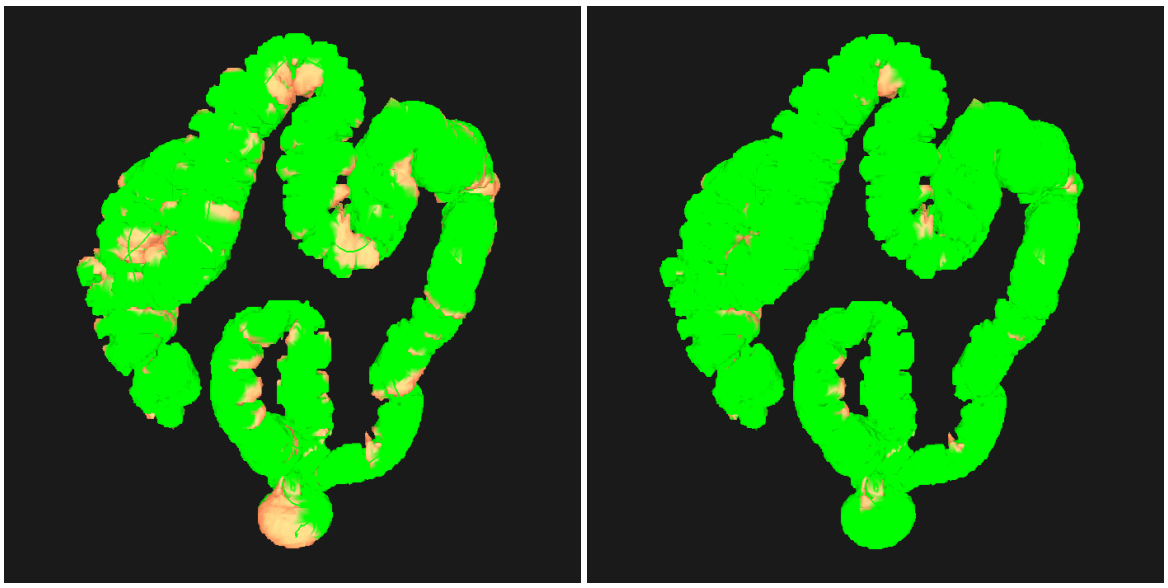
Kreeger et al. [73] proposed a method to automatically detect and sort unobserved patches to cover all clinically significant areas of the colon surface, fly-through navigation in antegrade and retrograde directions as well as examining unobserved patches. The colon surface coverage can be increased to 98% by reviewing the missed regions with an area of 300 mm^2 or larger and can potentially go to 100%.

Virtual colon flattening is an efficient visualization technique for colon polyps screening, in which the entire inner surface of the colon is displayed as a single 2D image. However, this method results in severe distortions. Several methods have been developed that are either area preserving [7] or angle preserving [43, 50]. We have developed an angle preserving method [50], in which the entire colon surface is mapped to a 2D rectangle. The flattened colon image can be easily integrated with the VC system to achieve up to 100% coverage.

2.6 Conclusions

By simulating a wide angle fisheye camera fly-through navigation along the hugging corner shortest path in the retrograde direction, we evaluate the average colon surface coverage in the standard OC examination. The average colon surface coverage is 77% in our simulated OC. Considering that the endoscopists can flex the endoscope end and/or depress the haustral folds to look behind, the real surface coverage of OC may be slightly higher than our estimation. However, it is still not adequate for colonic polyp detection.

We have also compared the simulated OC to the VC examination. A better result is



(a) (b)

Figure 2.6: The covered colon surface are painted in green: (a) the simulated OC, and (b) VC in both directions.

obtained by enabling endoscopic navigation in the VC along the colon centerline in both antegrade and retrograde directions, which can visualize an average of 91% of the colon surface. Combined bidirectional antegrade and retrograde 3D navigation, supplemented by rapid review of missed regions, effectively covers the entire clinically significant colon surface 100%. Virtual colon flattening is another promising technique to achieve 100% surface coverage.

Chapter 3

Segmentation and Digital Cleansing

3.1 Introduction

All current implementations of VC and CAD techniques require a rigorous cleansing of the colon prior to the virtual examination. With some stool residues and colonic fluids remaining during the patient image acquisition, the efficiency of VC and CAD will be lowered dramatically because the residues mimic polyps while the fluids may cover polyps.

Digital cleansing [107] is a relatively new technology that has been under development to remove remnants of stool and residual fluids from the acquired images. First, the patient undergos a less-stressful bowel preparation with oral contrast to tag out the colonic materials, so that the residue stool and fluid have an enhanced image density compared with the colon/polyp tissues [84]. Taking advantage of image segmentation and pattern recognition techniques, a digital cleansing method can identify the tagged colonic materials and restore a cleansed colon lumen for both VC navigation and CAD analysis. Digital removal of opacified residual fluid allows 3D evaluation of colonic mucosa that would have otherwise been obscured. With effective electronic cleansing, the entire anterior and posterior walls can be evaluated on 3D images obtained with patients in both the prone and supine positions.

In general, there exist two major challenges for digital cleansing. The first is the removal of the interface layer between the air and the tagged colonic materials. Due to the partial volume effect, this layer covers the density values of colon tissues and so it is impossible to distinguish the voxels of colonic materials in this layer from that of the colon tissues. Another challenge is the restoration of the CT density values of colon tissues in the enhanced mucosa layer and the removal of the portion with tagged colonic materials. Lakare et al. [78] have introduced a ray-based detection technique with utilizes a predefined profile pattern to detect the interfaces. Chen et al. [16] have explored image gradient information to address the partial volume effect. Zalis et al. [138] have presented a technique of using morphological and linear filters to mitigate the partial volume effect. Cai et al. [15] have developed a novel method based on the analysis of the soft tissue structures submerged in tagged fecal material. Their method is designed to preserve the soft tissue

structures while removing the tagged fecal material. Franaszek et al. [33] have proposed hybrid a algorithm using modified region growing, fuzzy connectedness and level set segmentation. In this chapter, we present an automatic segmentation and digital cleansing framework to handle both partial volume effect and topological noise.

3.2 Partial Volume Segmentation

In this section, we presented a statistical method for colon segmentation to handle the partial volume effect. Instead of labeling each voxel with a unique class type, the percentage of different tissue types are estimated within each voxel [28, 81, 131].

Let the acquired CT image density distribution Y be represented by a column vector $[y_1, y_2, \ldots, y_N]^T$, where y_i is the observed density value at voxel i and N is the total number of voxels in the image. Assume that the acquired image y_i contains K tissue types distributed inside the body. Within each voxel i, there possibly are K tissue types, where each tissue type has contribution to the observed density value y_i at that voxel. Let tissue type k contribute x_{ik} to the observation y_i at voxel i, then we have $y_i = \sum_{k=1}^{K} x_{ik}$. Assume the unobservable variable $x_i k$ follows a Gaussian distribution with mean μ_{ik} and variance σ_{ik}^2. Let $m_i k$ be the fraction of tissue type k inside that voxel, called *mixture*, then we have $\mu_{ik} = m_{ik}\mu_k$ and $\sigma_{ik}^2 = m_{ik}\sigma_k^2$, where $\sum_{k=1}^{n} m_{ik} = 1$ and $0 \leq m_{ik} \leq 1$. Therefore, the observed image density value at voxel i is expressed as:

$$y_i = \sum_{k=1}^{n} m_{ik}\mu_k + \varepsilon_i \tag{3.1}$$

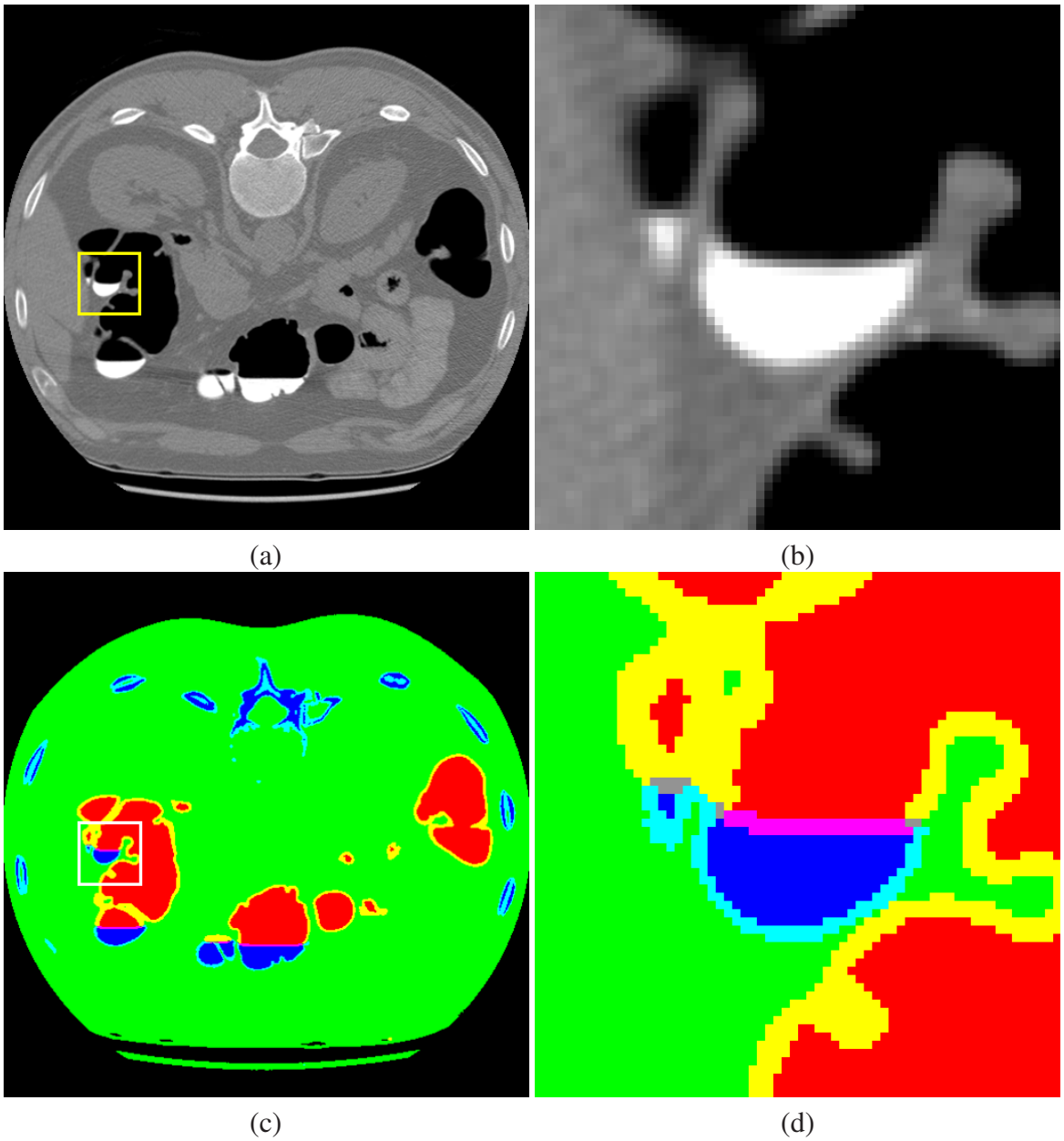
where ε_i is Gaussian noise associated with density value y_i at voxel i with its mean being zero. The probability distribution of sampling $x_i k$, given the parameters m_{ik}, μ_i, σ^2, is

$$Pr(X|M, \mu, \sigma) = \prod_{i,k} \frac{1}{\sqrt{2\pi m_{ik}\sigma_k^2}} exp[-\frac{(x_{ik} - m_{ik}\mu_k i)}{2m_{ik}\sigma_k^2}] \tag{3.2}$$

where $X = [x_1, x_2, \ldots, x_N]^T$ with $x_i = [x_{i1}, x_{i2}, \ldots, x_{iK}]^T$, $M = [m_1, m_2, \ldots, m_N]^T$ with $m_i = [m_{i1}, m_{i2}, \ldots, m_{iK}]^T$, $\mu = [\mu_1, \mu_2, \ldots, \mu_K]^T$, and $\sigma^2 = [\sigma_1^2, \sigma_2^2, \ldots, \sigma_K^2]^T$. This is a well-known problem of parameter estimation from incompletely observed data y_i. The well-established expectation-maximization (EM) algorithm is used to estimate the parameters via conditional expectation and maximization in an iterative manner.

When the mixture information within each voxel is estimated, a voxel can be classified as air, soft tissue, or bone/tagged material if the corresponding mixture is larger than a threshold, say 95%. If a voxel has two major tissue types, it has partial volume effect, and it can be classified as the boundary of these two tissue types. In Figure 3.1, we show the classification result for one CT image, in which air is shown in red, soft tissue is shown in green, bone and tagged material are shown in blue, and the interface between air and

tagged material is shown in pink. Our results demonstrated that the partial volume effects between different types have been precisely detected.



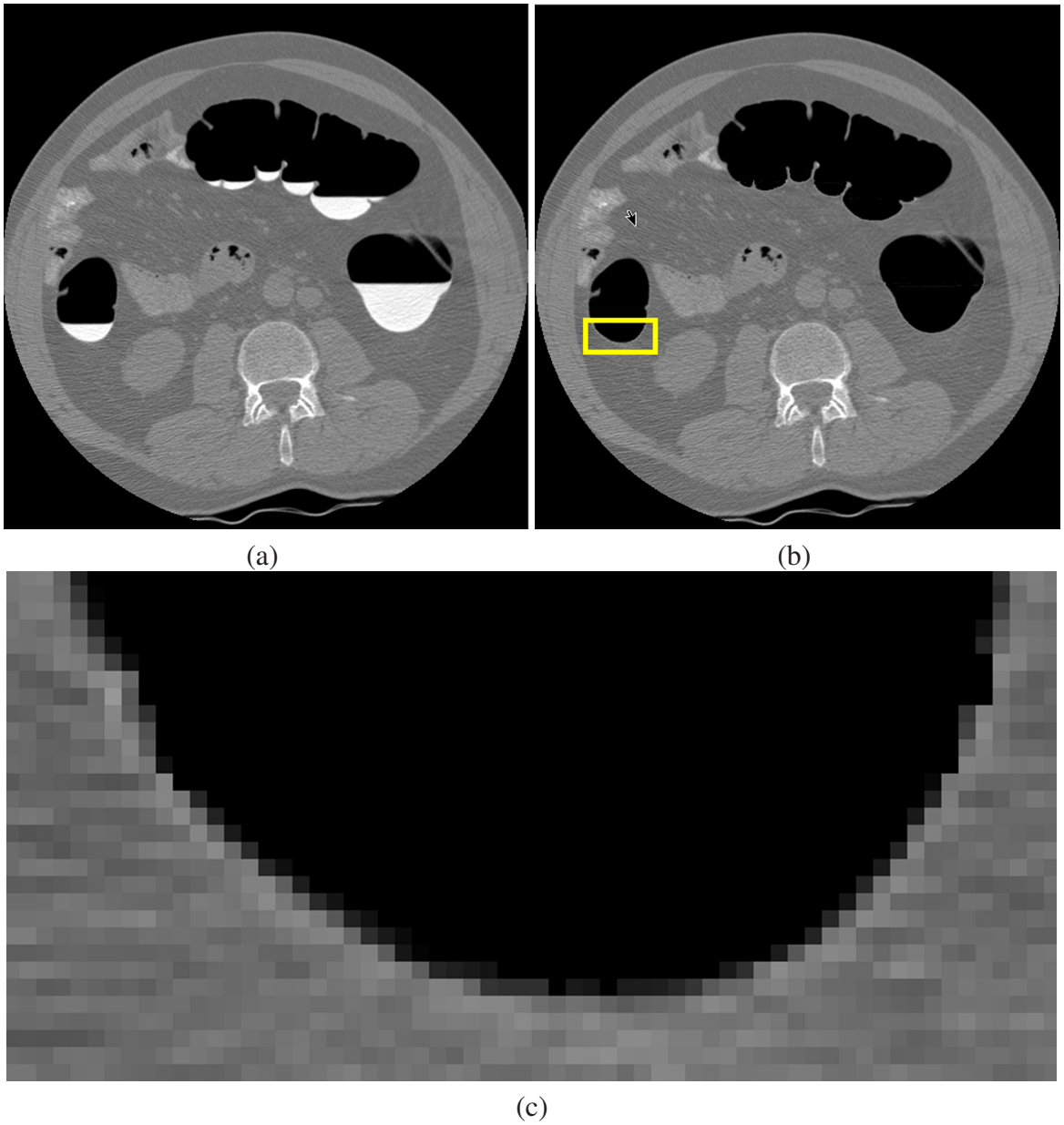
(a) (b)

(c) (d)

Figure 3.1: (a) An original contrast enhanced CT image, (b) A zoomed-in view of the marked rectangle in (a), (c) the classification result based on the mixture information, (d) A zoomed-in view of the marked rectangle in (c).

After classification, we need to segment the colon lumen and identify the interface layer between air and the tagged fluid based on the mixture information. The voxels in the colon lumen are classified as air, mixture of air with tissue, mixture of air with tagged materials,

or mixture of tissue with tagged materials. First, we use a labeling algorithm to extract all connected components for the whole data set. It is noted that the interface layers between air and the tagged fluid should be flat, and must be below its neighboring air component and above its neighboring tagged material component due to gravity. Therefore, we first identify these interface layers and mark them as colon. Then, the air and fluid components connected with these interface layers are also marked as colon.



(a) (b)

(c)

Figure 3.2: The result of digital cleansing. (a) An original CT image (slice), (b) the corresponding cleansed slice, and (c) a zoomed-in view of the marked rectangle in (b).

When we obtained the segmentation of the colon lumen, the tagged material within the colon lumen is cleansed based on Equation 3.3. The idea is that we can subtract the contribution of tagged material from the observed density value and add some contribution of air or soft tissue using the mixture information. The equation to remove the tagged material is expressed as:

$$y_{new} = y_{old} - m_{tag}\mu_{tag} + m_{air}\mu_{air} \tag{3.3}$$

The enhanced mucosa layer can also be restored using a similar equation. After this step, we obtain a segmentation of the colon and a clean colon lumen. One original CT axial image (slice) and the corresponding cleansed slice are shown in Figures 3.2(a) and 3.2(b), respectively, from which we can see that the interface layer has been removed and small features have been well preserved. A zoomed-in view of the region bounded by the yellow box in Figure 3.2(b) is shown in Figure 3.2(c).

3.3 Topological Denoising

After segmentation and digital cleansing, the colon surfaces need to be extracted for visualization and our flattening algorithm. The colon surfaces reconstructed from a CT data set usually have complicated topologies caused by the noise and inaccuracy of the reconstruction methods. In general several spurious handles will be introduced to a surface. This topological noise complicates our flattening algorithm, and introduces large distortions.

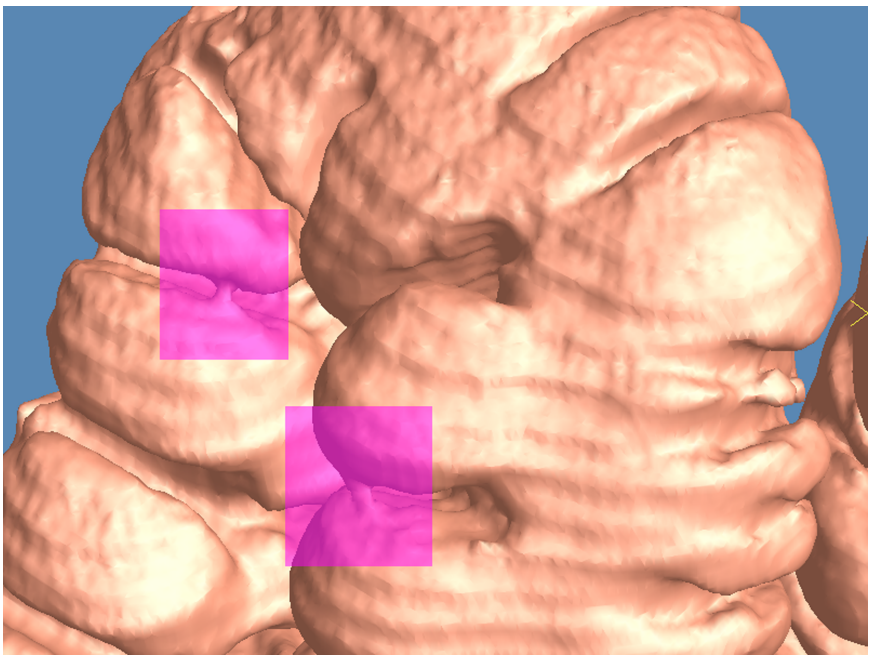


Figure 3.3: A zoom-in view of a colon surface with two handles, shown within the boxes.

It is challenging to locate these handles and remove them using some special "topology surgery". El-sana and Varshney [25] have proposed a topology controlled simplification method for polygonal models. Tiny tunnels are identified by rolling a sphere with small radius over the object. Guskov and Wood [40] have presented a local wave front traversal algorithm to discover the local topologies of the mesh and identify features such as small tunnels. The mesh is then cut and sealed along non-separating cuts, reducing the topological complexity of the mesh. These methods are efficient for tiny handle identification. However, we find that handles are not tiny in our colon data sets as shown in Figure 3.3. We have proposed two approaches to do topological denoising: surface-based method and volume-based method.

3.3.1 Surface-based Method

Intuitively, handles can be identified by locating the shortest loop for each homotopy class. The topology of a closed oriented surface is determined by its number of handles (genus). Two closed curves are *homotopic* if they can deform to each other on the surface. Homotopic equivalence classes form the so-called *homotopy* group, which has finite generators, that is, *homotopy basis*. Each handle corresponds to two generators. A handle can be removed by cutting the handle along one of its generators, and filling the resulting holes as shown in Figure 3.4.

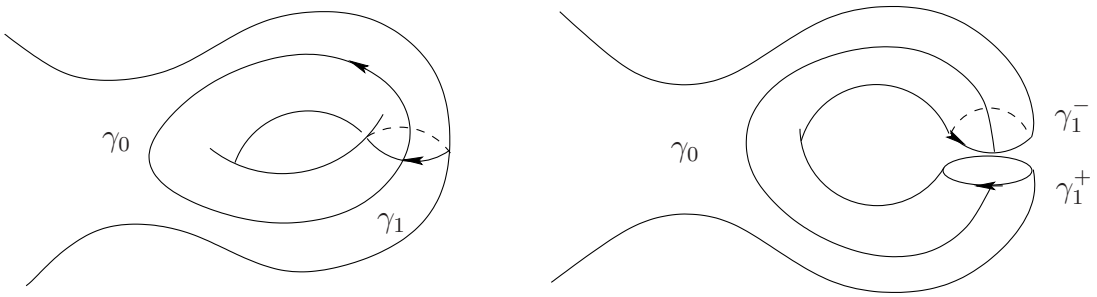


Figure 3.4: (a) Homotopy basis, and (b) topological surgery.

In order to remove a handle, it is highly desirable to locate the shortest loop. It is natural to compute the shortest loop using *universal covering space*. Suppose that $\bar{M}$ and M are two surfaces, then $(\bar{M}, \pi)$ is said to be a covering space of M if $\pi : \bar{M} \rightarrow M$ is a surjective continuous map with every $p \in M$ having an open neighborhood U such that every connected component $\pi^{-1}(U)$ is mapped homeomorphically onto U by π. If $\bar{M}$ is simply connected, then it is said to be a *universal covering space* of M. A simply connected region $\tilde{M} \subset \bar{M}$ is called a *fundamental domain*, if the restriction of π on $\tilde{M}$ is bijective. Intuitively, one can slice M along some curve set (cut graph) to obtain a topological disk (a fundamental domain), and glue fundamental domains coherently to form the universal covering space. For any point $p \in M$, its preimages are the discrete set

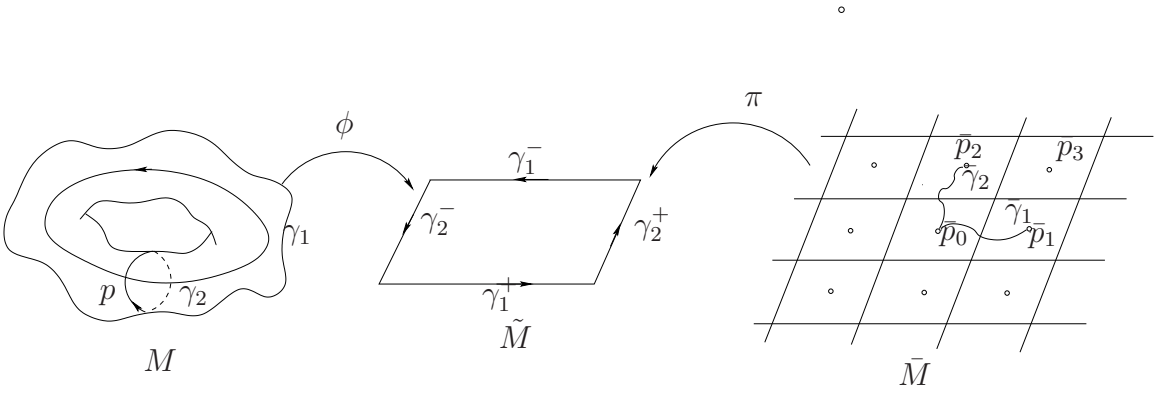


Figure 3.5: Topology concepts: Two curves γ_1, γ_2 on a surface M form a *cut graph*. M is sliced open along the cut graph to become a *fundamental domain* $\tilde{M}$, γ_i is mapped to γ_i^+ and γ_i^-. By gluing many copies of $\tilde{M}$ such that γ_i^+ is glued with γ_i^{-1}, the *universal covering space* $\bar{M}$ can be obtained. $\pi : \bar{M} \to M$ is the projection map. Any point p on M has a discrete preimage set $\pi^{-1}(p) = \{\bar{p}_0, \bar{p}_1, \bar{p}_2, \cdots\}$. Any closed curves through p on M are lifted as curve segments connecting two points in $\pi^{-1}(p)$, e.g., γ_1 is lifted as $\bar{\gamma}_1$, γ_2 is lifted as $\bar{\gamma}_2$. The shortest loops on M correspond to the shortest path on $\bar{M}$.

$\pi^{-1}(p) = \{\bar{p}_0, \bar{p}_1, \bar{p}_2, \bar{p}_3 \cdots\} \subset \bar{M}$. If $\bar{\gamma}_k$ is a curve connecting $\bar{p}_0$ and $\bar{p}_k$ in the universal covering space $\bar{M}$, then $\gamma_k = \pi(\bar{\gamma}_k)$ is a closed loop on M. By going through all end points $\bar{p}_k$, γ_k goes through all homotopy classes. In order to find the shortest loop γ_k, we can find the shortest path $\bar{\gamma}_k$ in the universal covering space instead. Figure 3.5 demonstrates the concepts of fundamental domain and universal covering space using a genus one surface. It illustrates the idea of lifting a loop to a path and converting the shortest loop problem to the shortest path problem. In computational topology, the algorithms to compute cut graph, homotopy basis, and fundamental domain have been well developed [21, 29, 38, 80].

However, it has been proven by Erickson and Har-Peled [29] that this problem is NP-hard. Erickson and Whittlesey [30] have presented a greedy algorithm to compute the shortest system of loops in $O(nlogn)$ time. However, each loop may not be the shortest loop in its *homotopy* group. Given a system of loops, Colin de Verdière and Lazarus [19] have proposed a method to compute the shortest simple loop homotopic to a given simple loop (a loop without self intersection). However, the shortest loop within each homotopy group may not be simple. In our case, the surfaces extracted from the segmented colon data sets usually only have a small number of handles as shown in Figure 3.3. In order to compute the shortest loop, we first simplify the mesh while preserving the topology of the finest mesh. A finite portion of the universal covering space is constructed using the coarsest mesh. The shortest loop is computed in the universal covering space and lifted back to the finest mesh, which approximates the shortest loop on the finest mesh. Our experiments show that this algorithm is manageable in our case.

The main procedure of our denoising algorithm is described as follows:

1. Compute the cut graph and homotopy basis.

2. Simplify the cut graph, then slice along the simplified cut graph to form the fundamental domain.

3. Glue finite copies of the fundamental domain coherently to construct a finite portion of the universal covering space.

4. Compute the shortest loop by finding the shortest path in the universal covering space.

After the shortest loop is obtained, we slice the mesh along the loop and fill the holes to remove a handle. This procedure is repeated until all handles are removed.

3.3.2 Volume-based Method

In this section, we discussed our volume-based method using the concept of *simple point*. Han et al. [44] have presented a topology preserving level set method, which achieves topology preservation by applying the simple point concept from *digital topology* [11]. They conclude that it is only necessary to be concerned with topological changes when the level set function is changing sign. Therefore, in their method the level set function sign changes are only allowed at a simple point. In this method, a cut is obtained at each handle of the digital object to preserve topology. However, this method cannot guarantee that the cut of the handle is minimized. Moreover, solving the partial differential equations in the level set method results in a significant computational burden, especially when it is applied to volumetric data.

3.3.2.1 Simple Point

In this section, we present a new volume based topological denoising algorithm to remove small handles (that is, topological noise) from the segmented colon. Because we already have a segmentation of the colon, we incorporate the simple point concept in a region growing based algorithm to extract a topologically simple segmentation of the colon lumen. A point is *simple* if its addition to or removal from a digital object does not change the object topology. In other words, a point is simple if it is adjacent to just one object component and one background component. We show a 2D example with (4, 8) connectivity in Figure 3.6, in which the red point is a simple point and the yellow point is a non-simple point. To avoid the connectivity paradox, (6, 18), (18, 6), (6, 26) and (26, 6) are four pairs of compatible connectivity used in 3D digital topology. In order to guarantee that the extracted colon surface using our topology preserving dual contouring algorithm [139] is a manifold, 6-connectivity is used for the colon lumen and 18-connectivity is used for the background. Thus, we use the topological numbers [11] corresponding to the compatible connectivity pair (6, 18) to determine whether a voxel is simple. The topological number is equal to the number of connected components within its geodesic neighborhood. Therefore, if both of them are equal to one, the voxel is simple. The following definitions are from [11].

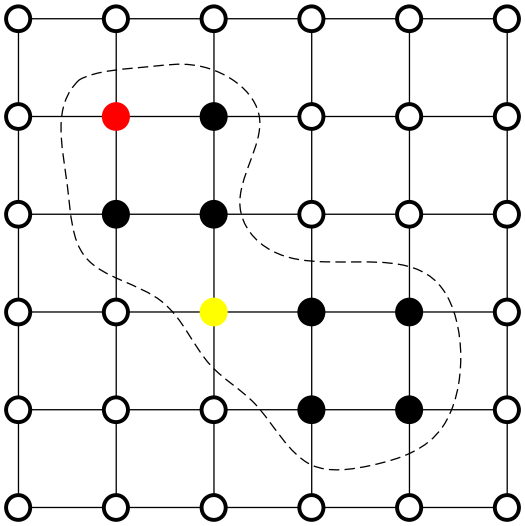


Figure 3.6: Illustration of a simple point (red) and a non-simple point (yellow).

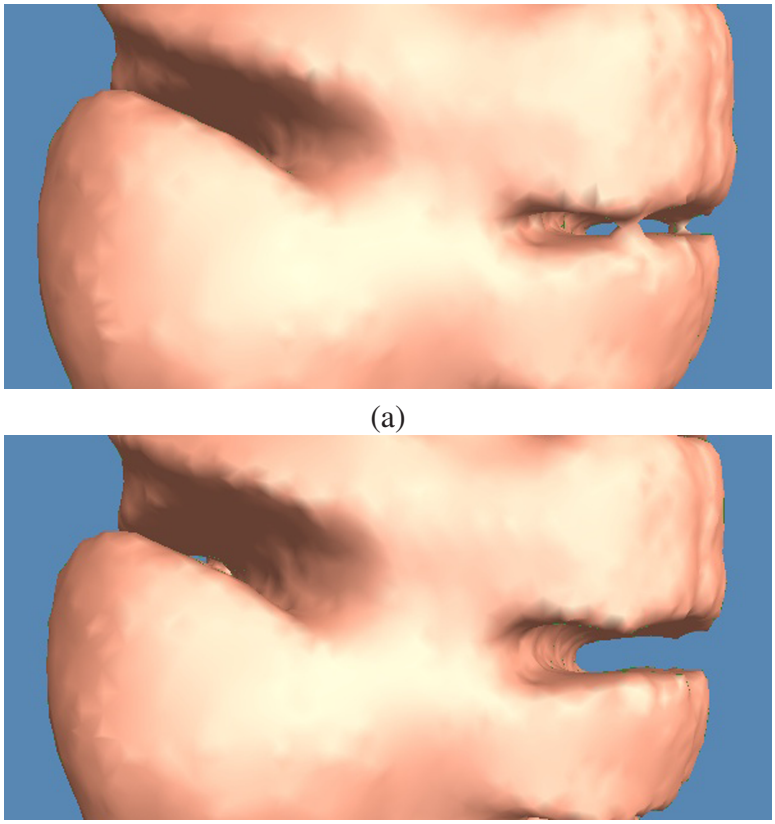
Definition 3.1. *Let $X \subset Z^3$ and $x \in Z^3$. The topological numbers relative to the point x and set X are:*
$T_6(x, X) = \#C_6[N_6^2(x, X)]$ *and*
$T_{18}(x, X) = \#C_{18}[N_{18}^2(x, X)]$,
where $C_n(X)$ stands for the set of all n-connected components of X, $\#C_n(X)$ stands for the cardinal of $C_n(X)$, N_n^k is the geodesic neighborhood of x inside X of order k and it is defined recursively by:
$N_n^1(x, X) = N_n^*(x) \cap X$ *and*
$N_n^k(x, X) = \cup N_n(y) \cap N_{26}^*(x) \cap X, y \in N_n^{k-1}(x, X)$,
where $N_n^*(x)$ is n-neighborhood of x.

3.3.2.2 Region Growing based Algorithm

In order to guarantee that each handle is minimally cut, we use the distance from the boundary as a weight to control the region growing algorithm. The voxel with a larger distance from the colon wall has a higher priority in our region growing algorithm. This can be implemented efficiently using a priority queue. In our region growing algorithm, only the simple point is removed from the priority queue and marked as colon. Because a non-simple point may become simple when some points are added to the object, we decrease its priority by a small value δ ($\delta = 0.1$ in our current implementation) and insert it into the priority queue again if its priority is larger than a predefined threshold T_{th} ($T_{th} = 0.8$ in our current implementation). Thus, we first compute an unsigned exact distance field using the segmented colon data obtained from the previous step. After that, we compute the skeleton (that is, centerline) of the colon using the unsigned distance field, which is then used as the initial seeds set for region growing. Our topology preserving region growing algorithm is

described as follows:

1. Mark the voxels of the input skeleton as colon.
2. For each voxel of the skeleton, put its six neighboring voxels into a priority queue Q.
3. While Q is not empty do
   (a) Let v be the top voxel in Q.
   (b) If v is a simple point, mark v as colon and put its six neighboring voxels into Q.
   (c) The priority of v is decreased by δ.
   (d) If the priority of v is greater than T_{th}, it is inserted into Q again.



(a)

(b)

Figure 3.7: A close up view of the colon surface (a) extracted without topological denoising, and (b) extracted with topological denoising.

After applying this algorithm, non-simple points are removed from the segmented colon. Thus, all handles are removed. Then, we use our enhanced dual contour method [139, 140] to extract a simplified smooth colon surface while preserving the topology of

the finest resolution colon surface. A close up view of the colon surface extracted without our topological denoising algorithm is shown in Figure 3.7(a). Figure 3.7(b) shows the surface with topological denoising, and we can see that the two handles on the right hand side of Figure 3.7(a) are removed by our topological denoising algorithm.

3.4 Conclusions

In this chapter, we have presented a segmentation and digital cleansing framework to handle partial volume effect and topological noise. Our algorithm automatically located and segmented all tagged material within the colon lumen. The performance of our method was demonstrated by visual judgment of the 2D slice show and 3D views. From the segmented colon data, the colon surface mesh with genus zero is extracted using a dual-contouring method. The colon surface mesh is used as the input of our flattening algorithm, which will be described in the next chapter.

Chapter 4

Conformal Virtual Colon Flattening

4.1 Introduction

In Chapter 2, we have shown that many regions behind haustral folds and around sharp bends are still missed even after fly-through navigation has been performed in both antegrade and retrograde directions. Polyps behind haustral folds or around sharp bends may be missed in the standard VC examinations. Colon flattening is an efficient visualization technique for polyp detection, in which the entire inner surface of the colon is displayed as a single 2D image. In this chapter, a conformal colon flattening method will be presented, by which the shape of polyps is preserved in the flattened colon image.

If two surfaces do not have the same Gaussian curvature, there does not exist a mapping which is length, area, and angle preserving at the same time. Therefore, all of the colon flattening methods introduce some kind of distortion. Several methods have been developed that are either area preserving [7] or angle preserving [43, 50]. We are specifically interested in an angle preserving method, because physicians identify polyps mainly based on the shape information.

Haker et al. [43] have proposed a method based on the discretization of the Laplace-Beltrami operator to flatten the colon surface onto the plane in a manner which preserves angles. The flattened colon surface is then colored according to its mean curvature. A morphological method is used to remove minute handles resulting from the segmentation algorithm, because their algorithm requires the input surface to be a topologically open-ended cylinder. However, the color-coded mean curvature of the extracted surface is not efficient for polyp identification, and it requires a highly accurate and smooth surface mesh to achieve a good mean-curvature calculation.

We propose a novel method for colon flattening by computing the conformal structure of the colon surface, represented as a set of holomorphic 1-form basis. The conformal mapping can be obtained by integration. Our method has the following advantages:

1. The algorithm is rigorous and theoretically solid, which is based on the Riemann surface theory and differential geometry;

2. It is general, so it can handle high genus surfaces;

3. The global distortion from the colon surface to the parametric rectangle is minimized, which is measured by harmonic energy;

4. It is angle preserving, so the shape of colonic polyps is preserved;

5. The topology noise is automatically removed by our shortest loop algorithm.

Combined with the direct volume rendering method, the flattened 2D colon image provides an efficient way to enhance VC systems.

4.2 Conformal Flattening

In our method, the colon surface is conformally mapped to a planar rectangle. Conformal maps are extremely valuable for medical applications because of their special properties as follows:

- Conformal maps are *angle preserving* (*local shape preserving*). Because analytic functions are angle preserving, therefore by definition, conformal maps preserve angles. The shape of polyps is preserved in the flattened colon image. Physicians can still identify polyps from the flattened colon image based on the shape information.

- Conformal maps minimize elastic energy (*harmonic energy*). One can treat colon surface as a rubber surface, and the mapping to another surface will introduce stretching distortion and generate the elastic energy. It has been proven [62] that conformal maps minimize the harmonic energy. It is highly desirable in practice to find the best match between the colon surfaces and the 2D rectangle which minimizes the distortion.

- Conformal maps are *intrinsic*. Conformal maps are determined by the metric, not the embedding. For example, one can change a surface by rotation, translation, folding, bending without stretching, the conformal parameterization is invariant. This is valuable for surface registration purpose.

- Conformal maps are *stable* and easy to compute. Computing conformal maps is equivalent to solve an elliptic geometric PDE [115], which is stable and insensitive to the noise and the resolution of the data. If two surfaces are similar to each other, then the corresponding conformal maps are similar too.

- Conformal parameterization simplifies geometric processing from 3D to 2D. By parameterizing a surface, we map it to the planar domain with local shape preservation, and it is easier to process in the planar domain than in the 3D domain. Some of the 3D geometric features are carried over by the mapping with high fidelity.

In the following sections, we first briefly introduce the major concepts and theorems used in our colon flattening algorithms. Thorough discussion can be found in the Riemann surface theory [62]. Then, the detail of the flattening algorithm will be presented.

4.2.1 Riemann Surface Theory

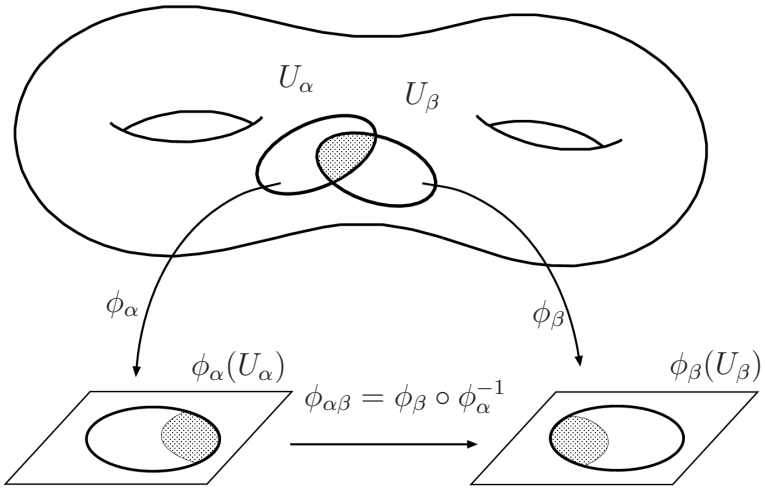


Figure 4.1: Riemann Surface: The manifold is covered by a set of charts (U_α, ϕ_α), where $\phi_\alpha : U_\alpha \to \mathbf{R}^2$. If two charts (U_α, ϕ_α) and (U_β, ϕ_β) overlap, the transition function $\phi_{\alpha\beta} : \mathbf{R}^2 \to \mathbf{R}^2$ is defined as $\phi_{\alpha\beta} = \phi_\beta \circ \phi_\alpha^{-1}$. If all transition functions are analytic, then the manifold is a Riemann surface. The atlas $\{(U_\alpha, \phi_\alpha)\}$ is a conformal structure.

A manifold can be treated as a set of open sets in $\mathbf{R}^2$ glued coherently.

Definition 4.1. *A 2-dimensional manifold is a connected Hausdorff space M for which every point has a neighborhood U that is homeomorphic to an open set V of $\mathbf{R}^2$. Such a homeomorphism $\phi : U \to V$ is called a coordinate chart. An atlas is a family of charts $\{(U_\alpha, \phi_\alpha)\}$, where U_α constitutes an open covering of M.*

Definition 4.2 (Analytic Function)**.** *A complex function $f : \mathbf{C} \to \mathbf{C}, (x, y) \to (u, v)$ is analytic (holomorphic), if it satisfies the following Riemann-Cauchy equation*

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}.$$

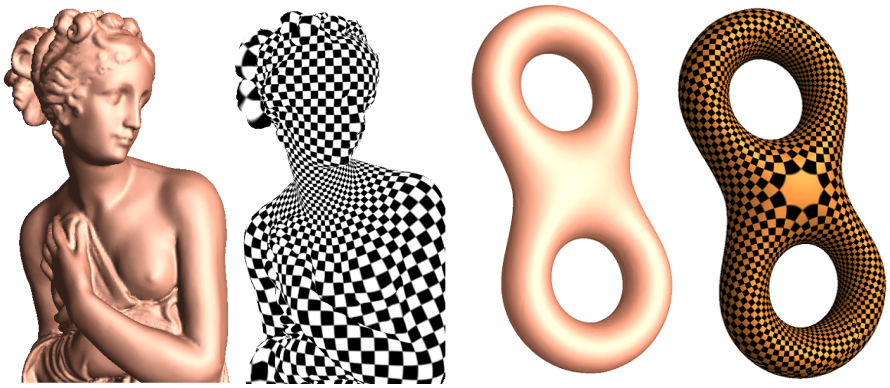
A conformal atlas is an atlas with special transition functions.

Definition 4.3 (Riemann Surface)**.** *Suppose M is a 2-dimensional manifold with an atlas $\{(U_\alpha, \phi_\alpha)\}$. If all chart transition functions*

$$\phi_{\alpha\beta} := \phi_\beta \circ \phi_\alpha^{-1} : \phi_\alpha(U_\alpha \bigcap U_\beta) \to \phi_\beta(U_\alpha \bigcap U_\beta)$$

are analytic, then the atlas is called a conformal atlas, and M is called a Riemann surface.

Two conformal atlases are *compatible* if their union is still a conformal atlas. All the compatible conformal atlases form an *conformal structure* of the manifold as shown in Figure 4.1. All oriented 2-dimensional manifolds with Riemannian metrics are Riemann surfaces and have conformal structures [62], such that on each chart (U_α, ϕ_α) with local parameter (u, v), the metric can be represented as $ds^2 = \lambda(u, v)(du^2 + dv^2)$.



(a) (b)

Figure 4.2: Holomorphic 1-form examples for (a) genus zero surface and (b) genus two surface.

4.2.1.1 Holomorphic 1-form

In order to flatten the surface, we need special differential forms defined on the conformal structure.

Definition 4.4 (Holomorphic 1-form)**.** *Given a Riemann surface M with a conformal structure $\mathcal{A}$, a holomorphic 1-form ω is a complex differential form, such that on each local chart $(U, \phi) \in \mathcal{A}$,*

$$\omega = f(z)dz,$$

where $f(z)$ is an analytic function, $z = u + iv$ is the local parameter in the complex form.

The holomorphic 1-forms of closed genus g surface form a g complex dimensional linear space, denoted as $\Omega(M)$. It is noted that a genus zero surface has no holomorphic 1-forms. A conformal atlas can be constructed by using a basis of $\Omega(M)$. Considering its geometric intuition, a holomorphic 1-form can be visualized as two vector fields $\omega = (\omega_x, \omega_y)$, such that the curls of ω_x and ω_y equal zero. Furthermore, one can rotate ω_x about the normal a right angle to arrive at ω_y,

$$\nabla \times \omega_x = 0, \nabla \times \omega_y = 0, \omega_y = n \times \omega_x.$$

4.2.1.2 Conformal Parameterization

Suppose $\{\omega_1, \omega_2, \cdots, \omega_g\}$ is a basis for $\Omega(M)$, where g is genus of M. We can find a collection of open disks $U_\alpha \subset M$, such that U_α form an open covering of M, $M \subset \cup U_\alpha$. We define $\phi_\alpha^k : U_\alpha \to \mathbf{C}$ using the following formula, first we fix a base point $p \in U_\alpha$, for any point $q \in U_\alpha$,

$$\phi_\alpha^k(q) = \int_\gamma \omega_k,$$

where the path $\gamma : [0,1] \to U_\alpha$ is arbitrary curve connecting p and q and inside U_α, $\gamma \subset U_\alpha, \gamma(0) = p, \gamma(1) = q$. It can be verified that, we can select a $\phi_\alpha^k, k = 1, 2, \cdots, g$, such that ϕ_α^k is a bijection, we simply denote it as ϕ_α. Then, the atlas $\{(U_\alpha, \phi_\alpha^k)\}$ is a conformal atlas.

For a genus one closed surface M, given a holomorphic 1-form $\omega \in \Omega(M)$, we can find two special curves $\Gamma = \gamma_1 \cup \gamma_2$, such that $\tilde{M} = M/\Gamma$ is a topological disk. Furthermore, on each open set U_α, if the curve $\int_{\gamma_1} \omega$ is a horizontal line in the parameter plane, then γ_1 is a *horizontal trajectory*. In the current work, we choose γ_2 such that $\int_{\gamma_2} \omega$ is a vertical line in the parameter plane, namely, γ_2 is a vertical trajectory. Γ is called a *cut graph*.

Then, by integrating ω on $\tilde{M}$, $\tilde{M}$ is conformally mapped to a parallelogram, as shown in figure 3.5. Figure 4.2 illustrates holomorphic 1-forms on surfaces. The texture coordinates are obtained by integrating the 1-form on the surface.

4.2.1.3 Conformal Maps

Suppose M_1 is a Riemann surface with a conformal atlas $\{(U_\alpha, \phi_\alpha)\}$, and M_2 is another Riemann surface with conformal atlas $\{(V_\beta, \tau_\beta)\}$.

Definition 4.5 (Conformal Map)**.** *A map $f : M_1 \to M_2$ is a* conformal map*, if its restriction on any local charts (U_α, ϕ_α) and (V_β, τ_β),*

$$f_\alpha^\beta := \tau_\beta \circ f \circ \phi_\alpha^{-1} : \phi_\alpha(U_\alpha) \to \tau_\beta(V_\beta)$$

is analytic.

4.2.2 Flattening Algorithm

The concepts of Riemann surface and conformal map are defined using continuous mathematics. Computing conformal parameterization is equivalent to solving an elliptic partial differential equation on surfaces.

Unfortunately, in reality, all surfaces are represented by discrete piecewise linear meshes, which are not differentiable in general. Fortunately, the solution to the elliptic PDE can be approximated accurately by piecewise linear functions using finite element method [110]. The convergence and accuracy have been thoroughly analyzed in finite element field.

Therefore, our algorithm is mainly based on the finite element method. The key step is to use piecewise linear functions defined on edges to approximate differential forms. Furthermore, the forms minimize the harmonic energy, the existence and the uniqueness are guaranteed by Hodge theory [115].

4.2.2.1 Double Covering

In our case, after the topological noise removal, the surface is a closed genus zero surface. Because the genus zero surface has no holomorphic 1-form, a *double covering* method is used to construct a genus one surface. Two holes are first punched on the input surface. Then, a mesh M with two boundaries is obtained. The algorithm to construct a closed genus one mesh is described as follows:

1. Make a copy of mesh M, denoted as M', such that M' has all vertices in M, if $[v_0, v_1, v_2]$ is a face in M, then $[v_1, v_0, v_2]$ is a face of M'.

2. Glue M and M' along their boundaries, if an halfedge $[v_0, v_1]$ is on the boundary of M $[v_0, v_1] \in \partial M$, then $[v_1, v_0]$ is on the boundary of M'. Glue $[v_0, v_1]$ with $[v_1, v_0]$.

The resulting mesh is a closed and symmetric, with two layers coincided. It is noted that general genus one surface can be conformally mapped to a planar parallelogram, but not a rectangle. In our case, the genus one surface is obtained by double covering method. The Riemann metric defined on the double covered surface is symmetric. Each boundary where we glue two surfaces is mapped to a straight line. Thus, the denoised genus zero colon surface can be conformally mapped to a rectangle.

4.2.2.2 Computing Harmonic and Holomorphic 1-form

After getting the homology basis $\{\gamma_1, \gamma_2, \cdots, \gamma_{2g}\}$, it is easy to compute the holomorphic 1-form basis.

1. Select γ_k, compute $\omega_k : K_1 \to \mathbf{R}$, form the boundary condition:

$$\sum_{e \in \gamma_i} \omega_k(e) = \delta_i^k, \omega_k(\partial f) = 0, \forall f \in K_2, \tag{4.1}$$

   where

$$\delta_i^k = \begin{cases} 1 & : \quad i = k \\ 0 & : \quad i \neq k \end{cases}$$

   K_1 is the edge set of M and K_2 is the face set of M.

2. Under above linear constraints, compute ω_k minimizing the quadratic energy,

$$E(\omega_k) = \sum_{e \in K_1} k_e \omega_k^2(e), \tag{4.2}$$

using linear constrained least square method, where k_e is the weight associated with each edge. Suppose the angles in the adjacent faces against edge e are α, β, then $k_e = \frac{1}{2}(\cot\alpha + \cot\beta)$ [109]. Solving this equation is equivalent to solve Riemann-Cauchy equation using finite element method.

3. On face $[v_0, v_1, v_2]$, its normal n is computed first, and a unique vector v in the same plane of v_0, v_1, v_2 is obtained by solving following equations:

$$\begin{cases} < v_1 - v_0, v > &= \omega_k([v_1, v_0]) \\ < v_2 - v_1, v > &= \omega_k([v_2, v_1]) \\ < n, v > &= 0 \end{cases} \tag{4.3}$$

Rotate v about n a right angle, $v^* = n \times v$, then define

$$\omega_k^*([v_i, v_j]) :=< v_j - v_i, v^* > .$$

The harmonic 1-form basis is represented by $\{\omega_1, \omega_2, \cdots, \omega_{2g}\}$, and the holomorphic 1-form basis is given by $\{\omega_1 + i\omega_1^*, \omega_2 + i\omega_2^*, \cdots, \omega_{2g} + i\omega_{2g}^*\}$. Figure 4.3 illustrates holomorphic 1-forms on a colon surface. The checkboard texture coordinates are obtained by integrating the 1-form on the colon surface. This figure demonstrates that our method is angle preserving.

4.2.2.3 Conformal Parameterization

Suppose we have selected a holomorphic 1-form $\omega : K_1 \to \mathbf{C}$, then we define a map $\phi : \tilde{M} \to \mathbf{C}$ by integration. The algorithm to trace the horizontal trajectory and the vertical trajectory on $\phi(\tilde{M})$ is as follows:

1. Pick one vertex $p \in \tilde{M}$ as the base vertex.
2. For any vertex $q \in \tilde{M}$, find the shortest path $\gamma \in D$ connecting p to q.
3. Map q to the complex plane by

$$\phi(q) = \sum_{e \in \gamma} \omega(e).$$

4. Pick a vertex $p \in M$, trace the horizontal line γ on the plane region $\phi(\tilde{M})$ through $\phi(p)$. If γ hits the boundary of $\phi(\tilde{M})$ at the point $\phi(q)$, q must be in the cut graph Γ, then there are two points q^+, q^- on the boundary of $\tilde{M}$, $\partial\tilde{M}$. Assume γ hits $\phi(q^+)$, then we continue to trace the horizontal line started from $\phi(q^-)$, until we return to the starting point $\phi(p)$. The horizontal trajectory is $\phi^{-1}(\gamma)$.
5. Trace vertical trajectory similar to step 4.

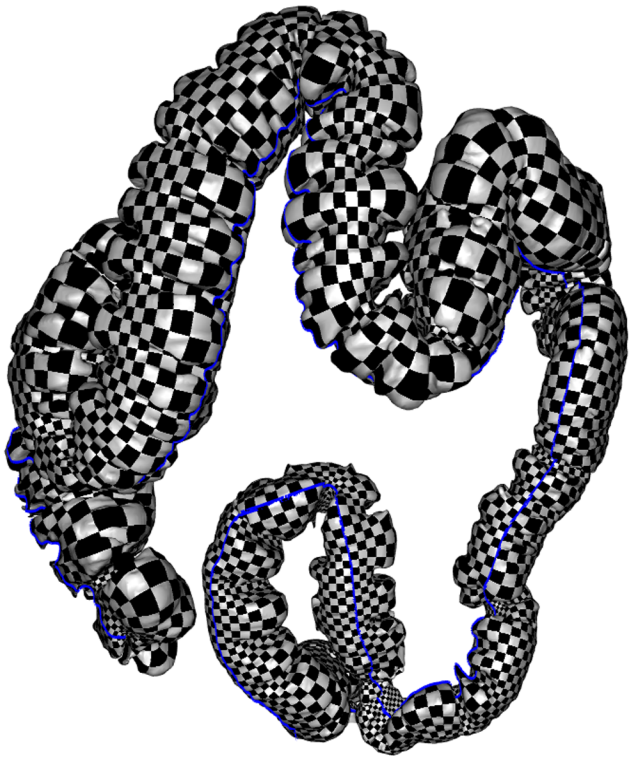


Figure 4.3: Holomorphic 1-form example for a colon surface.

6. The new cut graph $\tilde{\Gamma}$ is the union of the horizontal and vertical trajectories. Cut the surface along $\tilde{\Gamma}$ to get $\tilde{M}'$, and compute $\tilde{\phi}$. Then, $\tilde{\phi}(\tilde{M}')$ is a rectangle, $\tilde{\phi}$ is a conformal map.

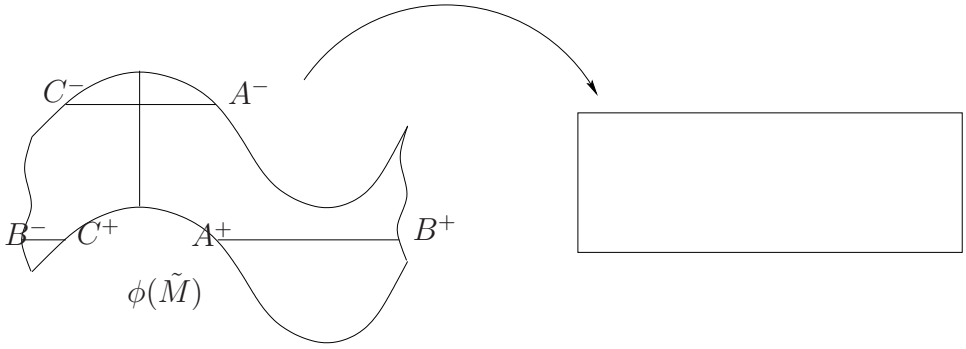


Figure 4.4: Trace horizontal trajectory.

4.3 Visualization of the Flattened Colon

The result of the flattening algorithm is a triangulated rectangle where the polyps are also mapped. The rendering of the flattened colon image is crucial for the detection of polyps. Haker et al. [43] have utilized color-coded mean curvature to visualize the flattened colon surface. Although it can show the geometry information of the 3D colon surface, it is still unnatural for the physicians to detect colonic polyps. The shape and texture of the polyps are good clues for polyp detection. In this section, we describe a direct volume rendering method to render the flattened colon image. Each pixel of the flattened image is shaded using a fragment program executed on the GPU, which allows the physician to move and zoom a viewing window in real-time and to inspect the entire flattened inner colon surface. The idea of our rendering algorithm is to map each pixel of the flattened image back to the 3D colon surface, that is, the volume space. The pixel is shaded using volumetric ray-casting algorithm in the volume space.

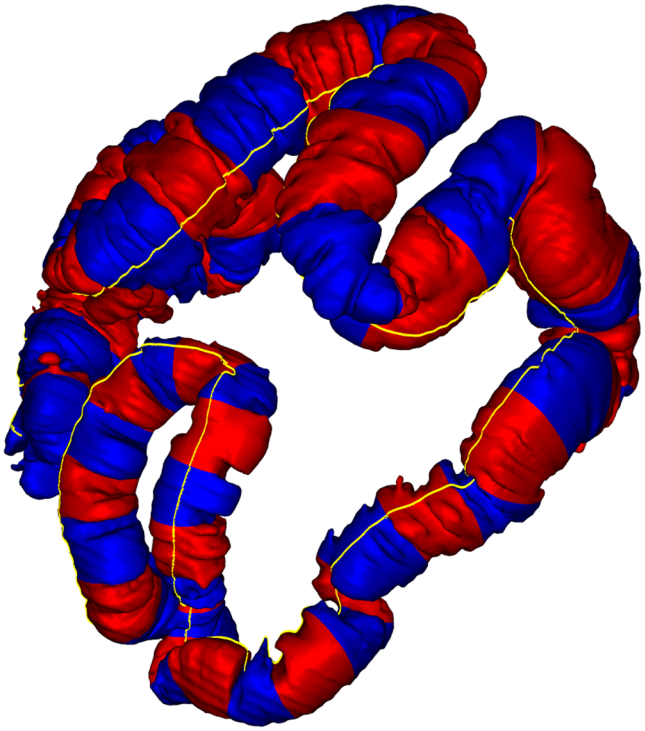


Figure 4.5: The colon is divided into segments colored in red and blue.

4.3.1 Camera Registration

In order to perform the ray-casting algorithm, the ray direction needs to be determined for each vertex of the 3D colon surface first. A number of cameras are uniformly placed on

the central path of the colon. The ray direction of a vertex is then determined by the nearest camera to that vertex.

Our camera registration algorithm starts with approximating the central path with a B-spline and re-sampling it into uniform intervals. Each sampling point represents a camera. Each vertex is then registered with a sampling point on the central path. The registration procedure is implemented efficiently by first dividing the 3D colon surface and the central path into N segments. The registration is then performed between the corresponding segments of colon and the central path. The division of the 3D colon is done by classifying the vertices of the flattened 2D mesh into uniform N segments based on their height. As a consequence, the vertices of the 3D colon mesh are also divided into N segments, as shown in Figure 4.5. We then trace $N-1$ horizontal lines on the flattened 2D mesh, which uniformly divide the 2D mesh into N segments. Each traced horizontal line corresponds to a cross contour on the 3D mesh. In fact, we do not need to really trace the horizontal lines. For each horizontal line, we only need to compute the intersection points of the horizontal line and the edges intersecting with it. For each intersection point, the corresponding 3D vertex of the 3D colon mesh is then interpolated. The centroid of these interpolated 3D vertices is computed and registered with a sampling point of the central path. Therefore, the central path is also divided into N segments, and each segment of the 3D colon mesh corresponds to a segment of the central path. Although the division of the 3D colon surface and the central path is not uniform as that of the 2D mesh, it does not affect the accuracy of the camera registration.

For each vertex of a colon surface segment, we find its nearest sampling point in its corresponding central path segment and the neighboring two segments. This algorithm is efficient because for each vertex the comparison is performed only with a small number of sampling points on the central path. For each vertex, we only record the B-spline index of the sampling points, instead of its 3D coordinates.

4.3.2 Volumetric Ray-Casting

To generate a high-quality image of the flattened colon, only coloring the vertices of the polygonal mesh and applying linear interpolation is not sufficient. We need to determine the color for each pixel of the 2D image. This can be performed efficiently using a fragment program on the GPU. For each vertex of the flattened polygonal mesh, we pass its corresponding 3D coordinates and camera index through texture coordinates to the fragment program. When the flattened polygonal mesh is rendered, each pixel of the flattened image will obtain its barycentric interpolated 3D coordinates and camera index. Its 3D position may not be exactly on the colon surface, but very close to the colon surface. Because we use a direct volume rendering method to determine the color for the pixel, it does not really affect the image quality. We use the interpolated camera index to look up its corespondent sampling point on the central path. Then, the ray direction is determined and volumetric ray casting algorithm is performed using an opaque transfer function. By this method, we can determine the color for each pixel on the flattened image to generate a high-quality

image.

Since our flattened image is colored per-pixel, we can in real-time provide the physician with a high-quality zoom-in views of a suspicious area on the flattened image. Because each vertex is registered with a sampling point on the central path, the flattened colon image can be easily correlated with the VC fly-through navigation. The correlated 3D view of the suspicious area can be also shown simultaneously.

4.4 Implementation and Results

We have implemented our conformal flattening and rendering algorithms in C/C++. All of the experiments have been performed on a uni-processor 3.6 GHz Pentium IV PC running Windows XP, with 2G RAM and NVIDIA Geforce 8800GTX graphics card. 98 colon CT data sets from the National Institute of Health and Stony Brook University have been used to test our algorithms. All the data sets have a large number of slices, and the resolution of each slice is 512×512. They all exhibit similar results.

4.4.1 Preprocessing

Before our colon flattening algorithm can be applied, we need to perform the following tasks to extract the colon surface from the CT data set. First, a partial volume segmentation algorithm [131] is applied, and a binary mask is generated, which labels the voxels belonging to the colon interior and the colon wall. This algorithm ensures a fast and accurate segmentation and electronic cleansing with the ability to consider the partial volume effect. Second, the rendering algorithm involves the central path of the colon. The central path is automatically extracted from the CT data set based on an accurate DFB-distance field with the exact Euclidian values [126]. The path is then approximated by a B-spline curve. Finally, given the binary mask and the CT data set, an enhanced dual contouring method [139] is used to extract the simplified colon surface while preserving the finest resolution isosurface topology. Since our algorithm can deal with small handles, we do not need to remove these handles in the preprocessing step. All these algorithms used in the preprocessing step are robust and efficient, and can be done in seconds on the PC platform. We list the timings of every stage of our flattening algorithm in Table 4.1.

The whole process of our algorithm can be completed in about 13 minutes. Most steps of our algorithm are done within seconds or minutes. The most time consuming part of our algorithm is computing the harmonic holomorphic 1-form using the conjugate gradient method for conformal mapping, which takes about seven minutes. The good news is that the conjugate gradient method can be accelerated with the GPU [13].

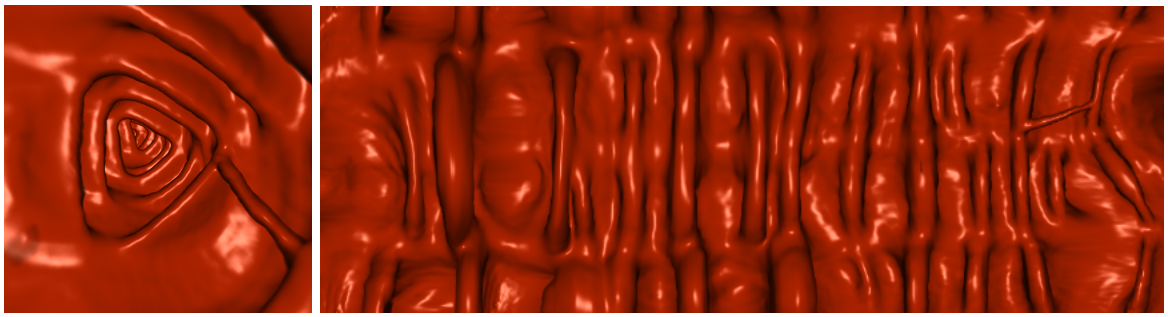
4.4.2 Discussion

During an VC fly-through navigation, the virtual endoscopic view of the physician may be blocked by haustral folds as shown in Fig. 4.6(a). The regions behind the haustral folds

Table 4.1: Average timings of every stage of our flattening algorithm.

No.	Stage	Timing
1.	Segmentation and Digital Cleansing	3 $mins$
2.	Centerline Extraction	1 min
3.	Topological Denoising	< 1 min
4.	Colon Surface Extraction	< 1 min
5.	Conformal Mapping	7 $mins$
6.	Flattened Colon Rendering	500 ms

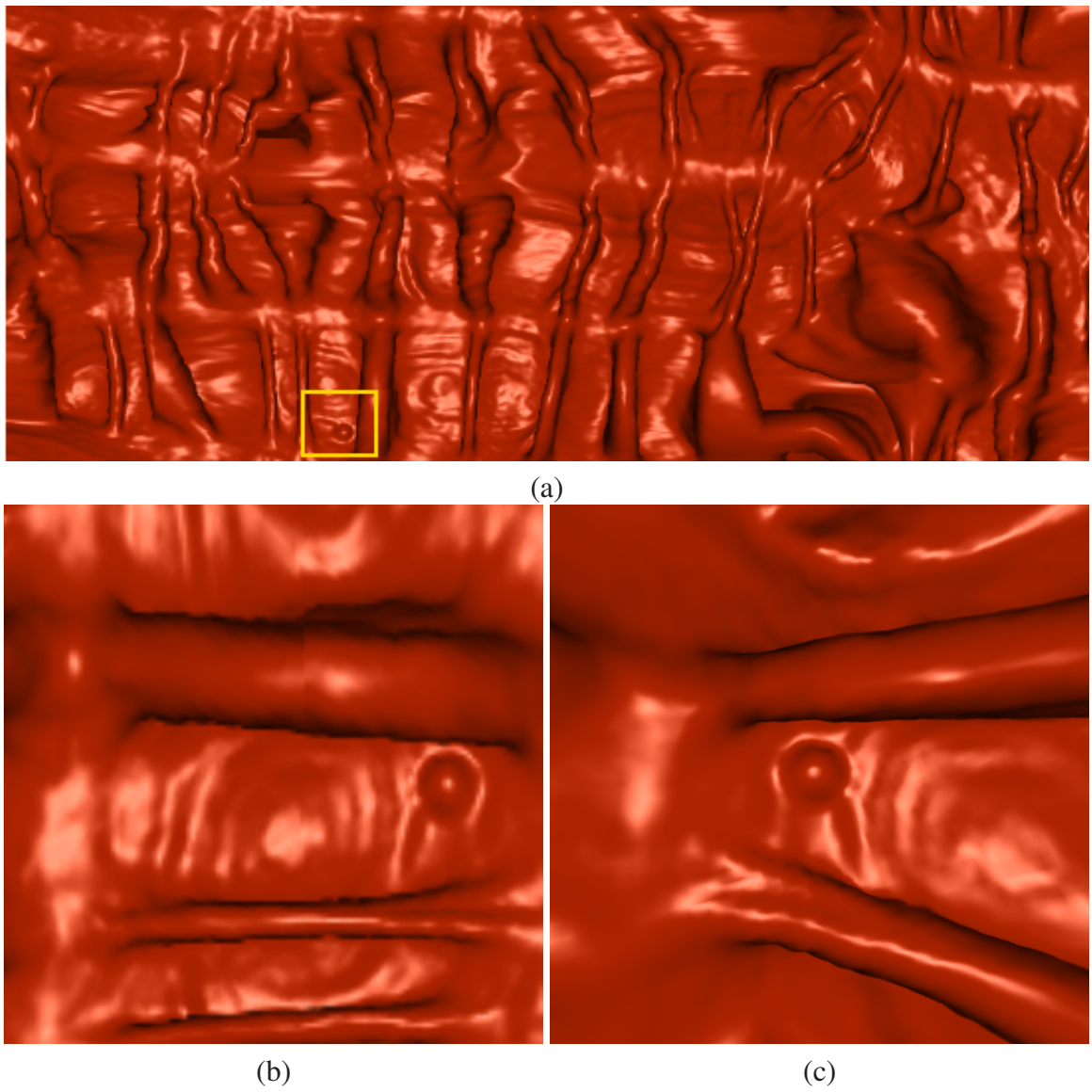
may be missed even when both antegrade and retrograde navigations are performed. The corresponding flattened colon of Fig. 4.6(a) is shown in Fig. 4.6(b), which demonstrates that our conformal colon flattening method is an efficient way to inspect the regions behind the haustral folds. It is clear that our virtual colon flattening method can provide 100% colon inner surface coverage for polyp screening.



(a) (b)

Figure 4.6: Colon haustral folds in (a) 3D endoscopic view and (b) corresponding flattened image.

For a typical 512^3 colon data, the resolution of the flattened image generated by our method is about 400×8000. The total rendering time for the whole flattened colon image is about $500ms$. A small part of a flattened colon image is shown in Fig. 4.7(a), which has a small polyp enclosed by a yellow rectangle. The zoom-in view of this small polyp is shown in Fig. 4.7(b). Since each pixel is registered with a virtual camera on the centerline, the corresponding 3D endoscopic view can be generated easily using our method. The 3D endoscopic view of this small polyp is shown in Fig. 4.7(c). Although the diameter of this small polyp is only 4mm, it can be directly identified from the flattened colon image without difficulty. In clinical applications, we recommend that the resolution of the flattened image should be at least four times higher than the one that we used in this paper for physicians to easily identify small polyps. In fact, it is unnecessary to pre-compute the entire high resolution flattened image. Our rendering algorithm accelerated by the commodity

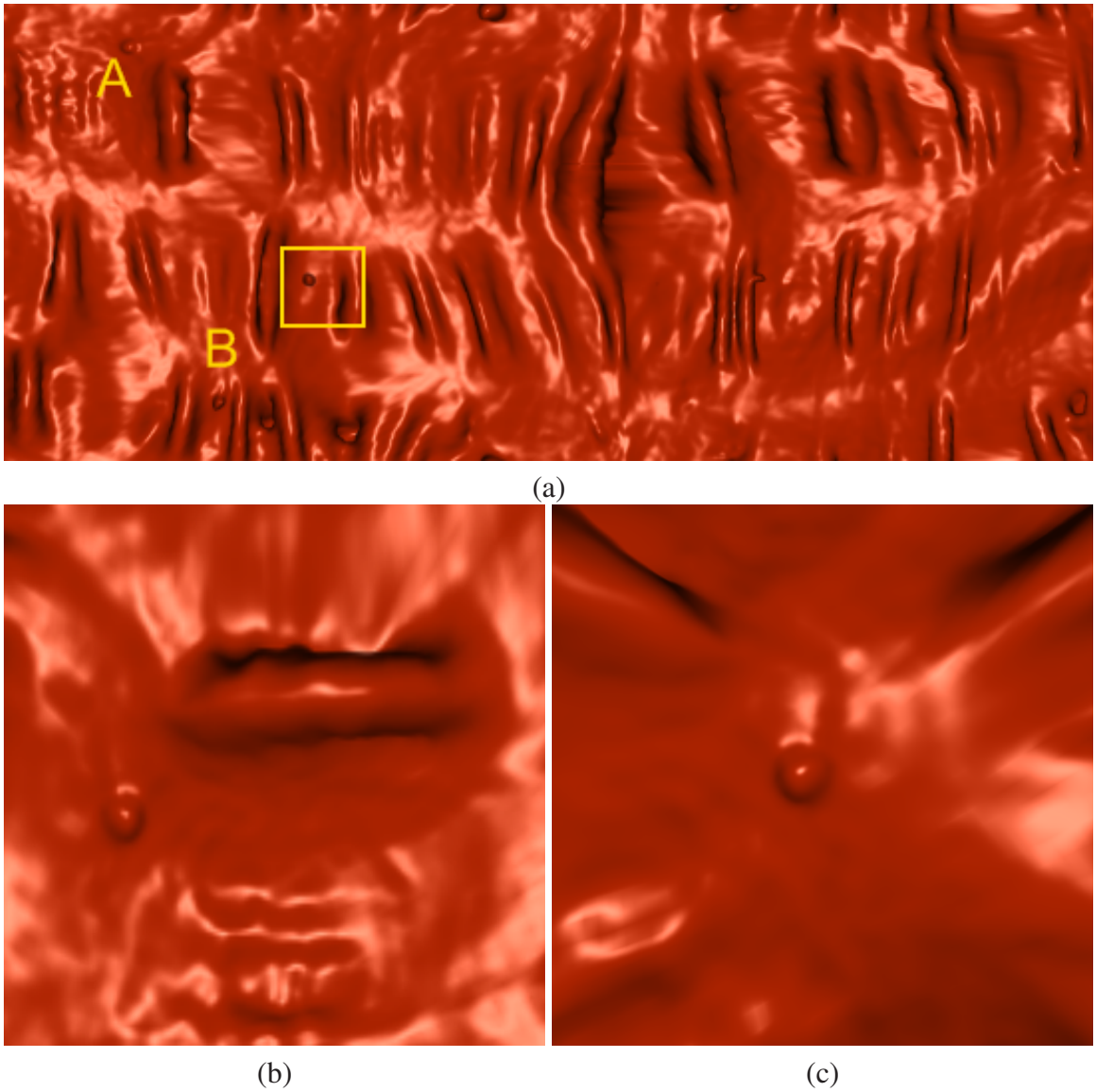


(a) (b) (c)

Figure 4.7: (a) A part of flattened colon image, (b) zoom-in view of the polyp enclosed by a yellow rectangle in (a), and (c) the corresponding 3D endoscopic view of the polyp enclosed by the yellow rectangle in (a).

graphics hardware provides real-time zoom-in function with high-quality, which allows the physician to interactively inspect the entire flattened colon at various resolutions.

Another example with three polyps is shown in Fig. 4.8(a). The zoom-in view of a small polyp enclosed by yellow rectangle is shown in Fig. 4.8(b), and its corresponding 3D endoscopic view is shown in Fig. 4.8(c). The other two polyps are also shown in 4.8(a) at location A and B, which demonstrates that virtual colon flattening technique is an efficient way for polyp screening.



(a)

(b) (c)

Figure 4.8: (a) A part of flattened colon image, (b) zoom-in view of the polyp enclosed by a yellow rectangle in (a), and (c) the corresponding 3D endoscopic view of the polyp enclosed by the yellow rectangle in (a).

Fig. 4.9 shows a comparison of polyps in 3D endoscopic views and zoom-in views of flattened colon image to demonstrate that our virtual colon flattening method is angle preserving. The 3D endoscopic views of four different polyps are shown in Fig. 4.9(a)-(c), and its corresponding zoom-in views on the flattened images are shown in shown in Fig. 4.9(d)-(f). We can see that polyps can be clearly identified by the shapes from the zoom-in views of flattened colon images as from the conventional endoscopic views. It is worth to note that the shape of the polyp on the haustral fold is also preserved in our method as

shown in Fig. 4.9(f).

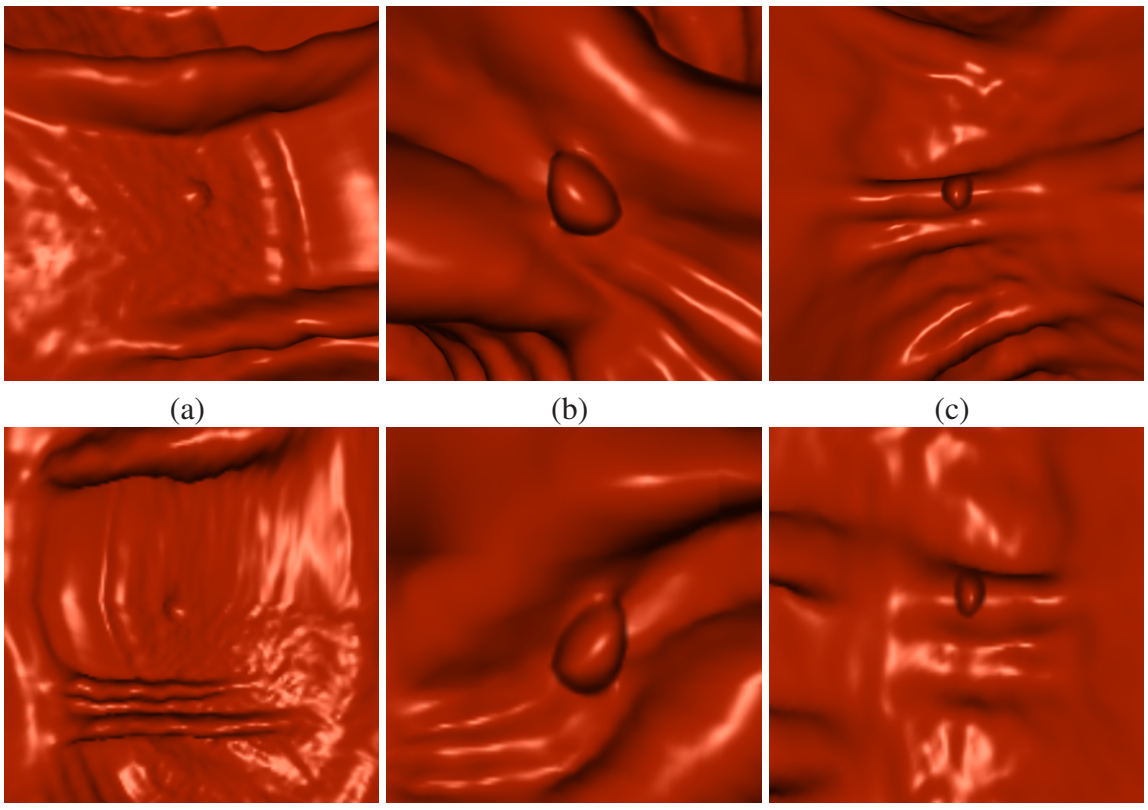


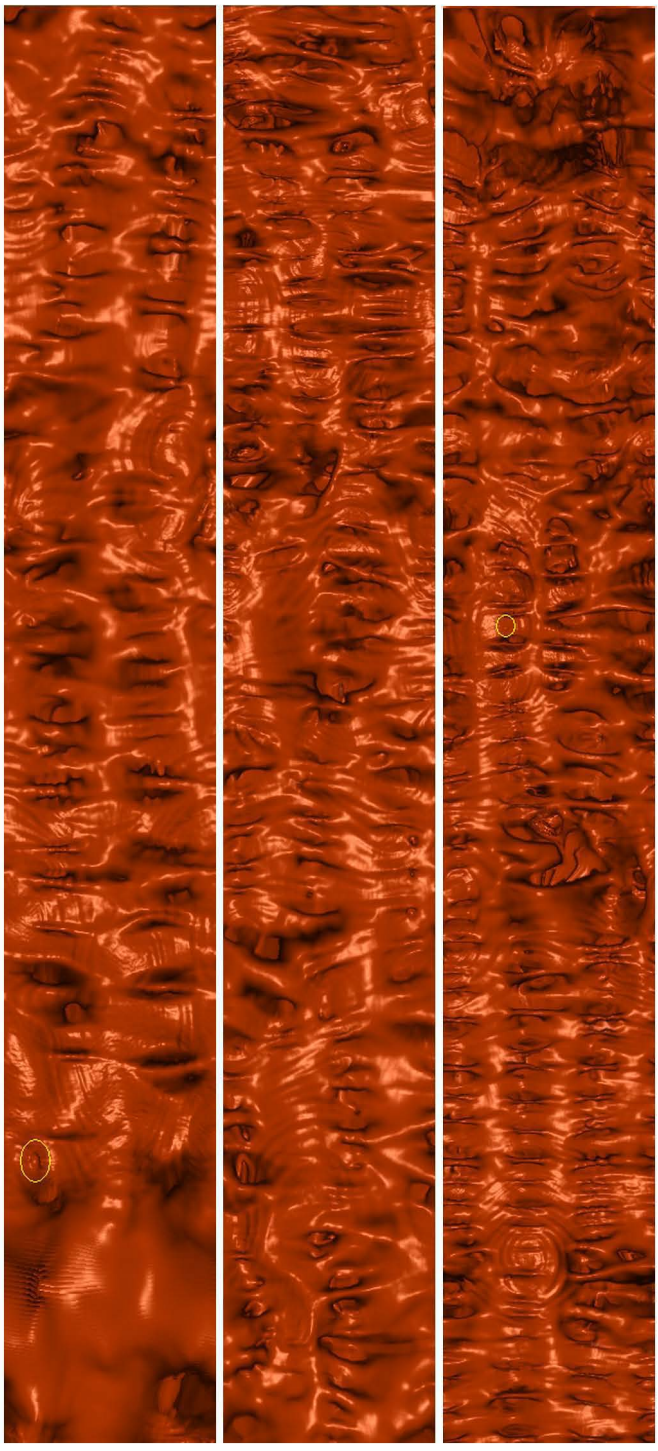
Figure 4.9: (a)-(c) are the 3D endoscopic views of four different polyps, and (d)-(f) are the zoom-in views of corresponding polyps on the flattened colon images.

Fig. 4.10 shows that our method can map the entire colon inner surface to a rectangle, which means that our method guarantees 100% surface visibility coverage for polyp screening. There is no blind spot on the flattened colon image. This is important for clinical applications. This data set contains two polyps enclosed by yellow ellipse. The diameter of the larger polyp is $9mm$, and the diameter of the smaller polyp is $4mm$.

4.5 Conclusions

In this chapter, we have presented an efficient colon flattening algorithm using a conformal structure. Our algorithm is general for arbitrarily high genus surfaces, and does not require the input surface to be a topological cylinder. We have proven that our algorithm is angle preserving and the global distortion is minimal. The shape of colonic polyps on the flattened colon image is well preserved, and can be easily identified by physicians. The flattened colon image is rendered with a direct volume rendering method accelerated with

commodity graphics hardware. We demonstrate that the conformal colon flattening image cooperates well with the fly-through VC system, which displays 100% of the endoluminal surface to the physician. This technique can reduce evaluation time with a more rapid 3D image assessment than with an antegrade and retrograde 3D endoluminal fly-through. It may also ultimately improve accuracy by reducing blind spots presented in 3D endoscopic views and by reducing reader fatigue.



(a) (b) (c)

Figure 4.10: A flattened image for a whole colon data set is shown in three images. The bottom of image (a) is the rectum of the colon, and the top of image (c) is the cecum of the colon. Two polyps are marked using yellow ellipse in (a) and (c).

Chapter 5

Volumetric Ray-Casting

5.1 Introduction

Virtual endoscopy applications require that the endoscopic view must be interactively rendered with high quality. In this chapter, we first introduce the idea of the basic ray-casting algorithm and its implementation on the GPU. Then, a GPU-based object-order ray-casting approach for rendering large volumetric data sets on the commodity computers, which is ideal for virtual endoscopy applications using large data sets. Furthermore, the cooperation and trade-off between the CPU and the GPU is exploited to obtain further acceleration.

Ray-casting [83] is a method for direct volume rendering, which can be seen as straightforward numerical evaluation of the volume rendering integral which sums up all optical effects such as color and opacity along viewing rays. For each pixel in the generated image, a single ray is cast into the volume. At equispaced intervals along the ray (the sampling distance), the discrete volume data is resampled, usually using trilinear interpolation as reconstruction filter. That is, for each resampling location, the scalar values of eight neighboring voxels are weighted according to their distance to the actual location for which a scalar data value is needed. After resampling, the scalar data values are mapped to optical properties by means of the transfer function, which yields an RGBA value for this resampling location. The volume rendering integral is approximated via alpha blending in back-to-front or front-to-back order. The following iterative formulation evaluates volume rendering integral in front-to-back order by stepping i from 1 to n:

$$\begin{aligned} C_i^{'} &= C_{i-1}^{'} + (1 - A_{i-1}^{'})C_i \\ A_i^{'} &= A_{i-1}^{'} + (1 - A_{i-1}^{'})A_i \end{aligned} \tag{5.1}$$

where the new color $C_i^{'}$ and opacity $A_i^{'}$ are calculated from the color C_i and opacity A_i at the current location i, and the composited color $C_{i-1}^{'}$ and opacity $A_{i-1}^{'}$ from the previous location $i-1$. The initial condition is $C_0^{'} = 0$ and $A_0^{'} = 0$.

The ray-casting algorithm can be described by the pseudocode in List 5.1. Accordingly,

Listing 5.1: Pseudocode for ray-casting.

```
Determine ray entry position
Compute ray direction
While ( ray position in volume )
    Do sampling at current position
    Compositing of color and opacity
    Advance position along the ray
End while
```

ray-casting can be split into the following major components.

- Ray Setup: First, a viewing ray needs to be set up according to given camera parameters and the respective pixel position. This component computes the volume entry position, which is the first intersection point between the bounding geometry of the volumetric data set. This component also determines the direction of the ray.
- Traversal Loop: This main component traverses along the ray, evaluating the volume rendering integral. The ray is sampled at discrete positions, and the traversal loop scans the rays along these positions. Each iteration of the loop consists of the following subcomponents.
  - Sampling: The data set is accessed at the current ray position, which might involve a reconstruction filter (that is, interpolation). The corresponding color and opacity are computed by applying the transfer function.
  - Compositing: The previously accumulated color and opacity are updated according to the front-to-back compositing equation (Equation 5.1).
  - Advance Ray Position: The current ray position is advanced to the next sampling location along the ray.
  - Ray Termination: The traversal loop ends when the ray leaves the data set volume. This subcomponent checks whether the current ray position is inside the volume and it only enters the next iteration of the loop when the ray is still inside.

Ray-casting exhibits an intrinsic parallelism in the form of completely independent light rays. This parallelism is compatible with hardware parallelism in GPUs. For example, by associating the operations for a single ray with a single pixel, the built-in parallelism for GPU fragment processing is used to achieve efficient ray-casting. In addition, volume data and other information can be stored in textures and thus accessed with the high internal bandwidth of a GPU.

Krüger and Westermann [74] have integrated the early ray termination and empty space skipping into texture based volume rendering on GPU to implement multi-pass volumetric

ray-casting. They exploit the early z-test provided by ATI graphics card to terminate fragment processing once sufficient opacity has been accumulated, and to skip empty space in the ray level. This method is only considering general case, that is the camera is put outside the data set. However, for some applications such as virtual endoscopy, the camera is located within a very large data sets (usually 512^3). In this case, using the longest ray to decide the rendering passes is not acceptable, because of the overhead of intermediate rendering pass to enable early-z test. With real dynamic branching and loop support, single-pass ray-casting algorithm can be efficiently implemented on the GPU using fragment program. Stegmaier et al. [117] have presented a framework for the hardware accelerated visualization of volumetric data based on a single-pass ray-casting approach. Their system exhibits very high flexibility and allows for an easy integration of non-trivial volume rendering techniques. However, if the whole volume data cannot be held in the GPU memory, the rendering performance of previous GPU ray-casting methods drop dramatically. In the next section, an object-order GPU ray-casting algorithm is described for the rendering of large data sets.

5.2 Object-Order GPU Ray-Casting

Volumetric data sets used in a variety of virtual endoscopy applications usually contain many regions that are classified as transparent or empty, for example, the colon lumen. The object-order approaches are well-suited for skipping empty regions, but usually the associated filters are too complex to be used for interactive rendering. And the hidden volume removal is also inefficient compared with the ray-casting method. Mora et al. [91] have proposed a CPU-based object-order ray-casting algorithm to take the advantages of both image-order and object-order approaches for interactive high-quality volume rendering. However, the cell projection implemented in this method can be efficiently performed only in orthogonal projection. Grimm et al. [37] have presented a CPU-based volume ray-casting approach based on image-ordered ray-casting with object-ordered processing. They introduced a memory efficient acceleration technique for on-the-fly gradient estimation and a memory efficient hybrid removal and skipping technique of transparent regions. Their method is also limited to orthogonal projection.

In our object-order ray-casting approach, we define a cell as a cubical region which corresponds to a sub-volume containing $N \times N \times N$ voxels. A cell is classified as empty, if all voxels of the cell are invisible based on the transfer function. Otherwise, it is classified as non-empty. The min-max ocree [134] is used to organize the cells for efficient classification. Each leaf node of the min-max octree contains a cell, as well as the minimum and maximum density values of the cell. Each interior node only contains the minimum and maximum density values found in that node's subtree.

In stead of projecting a reconstruction kernel for each voxel onto the image plane as in the splatting technique, we project the whole cell onto the image plane. Moreover, we use a fragment program to perform ray integration for the projected cell on-the-fly, in

which a volumetric ray-casting algorithm is performed. For each cell, we need to store its corresponding voxels in a 3D texture. Since the volumetric ray-casting algorithm requires a neighborhood of voxels for proper interpolations and gradient calculations, the neighboring voxels of the cell need to be stored in the 3D texture. Thus, for each cell the resolution of the corresponding 3D texture is $(N+2) \times (N+2) \times (N+2)$.

In order to obtain correct compositing result, we must first determine the visibility order of the cells so that the cells can be projected from front to back. Although the cells can be hierarchically sorted using the min-max octree structure, we devised a more efficient propagation algorithm to sort cells. As a result, the cells are front-to-back sorted and grouped into layers. The cells within the same layer can be projected simultaneously, which dramatically improves the performance of our cell projection algorithm on the GPU. Our cell sorting algorithm and cell projection algorithm can take the advantage of the parallelism between the CPU and the GPU. Thus, when a layer of cells are determined, they can be projected immediately to trigger fragment programs to be executed on the GPU. The CPU then can be used to generate the next layer of cells.

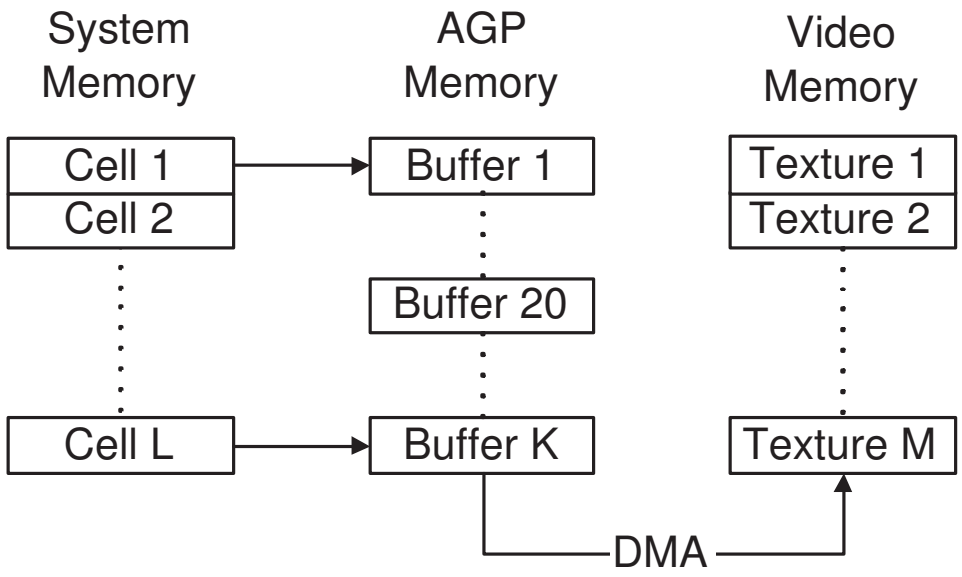


Figure 5.1: The three-layer structure used to store the cell data.

Although a large number of cells are classified as empty cells, which do not need to be uploaded to the GPU, the 3D textures corresponding to the non-empty cells are still too large to be fitted in the graphics card memory. We need to transfer some non-empty cells to the graphics card memory on-the-fly. The OpenGL extension pixel_buffer_object (PBO) defines an interface to using buffer objects for pixel data, which dramatically improves the texture uploading performance. By using this extension, the GPU can asynchronously pull the data from the AGP memory using DMA (Direct Memory Access). Thus, we use a three-level structure to store the cell data in the graphics card memory, the AGP memory, and the

system memory as shown in Figure 5.1. Suppose that we can allocate M 3D textures in the graphics card memory and N buffers of the same size in the AGP memory, and the first 20 buffers are used as a memory pool for transferring data on-the-fly. We first randomly choose N non-empty cells and upload them into the graphics card memory. We then copy the other M-20 non-empty cells into the AGP buffers. The rest of non-empty cells are still resident in the system memory. For each cell, we use a flag to indicate whether its corresponding data is resident in the graphics card memory, the AGP memory, or the system memory. Thus, the size of the data set that can be rendered by our algorithm is limited by the size of the system memory.

The overview of the proposed algorithm is shown in Figure 5.2. The min-max octree construction, classification, and texture loading are performed in the pre-processing step, which are view independent. Our cell sorting algorithm organizes cells into layers. When a layer of cells are generated, we first check whether all non-empty cells are resident in the graphics card memory. If any non-empty cell within the layer is not resident in the graphics card memory, we need to upload it on-the-fly. Before we can transfer the data, we must determine which 3D texture object is used to receive the data. In other words, the current data stored in that 3D texture is replaced by the new data. We use a replacement queue to hold the cells that are already projected and can be switched out. When a layer of cells are sent to the GPU, we can not put them into the replacement queue immediately. Because we do not know whether the corresponding fragment programs executed on the GPU are finished or not. We use a NVIDIA OpenGL extension NV_fence to determine whether the cell projection of a layer of cells is finished on the GPU. This extension introduces the concept of a ”fence” to the OpenGL command stream. Once the fence is inserted into the command stream, it can be queried whether it is finished. After all OpenGL commands for cell projection of the layer of cells are issued, we insert a fence into the commands. Then, we query the fence's state after every layer of cells are projected. If the fence is completed, the cells before the fence are inserted into the replacement queue, and a new fence is inserted into the OpenGL commands stream again. In case the replacement queue is empty, we randomly choose a 3D texture whose corresponding cell has not been projected to receive the data. We will discuss the detail of the cell projection algorithm and cell sorting algorithm in the following sections.

5.2.1 Cell Projection

When the orthogonal projection is used, every cell projection on the image plane is given by the same hexagon shape per viewing direction. This projection can be computed once, and then used as a template for all cells, which can be obtained by translation. The rays intersecting with the cell are then determined by the cell projection efficiently. However, when the perspective projection is applied, the situation becomes more complicated. The cell projections on the image plane are different, and the pre-computed template can not be used any more, which make the CPU-based object-order ray-casting algorithm infeasible. The good thing is that the cell projection can be efficiently implemented on the

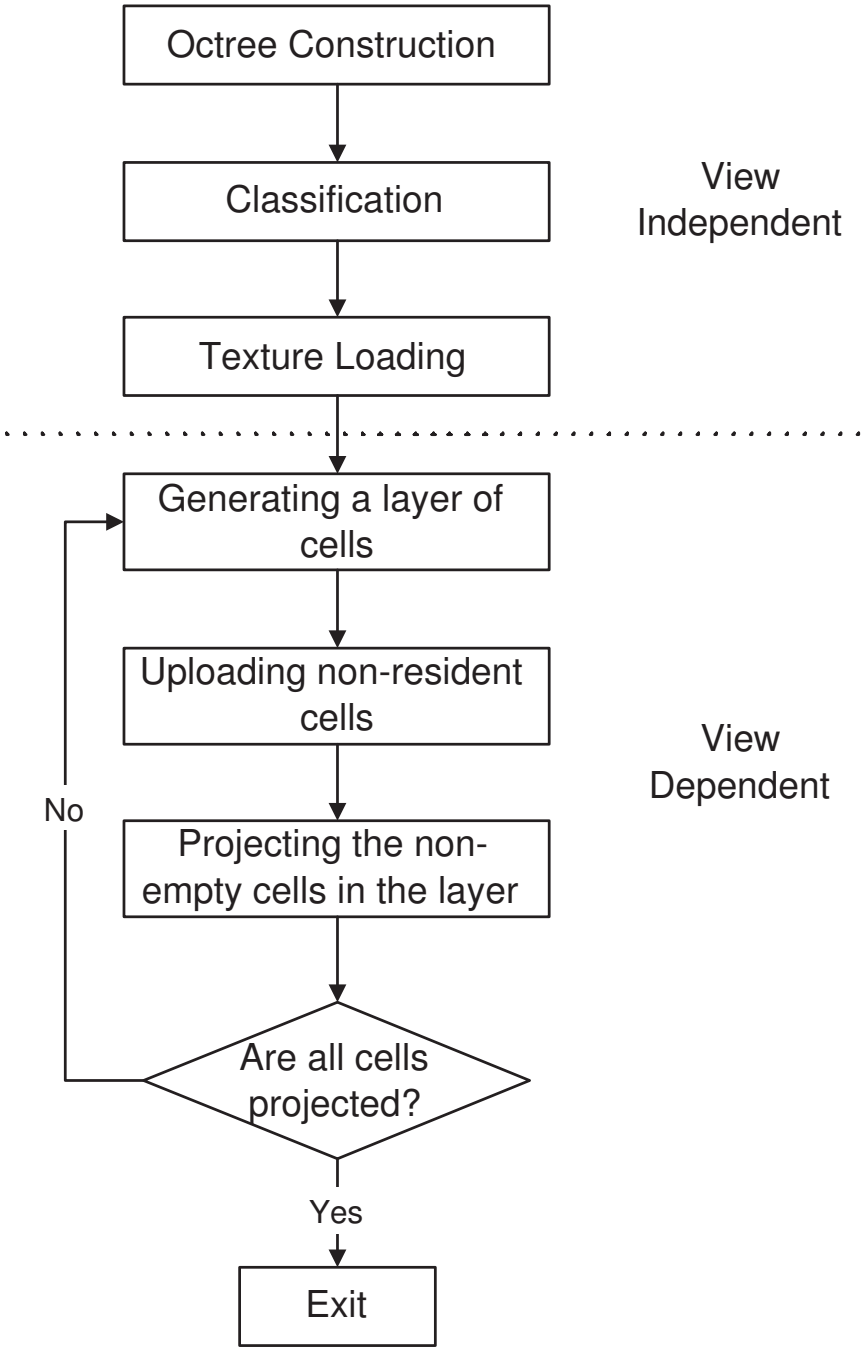


Figure 5.2: The overview of our GPU-based object-order ray-casting algorithm.

recent graphics card even when a perspective projection is used, which makes it possible to implement an object-order ray-casting algorithm on the GPU.

Our cell projection algorithm is implemented using fragment programs running on the

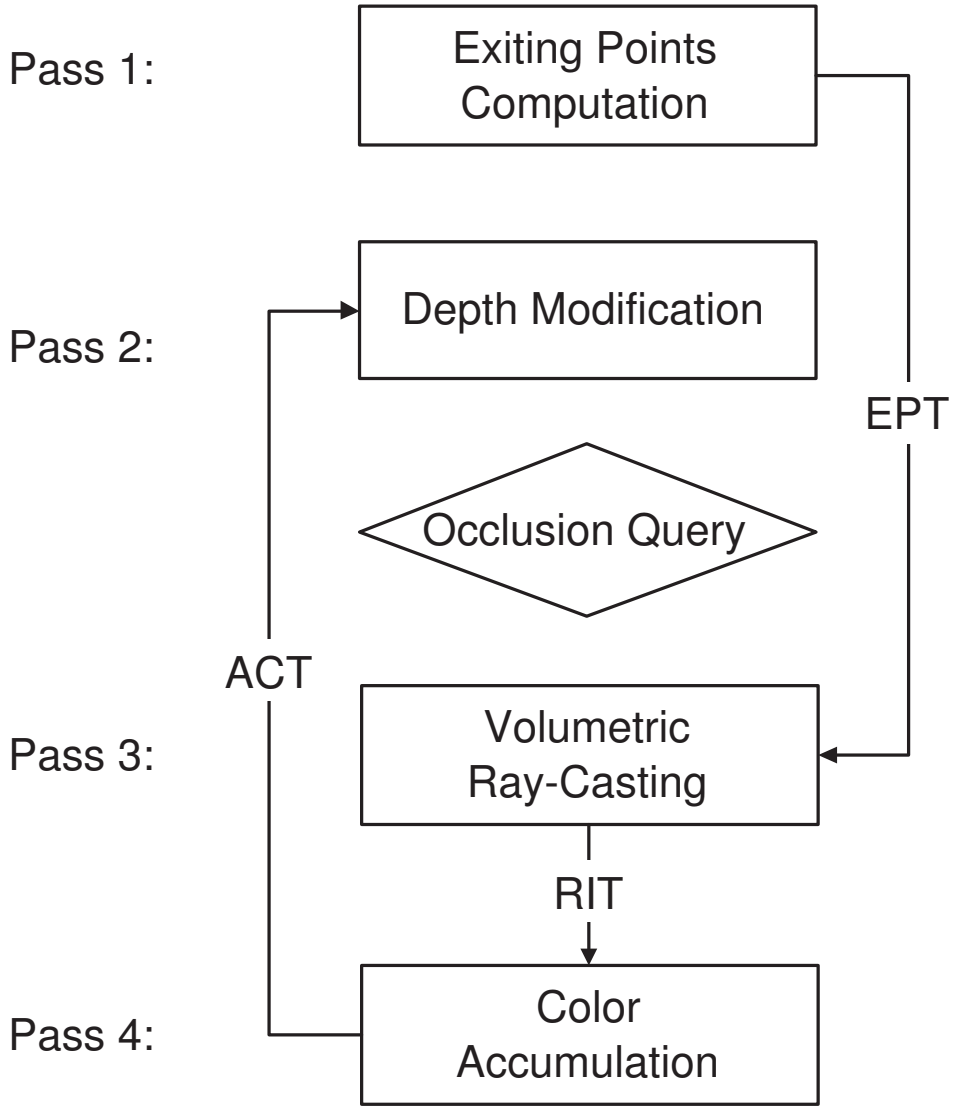


Figure 5.3: The pipeline of the cell projection.

GPU. When a cell is rendered, a number of fragments are generated, which are corresponding to the rays intersecting with that cell. For every non-empty cell that has to be projected, the rendering pipeline is shown in Figure 5.3. The proposed algorithm consists of four rendering passes for each cell. The modelview matrix and projection matrix remain unchanged for all four rendering passes. Hence, the fragments generated at the same window position in the four rendering passes correspond to the same ray intersecting with the cell.

OpenGL provides pixel buffers (pbuffer for short) for off-screen rendering. Combined with the render_texture extension, it allows the color buffer of the pbuffer to be used for both rendering and texturing. Three pbuffers are used as rendering targets for different render passes in our algorithm. The first pbuffer, the rendering target of the first rendering

pass, is used to store the exiting points of the rays that intersect with the projected cell. It is also bounded to a 2D RGB floating point texture, named *exiting points texture* (EPT), which is accessed in the third rendering pass to compute the length of each ray segment and normalized ray direction. The second pbuffer is made up of a depth buffer and a color buffer, which are the rendering targets of the second and third rendering pass, respectively. The depth buffer is used to implement early ray termination with the early-z test technique described in [74]. This optimization happens only if the fragment program is not going to modify the fragment depth. However, we need to modify the depth values based on the opacity values. We thereby use a separate rendering pass to modify the depth values. The color buffer is used to store the result of the ray integration, which is bound to a 2D RGBA floating point texture, named *ray integration texture* (RIT) and accessed in the last rendering pass. The third pbuffer is the rendering target of the last rendering pass, which is used to accumulate the color values. It is bound to a 2D RGBA floating point texture in the second rendering pass, which is named *color accumulation texture* (CAT). Its opacity values are accessed in the second rendering pass to modify the depth values accordingly for culling the fragments whose corresponding rays have already saturated their opacity values. The cell projection algorithm is described as follows:

- **Pass 1** (Exiting Points Computation): In the first rendering pass, the exiting points for the rays intersecting with the projected cell are computed by only rendering the back faces of the cell. For each vertex of the cell, we assign its texture coordinates in the corresponding 3D texture space as its primary color. The fragment program is straightforward, which just passes the fragment's primary color as output. As a result, we obtain a texture coordinate for each fragment, which is the coordinate for the exiting point of the ray in the texture space.

- **Pass 2** (Early Ray Termination): The opacity value of the fragment is accessed through the color accumulation texture (CAT). For any fragment whose opacity value exceeds 0.99, the depth value is set to one. As a consequence, if the depth test is set to GREATER, the corresponding fragment in the third rendering pass is discard.

- **Pass 3** (Volumetric Ray-Casting): The front faces of the cell are rendered to compute the entry points for the rays using the same method as Pass 1. In the fragment program, the exiting point is obtained through accessing the exiting point texture (EPT). The normalized ray direction and length of ray segment are computed in the 3D texture space. The ray is then evenly sampled with a sampling distance 0.5 to perform ray integration. It is impossible to pre-compute the gradient information and store them on the GPU for large data sets. Thus, we estimate the gradient on each sampling point on the fly. We use texture lookup to obtain the density at six neighboring positions, then estimate the gradient using central difference.

- **Pass 4** (Color Accumulation): The front faces of the cell are rendered again to generate corresponding fragments. In the fragment program, the color value and opacity

value of the projected cell are accessed through ray integration texture (RIT), and returned as color output directly. OpenGL blending is enabled in this rendering pass for accumulating the color and opacity values.

When all non-empty cells are projected, the color accumulation texture (CAT) holds the final image. In fact, the rendering Pass 2 is not need to be executed for every layer. In our implementation, we enable the rendering Pass 2 every other two layers.

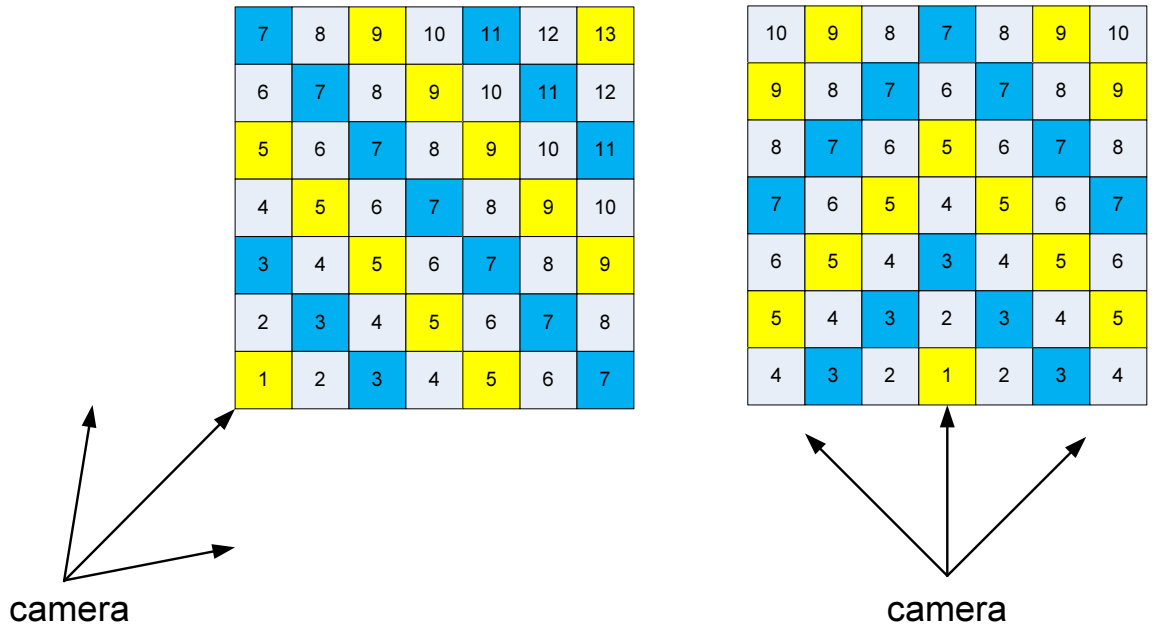
Because the rendering context are switched three times during the cell projection, this may cause a significant loss in performance on current GPUs. In order to decrease the number of rendering context switching, we need to project more cells in each rendering pass to improve the performance of the cell projection. Thus, we devise a cell sorting algorithm, which allows us to project a layer of cells each pass.

For virtual endoscopy applications, the virtual camera is always moved inside the volume, exploring the data set in a fly-through mode. When the camera is located inside the data set, the sorting algorithm of our object-order ray-casting approach becomes even simpler. The source cell is right the cell where the camera is currently located. Moreover, we only need to propagate the order information along with the viewing direction of the camera. The cell projection of the starting cell is implemented a little different from the of other cells. Only one rendering pass is needed to implement the projection of the starting cell. The rendering target is the third pbuffer used for color accumulation. The back faces of the starting cells are rendered to trigger the fragment program, which also give the exiting points of the corresponding rays. The camera position is passed to the fragment program as an uniform parameter. The ray direction is computed by using the exiting points and camera position. Then, the ray is evenly sampled to perform ray integration from the camera position. Our algorithm is efficient for virtual endoscopy applications using large volumetric data set. Because only a small number cells are used to generate the endoscopic view.

5.2.2 Cell Sorting

For a given viewing direction vector in the octree coordinate system, the signs of the coordinates determine the order in which the eight children are visited when parallel projection is used. When perspective projection is used, visibility order of the eight children can still be determined by the location of the camera to the octree. However, the octree structure only allows us to project at most four cells in one pass for some viewing directions. We need to project cells as more as possible to decrease the number of rendering context switching.

The main idea of our algorithm is to divide the cells into layers. We only need to determine the visibility order of layers. The cells within the same layer can be projected at the same time. It has been observed that the cells that have the same distance to the camera can be projected together. However, using the Euclidean distance from the cells to the camera to perform the cell sorting is inefficient. In order to improve the performance,



(a) (b)

Figure 5.4: Cells with the same Manhattan distance can be projected together. (a) The camera is located at the corner region, (b) The camera is located at the side region.

we use the Manhattan distance instead of the Euclidean distance. Moreover, we use the Manhattan distance between a source cell and the other cells to group the cells into layers. A source cell is determined first for a given view point, which is the closet cell to the camera. We then use a propagation method to compute the Manhattan distance for the other cells. The cells that have the same Manhattan distance to the source cell are put into the same layer. We first describe our cell sorting algorithm in the 2D case, and then extend it to 3D.

In the 2D case, the whole object can be represented with a square, and the camera can be set up around the square. We first find the closest cell to the camera based on the camera's location with respect to the square. If the camera is located at the corner region as shown in Figure 5.4(a), the closest cell is the corresponding corner cell shown in grey. Otherwise, the closest cell is on the edge of the square that is opposite to the camera as shown in Figure 5.4(b). The closest cell can be obtained by shooting a ray perpendicular to the edge. The intersected cell is the closest cell. If the ray intersects two cells, the two cells are both used as source cell. In Figure 5.4, we shows the Manhattan distance of each cell. It has been observed that the cells show a very clear layer structure. It is also noted that each layer consists of more cells by using the Manhattan distance than using the Euclidean distance. In fact, we do not need to explicitly compute the Manhattan distance. From the Figure 5.4, we can see that the source cell is made up of the first layer. And, the second layer consists of the edge neighboring cells of the source cell. Thus, we can use a propagation method to

group the cells into layers from the source cell C_0. The propagation algorithm is described as follows:

1. Let $C_0.visited = 1$ and put C_0 into a list L_0. Set the other cells to be un-visited.
2. For each cell C_i in the list L_0
   (a) Obtain the four edge neighboring cells $C_{ij}(j = 0, 1, 2, and 3)$ of C_i. If $C_{ij}.visited$ is 0, let $C_{ij}.visited = 1$ and put C_{ij} into the list L_1.
3. Project the non-empty cells of L_0. If all non-empty cells are projected, the algorithm is terminated.
4. Copy L_1 to L_0, and goto 2.

By using this sorting algorithm, each layer of the cells have the same Manhattan distance to the source cell. The cells within the same layer does not occlude with each other, which can be projected at the same time. This algorithm can be easily extended to the 3D case. In the 3D case, the closest cell still can be find efficiently based on the region where the camera is located with respect to the volumetric data set. The propagation process is almost same, except that we need to use the six face neighboring cells for propagation in the 3D case.

5.2.3 Implementation and Results

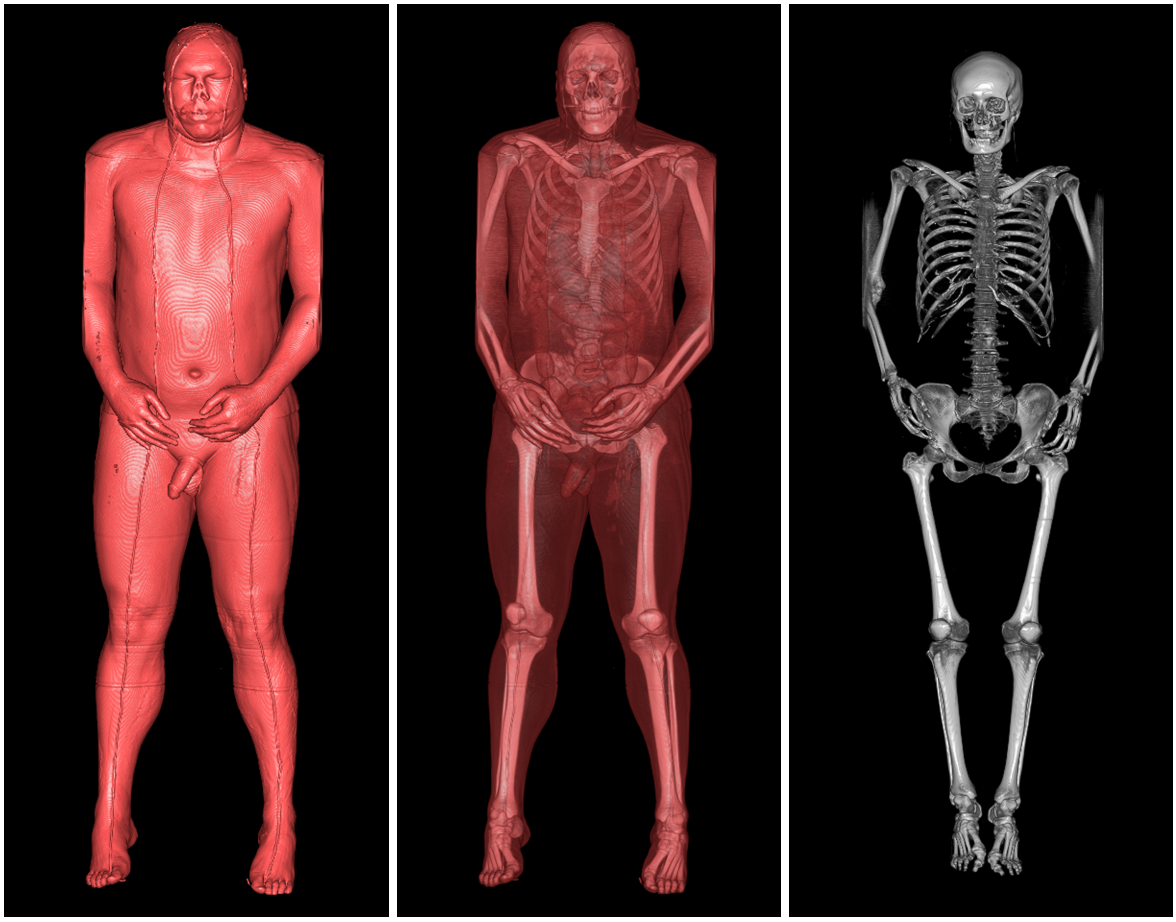
In this section, we present some implementation details and testing results. The presented algorithm is implemented using C/C++, and fragment programs are implemented using Cg [88]. The experiments have been conducted on a 3.0GHz Intel Pentium IV PC, with 2G RAM and a NVIDIA Quadro FX 3400 graphics card. We list the information of the data sets used in our experiments in Table 5.1.

The size N of the cell is crucial to our algorithm. A smaller N is efficient for empty space skipping, but inefficient for the cell projection executed on the GPU. Because using a smaller N will increase the number of rendering context switching, which decreases the performance. Furthermore, it also increase the number of texture objects switching because our cells are stored in separate 3D textures. A smaller N will result in the projection of the cell covering less pixels on the image plane, which degrade the efficiency of the volumetric ray casting because of the poor caching. Moreover, for each cell normalized ray direction and length of the ray segment are needed to be computed for the rays intersecting with that cell. A larger N can decrease such computation. We choose $N = 64$ in our implementation for the purpose of the trade-off between the empty space skipping and the cell projection on the GPU.

We use the full resolution Visible Human CT data sets to test our algorithm. About a half of cells are skipped after the classification. Thus, most cells are fitted into the graphics card memory and the AGP memory. Only a small number of cells are still resident in the

Table 5.1: The size of the data sets used in our experiments.

Data Set	Dimension	Size
Visible Male	$512 \times 512 \times 1887$	0.71GB
Visible Female	$512 \times 512 \times 1734$	0.65GB
Visible Korean Human Brain	$1080 \times 1110 \times 158$	0.93GB



(a) 3.1 fps (b) 1.8 fps (c) 3.5 fps

Figure 5.5: (a) and (b) are rendered using Visible Male data sets with opaque and semi-transparent transfer functions, (c) is rendered using Visible Female data sets with an opaque transfer function.

system memory. We can achieve several frames per second for such large data sets on a commodity PC. We show some resulting images in Figure 5.5, which are all rendered at the resolution of 512×1024. In Figure 5.5(a), we show the skin of the Visible Male using an opaque transfer function. In Figure 5.5(b), we show the bone structure and some organs of the Visible Male using a semi-transparent transfer function. In Figure 5.5(c), the bone

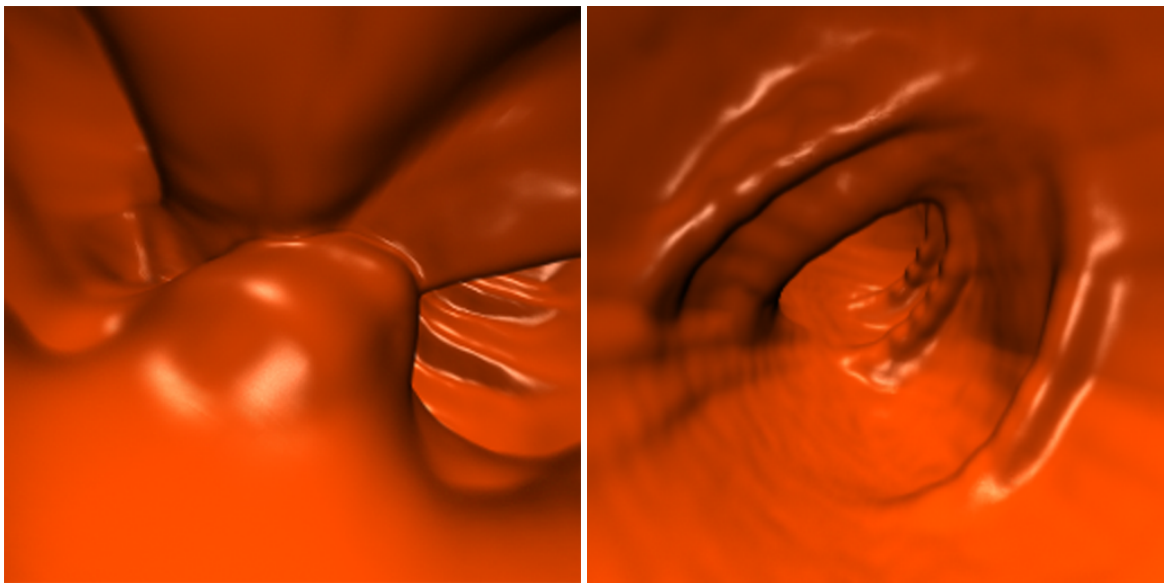
of the Visible Female is shown by using an opaque transfer function. It is natural that we achieve higher rendering speed when the opaque transfer functions are applied. Because more cells are skipped in the object-space and less cells need to be projected.

(a) 1.5 fps (b) 21.8 fps (c) 23.1 fps

Figure 5.6: (a) A top view of the full resolution brain data set rendered using our algorithm, (b) The segmented brain stem rendered in real-time using our algorithm, (c) The segmented brain ventricle rendered in real-time using our algorithm.

We also use a segmented photographic volumetric data set to demonstrate the efficiency of our algorithm. Compared with the CT data sets, the volume rendering for photographic data sets requires an opacity transfer function from the non-linear color space, which is more complicated than that for the CT data sets. We use the CIE *Luv* color space to obtain a perceptually uniform representation of the color volume, and assign an opacity value for each voxel using the method proposed by Ebert et al. [24]. Thus, each voxel of this data set contains RGB color, opacity and segmentation information. In order to render segmented data sets, for each cell we store a list of labels that are used for labeling the voxels in the cell. When an organ is chosen for rendering, only the cells containing the corresponding label are loaded into the graphics card memory for projection. These cells usually can be fitted into the graphics card memory without on-the-fly transferring data, which allows us to interactively explore the segmented organs. In Figure 5.6, we show some resulting images rendered from the segmented brain data set with a resolution of 512×512. A top view of the full resolution brain data set is shown in Figure 5.6(a). The segmented brain stem and ventricle can be rendered in real-time shown in Figure 5.6(b) and 5.6(c), because all the related cells can be fitted in the graphics card memory.

The size of the contemporary clinical CT data sets is usually 512^3. It is impossible to pre-compute the gradient information and store them on the GPU. Therefore, the gradient at each sampling point needs to be estimated using central difference method on-the-fly, which needs additional six texture lookups and lowers the rendering performance. By means of object-order GPU ray-casting, it is possible to pre-compute the gradient information and store them on the GPU, because only a small number of cells are utilized to generate an endoscopic view. Figure 5.7 shows two different endoscopic views inside a



(a) 24.3 fps (b) 21.8 fps

Figure 5.7: (a) A close view of a polyp rendered at 24.3 fps, and (b) A typical endoscopic view rendered at 21.8 fps.

human colon, rendered with a size of 512×512 in real-time.

5.3 Hybrid Volumetric Ray-Casting

The main bottleneck of our GPU-based ray casting algorithm for rendering large volumetric data sets is on-the-fly transferring data from the system memory to the graphics card memory, although we have already used OpenGL pixel buffer object (PBO) extension to accelerate it. Data transferring stalls the busy OpenGL pipeline, while the CPU is almost free. Although the CPU-based direct volume rendering algorithm are not suitable to generate high-quality images in real-time due to the lack of the ability of parallel processing and hardware support for trilinear interpolation and local illumination, many techniques have been exploited for accelerating volume rendering in software. Knittel [72] has described an architecture and implementation that makes extensive use of MMX and streaming SIMD instructions for perspective ray-casting on a PC. Moreover, various precomputed data structures have been proposed in software to rapidly traverse or skip over the empty voxels that have no contribution to the rendered image, such as octree [77, 83], K-d tree [119], bounding convex polyhedrons [5], and proximity clouds [17]. The min-max octree structure [77] allows changing the classification interactively, which have been used in our object-order GPU ray-casting algorithm to organize volumetric data sets.

Since the CPU is almost free when the GPU is used for data transferring and rendering, we can move some work from the GPU to the CPU to fully utilize the computation power of

the CPU. Westermann and Sevenich [132] have proposed to combine the processing power of the CPU and the GPU to accelerate volumetric ray-casting. They computed the ray entry points and exit points using texture mapping, and the results are read back from the GPU. Then, the ray traversal is performed in software. The main drawback of this method is that on current graphics hardware the performance of read back is poor and the whole pipeline is stalled. Thus, the CPU and the GPU can not work in parallel in their method.

We devise a method that fully exploits the advantages of both the CPU and the GPU, making them work in parallel to accelerate the volumetric ray-casting algorithm. The basic idea of our hybrid volumetric ray-casting method is as follows. We use the GPU to perform streamed trilinear interpolation, local illumination and compositing, and use the CPU to perform the ray traversal and maintain an elaborate data structure. For example, the ray determination and exiting points computation can be done by the CPU. This algorithm can be seamlessly integrated with our object-order GPU ray-casting algorithm.

In order to take advantage of the parallelism between the CPU and the GPU, the rays are grouped into small tiles. Each tile of rays corresponds to a square region on the image plane. The flowchart of our hybrid volumetric ray-casting algorithm is shown in Figure 5.8. After the tile construction, a ray determination step is executed on the CPU to compute ray entry points and normalized ray directions. We then apply a multi-pass slab rendering algorithm on the GPU, in which the CPU is only used to issue some non-block OpenGL commands. The detail of the ray determination, multi-pass slab rendering, and hole filling algorithms are described in the following sections.

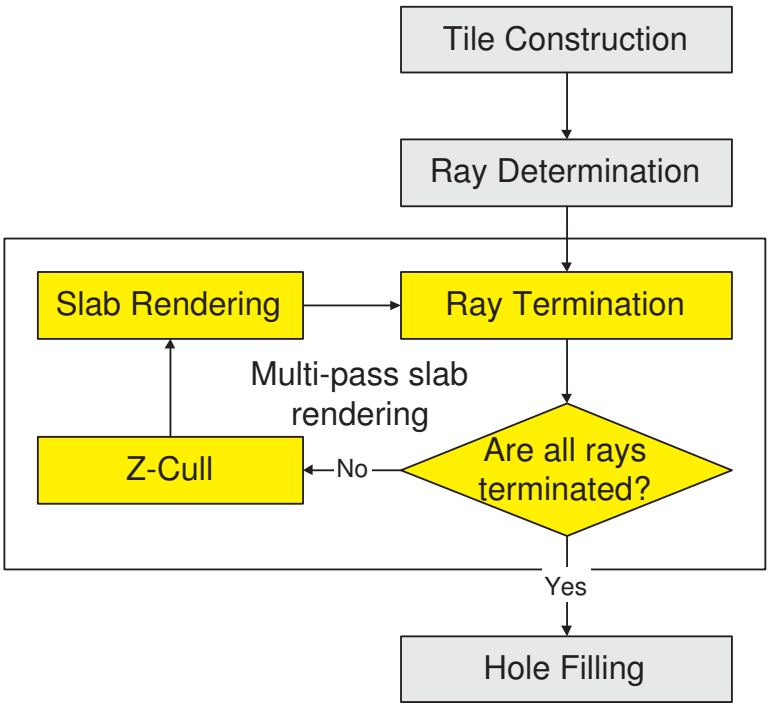


Figure 5.8: Flowchart of our hybrid volumetric ray-casting algorithm.

5.3.1 Ray Determination

In our hybrid volumetric ray-casting algorithm, the ray determination is done by the CPU. The computation results will be uploaded to the GPU. First, we need to compute the position of each pixel on the image plane, along with the normalized ray direction. The main problem is the normalization of the ray direction, which requires more instructions to compute the reciprocal of the distance between the camera and each pixel position. On the positive side, the distance between the camera and the image plane as well as the angle of the field of view are both fixed, the distance between the camera and each pixel on the image plane is also unchanged for every frame. Thus, we can pre-compute the reciprocal of distances between the camera and pixels on the image plane, and use them to scale the ray direction to obtain the normalized ray direction, which only needs three multiply instructions.

Second, we need to compute the first intersection point of each ray with the object boundary. By using the min-max octree structure, this computation can be done very efficiently. Instead of computing the first intersection point of each ray with the real object boundary, we compute the intersection point of each ray with the cell nodes of the min-max octree first. We use a parameter l_{min} to control the minimal level of cells that the ray can reach, which means that when the ray hits a non-empty cell with level equal to l_{min}, the ray traversal is stopped. Whether moving these points to the exact object boundary is based on the workload of the CPU.

5.3.2 Multi-pass Slab Rendering

After the ray determination step, all the ray entry points and normalized ray directions are computed and stored in two 2D floating point textures. After these two textures are uploaded to the GPU, we can perform ray integration using the GPU. However, at this stage we do not know how far each ray of the tile will travel before being terminated or leaving the volume. Our simple and efficient solution is using the GPU to perform multi-pass slab rendering, and simultaneously using the CPU to decide when the multi-pass slab rendering should be terminated on the GPU.

In order to make the multi-pass slab rendering more efficient, the rays within each tile are further divided into quads. Each ray quad consists of 2×2 rays. It has been observed that if one ray of the four rays are terminated, the other rays are likely terminated in the following slab rendering pass. Consequently, we treat the ray quad as one basic unit in the multi-pass slab rendering step.

After the empty space leaping in the ray determination step, some rays may already leave the volume. For each quad of the tile, we check whether its four rays have already left the volume or not. If so, the quad is marked as *terminated*. If only some part of the four rays leave the volume, the quad is marked as a *hole*. Otherwise, the quad is marked as *non-terminated*. If a tile does not contain any terminated quad and hole quad, the tile is called a *full tile*. Otherwise, it is called a *partial tile*. We first employ the multi-pass slab rendering algorithm to the full tiles until all full tiles become partial tiles. This is done by employing

slab rendering on the GPU, and in the meantime the ray termination is performed on the CPU. The ray termination and slab rendering algorithms are same for full tiles and partial tiles. The only difference is that the Z-cull step is skipped for the full tiles, because we do not need to modify the depth values to cull the terminated quads for the full tiles in the slab rendering.

5.3.2.1 Z-Cull

On the current graphics hardware, before a fragment reaches the fragment processor, the z-cull unit is used to compare the fragment depth with corresponding value that already exists in the depth buffer. If the fragment depth is greater, the fragment will not be visible, and the fragment program is not executed by the fragment processor. In our implementation, the depth buffer is initialized with one at the beginning. In the ray termination step, it will generates two lists containing the terminated quads and hole quads respectively. Because we only need to modify the depth buffer, the three color channels are masked in this step. For the terminated quad, we render its corresponding quad with depth value zero. As a result, the depth values corresponding to the terminated rays are all set to zero, and, the corresponding fragments will be culled before they reach the fragment processors in the following slab rendering passes. We perform the same for the hole quad using a different depth value 0.4. Because the hole quads still need to be rendered to trigger fragment programs in the hole filling step. By this technique a quad with depth value smaller than 0.4 can be rendered to trigger hole filling fragment programs for the hole quad in the final step. After the depth values are modified to enable z-cull, the quads corresponding to the partial tiles are rendered to trigger the slab rendering fragment program.

5.3.2.2 Slab Rendering

When the fragments pass the z-cull test and reach the fragment processor, our fragment program is executed on the GPU. In the fragment program, N uniformly sampled points along the rays of sight are processed. At each sampling point, we perform trilinear interpolation to obtain the density value and gradient, and perform the post-classification to obtain the color for the sampling point. The gradient is pre-computed for each voxel with central difference and uploaded to the GPU along with its density values using a 3D RGBA texture, if the gradient information can be held in the graphics card memory. Otherwise, we need to compute the gradient on-the-fly.

Since we only integrate N sampling points along each ray in one slab rendering pass, some rays may saturate their opacity values or leave the volume, which should be terminated. While others still need to be processed in the following slab rendering pass. After we issue the OpenGL commands to render the quads to trigger the fragment programs, we start to detect which non-terminated quad becomes terminated, and which non-terminated quad becomes hole quad after the slab rendering on the CPU. The slab rendering fragment program and ray termination are performed in parallel on the GPU and the CPU respectively.

5.3.2.3 Ray Termination

In order to accurately determine when a ray should be terminated on the GPU, the ray also needs to be uniformly sampled on the CPU using the same sampling distance as that on the GPU. At each sampling point, the density value should also be tri-linearly interpolated. The opacity value is then queried through the same opacity transfer function and accumulated along the ray. When the accumulated opacity value exceeds the predefined threshold, the ray should be terminated on the GPU. It is obvious that the CPU and the GPU perform some overlapping work in this method, which is inefficient. Moreover performing the trilinear interpolation on the CPU causes a loss in performance.

We describe here an efficient method to estimate when a ray is terminated on the GPU. As mentioned in the previous section, each leaf cell of the min-max octree is defined as the cubical region with voxels on its eight corners. Given a transfer function, it is classified as opaque, when the density values of its eight voxels are all greater than 0.99.

It is noted that the accumulated opacity value is greater than the source opacity value. When a ray pass through an opaque cell, the sampling point in this cell has an opacity value greater than 0.99, so does the accumulated opacity value. Thereby, a ray should be terminated if it passes through an opacity cell. In this method, the time consuming trilinear interpolation is avoided. The main task of this method is to compute the cells that are pierced by the ray, which can be efficiently obtained by using a 3D digital differential analyzer (3DDDA) [4]. Given two endpoints of the ray, this algorithm generates a 6-connected line, which includes all of the cells pierced by the ray.

For each non-terminated quad, we use the proposed method to check whether the quad contains any ray will be terminated after the corresponding slab rendering pass. If all four rays of the quad should be terminated after this slab rendering pass, it is marked as terminated and put into a list that stores the new generated terminated quads. If all four rays of the quad are not terminated after this slab rendering pass, we keep it unchanged. Otherwise, the quad is marked as hole and put into a list that stores the new generated hole quads. The two lists will be used to in the Z-Cull step to modify the corresponding depth values.

For each partial tile, if the number of the non-terminated quads is less than a predefined threshold, the whole tile of rays are terminated, and the tile is removed from the partial tile list. This threshold value can also be used to control the balance between the CPU and the GPU. We will discuss it in Section 5. If the partial tile list is empty, the multi-pass slab rendering algorithm is terminated.

For semi-transparent transfer functions, the early ray termination is not as effective as for opaque transfer functions. Consequently, we do not need to test whether a ray pierces an opaque cell. Because the volume only contains very few opaque cells or not. We only need to check whether the ray leaves the volume, which makes the ray termination algorithm even simpler. To make this checking more efficient, we also compute the length for each ray in the ray determination step.

5.3.3 Hole Filling

After the multi-pass slab rendering, the hole quads still need to be processed. For each non-terminated rays within the hole quads, we first compute the point where the ray leaves the volume. Then, we employ the space leaping from both ends of the ray based on the workload of the CPU and the GPU. We will discuss this in detail in the next section. Then, we store the length of ray segment in a 2D texture and upload it to the GPU.

Before we execute the hole filling fragment programs on the GPU, we need to modify the depth values of the terminated rays within the hole quads to cull the corresponding fragments. Then, a bounding box enclosing all hole quads is computed and rendered with depth value 0.2 to trigger the hole filling fragment programs. The hole filling fragment program is very similar to the slab rendering fragment program. The only difference is that the hole filling fragment programs have different travel steps based on the the length of the corresponding ray segment.

5.3.4 Dynamic Workload Balancing

The workload balance between the CPU and the GPU is crucial to our hybrid algorithm. The ideal situation is that the programs running on the CPU and the GPU take almost the same time for rendering one frame. In our method, we use NVIDIA performance toolkit to access the *gpu_idle* counter to determine if the GPU is underloaded. The *gpu_idle* counter contains the percentage of time the GPU is idle since the last call. If the *gpu_idle* counter is greater than zero, the workload of the CPU should be reduced and some work passed to the GPU. On the other hand, if the GPU is always busy, some work needs to be passed to the CPU. The basic idea to balance the workload of the CPU and the GPU is controlling the degree of the empty space skipping on the CPU. The more empty voxels are skipped, the less work needs to be done by the GPU. On the contrary, the CPU can do less work.

The ways to adjust the workload of the CPU and the GPU are described as follows:

1. The first place that we can control the empty space skipping is the ray traversal in the min-max octree. If we want to reduce the workload on the CPU, we stop the rays at a high level partial cell before reaching the object cell, when we compute the first intersection point.

2. When we compute the first intersection point, we compute the intersection point between the ray and the cell. The ray does not go inside the cell. If we want to increase the workload on the CPU, we can let the ray move into the cell and reach the real object boundary.

3. For each ray, we can also compute the existing point and employ empty space skipping from existing point in the reverse direction. This way increases the workload on the CPU and efficiently decreases the workload on the GPU.

4. For a partial tile, if the number of the non-terminated quads is less than a threshold, the whole tile rays are terminated. A larger threshold can be used if the workload of the GPU needs to be reduced. A smaller threshold results in the tiles are terminated quickly on the CPU. Therefore, the workload of the GPU is increased.

5.3.5 Implementation and Results

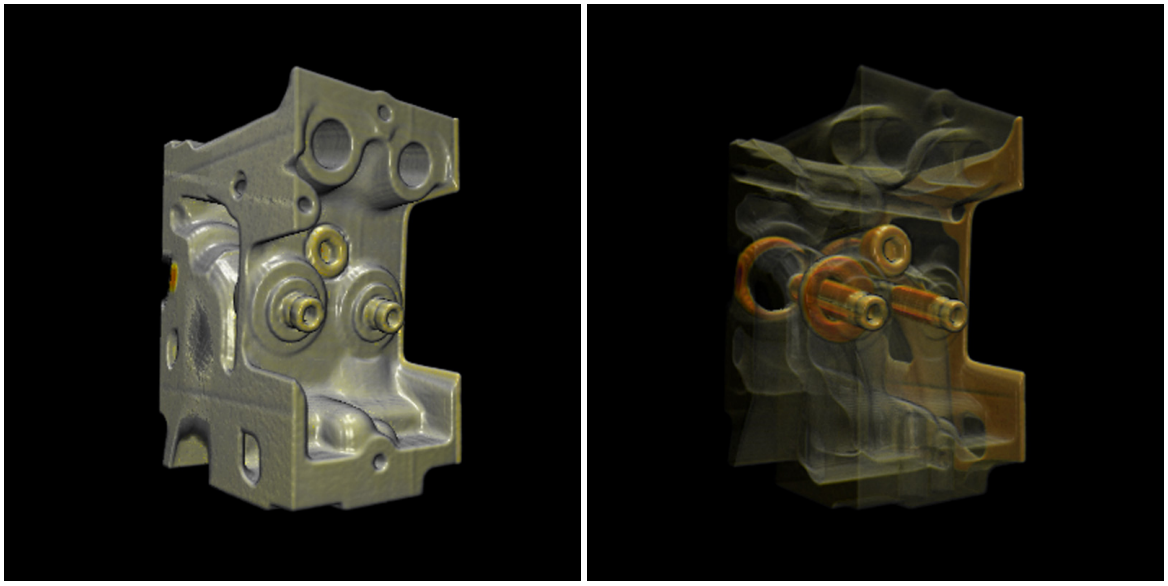
In this section, we present some implementation details and testing results of our hybrid volumetric ray-casting. All images shown in this section have a resolution of 512×512. We use 0.5 as the sampling distance in our implementation, which is good enough to generate high quality images for all tested data sets. Most of the experiments have been conducted on a 3.0GHz Intel Pentium IV PC, with 1G RAM and an NVIDIA Quadro FX 3400 graphics card (PC1). We have also used a 2.4GHz Intel Pentium IV PC, with 1G RAM and an NVIDIA Geforce 6800 Ultra graphics card (PC2) to demonstrate workload balancing. Both PCs are running Windows XP operating system.

We use CT engine and CT human foot data sets to demonstrate the proposed hybrid volumetric ray-casting for general volume rendering on PC1. The image resolutions are all 512×512. The two data sets are rendered both with an opaque transfer function and a semi-transparent transfer function on both PC1 and PC2. We list the rendering timings in Table5.2. The CPU on PC1 is faster than that on PC2. Thus, we employ accurate empty space skipping on PC1 and a coarse empty space skipping on PC2. Because the GPU on PC2 is faster than that on PC1, we obtain similar performance on both PC1 and PC2. The performance on PC1 is a little faster than that on PC2, because the PC1 uses PCI Express which is faster than AGP8 used by PC2, and the CPU on PC1 is also faster than that on PC2. We can see that when the semi-transparent transfer function is applied on PC1, the performance is dropped a little, because the early ray termination is not effective at this situation. While the performance on the PC1 for two cases are nearly same. Because the 3DDDA algorithm is not performed on the CPU when semi-transparent transfer function is applied.

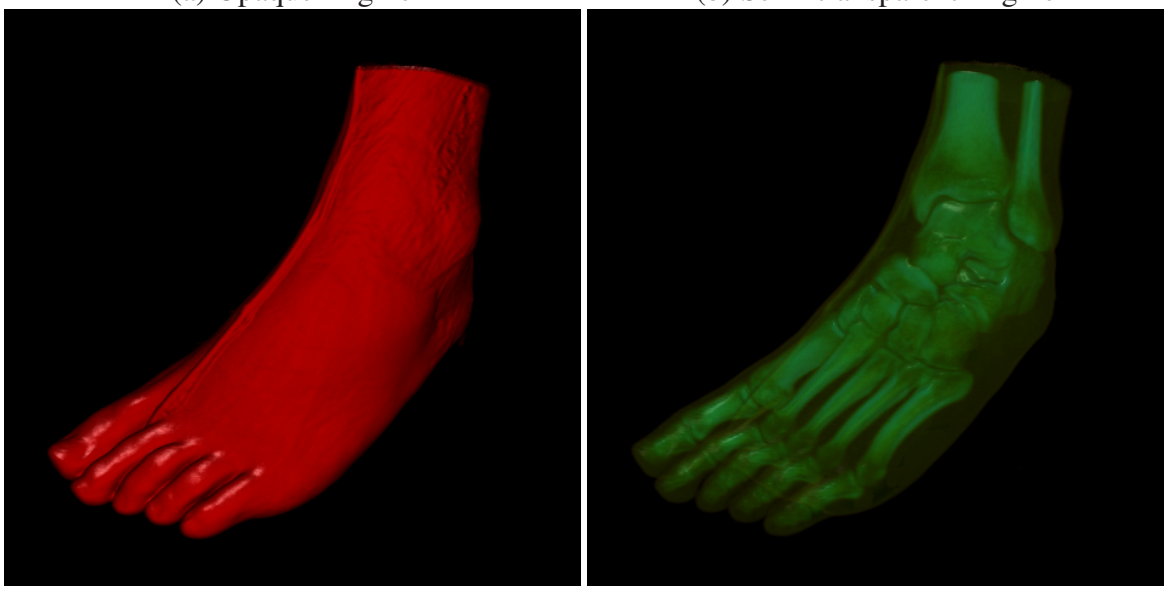
Table 5.2: Average rendering frame rates per second (fps) for the engine and human foot data sets.

Data Set	Size	Opaque		Semi-transparent	
		PC1	PC2	PC1	PC2
Engine	$256 \times 256 \times 128$	21.9	17.9	17.8	18.0
Foot	$152 \times 256 \times 220$	19.8	14.5	14.3	13.6

Nvidia Geforce 6 series cards support dynamic branching in the fragment program, which makes it possible to implement one-pass GPU-based volumetric ray-casting algorithm. We use the lego car, lobster and human tooth data sets to compare our hybrid algorithm with the pure GPU-based algorithm. In Figures 5.10(a) and 5.10(b), we show



(a) Opaque Engine (b) Semi-transparent Engine



(c) Opaque Foot (d) Semi-transparent Foot

Figure 5.9: Volume rendering of the engine (a-b) and foot (c-d) data sets with opaque and semi-transparent transfer functions.

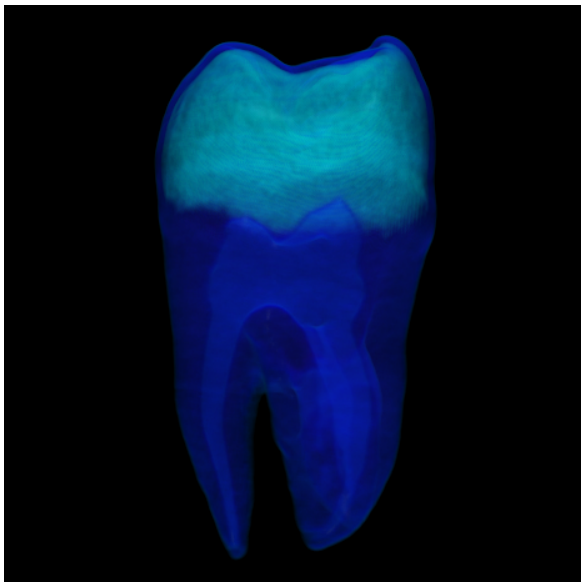
two semi-transparent lobsters rendered with the two different methods from the same view point. We can see that we obtain the same image quality, the difference between the two images can not be observed. For the lego car and human tooth data sets, we only show the volume rendering images by our hybrid method in Figures 5.11(a) and 5.11(b). The performance of the two algorithms are list in Table 5.3, which show that our hybrid algorithm is faster than pure GPU-based algorithm, because we also use the power of the CPU to assist

(a) Lobster (HRC)

(b) Lobster (GRC)

Figure 5.10: Volume rendering of a semi-transparent lobster with out hybrid volumetric ray casting (a) and with a pure GPU ray casting (b).

(c) Lego Car



(d) Human Tooth

Figure 5.11: Volume rendering of the lego car data set with an opaque transfer function (a) and the human tooth data set with a semi-transparent transfer function (b).

the GPU. The experiments are conducted on the PC1.

Table 5.3: Average rendering frame rates per second (fps) for car, lobster and tooth data sets by using our hybrid volumetric ray-casting algorithm (HRC) and pure GPU-based volumetric ray-casting algorithm (GRC).

Data Set	Size	HRC	GRC	Speedup
Lego Car	$256 \times 256 \times 128$	19.1	16.5	15.8%
Lobster	$152 \times 256 \times 220$	28.3	20.4	39.1%
Tooth	$128 \times 128 \times 256$	32.8	16.9	94.1%

5.4 Conclusions

We have presented an object-order GPU ray-casting algorithm for rendering large volumetric data sets such as the Visible Human CT data sets and 16bit CT data sets with pre-computed gradient information. The volume data set is decomposed into small cells, and organized using a min-max octree structure. The empty cells are skipped immediately after the classification. The volumetric ray-casting algorithm is performed on the GPU for each non-empty cell projection, and the resulting integration of the cell are front-to-back composited to generate the final image. We devised a cell sorting algorithm to allow us project a layer of cells at the same time, which improves the performance of the fragment programs on the GPU. While the hybrid volumetric ray-casting algorithm exploits the parallelism between the CPU and the GPU to obtain further acceleration.

Chapter 6

Computer Aided Polyp Detection

6.1 Introduction

Because of the complex structure of the colon surface, the inspection is prone to errors, and the physician needs to navigate antegrade (from rectum to cecum) and retrograde (from cecum to rectum) to improve the coverage and accuracy of the inspection [57]. A complete inspection by a radiologist conducting 3D VC takes 10-15 minutes [60]. The long interpretation effort of the VC screening procedure suggests a CAD approach. A CAD scheme that automatically detects the locations of the potential polyp candidates could substantially reduce the physicians' interpretation time and improve their diagnostic performance with higher accuracy. However, the automatic detection of colonic polyps is a very challenging task because polyps can occur in various sizes and shapes. Moreover, there are numerous colon folds and residual colonic materials on the colon wall that mimic polyps and could result in FPs. A CAD scheme should have the ability to identify true polyps and eliminate the FPs.

In our earlier work [125], we have observed that the internal tissues of polyps have a slightly higher density and different texture than healthy tissues. These high density areas are beneath the colon wall and cannot be seen with an optical colonoscopy. However, the internal structure of polyps can be revealed through volume rendering with a translucent transfer function, called *electronic biopsy*. Pickhardt [106] has presented that translucency rendering effectively improves polyp specificity and increases overall diagnostic confidence, especially when barium tagging of residual stool is used to maximize the full benefit of the technique. By significantly reducing the need for 2D correlation, translucency rendering greatly decreases interpretation time for primary 3D virtual colonoscopy.

The four images of Figures 6.1(e)-(h) are the electronic biopsy images of the four corresponding objects of Figures 6.1(a)-(d). Although polyps and normal tissues may have similar shapes, it is observed that adenomatous and malignant polyps have a higher density and different texture beneath the surface. As shown in Figure 6.1, four different objects including retained stool, a hyperplastic polyp, an adenoma, and a tubulovillous adenoma have different rendering results for the same transfer function. The retained stool has a

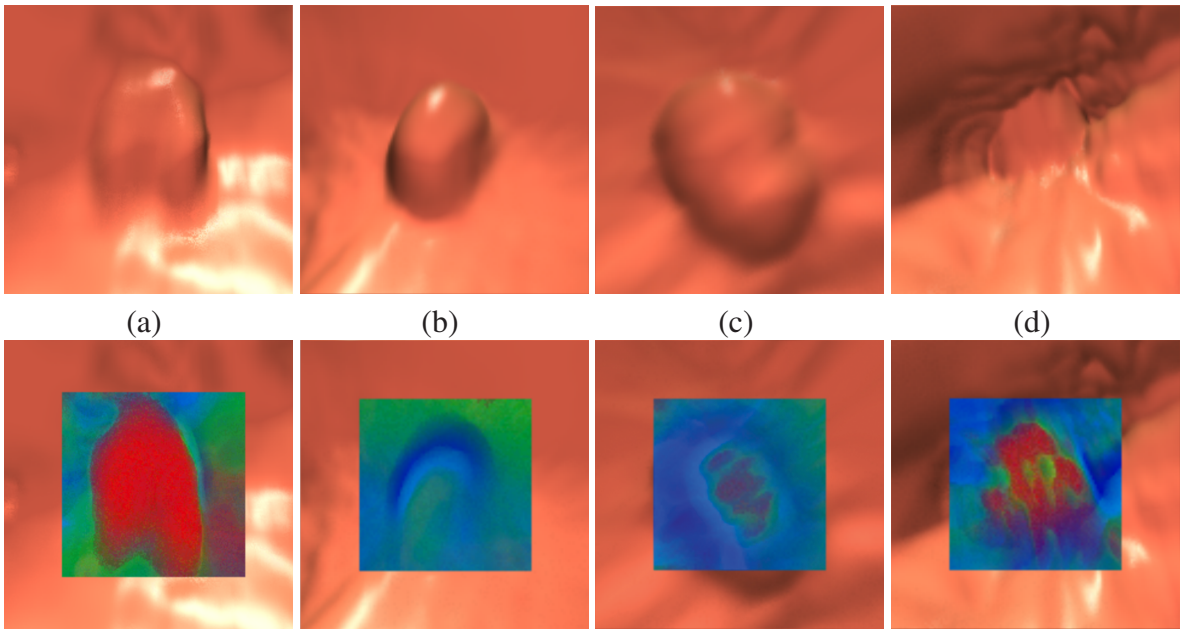


Figure 6.1: (a)-(d) are the surface rendering of (a) retained stool, (b) a hyperplastic polyp, (c) an adenoma, and (d) a tubulovillous adenoma. The small square images in (e)-(h) are the electronic biopsy rendering of the respective objects in (a)-(d), all with the same transfer function. In the electronic biopsy images, the red color represents the highest densities and blue represents the lowest densities. Green represents tissues of middle densities. Normal tissues have low to middle densities.

uniform high density inside the whole object with sharp boundaries due to the oral agent tagging. The hyperplastic polyp is benign and does not have any high density voxels. The adenoma and tubulovillous adenoma are neoplastic with irregular internal structures and high density voxels gradually change to normal tissues towards the boundary. These observations suggest that polyps can be detected by analyzing the electronic biopsy images of the whole colon.

In our method, we conformally map the colon surface to a 2D rectangle, which simplifies the polyp detection problem from 3D to 2D. Our polyp detection method is then applied on high-quality 2D electronic biopsy images generated with a volumetric ray-casting algorithm. Unlike previous shape based methods, in which shape information is computed for polyp detection in the entire colon, we only compute the shape information at suspicious regions in order to reduce FPs.

6.2 Our CAD Pipeline

A diagram of our CAD pipeline is shown in Figure 6.2. First, for segmentation and digital cleansing of the colon, an iterative partial volume segmentation algorithm is applied. Then, a topologically simple colon surface is extracted for conformal colon flattening. The electronic biopsy colon image is then generated using a volumetric ray casting algorithm on the entire flattened colon. After that, our clustering algorithm and reduction of FPs are performed. All of these processes are performed automatically in our pipeline. The details of each step are discussed in the following subsections.

6.2.1 Segmentation and Digital Cleansing

The first step in our pipeline aims to segment the colon lumen from the CT scan of the patient's abdomen. The day prior to the scan, the patient drinks an oral contrast agent in order to tag colonic material, making it unnecessary for the colon to be cleaned out physically. The tagged material is enhanced in the CT scan, allowing it to be identified. Care must be taken during electronic cleansing to restore CT density values where the partial-volume effect occurs. A partial volume segmentation algorithm has been proposed in Chapter 3 to handle partial volume effect. In this algorithm, the interface layer between the air and the tagged colonic materials is identified and removed digitally. Moreover, the CT density values of colonic tissues in the enhanced mucosa layer are restored based on the mixture information. After this step, we obtained a segmentation of the colon and a clean colon lumen. The segmentation of the colon will be used to compute centerline and extract colon surface. The clean colon lumen will be used for visualization and automatic detection of colonic polyps.

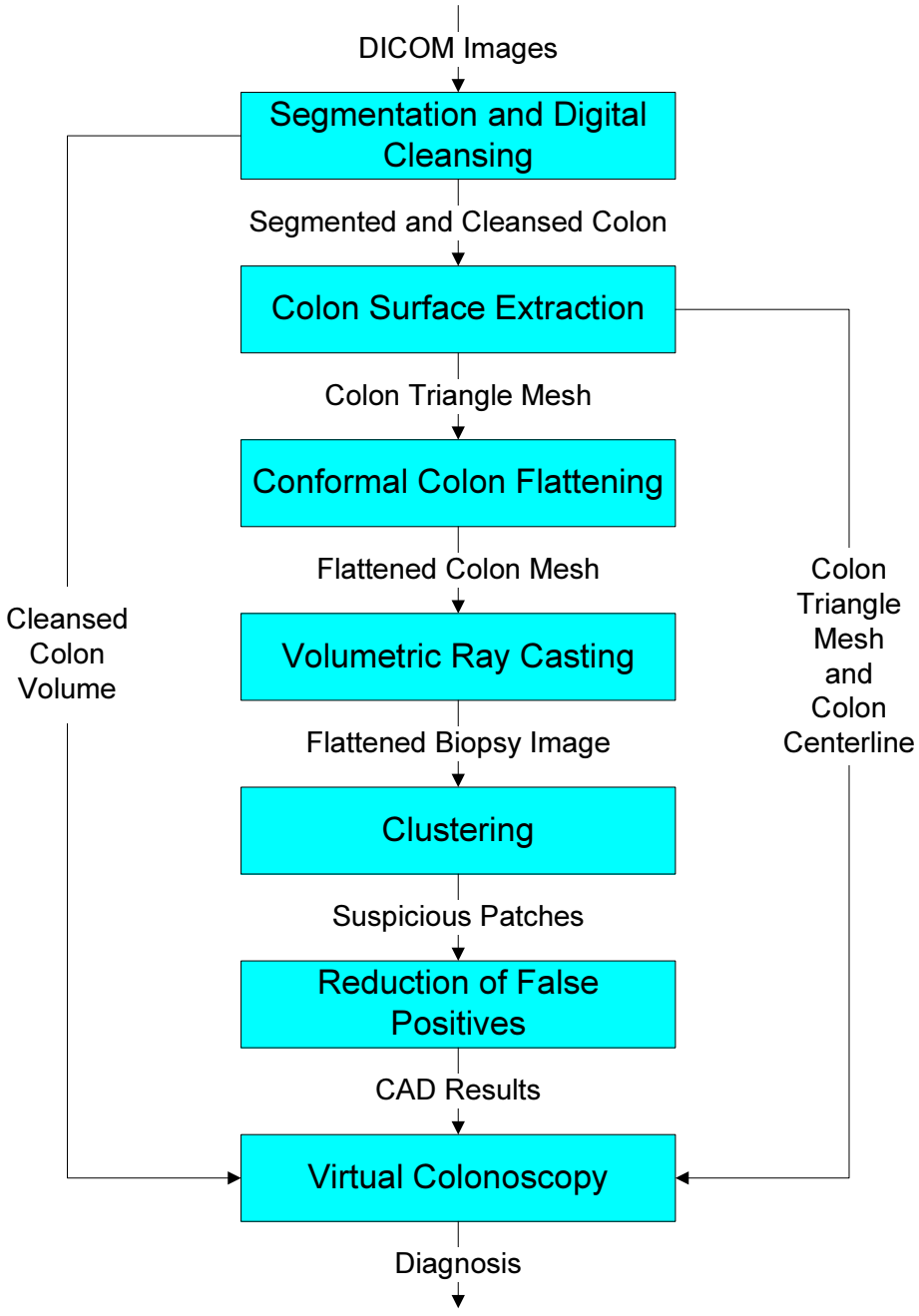


Figure 6.2: Overview of our CAD pipeline.

6.2.2 Colon Surface Extraction

After segmentation and digital cleansing, we need to extract the colon surface for our conformal virtual colon flattening algorithm. The topological noise makes our flattening

algorithm complex and introduces distortion. In Chapter 3, we have presented a volume-based method to do topological denosing based on the concept of a *simple point*. In this algorithm, non-simple points are removed from the segmented colon. Thus, all handles are removed automatically. Then, we use our enhanced dual contour method [139] to extract a simplified smooth colon surface while preserving the topology of the finest resolution colon surface.

6.2.3 Conformal Virtual Colon Flattening

In the 3D endoscopic view of the virtual colonoscopy, we can only see a small part of the colon. Moreover, our views are blocked by the colon haustral folds, so many regions will be invisible in the endoscopic view. The worst scenario is that polyps may be missed during navigation. Virtual colon flattening is an efficient visualization technique for polyp detection, in which the entire inner surface of the colon is displayed as a single 2D image. However, if two surfaces do not have the same Gaussian curvature, there does not exist a mapping which achieves both area and angle preservation. We have presented our conformal virtual colon flattening algorithm to achieve angle preserving in Chapter 4.

Instead of directly computing the conformal map between the 3D colon surface and a 2D rectangle, we compute its gradient field first. Mathematically, this gradient field is called *holomorphic 1-form*. Then, the conformal mapping can be obtained by integration. Each gradient field of a conformal map is a pair of tangential vector fields with special properties, such that the curl and Laplace are zero everywhere. All such vector fields form a linear space. We construct a basis of this linear space by solving a linear system derived from these properties. The global distortion from the colon surface to the parametric rectangle is minimized, which is measured by harmonic energy. The details of our flattening algorithm can be found in Chapter 4.

As postulated in previous CAD papers [136, 137], the colonic polyps usually have an elliptic curvature of the peak subtype, that is, the shape at the top section of a regular polyp (toward the colon wall) is more likely to be a spherical cap. Because of the angle preservation of our colon flattening algorithm, the elliptic shape of a colonic polyp is preserved in the flattened image. It is noted that our conformal colon flattening has area distortion, yet minimizes the global distortion. Consequently, polyps cannot be directly measured on the 2D flattened colon image. Since we maintain a one-to-one mapping between the 3D vertices and 2D vertices of the colon mesh, polyps can still be measured in 3D. Geometric features and texture features [45] can also be computed in 3D and mapped to 2D. This conformal mapping simplifies our polyp detection problem from 3D to 2D.

6.2.4 2D Electronic Biopsy Image Generation

The electronic biopsy technique uses a volume rendering algorithm to present the information inside the colon wall on a 2D image, the biopsy image. Each voxel is assigned a specific color and opaque values according to its CT intensity. Then, the 3D volume is

volume rendered and transformed into a 2D texture image based on our conformal mapping. This 2D texture image provides the intensity distribution information along each ray, which is hidden behind the colon surface.

In the canonical volumetric ray casting algorithm, a ray is shot for each pixel on the image plane. The direction of the ray is defined by the locations of the viewpoint and the pixel. When the ray hits the boundary of the volume, the ray starts to accumulate color and opacity values while stepping inside the volume. In our pipeline, a constrained volumetric ray casting algorithm is used to generate the 2D biopsy image. Each vertex of the mesh of the flattened colon has a 3D coordinate in the volume space. The coordinate of the first intersection point of each pixel is linearly interpolated from the three vertices of the triangle with which the ray intersects. Because flattening the colon into a 2D mesh is a nonlinear transformation, no one point can be defined as the viewpoint in the volume space for all rays. Therefore, we define the gradient at the intersection point as the direction of the ray. In our volumetric ray-casting algorithm, the sampling distance is 0.5 *mm*. Because we are only interested in a thin layer (about 20 *mm*) beneath the colon surface, each ray is only allowed to traverse up to 40 steps. Moreover, because the colon wall protrudes into the lumen, some rays may enter the colon lumen again. In order to avoid rays re-entering the colon lumen, these rays are terminated in our ray-casting algorithm using the segmentation information of the colon lumen. We can efficiently generate high resolution biopsy images accelerated on the GPU, where the thin layer beneath the colon wall is super-sampled. An electronic biopsy image is shown in Figure 6.5(a).

6.2.5 Clustering

It is observed that similar color features appear in contiguous areas in several regions of the 2D electronic biopsy image. It is reasonable to classify these features within a certain range in the 2D image. The RGB values of the given pixel and its twelve neighboring pixels form a 39-dimensional local feature vector. Consequently, a high resolution flattened electronic biopsy image is used in our CAD system, where each pixel has a 39-dimensional local feature vector. It requires intensive computational effort to manipulate such a large quantity of vectors. To reduce the computational burden, a feature analysis of the local vector series is necessary. The principal component analysis (PCA) is applied to the local vector series to determine the dimension of the feature vectors and the associated orthogonal transformation matrix (that is, the K-L transformation matrix). The PCA on the training data sets shows that a reasonable dimension of the feature vectors is 7, where the summation of the first 7 principal components variances is more than 96.5% of the total variance.

The K-L transformation matrix is applied to the local vector series belonging to hand segmented polyps on the 2D flattened electronic biopsy images. In the K-L domain, the feature vectors are formed by the first 7 principal components from the transformed vector series. The mean vector of these feature vectors is computed and used as the representative vector V of the feature vectors belonging to polyps. The square root of the variance of these feature vectors is also computed and used as a threshold T for vector similarity in the

clustering.

For a given testing data set, we use the representative vector V and similarity threshold T to classify the feature vectors in the K-L domain. If the Euclidean distance between a feature vector and V is less than T, the corresponding pixel is classified as belonging to a polyp. A 2D image is generated where the pixels classified belonging to a polyp are colored in red. The red regions in this 2D image are highly suspicious for being polyps, indicating that the physicians should observe these areas in the 3D view very carefully.

After the clustering algorithm, the pixels classified belonging to a polyp are marked. We first use a labeling algorithm to extract the connected components on this image. Since we only consider the polyps with a diameter larger than 5 *mm*, a component whose pixel count is below such a threshold is classified as a false-positive finding. Consequently, many small components are removed.

6.2.6 Reduction of False Positives

The false-positive findings can be further reduced by analyzing the shape features, such as the shape index and curvedness [137], as well as volumetric texture features [136]. The shape index is a measure of the shape. Every distinct shape, except for the plane, corresponds to a unique value of the shape index. The shape index values increase smoothly from the top section to the bottom peripheral region of a polyp on the colon wall inner surface. The curvedness represents how gently curved the surface is. Curvedness is a dual feature to the shape index in that the shape index measures which shape the local neighborhood of a voxel has, whereas the curvedness measures how much shape the neighborhood includes. The curvedness also provides scale information: a large negative value implies a very gentle change, whereas a large positive value implies a very sharp edge. In the 3D volumetric data, polyps generally appear as bulbous, cap-like structures adhering to the colonic wall, with small to medium curvedness, whereas folds appear as elongated, ridge-like structures with large curvedness. The colonic walls appear as nearly flat, cup-like structures with small curvedness. Therefore, the shape index and the curvedness can differentiate polyps from folds and colonic walls effectively.

Because of the partial volume effect, the soft-tissue density values within a polyp tend to smoothly increase from the colonic air toward the center of the polyp. Therefore, most density gradient vectors within a polyp tend to point toward the polyp center. A gradient concentration feature that characterizes the overall direction of the gradient vectors around a point is used for further reducing FPs.

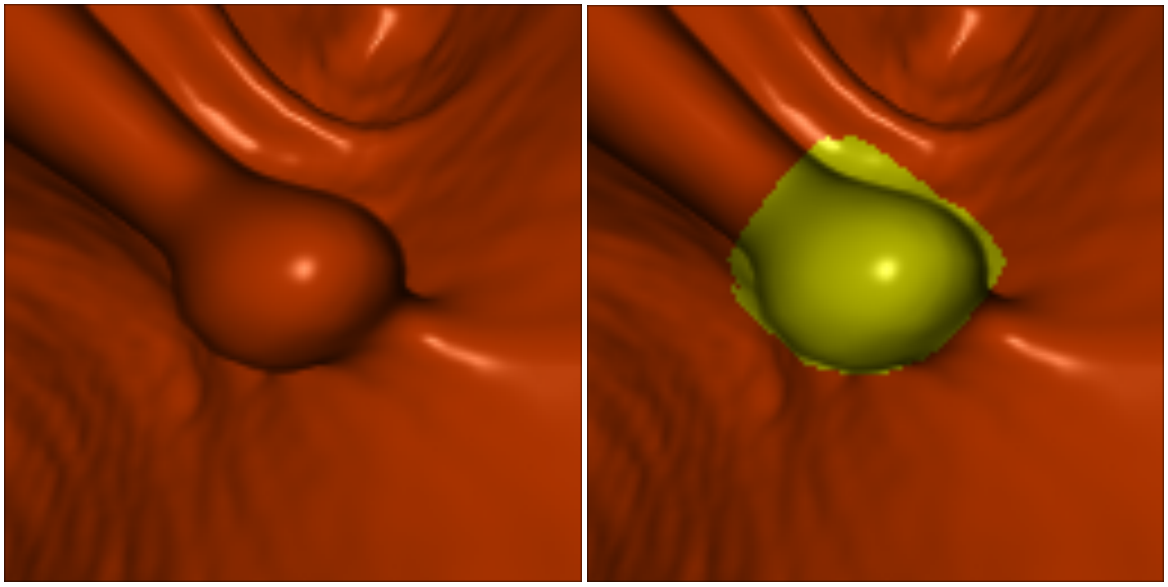
The computation of these features for the entire volume is time consuming. In our pipeline, we compute these features in a way similar to the shape based CAD methods. However, the critical difference is that we only compute these features on several suspicious areas for FP reduction, rather than for the entire colon.

6.3 Integration with Virtual Colonoscopy

The polyp detection results of our CAD pipeline are also stored with the flattened colon image, which can be used for highlighting the corresponding VC endoscopic view. The colon mesh extracted in our pipeline can also be used to accelerate the direct volume rendering of the VC endoscopic view.

6.3.1 Polygonal Assisted Volume Rendering

When navigating or flying through the colon lumen, the colon wall is rendered with direct volume rendering. Because of the large size of the colon volume data and the inherent complexity of volume rendering, it is very hard to achieve interactive frame rates with a software implementation. 3D texture-based volume rendering [14] is a popular volume rendering method that can achieve real-time speed on commodity graphics hardware (GPU). However, the rays shot from the image plane have different sampling rates due to the planar proxy geometry. Ray casting has been implemented on the GPU, which has a coherent sampling rate for all rays [74]. They have achieved interactive speed by using the two common acceleration techniques, empty-space skipping and early ray termination.



(a) (b)

Figure 6.3: A close up view of a polyp rendered with volumetric ray casting (a) without coloring, and (b) with coloring.

The polygonal mesh has been exploited by us to accelerate direct volume rendering [5]. The polygonal mesh representing the object boundary is extracted from the volume in the second step of our pipeline. Each vertex is associated with its coordinates in volume texture space. The mesh is projected onto the image plane for calculating the entry points

of rays, and the empty space between the image plane and the object boundary is skipped. Our detection result is also stored in a 2D image, in which we use yellow color for polyps and red color for normal colon wall. When the flattened mesh is projected on the image, we also obtain a 2D texture coordinates by interpolation. We use this 2D texture coordinates to access the resulting image to determine the color of the ray. This method is very efficient because the GPU is very efficient in rasterizing triangles onto the image plane.

Our algorithm has two passes. In the first pass, the mesh is rendered and the rasterization hardware interpolates the texture coordinates for each fragment. In this pass, the depth test is enabled so that only the nearest intersection points are preserved in the framebuffer. In the second pass, the fragment shader reads back the intersection point for each pixel on the image plane and a standard ray casting is performed from this point. A polyp rendered with our method with and without coloring is shown in Figure 6.3. The rendering frame rates is 17-20 per second for a 512×512 image.

6.3.2 Enhanced Virtual Colonoscopy

Our interactive user interface shown in Figure 6.4 provides multiple views of the colon CT data. In the center is the 3D volume rendered endoscopic view. A flattened colon image is shown on the right of the endoscopic view. Bookmarks for suspicious regions can be stored on the flattened colon image. A zoom-in view to display the corresponding part of the flattened colon image at the current camera position is provided under the endoscopic view. The 2D mutually perpendicular slice views oriented axial, coronal, and sagittal are shown on the left hand side. An outside overview of the patient's colon and z zoom-in slice view are shown on the right hand side. All these 2D and 3D images are correlated and interlinked so that position in 3D is overlaid on the 2D images and the position of 2D slices and the flattened colon image can be overlaid on the 3D images. This provides a quick and simple mechanism to easily analyze suspicious patches in both 2D and 3D. It is noted that the correlation between the flattened zoom-in view and the 3D endoscopic view is established by our conformal colon flattening algorithm.

In our enhanced VC system, we provide a new mode to allow the physician to go over solely the flattened colon image for polyp screening. In this mode, the physician only need to fly over the flattened image from one end to the other. The physician only need to go over the flattened colon image in one direction through the zoom-in view, because our conformal colon flattening method guarantee 100% surface visibility coverage. Since the 3D endoscopic view is correlated with the zoom-in view of the flattened colon image, if any suspicious polyp is found on the zoom-in view of the flattened colon, the physician can double click the suspicious polyp to update the 3D endoscopic view and confirm it in the 3D endoscopic view. In this way, the physician mainly focus on the zoom-in view of the flattened colon image, which can also reduce reader fatigue.

Although our conformal colon flattening method does not preserve area information, measurement can still be performed on the flattened colon image by mapping points back to the 3D world coordinate system and computing the diameter of the polyp in the 3D

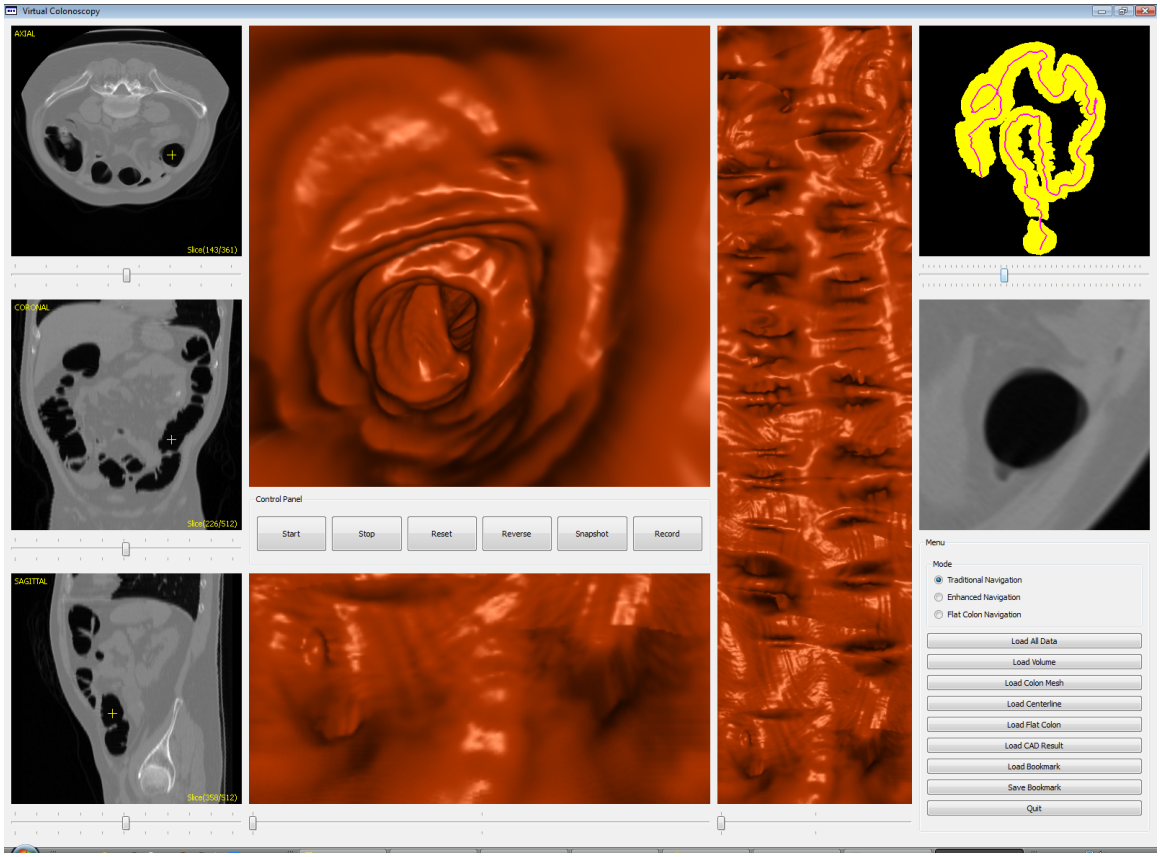


Figure 6.4: The user interface of our CAD system.

world coordinate system. It is much more convenient for the physician to perform the measurement on the 2D flattened colon image than that on the 3D endoscopic image.

We have integrated the detection result of our CAD pipeline into our VC system. Our new VC system is enhanced in the following ways:

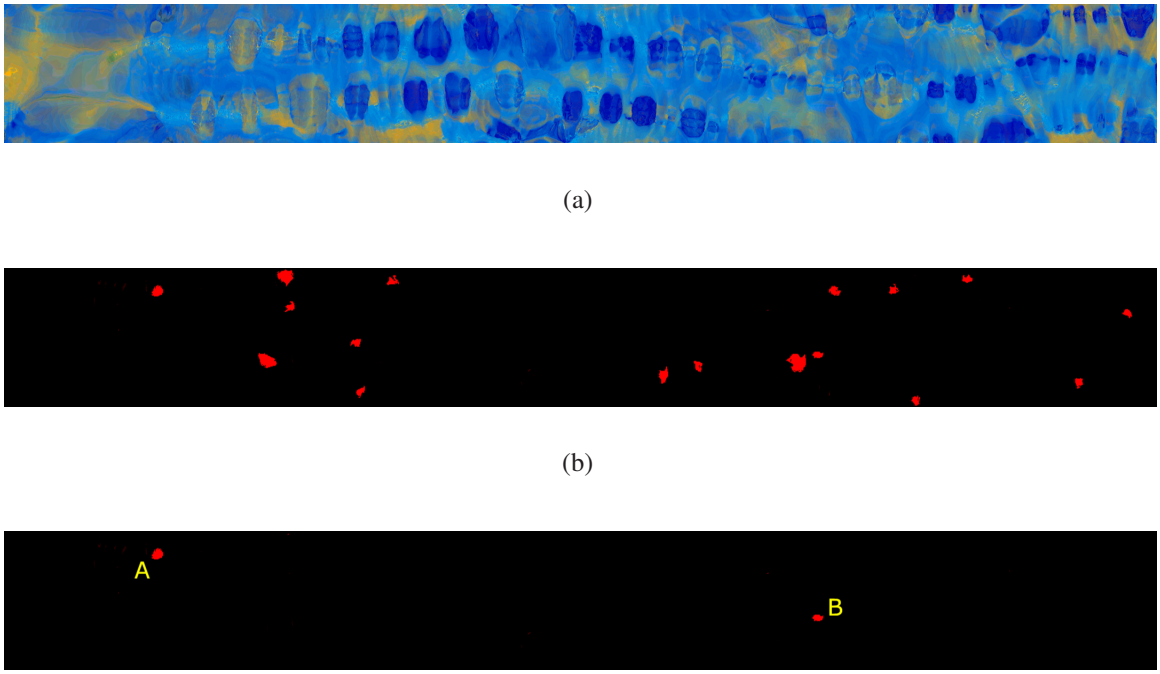
1. In the navigation mode, the suspicious patches are highlighted in the endoscopic view to attract the attention of the physicians during navigation. Since our detection algorithm is 100% sensitive to polyps, the missed polyps in the conventional VC system will not be missed in our system.

2. All suspicious polyp candidates are also highlighted on the 2D flattened colon view. Physicians can directly inspect these suspicious regions by clicking on them. All other views are updated simultaneously.

3. Bookmarks for suspicious regions are stored on the flattened colon image and on the colon overview image. From either image, the physicians can sequentially or randomly go through all bookmarks of suspicious regions, which are automatically provided by our CAD pipeline.

Our initial feedback from a physician using our prototype system has been very positive, where the CAD results serving as a second reader guarantee a low miss examination and that the user interface features indeed enhance the VC system.

6.4 Results



(a)

(b)

(c)

Figure 6.5: (a) The electronic biopsy image generated using our conformal colon flattening and volumetric ray casting algorithm. (b) The result of our clustering algorithm. (c) The result of the reduction of FPs with shape analysis and 3D texture analysis. Two polyp candidates are obtained using this data set, the real polyp at location A and a FP at location B.

We implemented our polyp detection pipeline in C/C++ and ran all of the experiments on a 3.6 GHz Pentium IV PC running Windows XP with 3G RAM and one NVIDIA Quadro 4500 graphics board. We have been collaborating closely with physicians and gastroenterologists in developing and evaluating our methods. Our pipeline was tested using a total of 198 CT data sets. We used 152 CT data sets from the National Institute of Health (NIH) to demonstrate and test our CAD pipeline. Along with the raw DICOM images, there are VC reports, OC reports, pathology reports, and OC videos. In addition, we used another 46 CT data sets along with VC reports and OC reports obtained from Stony Brook University

Hospital (SBUH) to test and demonstrate our pipeline. We used the specialists' VC and OC reports for the NIH and SBUH data sets to evaluate our CAD pipeline.

Twenty data sets from the NIH were used in the training of our pipeline, to compute the K-L matrix and the representative vector V. The rest of the data sets were used to test our CAD pipeline, which generated consistent results and is 100% sensitive to polyps. No polyp in the 132 NIH data sets or the 46 SBUH data sets used for testing was missed by our system. The polyps are colored using our volumetric ray casting algorithm with a translucent biopsy transfer function. All the polyps are shown in similar colors on the 2D image, which will not be missed by our clustering algorithm.

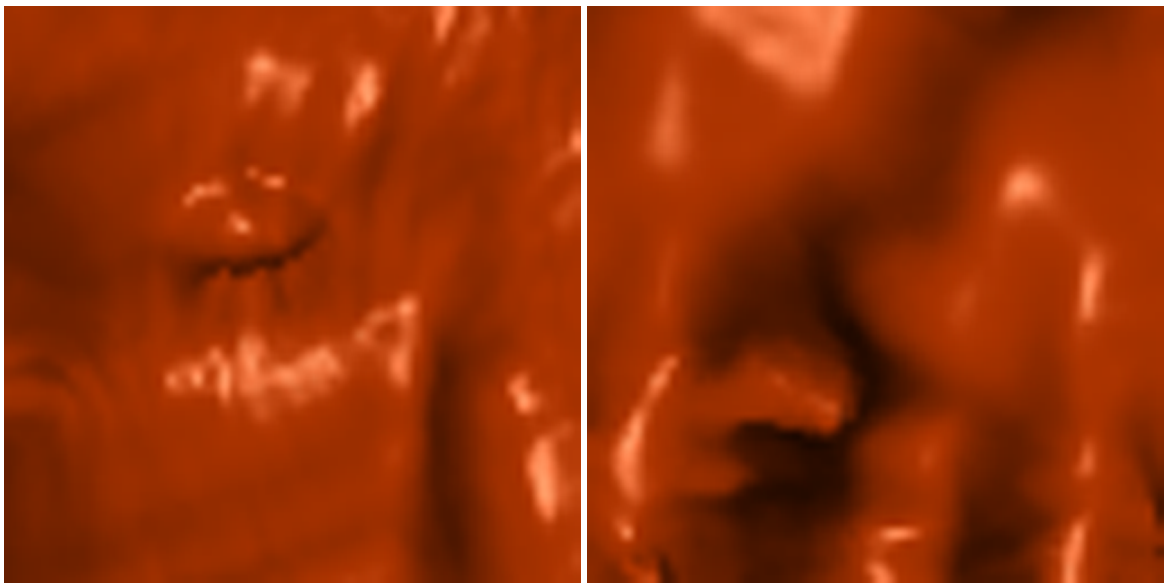
Table 6.1: Experimental results of our CAD pipeline.

Data Source	Total Polyps	FP per Data Set	FP Reduction
SBUH	65	2.9	97.1%
NIH	82	3.5	96.1%

The experimental results of testing our pipeline are depicted in Table 6.1, which are confirmed using VC reports and OC reports. In addition to detecting all polyps, our pipeline also significantly reduced the number of FPs for each data set. The 132 NIH data sets used for testing contain 82 polyps, all of which were identified by our pipeline. An average of 3.5 FPs was identified in each data set, after FP reduction. The FP reduction step removed 96.1% of the FPs. The 46 SBUH data sets contained 65 polyps, all of which were identified by our pipeline. An average of 2.9 FPs was identified in each data set, after FP reduction. The FP reduction step removed 97.1% of the FPs. The best shape analysis based systems [36, 122, 130, 137] achieved $2-3$ FPs per dataset with 100% sensitivity. Our experiment results show that our method achieved similar results as these systems.

One of the SBUH CT data sets of size $512 \times 512 \times 460$ has a polyp near the rectum. The resolution of the flattened electronic biopsy image is 4000×200, which is shown in Figure 6.5(a). The rectum is at the left end of Figure 6.5(a). There is a polyp of 8 *mm* diameter near the rectum of this colon data set. The suspicious polyp candidates from our clustering algorithm are shown in red in Figure 6.5(b). The FPs are reduced by shape analysis and 3D texture analysis applied at these suspicious areas. As a result, we obtain 2 polyp candidates, the real polyp at location A and a FP at location B, both shown in red in Figure 6.5(c). The corresponding 3D VC views of these two locations are shown in Figures 6.6(a) and 6.6(b), respectively. It is noted that the FP B is resulted from the protuberance on the colon haustral fold.

Compared with shape-based CAD methods, our system is much faster. Our topological denoising algorithm and colon surface extraction algorithm costs less than 1 minute. The most time consuming step of our pipeline is the conformal colon flattening, which takes about 7 minutes. The electronic biopsy image rendered with a resolution of 4000×200 costs only about 300 milliseconds accelerated on the GPU. Therefore, it takes our pipeline about 8 minutes to gather features for polyp detection. In the shape-based CAD methods,



(a) (b)

Figure 6.6: (a) The 3D view of the detected polyp A. (b) The 3D view of the false-positive finding B on a colon fold.

the computation step of shape index and curvedness is the most time consuming step. It took us about 20 minutes to compute shape index and curvedness with a mask size of $5 \times 5 \times 5$ for the entire mucosa layer, using the same data set on the same platform as our approach. The computation time is about one hour when the mask size is $7 \times 7 \times 7$. We achieved an average of 3 FPs per data set with 100% sensitivity, which is equal or better than the shape based methods. Our detection results are stored in 2D flattened images, which are much easier to integrate into the VC system than that of other CAD systems. Our pipeline provides a flattened colon view in the user interface of the VC system, which is much more friendly than the other systems.

6.5 Conclusions

The pipeline we have presented here is a novel method for the CAD of colonic polyps. Unlike previous shape-based method, our method uses a 2D volume rendered flattened biopsy image of the colon to detect suspicious patches by a clustering method. This is due to the fact that the adenomatous and malignant polyps in the volume rendered biopsy images have different densities compared with normal tissues. The FPs are further reduced in a subsequent step by performing shape analysis and 3D texture analysis only on these patches, not on the entire endoluminal colon surface. Our system detects 100% of the adenomatous polyps, and yields a low FP rate in only several minutes. The results are easily integrated into any VC system, which allows physicians to perform their diagnoses

more accurately and efficiently. Since the suspicious areas are clearly identified to the user, the physician needs only traverse the colon in one direction, without fear of missing a polyp.

Chapter 7

Conclusions

7.1 Summary

In this dissertation, we have presented our solution to one of the attractive and challenging research topics in the visualization and medical imaging communities. Our conformal colon flattening algorithm can display the entire inner colon surface as a single 2D image, and our CAD pipeline is able to identify 100% of the polyps with a low FP rate, which makes our VC system much more powerful and efficient. The primary research contributions of this dissertation are described as follows:

- A simulation method to estimate the percentage of the colon surface is missed in the standard OC examination. An Olympus colonoscope, a wide angle fisheye camera, is calibrated and simulated in our method. The simulated fisheye camera is moving along the hugging corner shortest path, rather than the centerline of the colon as for VC fly-through navigation. Our simulation study reveals that about 23% of the colon surface is missed in the standard OC examination and about 9% of the colon surface is missed in the VC examination when navigating in both antegrade and retrograde directions.

- A fully automatic segmentation and digital cleansing framework with the capability to handle the partial volume effect and topological noise. The mixture information is estimated for each voxel using the well-developed EM algorithm in an iterative manner. It allows us to accurately segment the colon lumen and restore the CT density values of the tagged materials for digital cleansing. The topological noise is automatically removed using a region growing based method. A topologically simple colon surface can be extracted from the segmented colon and simplified for visualization as well as virtual colon flattening.

- A conformal colon flattening algorithm based on Riemann surface theory and differential geometry. Our algorithm is general, which can be applied to arbitrarily high genus surface. The global distortion from the colon surface to the parametric rectangle is minimized, which is measured by the harmonic energy. We have proved and

shown that our algorithm is angle preserving (local shape preserving). The shape of colonic polyps on the flattened colon image is well preserved. Even a small polyp in the high resolution flattened colon image and can be easily identified by a physician. The flattened colon image has been integrated into our VC system to enhance the user interface and accuracy of the VC system.

- A GPU-based object-order ray-casting algorithm for rendering large volumetric data sets. The volumetric data set is decomposed into small cells, and organized using a min-max octree structure. The empty cells are skipped immediately after the classification. The volumetric ray-casting algorithm is performed on the GPU for each non-empty cell. The cooperation and trade-off between the CPU and the GPU are exploited in our hybrid volumetric ray-casting method to obtain further acceleration. Our algorithm allows large volumetric data set to be rendered for virtual endoscopy applications in real-time with high quality.

- A novel CAD pipeline for polyp detection integrating texture and shape analysis with volume rendering as well as conformal colon flattening. Polyps are identified by a clustering method on the 2D electronic biopsy images. The false positive findings are further reduced by shape analysis. The polyp detection results can be seamlessly incorporated into the VC system to highlight the suspicious regions during the fly-through navigation. The CAD enhanced VC system can reduce physicians' perceptual errors, thereby improving the accuracy of VC.

7.2 Near Future Work

7.2.1 General Volume Processing Framework

Compute Unified Device Architecture (CUDA), a technique developed by NVIDIA, is a new hardware and software architecture for issuing and managing computations on the GPU as a data-parallel computing device without the need of mapping them to a graphics API. CUDA features a parallel data cache or on-chip shared memory with very fast general read and write access, that threads use to share data with each other. When programmed through CUDA, the GPU is viewed as a compute device capable of executing a very high number of threads in parallel. It operates as a coprocessor to the main CPU, or host: In other words, data-parallel, compute-intensive portions of applications running on the host are off-loaded onto the device. More precisely, a portion of an application that is executed many times, but independently on different data, can be isolated into a function that is executed on the device as many different threads. To that effect, such a function is compiled to the instruction set of the device and the resulting program, called a *kernel*, is downloaded to the device.

The mixture information of each voxel is estimated using the EM algorithm in an iterative manner, which is the most time-consuming part in our partial volume segmentation

algorithm. The most time-consuming part of our conformal flattening algorithm is the minimization of the harmonic energy using the conjugate gradient method. Applications such as the EM algorithm and the conjugate gradient algorithm that require mathematically intensive computing on large amounts of data are ideal targets for GPU computing with CUDA. We would like to implement our partial volume segmentation algorithm and conformal surface flattening algorithm using CUDA to further improve the overall performance of our system.

Furthermore, we would also like to design a general framework for 3D medical image processing using CUDA. In this framework, currently time-consuming algorithms, such as volume filtering algorithms and 3D level set methods, can be implemented easily and efficiently as kernel programs.

7.2.2 Volume Rendering for Very Large Data Sets

The newly released NVIDIA graphics hardware also provides the capability of *render to 3D texture*. It has the great potential to improve the state-of-the-art rendering and simulation in computer graphics and visualization, which would benefit researchers and end-users in a variety of applications. Due to the large size of the volume data, the gradient is estimated on-the-fly on the GPU in our current system, which needs six texture lookups for each sampling point and is not efficient. The rendering performance and quality of our GPU-based volumetric ray-casting algorithm can be further improved, if we can compute the gradient information for the whole cell on-the-fly using the *render to 3D texture* feature and use tri-linear interpolation to obtain the gradient for each sampling point.

7.2.3 Unified Colon Flattening

If two surfaces do not have the same gaussian curvature, there is no way to achieve both area and angle preservation for surface mapping. Therefore, in our conformal colon flattening method, only angle preservation is achieved. The shape of colonic polyps on the 2D flattened colon image can be identified by physicians. However, our method suffers from the area distortion, that is, physicians cannot measure the size of the polyp directly. Therefore, how to alleviate the area distortion in our harmonic energy minimization process is another near future work. We would like to investigate a unified colon flattening algorithm to minimize the overall distortion: angle distortion and area distortion.

7.2.4 Conformal Volumetric Colon Flattening

In our current conformal colon flattening algorithm, only the colon surface is mapped to a 2D rectangle. We would like to flatten the colon surface as well as the soft tissues. In other words, we would like to obtain a volume containing the flattened colon wall along with soft tissues. The main idea is that we shrink and expand the colon surface along the gradient direction to obtain two surfaces. Then, we tetrahedralize the space between the

two surfaces. We can compute a harmonic function on this domain by constraints that the inner surface has a value zero and the outer surface has a value one. Then, we can extract a set of isosurfaces at different values. We can obtain volumetric parameterizations by conformally mapping them to a set of coaxial cylinder surfaces. The flattened volume can be obtained by unfolding the cylinder. Furthermore, CAD algorithms can be applied to the flattened volume to improve their computation performance.

7.2.5 Automatic Transfer Function Generation for Polyp Detection

How to define a transfer function in order to generate meaningful results is a common problem in volume rendering [104]. In our current implementation for the electronic biopsy [125], the semi-translucent transfer function is not automatically generated. We would like to study the features of colonic polyps to devise a method to generate the transfer function automatically. This transfer function will be multi-dimensional and use various features such as density values, curvature, and eigenvalues of Hessian matrix [66]. In fact, the features used in the shape-based CAD algorithms can be considered for the design of the transfer function used for electronic biopsy. This can further improve the performance of our VC and CAD system.

7.2.6 Supine and Prone Registration

Supine and prone registration allows the user to easily correlate between the supine and prone scan. This technique empowers the user to switch at any time between the supine and prone scan enabling a faster verification of findings or parts that might be collapsed or hidden by residual stool and fluid in one of the two scans.

Supine and prone registration is difficult to be done directly in 3D due to the deformation of colon. In previous methods [2, 120], supine and prone is registered based on the centerline, which simplifies the 3D registration problem to 1D. However, the registration results based on the centerline cannot be used to correlate 3D endoscopic views. We would like to register supine and prone data sets using our flattened feature colon images, which supposedly provides better results than the method based on the colon centerline. If we can establish a one-to-one mapping between two flattened colon images, the one-to-one mapping between two 3D colon surfaces can also be computed. Results from this registration method could be used to double check for polyp detection. This registration method can also be used to align two scans taken at different times.

7.2.7 Image-based Path Planning

In the VC fly-through navigation, it is crucial to generate an optimal camera path for efficient colonic polyp screening. Although it is useful for describing the shape of an object, the centerline is not always the optimal camera path for observing the object. Hence, conventional methods in which the centerline is directly used as a path produce considerable

blind areas, especially in areas of high curvature. Many automatic path planning algorithms [49, 63] have been developed to improve visibility coverage. For comfortable user navigation, the camera path is usually smoothed and the amount of rotation between consecutive endoscopic views needs to be minimized. However, all these algorithms require some time consuming pre-processings such as colon lumen segmentation and distance transformation computation [89] before the fly-through navigation.

We would like to design an image-based path planning automatic navigation algorithm without the requirement of performing any pre-processing. Therefore, the VC fly-through navigation can be performed immediately after the colon data set is loaded from the computer hard drive. We also would like to integrate the CAD results into the path planning algorithm, which can guarantee that no suspicious region is missed during the fly-through navigation.

7.3 Long-term Future Work

In this dissertation, we focus on CT colon data sets for VC applications, although our techniques are general. Our techniques can be used with a variety of human organs, such as blood vessels and bladder, which includes topics like:

- Virtual angioscopy [35] is primarily used for detecting stenoses and calcifications in blood vessels. Virtual angioscopy can aid in the characterization of broad-based aneurysms, which can help to determine whether surgical treatment is preferable to coil embolization. With many blood vessels being too narrow for a normal endoscope, virtual angioscopy is in many cases the only alternative. Our techniques can be applied to blood vessels to enhance the virtual angioscopy applications. However, blood vessels cannot be mapped to a 2D rectangle, due to their complex topology. A new method to visualize the flattened blood vessels need to be designed. Moreover, the corresponding user interface for virtual angioscopy also need to be investigated.

- Virtual cystoscopy [39] is a promising new technique based on the rendering of the inner surface of the urinary bladder using volumetric MRI data sets, thus enabling maneuvers that normally are not feasible with conventional cystoscopy. Therefore, this method could potentially provide a means for the screening and surveillance of bladder tumors, which tend to recur. Our conformal flattening technique can be directly applied to the inner surface of the bladder to improve the performance of virtual cystoscopy. However, the partial volume effect must be specially taken care of during the segmentation of the bladder.

An abdominal Aortic Aneurysm (AAA) [3] is a bulge in the wall of an artery. It is estimated that 1.5 million Americans have AAA, though only approximately $200,000$ are diagnosed each year. AAA's are almost always caused due to arteriosclerosis. As plaque accumulates, the pressure of the blood blowing through the weakened section of the artery

causes the artery to balloon, forming an aneurysm.If the aneurysm is not detected in time, the weakened aorta will rupture, often causing death.

It has become standard practice to treat AAA through minimally invasive surgery. The procedure consists of placing a catheter into the iliac artery, inserted up to the kidney junction. A stent is then extracted from the catheter to protect the aneurysm from a rupture. The entire operation is generally performed with the aid of an intraoperative X-ray scanner, sometimes combined with an ultrasound probe. There are two primary problems with this approach. First, selecting a properly fitting prosthesis for each patient is difficult. Second, placing the stent quickly and accurately is hard to accomplish. Virtual AAA may aid surgeons in both diagnosis and treatment. A virtual AAA system should provide the following functions:

- automatically extract the aorta, iliac arteries and aneurysm in order to build a 3D model.
- enable surgeons to examine possible prosthesis locations and catheter trajectories.
- offer advanced measurement tools to assist evaluation of the surgical area. Our conformal flattening technique has the potential to be applied to aorta to provide efficient measurement tools.
- automatically suggest a set of candidate prostheses and allow the surgeon to validates his choice by simulating the prosthesis contact.
- noninvasively assess the wall stresses acting in individual aneurysms based on the patient's blood pressure and the 3D model.

The success of a surgical procedure is in direct relation to experience and intuition of the surgeon. Visualization tools such as computer-aided planning of operations, surgery simulation for training, and intra-operative surgery assistance are mostly still at an experimental level and not yet well established in daily clinical routine. Many tasks connected with the development of such tools can be accomplished by applying state of the art 3D visualization techniques to high quality radiological data sets. Nevertheless, there exists still a number of challenging open problems:

- Advanced visualization techniques have to be combined with high-level physics-based simulation to get realistic images for surgery simulation. Moreover, the performance of physics-based simulation should be improved.
- Quantization, manipulation, simulation and fast visualization very often requires a geometric reconstruction of volumetric structures.
- Augmented reality allows the combination of intra-operative with pre-operative data and enables the surgeon to realize a pre-operative plan exactly. The combination of virtual reality, physics based simulation and the use of haptic feedback devices open new methods for realistic surgery training.

- Speed and accuracy of visualization, simulation and tracking play a crucial role in having the necessary interactivity for surgery training and intra-operative use of the techniques in mind. A highly optimized software design and intelligent algorithms in combination with the expected development of the hardware will lead to more and more realistic simulations and visualizations.

Bibliography

[1] B. Acar, C. Beaulieu, S. Gokturk, C. Tomasi, D. Paik, R. B. Jeffrey, J. Yee, and S. Napel. Edge displacement field-based classification for improved detection of polyps in CT colonography. *IEEE Transactions on Medical Imaging*, 21:1461–1467, 2002.

[2] B. Acar, S. Napel, D. Paik, P. Li, J. Yee, R. Jeffrey, and C. Beaulieu. Medial axis registration of supine and prone ct colonography data. *IEEE Engineering in Medicine and Biology Society*, 32:2433–2436, 2001.

[3] G. Ailawadi, J. Eliason, and G. Upchurch. Current concepts in the pathogenesis of abdominal aortic aneurysm. *Journal of Vascular Surgery*, 38(3):584–588, 2003.

[4] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. *EUROGRAPHICS*, pages 3–9, 1987.

[5] R. Avila, L. Sobierajski, and A. Kaufman. Towards a comprehensive volume visualization system. *IEEE Visualization*, pages 13–20, 1992.

[6] E. Balogh, E. Sorantin, L. G. Nyul, K. Palagyi, A. Kuba, G. Werkgartner, and E. Spuller. Virtual dissection of the colon: technique and first experiments with artificial and cadaveric phantoms. *Proceedings SPIE*, 4681:713–721, 2002.

[7] A. Bartrolí, R. Wegenkittl, A. König, and E. Gröller. Nonlinear virtual colon unfolding. *IEEE Visualization*, pages 411–418, 2001.

[8] A. Bartrolí, R. Wegenkittl, A. König, E. Gröller, and E. Sorantin. Virtual colon flattening. *VisSym Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 127–136, 2001.

[9] D. Bartz. Virtual endoscopy in research and clinical practice. *Computer Graphics Forum*, 24(1):111–126, 2005.

[10] C. Beaulieu, R. Jeffrey, D. P. C. Karadi, and S. Napel. Display modes for CT colonography part ii. blinded comparison of axial CT and virtual endoscopic and panoramic endoscopic volume-rendered studies. *Radiology*, 212:202–212, 1999.

[11] G. Bertrand. Simple points, topological numbers and geodesic neighborhoods in cubic grids. *Pattern Recognition Letters*, 15:1003–1011, 1994.

[12] I. Bitter, A. Kaufman, and M. Sato. Penalized-distance volumetric skeleton algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):195–206, 2001.

[13] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder. Sparse matrix solvers on the gpu: Conjugate gradients and multigrid. *ACM Transactions on Graphics*, 22(3):917–924, 2003.

[14] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. *Symposium on Volume Visualization*, pages 91–98, 1994.

[15] W. Cai, J. Näppi, M. E. Zalis, G. J. Harris, and H. Yoshida. Digital bowel cleansing for computer-aided detection of polyps in fecal tagging ct colonography. *SPIE Medical Imaing*, 6144:1–9, 2006.

[16] D. Chen, Z. Liang, M. R. Wax, L. Li, B. Li, and A. Kaufman. A novel approach to extract colon lumen from CT images for virtual colonoscopy. *IEEE Transactions on Medical Imaging*, 19(12):1220–1226, 2000.

[17] D. Cohen and Z. Sheffer. Proximity clouds: An acceleration technique for 3D grid traversal. *The Visual Computer*, 11:27–38, 1994.

[18] S. Dave, G. Wang, B. Brown, E. McFarland, Z. Zhang, and M. Vannier. Straighening the colon with curved cross section: an approach to ct colonography. *Academic Radiology*, pages 398–410, 1999.

[19] É. C. de Verdière and F. Lazarus. Optimal system of loops on an orientable surface. *Discrete and Computational Geometry*, 33(3):507–534, 2005.

[20] T. Deschamps and L. Cohen. Fast extraction of minimal paths in 3D images and applications to virtual endoscopy. *Medical Image Analysis*, 5:281–299, 2001.

[21] T. K. Dey and H. Schipper. A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection. *Discrete and Computational Geometry*, 14:93–110, 1995.

[22] E. Dijkstra. A note on two problems in connection with graphs. *Nuerische Mathematik*, 1:269–271, 1959.

[23] R. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *ACM SIGGRAPH*, pages 65–74, 1988.

[24] D. S. Ebert, C. J. Morris, P. Rheingans, and T. S. Yoo. Designing effective transfer functions for volume rendering from photographic volumes. *IEEE Transactions on Visualization and Computer graphics*, 8:183–197, Apr. 2002.

[25] J. EI-Sana and A. Varshney. Controlled simplification of genus for polygonal models. *IEEE Visualization*, pages 403–412, 1997.

[26] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-Time Volume Graphics*. A K Peters, Ltd., 2006.

[27] K. Engel, M. Kraus, and T. Ertl. High-quality volume using hardware-accelerated pixel shading. *Graphics Hardware*, pages 9–16, 2001.

[28] D. Eremina, X. Li, W. Zhu, J. Wang, and Z. Liang. Investigation on an EM framework for partial volume image segmentation. *SPIE Medical Imaging*, 6144:1398–1406, 2006.

[29] J. Erickson and S. Har-Peled. Optimally cutting a surface into a disk. *ACM Symposium on Computational Geometry*, pages 215–228, 2003.

[30] J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. *ACM-SIAM Symposium on Discrete Algorithms*, pages 1038–1046, 2005.

[31] J. Fletcher, C. Johnson, R. MacCarty, T. Welch, J. Reed, and A. Hara. Ct colonography: potential pitfalls and problem-solving techniques. *American Journal of Roentgenology*, 172:1271–1278, 1999.

[32] J. Fletcher, C. Johnson, J. Reed, and J. Garry. Feasibility of planar virtual pathology: a new paradigm in volume rendered ct colonography. *Journal of Computer Assisted Tomography*, pages 864–869, 2001.

[33] M. Franaszek, R. M. Summers, P. J. Pickhardt, and J. R. Choi. Hybrid segmentation of colon filled with air and opacified fluid for CT colonography. *IEEE Transactions on Medical Imaging*, 25(3):358–368, 2006.

[34] A. Ghosh, P. Prabhu, A. Kaufman, and K. Mueller. Hardware assisted multichannel volume rendering. *Proceedings of the Computer Graphics International Conference*, pages 2–7, July 2003.

[35] E. Gobbetti, P. Pili, A. Zorcolo, and M. Tuveri. Interactive virtual angioscopy. *IEEE Visualization*, pages 435–438, 1998.

[36] S. B. Göktürk, C. Tomasi, B. Acar, C. F. Beaulieu, D. S. Paik, R. B. J. Jr., J. Yee, and S. Napel. A statistical 3D pattern processing method for computer aided detection of polyps in CT colonography. *IEEE Trans. Med. Imaging*, 20(12):1251–1260, 2001.

[37] S. Grimm, S. Bruckner, A. Kanitsar, and E. Groller. Memory efficient acceleration structures and techniques for CPU-based volume raycasting of large data. *IEEE Symposium on Volume Visualization and Graphics*, pages 1–8, Oct. 2004.

[38] X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. *ACM Transactions on Graphics*, 21(3):355–361, 2002.

[39] G. Gualdi, E. Casciani, M. Rojas, and E. Polettini. Virtual cystoscopy of bladder neoplasms: preliminary experience. *Radiol Med*, 97:506–509, 1999.

[40] I. Guskov and Z. Wood. Topological noise removal. *Graphics Interface*, pages 19–26, 2001.

[41] S. Guthe and W. Strasser. Real-time decompression and visualization of animated volume data. *IEEE Visualization*, pages 349–356, 2001.

[42] S. Guthe, M. Wand, J. Gonser, and W. Strasser. Interactive rendering of large volume data sets. *IEEE Visualization*, pages 53–60, 2002.

[43] S. Haker, S. Angenent, A. Tannenbaum, and R. Kikinis. Nondistorting flattening maps and the 3D visualization of colon CT images. *IEEE Transactions on Medical Imaging*, 19:665–670, Dec. 2000.

[44] X. Han, C. Xu, and J. L. Prince. A topology preserving level set method for geometric deformable models. *IEEE Transactions on PAMI*, 25(6):755–768, 2003.

[45] R. Haralick, K. Shanmugam, and I. Dinstien. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 6:610–621, 1973.

[46] M. Hassouna and A. Farag. Robust centerline extraction framework using level sets. *IEEE Computer Vision and Pattern Recognition*, pages 458–465, 2005.

[47] J. Hladuvka, A. König, and E. Gröller. Curvature-based transfer functions for direct volume rendering. *Spring Conference on Computer Graphics*, pages 58–65, 2000.

[48] L. Hong, A. Kaufman, Y. Cai, A. Viswambharan, M. Wax, and Z. Liang. 3D virtual colonoscopy. *Biomedical Visualization*, pages 26–33, 1995.

[49] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He. Virtual voyage: Interactive navigation in the human colon. *ACM SIGGRAPH*, pages 27–34, Aug. 1997.

[50] W. Hong, X. Gu, F. Qiu, M. Jin, and A. Kaufman. Conformal virtual colon flattening. *ACM Symposium on Solid and Physical Modeling*, pages 85–94, 2006.

[51] W. Hong, X. Gu, F. Qiu, and A. Kaufman. Conformal colon flattening for virtual colonoscopy. *Submitted for publication*, 2007.

[52] W. Hong and A. Kaufman. Feature preserved volume simplification. *ACM Symposium on Solid Modeling and Applications*, pages 334–339, 2003.

[53] W. Hong, F. Qiu, and A. Kaufman. GPU-based object-order ray-casting for large data sets. *International Workshop on Volume Graphics*, pages 177–186, 2005.

[54] W. Hong, F. Qiu, and A. Kaufman. A pipeline for computer aided polyp detection. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):861–868, 2006.

[55] W. Hong, F. Qiu, and A. Kaufman. Hybrid volumetric ray-casting. *Submitted for publication*, 2007.

[56] W. Hong, F. Qiu, J. Marino, and A. Kaufman. Computer-aided detection of colonic polyps using volume rendering. *SPIE Medical Imaging*, 6514:1–8, 2007.

[57] W. Hong, J. Wang, F. Qiu, A. Kaufman, and J. Anderson. Colonoscopy simulation. *SPIE Medical Imaging*, 6511:1–8, 2007.

[58] H. Hoppe, C. Quattropani, A. Spreng, J. Mattich, P. Netzer, and H.-P. Dinkel. Virtual colon dissection with CT colonography compared with axial interpretation and conventional colonoscopy: Preliminary results. *American Journal of Roentgenology*, 182:1151–1158, 2004.

[59] C. Johnson and A. Dachman. Ct colonography: the next colon screening examination. *Radiology*, 216:331–341, 2000.

[60] C. D. Johnson and A. H. Dachman. CT colonography: The next colon screening examination? *Radiology*, 216(2):331–341, 2000.

[61] K. Johnson, C. Johnson, J. Fletcher, R. MacCarty, and R. Summers. CT colonography using 360 degree virtual dissection: a feasibility study. *American Journal of Roentgenology*, pages 90–95, 2006.

[62] J. Jost. *Compact Riemann Surfaces*. Springer, 2002.

[63] D.-G. Kang and J. B. Ra. A new path planning algorithm for maximizing visibility in computed tomography colonography. *IEEE Transactions on Medical Imaging*, 24(8):957–968, 2005.

[64] J. Kannala and S. Brandt. A generic camera calibration method for fish-eye lenses. *17th International Conference on Pattern Recognition*, pages 10–13, 2004.

[65] C. Kay, D. Kulling, R. Hawes, J. Young, and P. Cotton. Virtual endoscopy–comparison with colonoscopy in the detection of space-occupying lesions of the colon. *Endoscopy*, 32:226–232, 2000.

[66] S. Kim, J. Lee, J. Lee, J. Kim, P. Lefere, J. Han, and B. Choi. Computer-aided detection of colonic polyps at ct colonography using a hessian matrix-based algorithm: Preliminary study. *American Journal of Roentgenology*, 189:41–51, 2007.

[67] G. Kindlmann and J. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. *IEEE Visualization*, pages 79–86, 1998.

[68] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. *IEEE Visualization*, pages 513–520, 2003.

[69] G. Kiss, J. Cleynenbreugel, M. Thomeer, P. Suetens, and G. Marchal. Computer-aided diagnosis in virtual colonography via combination of surface normal and sphere fitting methods. *European Journal of Radiology*, 12:77–81, 2002.

[70] J. Kniss, G. Kindlmann, and C. Hansen. Multi-dimentional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[71] J. Kniss, P. McCormick, A. McPherson, J. Ahrens, J. Painter, A. Keahey, and C. Hansen. Interactive texture-based volume rendering for large data sets. *IEEE Computer Graphics and Applications*, 21:52–61, July 2001.

[72] G. Knittel. The ultravis system. *Proceedings IEEE Symposium on Volume Visualization*, pages 71–79, 2000.

[73] K. Kreeger, F. Dachille, M. Wax, and A. Kaufman. Covering all clinically significant areas of the colon surface in virtual colonoscopy. *SPIE Medical Imaging*, 4683:198–206, 2002.

[74] J. Kruger and R. Westermann. Acceleration techniques for GPU-based volume rendering. *IEEE Visualization*, pages 38–44, 2003.

[75] K. Kwon and B. Shin. An efficient camera path computation using image-space information in virtual endoscopy. *Lecture Notes in Computer Science*, 3280:118–125, 2004.

[76] P. Lacroute and M. Levoy. Fast volume rendering using a shear–warp factorization of the viewing transformation. *ACM SIGGRAPH*, pages 451–458, July 1994.

[77] P. Lacroute and M. Levoy. Fast volume rendering using a shear–warp factorization of the viewing transformation. *ACM SIGGRAPH*, pages 451–458, July 1994.

[78] S. Lakare, M. Wan, M. Sato, and A. Kaufman. 3D digital cleansing using segmentation rays. *IEEE Visualization*, pages 37–44, Oct. 2000.

[79] E. LaMar, B. Hamann, and K. Joy. Multiresolution techniques for interactive texture-based volume visualization. *IEEE Visualization*, pages 355–361, 1999.

[80] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust. Computing a canonical polygonal schema of an orientable triangulated surface. *ACM Symposium on Computational Geometry*, pages 80–89, 2001.

[81] K. V. Leemput, F. Maes, D. Vandermeulen, and P. Suetens. A unifiying framwork for partial volume segmentation of brain MR images. *IEEE Transactions on Medical Imaging*, 22(1):105–119, 2003.

[82] M. Levoy. Display of surfaces from volume data. *ACM SIGGRAPH*, pages 29–37, 1988.

[83] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990.

[84] Z. Liang, S. Lakare, M. Wax, D. Chen, J. Anderson, A. Kaufman, and D. Harrington. A pilot study on less-stressful bowel preparation for virtual colonoscopy screening with follow-up biopsy by optical colonoscopy. *SPIE Medical Imaging*, 5746:810–816, 2005.

[85] B. Lichtenbelt, R. Crane, and S. Naqvi. *Introduction to Volume Rendering*. Prentice Hall, 1998.

[86] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH*, pages 163–169, 1987.

[87] J. S. Mandel, J. H. Bond, T. R. Church, D. C. Snover, G. M. Bradley, L. M. Schuman, and F. Ederer. Reducing mortality from colorectal cancer by screening for fecal occult blood. *New England Journal of Medicine*, 328(19):1365–1371, 1993.

[88] W. R. Mark, R. S. Glanville, K. Akeley, and M. J. Kilgard. Cg: A system for programming graphics hardware in a C-like language. *ACM SIGGRAPH*, pages 896–907, 2003.

[89] C. R. Maurer, R. Qi, and V. Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

[90] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, pages 100–107, 1995.

[91] B. Mora, J. P. Jessel, and R. Caubet. A new object-order ray-casting algorithm. *IEEE Visualization*, pages 203–210, Oct. 2002.

[92] K. Mori, Y. Hayahsi, Y. Suenaga, J. Toriwaki, J. Hasegawa, K. Katada, C. Chen, and A. Clough. A method for detecting unobserved regions in virtual endoscopy system. *SPIE Medical Imaging*, 4321:134–145, 2001.

[93] M. M. Morrin and J. T. LaMont. Screening virtual colonoscopy – ready for prime time? *The New England Journal of Medicine*, 349(23):2261–2264, 2003.

[94] B. Morson. The evolution of colorectal carcinoma. *Clinical radiology*, 35:425–431, 1984.

[95] S. Muraki, M. Ogata, K.-L. Ma, K. Koshizuka, K. Kajihara, X. Liu, Y. Nagano, and K. Shimokawa. Next generation supercomputing using PC clusters with volume graphics hardware devices. *IEEE Supercomputing*, pages 51–58, Nov. 2001.

[96] J. Näppi, H. Frimmel, A. Dachman, and H. Yoshida. Computerized detection of colorectal masses in CT colonography based on fuzzy merging and wall-thickening analysis. *Medical Physics*, 31:860–872, 2004.

[97] J. Näppi and H. Yoshida. Feature-guided analysis for reduction of false positives in cad of polyps for CT colonography. *Medical Physics*, 30:1592–1601, 2003.

[98] D. Paik, C. Beaulieu, R. Jeffery, G. Rubin, and S. Napel. Automated flight path planning for virtual endoscopy. *Medical Physicis*, 25(5):629–637, 1998.

[99] D. S. Paik, C. F. Beaulieu, R. B. J. Jeffrey, C. A. Karadi, and S. Napel. Visualization modes for CT colonography using cylindrical and planar map projections. *Journal of Computer Assisted Tomography*, 24:179–188, 2000.

[100] D. S. Paik, C. F. Beaulieu, G. D. Rubin, B. Acar, R. B. Jeffery, J. Yee, J. Dey, and S. Napel. Surface normal overlap: a computer-aided detection algorithm with application to colonic polyps and lung nodules in helical CT. *IEEE Transactions on Medical Imaging*, 23(6):661–675, June 2004.

[101] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. *IEEE Visualization*, pages 233–238, Oct. 1998.

[102] P. Pescatore, T. Glucker, J. Delarive, R. Meuli, D. Pantoflickova, B. Duvoisin, P. Schnyder, A. Blum, and G. Dorta. Diagnostic accuracy and interobserver agreement of ct colonography. *Gut*, 47(1):126–130, 2000.

[103] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The volumepro real-time ray-casting system. *ACM SIGGRAPH*, pages 251–260, July 1999.

[104] H. Pfister, B. Lorensen, C. Baja, G. Kinklmann, W. Schroeder, L. S. Avila, K. Martin, R. Machiraju, and J. Lee. Visualization viewpoints: The transfer function bake-off. *IEEE Computer Graphics and Applications*, 21(3):16–23, 2001.

[105] P. Pickhardt, A. Taylor, and D. Gopal. Surface visualization at 3D endoluminal CT colonography: Degree of coverage and implications for polyp detection. *Gastroenterology*, 130:1582–1587, 2006.

[106] P. J. Pickhardt. Translucency rendering in 3D endoluminal CT colonography: A useful tool for increasing polyp specificity and decreasing interpretation time. *American Journal of Roentgenology*, 183(2):429 – 436, 2004.

[107] P. J. Pickhardt and J.-H. R. Choi. Electronic cleansing and stool tagging in CT colonography: Advantages and pitfalls with primary three-dimensional evaluation. *American Journal of Roentgenology*, 181:799–805, 2003.

[108] P. J. Pickhardt, J. R. Choi, I. Hwang, J. A. Butler, M. L. Puckett, H. A. Hildebrandt, M. Roy K. Wong, P. A. Nugent, P. A. Mysliwiec, and W. R. Schindler. Computed tomographic virtual colonoscopy to screen for colorectal neoplasia in asymptomatic adults. *New England Journal of Medicine*, 349(23):2191–2200, 2003.

[109] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[110] J. N. Reddy. *An Introduction to Nonlinear Finite Element Analysis*. Oxford University Press, 2004.

[111] D. K. Rex, C. S. Cutler, G. T. Lemmel, E. Y. Rahmani, D. W. Clark, D. J. Helper, G. A. Lehman, and D. G. Mark. Colonoscopic miss rates of adenomas determined by back-to-back colonoscopies. *Gastroenterology*, 112(1):24–28, 1997.

[112] R. Sadleir and P. Whelan. Fast colon centerline calculation using optimized 3D topological thinning. *Computerized Medical Imaging and Graphics*, 29:251–258, 2005.

[113] Y. Sato, C. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis. Tissue classification based on 3d local intensity structures for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):160–180, 2000.

[114] S. Schaefer and J. Warren. Dual marching cubes: Primal contouring of dual grids. *Pacific Graphics*, pages 70–76, 2004.

[115] R. Schoen and S.-T. Yau. *Lectures on Harmonic Mpas*. International Press, 1997.

[116] W. Sidney. Screening of colorectal cancer. *The Surgical clinics of North America*, 14:699–722, 2005.

[117] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. A simple and flexiable volume rendering framework for graphics-hardware-based raycasting. *International Workshop on Volume Graphics*, pages 187–195, 2005.

[118] M. Strengert, M. Magallon, D. Weiskopf, S. Guthe, and T. Ertl. Hierarchical visualization and compression of large volume datasets using GPU clusters. *Eurographics Symposium on Parallel Graphics and Visualization*, pages 41–48, 2004.

[119] K. R. Subramanian and D. S. Fussell. Applying space subdivision techniques to volume rendering. *IEEE Visualization*, pages 150–159, 1990.

[120] J. Suh and C. Wyatt. Deformable registration of supine and prone ccolons using centerline analysis. *IEEE International Symposium on Biomedical Imaging*, pages 708–711, 2007.

[121] R. Summers. Navigational aids for real-time virtual bronchoscopy. *American Journal of Roentgenology*, 168:1165–1170, 1997.

[122] R. M. Summers, C. D. Johnson, L. M. Pusanik, J. D. Malley, A. M. Youssef, and J. E. Reed. Automated polyp detection at CT colonography: Feasibility assessment in a human population. *Radiology*, 219(1):51–59, 2001.

[123] C. Tomasi and S. B. Göktürk. A graph method for the conservative detection of polyps in the colon. *2nd International Symposium on Virtual Colonoscopy*, 2000.

[124] D. Vining, Y. Ge, D. Ahn, and D. Stelts. Virtual colonoscopy with computer-assisted polyps detection. *Computer-Aided Diagnosis in Medical Imaging*, pages 445–452, 1999.

[125] M. Wan, F. Dachille, K. Kreeger, S. Lakare, M. Sato, A. Kaufman, M. Wax, and J. Liang. Interactive electronic biopsy for 3D virtual colonoscopy. *SPIE Medical Imaging*, 4321:483–488, 2001.

[126] M. Wan, Z. Liang, Q. Ke, L. Hong, I. Bitter, and A. Kaufman. Automatic centerline extraction for virtual colonoscopy. *IEEE Transactions on Medical Imaging*, 21:1450–1460, Dec. 2002.

[127] G. Wang, S. B. Dave, B. P. Brown, Z. Zhang, E. G. McFarland, J. W. Haller, and M. W. Vannier. Colon unraveling based on electronic field: Recent progress and future work. *SPIE Medical Imaging*, 3660:125–132, 1999.

[128] G. Wang, E. G. McFarland, B. P. Brown, and M. W. Vannier. GI tract unraveling with curved cross section. *IEEE Transactions on Medical Imaging*, 17:318–322, Apr. 1998.

[129] G. Wang and M. W. Vannier. GI tract unraveling by spiral CT. *SPIE Medical Imaging*, 2434:307–315, 1995.

[130] Z. Wang, Z. Liang, L. Li, X. Li, B. Li, J. Anderson, and D. Harrington. Reduction of false positives by internal features for polyp detection in CT-based virtual colonoscopy. *Medical Physics*, 32(12):3602–3616, 2005.

[131] Z. Wang, Z. Liang, X. Li, L. Li, D. Eremina, and H. Lu. An improved electronic colon cleansing method for detection of colonic polyps by virtual colonoscopy. *IEEE Transactions on Biomedical Engineering*, 53:1635–1646, 2006.

[132] R. Westermann and B. Sevenich. Accelerated volume ray-casting using texture mapping. *IEEE Visualization*, pages 271–278, Oct. 2001.

[133] L. Westover. Footprint evaluation for volume rendering. *Computer Graphics*, 24(4):367–376, Aug. 1990.

[134] J. Wilhelms and A. V. Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11:201–227, July 1992.

[135] J. Yao, M. Miller, M. Franaszek, and R. Summers. Colonic polyp segmentation in CT colonoscopy-based on fuzzy clustering and deformable models. *IEEE Transactions on Medical Imaging*, 23:1344–1352, 2004.

[136] H. Yoshida, Y. Masutani, P. MacEneaney, D. T. Rubin, and A. H. Dachman. Computerized detection of colonic polyps in CT colonography based on volumetric features: A pilot study. *Radiology*, pages 327–336, Jan. 2002.

[137] H. Yoshida and J. Näppi. Three-dimensional computer-aided diagnosis scheme for detection of colonic polyps. *IEEE Transactions on Medical Imaging*, 20(12):1261–1274, 2001.

[138] M. Zalis, J. Perumpillichira, and P. hahn. Digital subtraction bowel cleansing for CT colonography using morphological and linear filteration methods. *IEEE Transactions on Medical Imaging*, 23(11):1335–1343, 2004.

[139] N. Zhang, W. Hong, and A. Kaufman. Dual contouring with topolgy-preserving simplification using enhanced cell representation. *IEEE Visualization*, pages 505–512, 2004.

[140] N. Zhang, H. Qu, W. Hong, and A. Kaufman. SHIC: A view-dependent rendering framework for isosurfaces. *IEEE/SIGGRAPH Symposium on Volume Visualization*, pages 63–70, 2004.

[141] Z. Zhang, G. Wang, B. Brown, E. Mcfarland, J. Haller, and M. Vannier. Fast algorithm for soft straightening of the colon. *Academic Radiology*, pages 142–148, 2000.

[142] Z. Zhang, G. Wang, B. Brown, and M. Vannier. Distortion reduction for fast soft straightening of the colon. *Academic Radiology*, pages 506–515, 2000.