

Virtual Voyage: Interactive Navigation in the Human Colon

Lichan Hong* Shigeru Muraki† Arie Kaufman* Dirk Bartz‡ Taosong He§

Center for Visual Computing
State University of New York at Stony Brook

Abstract

Virtual colonoscopy is a non-invasive computerized medical procedure for examining the entire colon to detect polyps. We present an interactive virtual colonoscopy method, which uses a physically-based camera control model and a hardware-assisted visibility algorithm. By employing a potential field and rigid body dynamics, our camera control supplies a convenient and intuitive mechanism for examining the colonic surface while avoiding collisions. Our Z-buffer-assisted visibility algorithm culls invisible regions based on their visibility through a chain of portals, thus providing interactive rendering speed. We demonstrate our method with experimental results on a plastic pipe phantom, the Visible Human, and several patients.

CR Categories: I.3.3 [Picture/Image Generation]: Display Algorithms; I.3.5 [Computational Geometry and Object Modeling]: Physically Based Modeling; I.3.6 [Methodologies and Techniques]: Interaction Techniques; I.3.7 [Three-Dimensional Graphics and Realism]: Hidden Line/Surface Removal; I.3.8 [Applications];

Keywords: Virtual Colonoscopy, Endoscopy, Camera Control, Visibility, Physically-Based Navigation, Potential Field, Interactive Rendering, Virtual Environment

1 Introduction

Cancer of the colon and rectum is the second leading cause of cancer deaths in the USA. Approximately 150,000 new cases of colorectal cancer are diagnosed every year. Consequently, it is imperative that an effective diagnostic procedure be found to detect colonic polyps or tumors at an early stage. Currently, optical colonoscopy and barium enema are the only two procedures available for examining the entire colon to detect polyps larger than 5mm in diameter,

*Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, USA. Email: {lichan|ari}@cs.sunysb.edu

†Machine Understanding Division, Electrotechnical Laboratory, 1-1-4 Umezono, Tsukuba, 305 Japan. Email: muraki@etl.go.jp

‡University of Tübingen, WSI/GRIS, Auf der Morgenstelle 10/C9, D72076 Tübingen, Germany. Email: bartz@gris.uni-tuebingen.de

§Software Production Research Department, Bell Laboratories, Lucent Technologies, Naperville, IL 60566, USA. Email: taosong@research.bell-labs.com

which are clinically considered to have a high probability of being malignant. In optical colonoscopy, a fiber optical probe is introduced into the colon through the rectum. By manipulating the tiny camera attached to the tip of the probe, the physician examines the inner surface of the colon to identify abnormalities. This invasive procedure requires intravenous sedation, takes about one hour, and is expensive. Barium enema requires a great deal of physical cooperation from the patient when the X-ray radiographs of the colon are taken at different views. Additionally, its sensitivity can be as low as 78% in detecting polyps in the range of 5mm to 20mm [13].

In the past few years, at Stony Brook we have been independently developing an innovative technique called *virtual colonoscopy* [7], which was proposed as an alternative procedure for examining the entire colon [7, 11, 14, 20]. In general, this procedure consists of three steps. First, the patient's colon is cleansed and inflated with air in a way similar to that of optical colonoscopy. Second, while the patient is holding his or her breath, a helical CT scan of the patient's abdomen is taken, capturing a sequence of 2D slices which covers the entire range of the colon. This scan takes 30 – 45 seconds and produces several hundred transaxial slices of 512×512 pixels, which are subsequently reconstructed into a 3D volume of 100 – 250 MBytes. Finally, the colonic surface is extracted, and the physician virtually navigates inside the colon to examine the surface for possible polyps. This non-invasive procedure can potentially improve the diagnostic sensitivity and specificity with fewer complications, lower expense, shorter examination time, and reduced patient discomfort [20].

Previous work [7, 11, 14] has focused primarily on generating a fly-through animation (i.e., planned navigation) inside the colon, by moving a virtual camera from one end of the colon to the other. Specifically, after the camera parameters at the keyframes have been either manually specified [14] or automatically calculated [7, 11], intermediate frames are rendered *off-line* in several hours. Although this planned navigation provides a general overview of the colonic surface, helping to quickly exclude many benign cases, it is rather limited because no user interaction is possible. A detailed study requires close interactions such as observing the shape of an abnormality from different angles, measuring its size, and even examining the tissues beneath the abnormality.

2 Interactive Virtual Colonoscopy

In this paper, we describe a technique called *interactive virtual colonoscopy*. In addition to providing an overview of the colonic surface as in planned navigation, our technique allows the physician to interactively manipulate the virtual camera to explore detailed structures as desired. The resulting virtual voyage inside the colon may remind one of *Fantastic Voyage* (20th Century Fox, 1966), an Academy-Award winning science fiction movie, in which a team of doctors aboard a miniaturized submarine cruised inside a patient's arteries to perform brain surgery. Fig. 1 shows the process of interactive virtual colonoscopy comprised of two primary components: camera control and interactive rendering. These two technical challenges are the main focus of this paper.

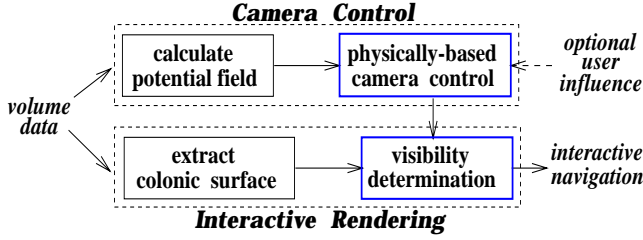


Figure 1: The process of interactive virtual colonoscopy.

Camera control essentially defines how the physician navigates inside the colon. A desirable camera control should enable the physician to examine the surface easily and intuitively, and prevent the camera from penetrating through the surface (an inescapable concern while navigating inside the colon). In this work, we have developed a physically-based camera control model. To balance between guiding the physician through the colonic interior and the physician’s freedom to manipulate the camera, our model employs a potential field and rigid body dynamics. Section 3 describes the general concept of our camera control model, how we calculate the potential field for the colonic interior from two volumetric distance fields, and move the camera using kinematic rules.

Interactive rendering speed is indispensable for virtual colonoscopy to be accepted by the medical community. For an effective navigation, at least 10 frames/second is required. From the acquired CT data, in a preprocess, we reconstruct the colonic surface using the Marching Cubes algorithm [10]. During navigation, based on the camera parameters supplied by the camera control model, we render the isosurface triangles on-the-fly to generate an image. Unfortunately, the number of triangles is enormous and can not be processed at interactive speed. Section 4 describes how we achieve high frame rates by reducing the number of triangles delivered to the graphics engine, without compromising image quality. Specifically, we present a hardware-assisted visibility algorithm which exploits the twisted nature of the colon. In Section 5, we briefly describe our user interface and present our experimental results on a pipe phantom, the Visible Human data, and patient studies.

3 Camera Control

3.1 Design Concepts

The following properties are desirable for our camera control:

- (1) Given a user-specified source point (e.g., the rectum) and a user-specified target point (e.g., the appendix), the camera automatically moves from the source point towards the target point, or vice versa.
- (2) When necessary, the physician can intuitively and easily change the camera position and direction.
- (3) To obtain a wide view of the colonic surface, the camera stays away from the surface.
- (4) Since in virtual colonoscopy the concern is the inner surface of the colon, the camera should never penetrate through the surface, even when incorrectly handled by the physician.

There has been a great deal of research on camera control for navigating within a 3D virtual environment. Roughly speaking, there are three groups of camera control techniques: planned navigation, manual navigation, and guided navigation. Planned navigation (e.g., [7, 11, 14]) does not satisfy properties 2 and 4. Manual navigation (e.g., [4, 19, 21]), which requires the user to control the camera parameters at every step, does not satisfy properties 1, 3 and 4. Guided navigation [3], which provides some guidance for

the camera and allows the user to control it when desired, does not satisfy property 4 but satisfies properties 1 – 3 quite well. Nevertheless, Galyean’s technique [3], which uses a pre-computed path as a guide and employs a spring-based model for the user control, lacks implementation details. It is unclear how the camera parameters (particularly the orientation) are interactively influenced by the user.

In this section, we present our guided-navigation camera control model which satisfies all four properties. Our camera is mounted on a *submarine*, which is immersed within a potential field [9] and moved according to a set of kinematic equations (see Fig. 2). With

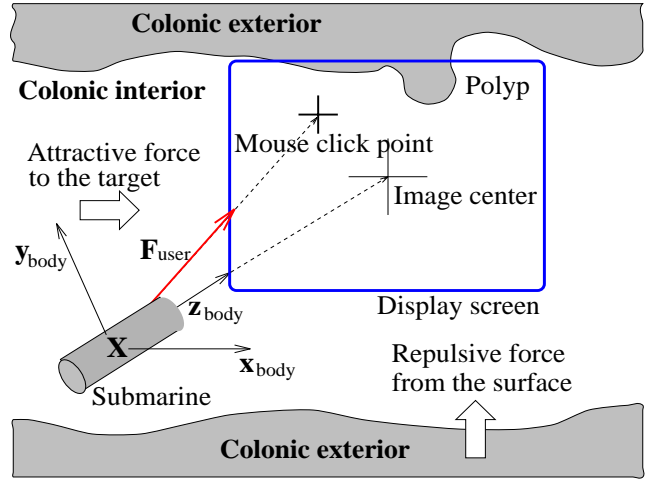


Figure 2: Our physically-based submarine model.

this physically-based model, the colonic interior mimics a “water tunnel”, with the water running downstream. The submarine is moved by the water flow as well as by any external influence imposed by the physician, but never collides with the tunnel walls. As the end user, the physician watches the display screen on which the view from the submarine is projected. When desired, the physician repeatedly clicks the mouse at a spot on the screen to maneuver the submarine closer to that spot.

3.2 Potential Field

As shown in Fig. 2, we approximate the submarine with a small cylinder, whose mass is 1. The center of mass is \mathbf{X} , where the camera is attached. \mathbf{x}_{body} , \mathbf{y}_{body} , and \mathbf{z}_{body} define the body-space coordinate system of the submarine. The major axis of the cylinder, \mathbf{z}_{body} , corresponds to the camera direction, and \mathbf{y}_{body} corresponds to the up-vector of the camera. The submarine is under the influence of the potential field $V(\mathbf{X})$, which is a volume data set of the same resolution as the CT data. Assuming that the submarine is sufficiently small, the following equation of motion is satisfied for the submarine, which is pushed toward the steepest descending direction of $V(\mathbf{X})$:

$$\dot{\mathbf{P}}(t) = -\nabla V(\mathbf{X}) - k_f \mathbf{P}(t) \quad (1)$$

$\dot{\mathbf{P}}(t)$ is the time derivative of the submarine linear momentum $\mathbf{P}(t)$ at time t ; $\nabla V(\mathbf{X})$ is the gradient of $V(\mathbf{X})$ at point \mathbf{X} ; and $k_f \mathbf{P}(t)$ is a dissipative force to prevent the submarine from moving too fast, where k_f is the friction coefficient.

To make the submarine motion satisfy properties 1, 3 and 4, we define $V(\mathbf{X})$ by using two distance fields: distance from the colonic surface $D_s(\mathbf{X})$ and distance from the target point $D_t(\mathbf{X})$. $D_s(\mathbf{X})$ and $D_t(\mathbf{X})$ are calculated using 3D image processing techniques. Since

the colon is inflated with air, the voxels inside the colon are easily extracted with a region growing method (for simplicity, we explain the process with 2D examples; see Fig. 3a). Then, for these inside voxels, we compute $D_t(\mathbf{X})$ using the single-source shortest path algorithm [2]. Fig. 3b shows the resulting $D_t(\mathbf{X})$ with the appendix as the target point. Similarly, $D_s(\mathbf{X})$ is calculated as an Euclidean distance map [15] (see Fig. 3c).

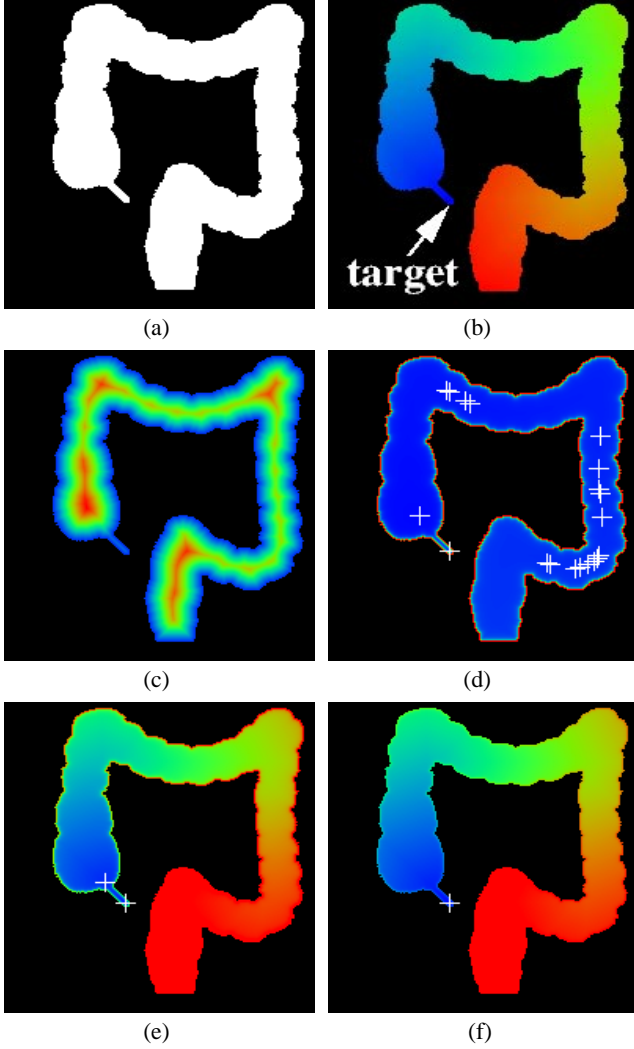


Figure 3: 2D example of potential field generation: (a) Inside voxels (white). (b) Distance from the target point (*min*: blue, *max*: red). (c) Distance from the surface (*min*: blue, *max*: red). (d) $V(\mathbf{X})$: $C_t = 1, C_s = 5, \rho = 30$ (the white cross markers show the local minima). (e) $V(\mathbf{X})$: $C_t = 30, C_s = 5, \rho = 30$. (f) $V(\mathbf{X})$: $C_t = 30, C_s = 5, \rho = 15$.

Based on $D_t(\mathbf{X})$ and $D_s(\mathbf{X})$, we generate an attractive force to the target point and a repulsive force from the surface, respectively. Specifically, $V(\mathbf{X})$ is defined as:

$$V(\mathbf{X}) = \begin{cases} C_t D_t(\mathbf{X}) + C_s (\rho / D_s(\mathbf{X}) - 1)^2, & 0 < D_s < \rho \\ C_t D_t(\mathbf{X}), & \text{otherwise} \end{cases} \quad (2)$$

where C_t and C_s are coefficients controlling the strength of the attractive and repulsive forces, respectively, and ρ is a parameter changing the distance-of-influence of the repulsive force. The repulsive potential term in Eq. 2 is a variant of the function used in robot path planning [9], which keeps the submarine away from

the surface. As a result, when the submarine approaches the surface, the value of $V(\mathbf{X})$ drastically increases and the submarine is bounced back as if there was a solid wall.

During preprocessing, $D_t(\mathbf{X})$ and $D_s(\mathbf{X})$ are computed as two volumes of the same resolution as the CT data. In order to save memory space, we only store $D_t(\mathbf{X})$ and $D_s(\mathbf{X})$ for the inside voxels along with their coordinate indices and sort them by the indices, since the submarine is inside the colon and only the distance values for the inside voxels are needed in the evaluation of $V(\mathbf{X})$. Subsequently, to calculate $V(\mathbf{X})$ during navigation, we access the distance values of an arbitrary voxel by performing a binary search on the voxel index. For a location not coinciding with any voxel vertex, trilinear interpolation is used to reconstruct the distance values from those of its eight neighboring voxels. Additionally, central differences are employed to estimate $\nabla V(\mathbf{X})$ for Eq. 1.

Like any other potential field method, the submarine may get stuck at a local minimum of $V(\mathbf{X})$. Eliminating local minima by modifying a particular $V(\mathbf{X})$ is possible [9]. However, our goal is not only to reach the target point, but also to provide a flexible submarine model. For example, if the physician increases the value of C_t , the submarine accelerates towards the target point. On the other hand, if the physician wants to examine an abnormality on the surface from close proximity, he or she can stop the influence of the attractive force by setting $C_t = 0$ and weaken the repulsive force by reducing the values of C_s and/or ρ . Thus, instead of using a static $V(\mathbf{X})$, we allow the physician to interactively adjust the values of C_t , C_s , and ρ during navigation. Figs. 3d, 3e, and 3f show how local minima depend on these values. From our experience, a small ρ and a large C_t usually reduce the number of local minima. Another way to propel the submarine out of local minima is to apply an external force, as described next.

3.3 Submarine Dynamics

To allow the physician to control the submarine, Eq. 1 is extended to:

$$\dot{\mathbf{P}}(t) = -\nabla V(\mathbf{X}) - k_f \mathbf{P}(t) + \mathbf{F}_{user}(t), \quad (3)$$

where $\mathbf{F}_{user}(t)$ is the external force applied to the submarine by mouse-clicking (see Fig. 2). Based on the geometry of the viewing frustum of the camera, the 3D location of the screen pixel clicked on by the mouse is computed. The direction of $\mathbf{F}_{user}(t)$ is then assigned as a unit vector pointing from the camera at \mathbf{X} to the screen pixel. The magnitude of $\mathbf{F}_{user}(t)$ is a parameter which can be interactively modified by the physician. $\mathbf{F}_{user}(t)$ changes not only the submarine location, but also its orientation. As a result, when the physician wants to examine an abnormality on the surface from close proximity, he or she can simply point the mouse on that region and click.

In our model, a small cylinder with six degrees of freedom is used to approximate the submarine. To represent the submarine orientation, instead of Euler's angles which tend to cause gimbal lock problem [17], we employ the quaternion $\mathbf{q}(t)$ [17] as follows:

$$\mathbf{q}(t) = [s(t), \mathbf{v}(t)] = [s, (v_x, v_y, v_z)^T]$$

The quaternion is subsequently converted into the orientation matrix $R(t)$:

$$\begin{pmatrix} 1 - 2v_y^2 - 2v_z^2 & 2v_x v_y - 2sv_z & 2v_x v_z + 2sv_y \\ 2v_x v_y + 2sv_z & 1 - 2v_x^2 - 2v_z^2 & 2v_y v_z - 2sv_x \\ 2v_x v_z - 2sv_y & 2v_y v_z + 2sv_x & 1 - 2v_x^2 - 2v_y^2 \end{pmatrix},$$

where the three column vectors correspond to \mathbf{x}_{body} , \mathbf{y}_{body} , and \mathbf{z}_{body} , respectively. Additionally, to handle the submarine as a rigid

object [1], we need to define its inertia tensor I_{body} . We approximate I_{body} with the inertia tensor of a cylinder as follows:

$$\begin{pmatrix} \frac{3r^2+h^2}{12} & 0 & 0 \\ 0 & \frac{3r^2+h^2}{12} & 0 \\ 0 & 0 & \frac{r^2}{2} \end{pmatrix},$$

where r and h are the radius and the length of the submarine, respectively. Note that the values of r and h are used only for the orientation calculation. In other words, the submarine can actually navigate through any constriction that is less than its size.

In addition to the translation depicted by Eq. 3, the rotation of the submarine is described as:

$$\dot{\mathbf{L}}(t) = \mathbf{z}_{body} \times \mathbf{F}_{user} - k_a \boldsymbol{\omega}(t) + \boldsymbol{\tau}_{option}(t). \quad (4)$$

$\dot{\mathbf{L}}(t)$ is the time derivative of the angular momentum $\mathbf{L}(t)$; $\mathbf{z}_{body} \times \mathbf{F}_{user}$ is the torque caused by \mathbf{F}_{user} ; $k_a \boldsymbol{\omega}(t)$ is a dissipative torque to stop the rotation of the submarine, where $\boldsymbol{\omega}(t)$ is the angular velocity and k_a is the friction coefficient; and $\boldsymbol{\tau}_{option}(t)$ is an optional torque which can be applied by the physician.

The differential equations, Eqs. 3 and 4, are integrated from certain initial values with the Euler method:

$$\mathbf{P}(t + \Delta t) = \mathbf{P}(t) + \dot{\mathbf{P}}(t)\Delta t, \quad (5)$$

$$\mathbf{X}(t + \Delta t) = \mathbf{X}(t) + \mathbf{P}(t + \Delta t)\Delta t, \quad (6)$$

$$\mathbf{L}(t + \Delta t) = \mathbf{L}(t) + \dot{\mathbf{L}}(t)\Delta t, \quad (7)$$

$$\boldsymbol{\omega}(t + \Delta t) = \mathbf{R}(t)I_{body}^{-1}\mathbf{R}(t)^T\mathbf{L}(t + \Delta t), \quad (8)$$

$$\mathbf{q}(t + \Delta t) = \Delta\mathbf{q}(t + \Delta t) \cdot \mathbf{q}(t). \quad (9)$$

Accordingly, we obtain the location and orientation of the submarine at every time step Δt . Eq. 9 computes the quaternion product [17] of $\Delta\mathbf{q}(t + \Delta t)$ and $\mathbf{q}(t)$, where

$$\Delta\mathbf{q}(t) = \left[\frac{\cos|\boldsymbol{\omega}(t)|\Delta t}{2}, \frac{\sin|\boldsymbol{\omega}(t)|\Delta t}{2} \frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|} \right]$$

is the quaternion representation of the submarine rotation during time step Δt . To prevent the submarine from moving erratically, we regularly check the change of \mathbf{X} in Eq. 6 and subdivide Δt when necessary.

There are a few optional controls, which are also provided by our submarine model:

- (1) **Compass mode:** When using the torque

$$\boldsymbol{\tau}_{compass}(t) = -\mathbf{z}_{body} \times \nabla V(\mathbf{X}),$$

as part of $\boldsymbol{\tau}_{option}(t)$, the submarine rotates its \mathbf{z}_{body} to match the descending direction of $V(\mathbf{X})$. This mode is useful for keeping the submarine orientation along the traveling direction of the submarine.

- (2) **Leveling mode:** If we define a certain gravity direction \mathbf{G} and apply the torque

$$\boldsymbol{\tau}_{leveling}(t) = -\mathbf{y}_{body} \times \mathbf{G},$$

the submarine rotates its \mathbf{y}_{body} towards the opposite direction of the gravity. This mode is useful for maintaining the camera up-vector pointing upward during navigation.

- (3) **Still mode:** If we do not execute Eqs. 5 and 6 at every time step, we essentially fix the submarine location while enabling other functionalities. This mode is useful for panning the camera around, yet keeping its location fixed.

4 Interactive Rendering

4.1 Reducing the Number of Triangles

Given the camera positions and orientations generated by the submarine model, we need to rapidly render images of the colonic surface. To reduce the number of triangles submitted to the graphics pipeline, one may suggest using surface simplification (e.g., [8, 16]). However, we strongly believe that in virtual colonoscopy, image fidelity should always be considered a top priority, not to be compromised. Essentially, this requires that the simplified surface should not lose any detail contained in the original highly curved surface. Otherwise, during navigation, since the camera might be very close to the colonic wall, the appearance of the surface could be degraded. More seriously, a small polyp on the surface may be faded out into the surrounding area and no longer recognizable. Such an image fidelity requirement means a very low, if any, simplification rate. Hence, no simplification has been employed in this work.

Fortunately, the twisted nature of the colon can be exploited to reduce the triangle count. As shown in Fig. 4, at any particular instant, what the camera sees is usually a small fraction of the entire surface. Therefore, to generate that snapshot, we do not need to ren-

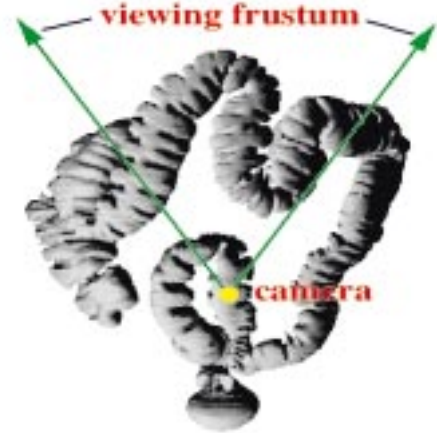


Figure 4: At any given camera position, only a small fraction of the colonic surface is visible to the camera.

der all the triangles, since most of them are invisible to the camera anyway. This essentially becomes an issue of visibility determination (e.g., [5, 6, 12, 18]). Commonly, in a visibility algorithm, the virtual environment is subdivided into a set of cells. Then, by conservatively estimating the *potentially visible set (PVS)* of cells from the camera, the number of polygons submitted to the graphics pipeline is substantially reduced. Prior work on visibility can be roughly classified into two approaches. One (e.g., [18]) determines the PVS for viewpoints inside a cell in a preprocess. The other approach (e.g., [5, 6, 12]) computes the PVS for the camera on-the-fly during rendering. In this work we have adopted the latter approach because of concerns with large memory space and long preprocessing time required by the first approach.

To determine the PVS, Greene et al. [5, 6] employed a hierarchical Z-buffer algorithm. Potentially, this algorithm can solve the visibility problem quite efficiently. Unfortunately, it relies on a Z-buffer query which is not supported in most of the currently-available graphics hardware and thus has to be implemented in software. In Luebke and Georges [12], the problem of determining which cell is potentially visible is reduced to the problem of checking whether the intersection area of the projections from the portal sequence is empty. This algorithm is particularly suitable for an architectural model [18], where cells are convex. Yet, as described

below, our cells are likely to be *concave*. All the triangles in a concave cell can potentially occlude the “see through” of a portal sequence and thus be exploited to improve rendering performance.

4.2 Subdividing into Cells

Our objective is to use graphics hardware to scan-convert the triangles and consider cell concavity to substantially reduce the size of PVS. Due to the twisted nature of the colon, instead of a BSP tree or a k -D tree [6, 18], we employ a simple subdivision method which is based on the *center-line* (or *skeleton*) of the colon. To determine the center-line, we use the distance field from the colonic surface $D_s(\mathbf{X})$, described in Section 3.2. We denote the maximal value of $D_s(\mathbf{X})$ as D_{max} . For each inside voxel, we assign a cost value of $D_{max} - D_s(\mathbf{X})$. As a result, voxels close to the surface have high cost values, while the cost values of voxels near the center of the colon are relatively low. Based on this cost assignment, we subsequently employ the single-source shortest path algorithm [2] to efficiently compute a minimum cost path from the user-specified source point to the target point. This is exactly the colonic center-line that we want to compute.

In the colon subdivision step, we start from the source point, and march along the center-line towards the target point, while continuously trying to partition the colon with the cross-section through the current position and perpendicular to the center-line (see Fig. 5). As a “cut” is desired (for example, when we have traversed enough

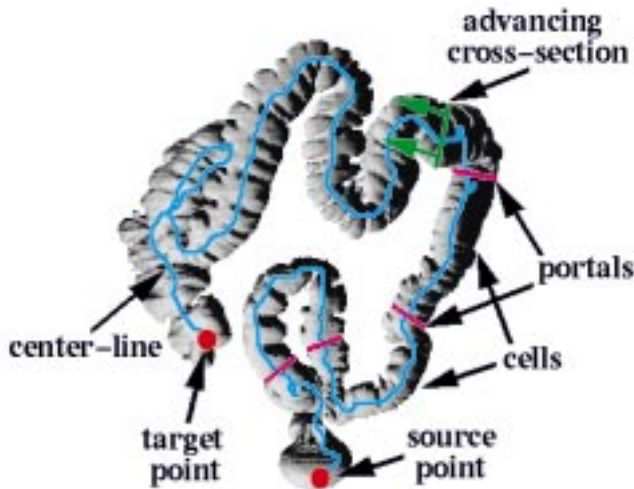


Figure 5: The patient’s colon is subdivided into a set of cells based on the colonic center-line.

distance along the center-line or the accumulated number of surface polygons has reached a certain threshold), we generate a cell which has two cross-sections as portals. Consequently, each inside voxel belongs uniquely to one cell and a list of isosurface triangles is associated with each cell. Note that the boundary of the concave cells, particularly the *haustra* which appears frequently (see Fig. 6), can be used to reduce the size of PVS, as described below.

4.3 Hardware-Assisted Visibility

During navigation, given a certain camera location, we identify the cell that contains the camera using a binary search on the location index, similar to that in Section 3.2. Then, starting from that cell, we traverse two portal sequences (towards the source and target points, respectively), in a near-to-far order. Specifically, for each portal sequence, we set up an *aggregate cull rectangle (ACR)*, which is initialized as the whole screen. As we traverse a portal

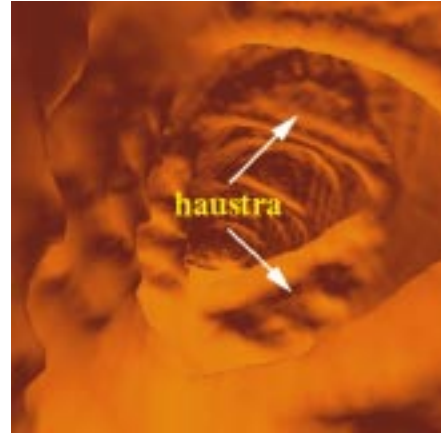


Figure 6: The concave cell boundary, especially the *haustra* which protrudes from the surface, potentially blocks the view of distant cells.

sequence, we render the triangles of the cells and shrink the ACR in an *alternating* pattern.

In other words, as we encounter a new cell, we first deliver all the triangles in the cell to the graphics engine. Note that at the farther end of this cell, we also have a new portal, which controls what can be seen beyond this cell. The ACR size is reduced in two steps (see Figs. 7a and 7b). At the first step, we project the vertices of the new

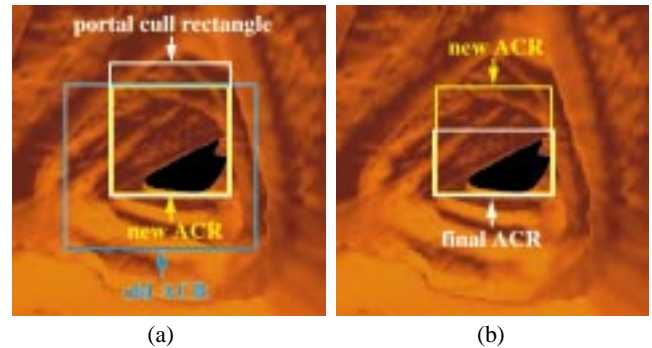


Figure 7: (a) The aggregate cull rectangle (ACR) is updated by intersecting it with the cull rectangle of the portal. (b) The size of the ACR is further reduced by checking the Z-buffer content.

portal onto the screen and compute the 2D axis-aligned bounding rectangle of the projected vertices. This bounding rectangle, called *cull rectangle*, represents a conservative bound of the portal. We update the ACR by intersecting it with the cull rectangle of the portal, as illustrated in Fig. 7a. At the second step, if the ACR has not fully degenerated, the ACR size is further reduced by checking the Z-buffer. Otherwise, no triangle is visible beyond the current portal, so we simply terminate the portal sequence.

As shown in Fig. 7b, if the first or last several scanlines of the ACR have been covered (i.e., the Z values of the pixels have been changed due to the projection of previous cells), the ACR size can be reduced. When we check the Z-buffer, instead of loading in the whole content of the Z-buffer (which takes milliseconds according to our experiments and [5]), we only read from the Z-buffer to the memory those pixels inside the ACR. Since an initialization cost is imposed in communicating with the Z-buffer, it is more efficient to read a block of pixels than reading them one-by-one. Therefore, at each step, we read in n consecutive scanlines of the ACR (top-down or bottom-up), where n is inversely proportional to the ACR width

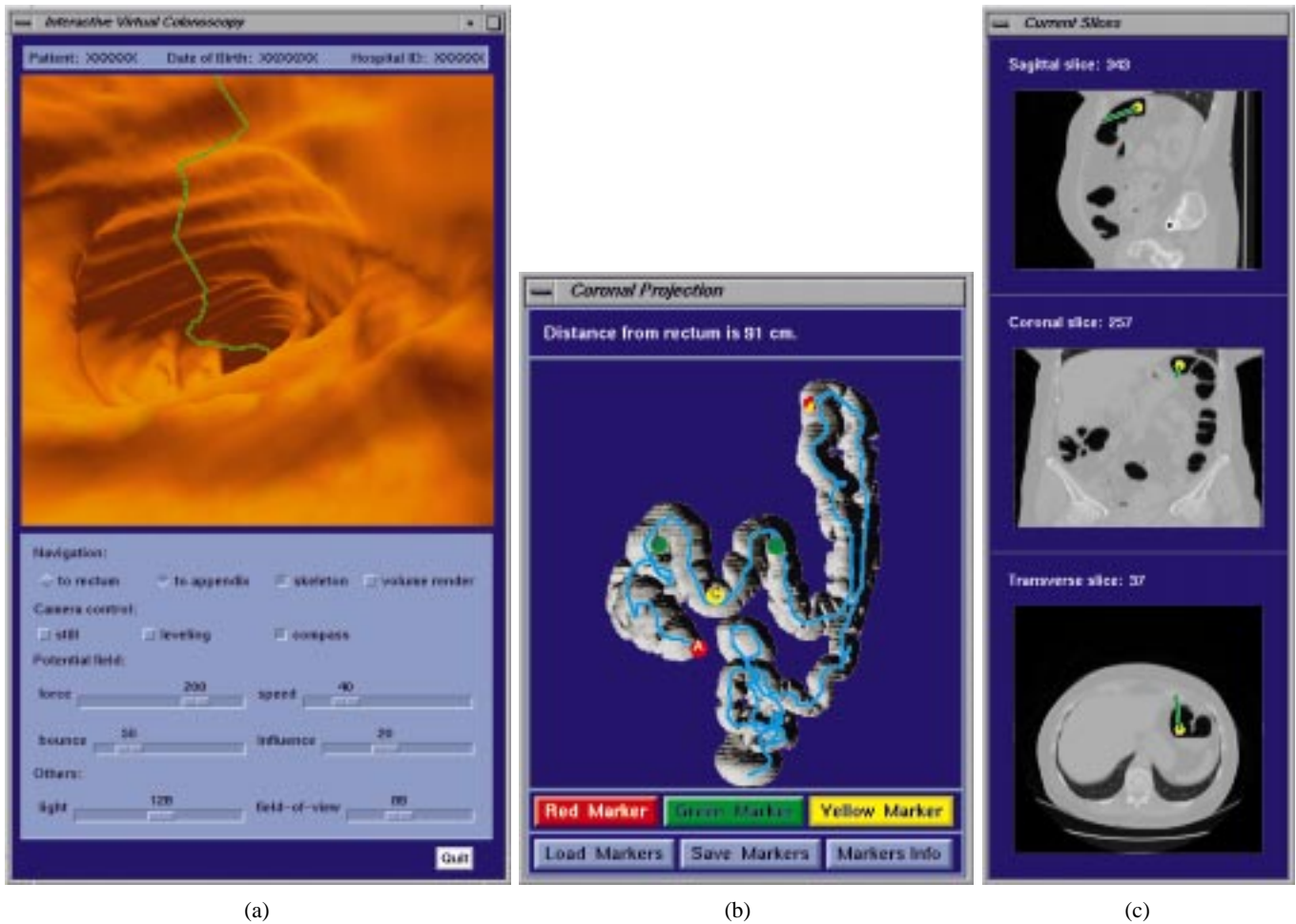


Figure 8: User interface of our interactive virtual colonoscopy: (a) endoscopy panel; (b) scout panel; (c) slice panel.

w . In the extreme case, we need to read $p + h/n$ times from the Z-buffer to generate an image, where h is the height of the ACR and p is the number of visible cells.

During navigation, since the camera parameters change only slightly and smoothly, most of the cells visible in the previous frame are probably still visible in the current frame. In other words, to generate the new frame, we encounter most of the same portals of the previous frame as we traverse the portal sequences. We have exploited this temporal coherence by reusing the visibility information of the previous image to render the new image. Specifically, without updating the ACRs, we first render 80% of the cells along the portal sequences of the previous frame. Then, we start to render the farther cells and compute the ACRs alternately, as described above.

5 Experimental Studies

Based on feedback from the two physicians involved in developing the navigation system, we have designed a user interface mimicking the visual effects of optical colonoscopy. As shown in Fig. 8, our interface consists of three components: endoscopy panel, scout panel, and slice panel. The endoscopy panel provides the patient's medical record, endoscopic view, and navigation controls. The physician can choose to navigate from the rectum to the appendix, or vice versa. The colonic center-line (shown in green in Fig. 8a) can be turned on to help the physician to better monitor the movement of

the submarine. The "volume render" option, turned on if desired, allows the physician to visualize the tissues beneath the surface to differentiate residual stool from polyps (see Fig. 9). The scout panel shows the actual shape of the patient's entire colon, with markers placed to identify points of interest. The current position and orientation of the submarine is also depicted on the scout panel. The slice panel displays the sagittal, coronal, and transverse slices through the current submarine position.

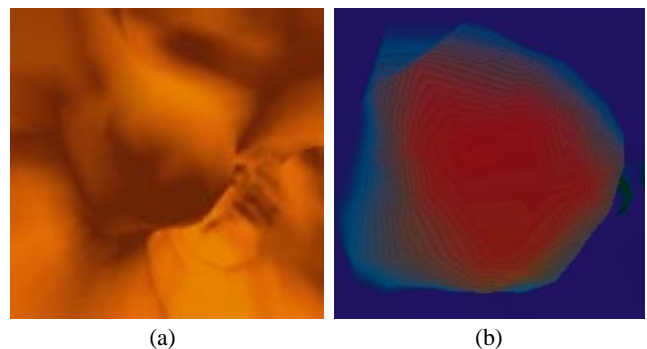


Figure 9: (a) An abnormality on the surface. (b) With close-up volume rendering, the abnormality of (a) was confirmed to be a polyp.

Our initial experiments with virtual colonoscopy used a pipe phantom and the Visible Male data. In the pipe simulation, we looped a plastic respirator pipe of 20mm radius inside a water tank. To simulate colonic polyps, we attached three small rubber objects of sizes 7mm, 5mm, and 3mm to the inner surface of the pipe. The CT scan of the pipe produced a volume of $512 \times 512 \times 107$. The image in Fig. 10a, which was captured during a navigation inside the pipe, clearly depicts the three simulated polyps. In the Visible Male data set, we primarily focused on the physical cross-sections (24-bit color slices) of the abdomen, from slice 1595 to slice 1848. For the navigation, the 24-bit color slices were downsampled, converted into 8-bit grayscale, segmented, and subsequently assembled into a volume of $683 \times 406 \times 254$. Fig. 10b shows an abnormality on the colonic surface of the Visible Male.

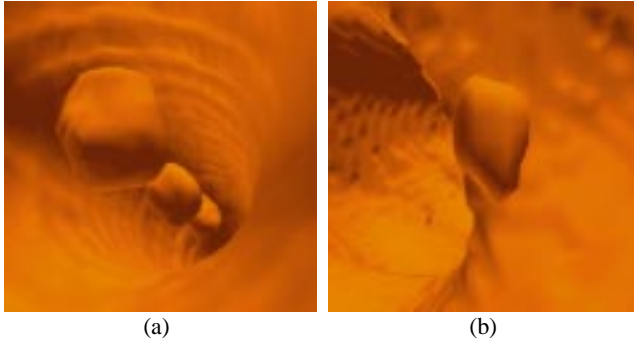


Figure 10: (a) An image from the pipe phantom clearly depicts the three simulated polyps. (b) An image from the Visible Male colon shows an abnormality on the surface.

Due to the encouraging results obtained from the previous two experiments, we acquired about a dozen patient data sets (see Figs. 4-9, 11). Polyps discovered by our collaborating physicians using our interactive virtual colonoscopy technique have been documented. They were subsequently evaluated and compared with the results of optical colonoscopy, performed by gastroenterologists and recorded on video immediately following the CT scan. In Fig. 11 we compare two diagnostic results of virtual colonoscopy with those of optical colonoscopy.

Table 1 shows the results that we obtained on three patient data sets. Column *MC* is the number of isosurface triangles generated by the Marching Cubes algorithm [10]; column *LG* shows the number of triangles per frame rendered with Luebke's and Georges' algorithm [12]; and column *Our* is the average triangles per frame using our visibility algorithm. Note that the number of visible triangles is determined mainly by the twisted degree of the colon as well as the amount of air inflated into the colon in the preparation routine.

Table 1: Triangles/frame rendered with the Marching Cubes algorithm (*MC*), Luebke's and Georges' method (*LG*), and our visibility scheme (*Our*).

Data Set	MC	LG	Our
P1 ($512 \times 512 \times 361$)	1,339K	212K	146K
P2 ($512 \times 512 \times 370$)	1,117K	132K	72K
P3 ($512 \times 512 \times 358$)	1,249K	96K	53K

We have achieved adequate interactive speeds for image size 512×512 on a single SGI R10000 processor with InfiniteReality. Average frames/second for patient data sets P1 – P3 were 14.8, 18.7, and 22.7, respectively. Rendering performances on an SGI RealityEngine2 were 5.6, 7.9, and 10.0 frames/second, respectively. The timings include not only the rendering expense but also the cost involved in solving the submarine motion equations.

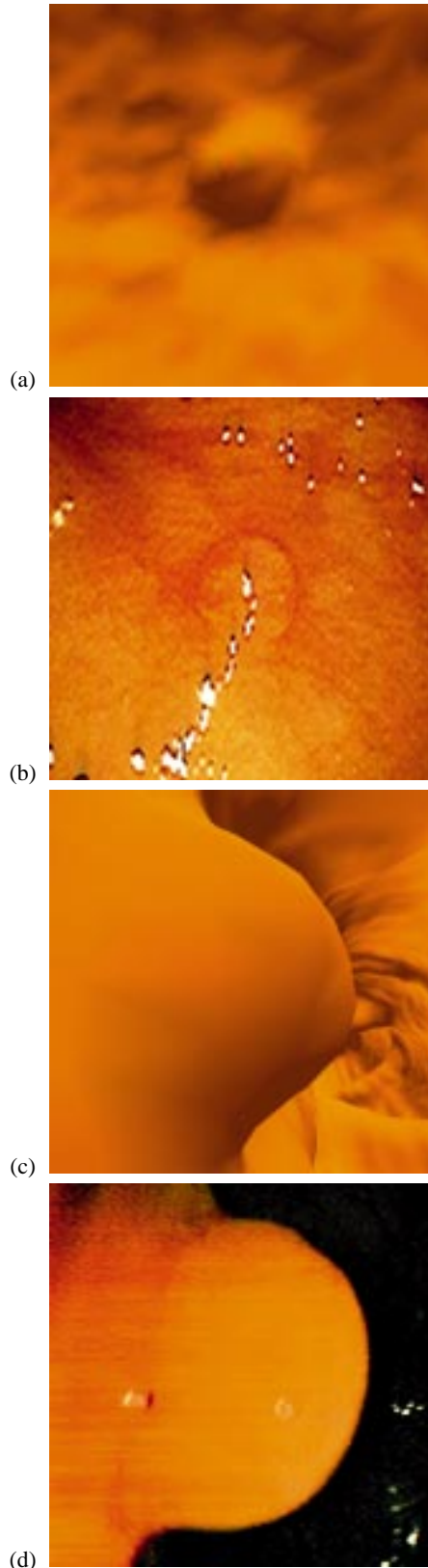


Figure 11: (a) A polyp of size 4mm at the center of the image discovered in virtual colonoscopy. (b) The corresponding polyp of (a) confirmed by optical colonoscopy. (c) Close-up of another polyp of size 8mm from virtual colonoscopy. (d) The corresponding polyp of (c) confirmed by optical colonoscopy.

6 Conclusions

In this work our primary contributions have been an intuitive and physically-based submarine model with collision avoidance using a potential field and kinematic rules, and an efficient Z-buffer-assisted visibility method which both employs a center-line-based subdivision and exploits cell concavity. We have harnessed these interactive computer graphics advances for the benefit of the important application of interactive virtual colonoscopy.

As a cost-effective, non-invasive, and patient-comfort procedure, interactive virtual colonoscopy provides a potential alternative for mass screening of patients for possible colon cancer. Unlike optical colonoscopy, our technique allows the physician to examine the tissues underneath abnormalities and measure polyp size, without physical biopsy and surgery. Our physically-based submarine model provides an intuitive mechanism to navigate inside the colon for examining the surface, as testified by our collaborating physicians. From their experience, usually after one hour of training, a computer-illiterate physician is able to master the system quite well, while two years of training is commonly required for optical colonoscopy. In addition, due to the accelerated rendering speed achieved by our technique, diagnostic time has been greatly reduced. Usually only several minutes are required to navigate through the entire colon for locating abnormalities, in contrast to one hour for optical colonoscopy.

In general, the preprocessing stage takes a few hours on our SGI workstations. It includes computing the two distance fields $D_s(\mathbf{X})$ and $D_t(\mathbf{X})$, extracting the center-line of the colon, subdividing the colon into cells, and generating isosurface triangles from the CT data. We are currently working on a variety of techniques to speed up the preprocess. Additionally, we plan to improve the speed of the volume rendering option by exploiting the depth information produced by our visibility algorithm. Together with our collaborating physicians, we are evaluating our technique on clinical cases. We are also working on "subtracting" residual stool and liquid from the colonic interior, and highlighting abnormalities with color mapping to attract the physician's attention. Furthermore, our technique is being extended to virtual endoscopy for examining other human organs where traditional medical procedures are invasive or difficult to perform.

Acknowledgments

This work has been supported by the Center for Biotechnology of New York State and E-Z-EM Inc. The Visible Human data set is courtesy of the National Library of Medicine and the Visible Human Project. The pipe and patient data were provided by our collaborating physicians at the Stony Brook University Hospital. We thank Ajay Viswambharan, Mark Wax, Jerome Liang, Yong Zhou, and Suya You for their contributions to this project. Special thanks to Amitabh Varshney, Claudio Silva, Lan Yu, Edmond Prakash, Ming Wan, Nilo Stolte, and Kathleen McConnell for their helpful comments on drafts of this paper.

References

- [1] D. Baraff. Rigid Body Simulation. *SIGGRAPH 95 Course Note 34*. ACM SIGGRAPH, August 1995.
- [2] E. Dijkstra. A Note on Two Problems in Connection with Graphs. *Numerische Mathematik*, vol. 1, 1959, pp. 269-270.
- [3] T. Galyean. Guided Navigation of Virtual Environments. *ACM Symposium on Interactive 3D Graphics*, pp. 103-104. ACM, April 1995.
- [4] M. Gleicher and A. Witkin. Through-the-Lens Camera Control. *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, vol. 26, pp. 331-340. ACM SIGGRAPH, July 1992.
- [5] N. Greene, M. Kass, and G. Miller. Hierarchical Z-Buffer Visibility. *SIGGRAPH 93 Conference Proceedings*, Annual Conference Series, pp. 231-236. ACM SIGGRAPH, August 1993.
- [6] N. Greene. Hierarchical Polygon Tiling with Coverage Masks. *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pp. 65-74. ACM SIGGRAPH, August 1996.
- [7] L. Hong, A. Kaufman, Y. Wei, A. Viswambharan, M. Wax, and Z. Liang. 3D Virtual Colonoscopy. *IEEE Symposium on Biomedical Visualization*, pp. 26-32. IEEE, October 1995.
- [8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh Simplification. *SIGGRAPH 93 Conference Proceedings*, Annual Conference Series, pp. 19-26. ACM SIGGRAPH, August 1993.
- [9] J. Latombe. Robot Motion Planning. *Kluwer Academic Publishers*, 1991.
- [10] W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics (SIGGRAPH 87 Conference Proceedings)*, vol. 21, pp. 163-169. ACM SIGGRAPH, July 1987.
- [11] W. Lorensen, F. Jolesz, and R. Kikinis. The Exploration of Cross-Sectional Data with a Virtual Endoscope. In R. Satava and K. Morgan (eds.), *Interactive Technology and New Medical Paradigm for Health Care*, IOS Press, 1995, pp. 221-230.
- [12] D. Luebke and C. Georges. Portals and Mirrors: Simple, Fast Evaluation of Potential Visible Sets. *ACM Symposium on Interactive 3D Graphics*, pp. 105-106. ACM, April 1995.
- [13] C. Morosi, G. Ballardini, and P. Pisani. Diagnostic Accuracy of the Double-Contrast Enema for Colonic Polyps in Patients with or without Diverticular Disease. *Gastrointest Radiology*, vol. 16, 1991, pp. 346-347.
- [14] G. Rubin, C. Beaulieu, V. Argiro, H. Ringl, A. Norbash, J. Feller, M. Dake, R. Jeffrey, and S. Napel. Perspective Volume Rendering of CT and MR Images: Applications for Endoscopic Imaging. *Radiology*, vol. 199, May 1996, pp. 321-330.
- [15] T. Saito and J. Toriwaki. New Algorithms for Euclidean Distance Transformation of an N-Dimensional Digitized Picture with Applications. *Pattern Recognition*, vol. 27, no. 11, 1994, pp. 1551-1565.
- [16] W. Schroeder, J. Zarge, and W. Lorensen. Decimation of Triangle Meshes. *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, vol. 26, pp. 65-70. ACM SIGGRAPH, July 1992.
- [17] K. Shoemake. Animation Rotation with Quaternion Curves. *Computer Graphics (SIGGRAPH 85 Conference Proceedings)*, vol. 19, pp. 245-254. ACM SIGGRAPH, July 1985.
- [18] S. Teller and C. Sequin. Visibility Preprocessing For Interactive Walkthroughs. *Computer Graphics (SIGGRAPH 91 Conference Proceedings)*, vol. 25, pp. 61-69. ACM SIGGRAPH, July 1991.
- [19] R. Turner, F. Balaguer, E. Gobbetti, and D. Thalmann. Physically-Based Interactive Camera Motion Control Using 3D Input Devices. *Computer Graphics International '91*, pp. 135-145. Springer-Verlag, June 1991.
- [20] D. Vining, D. Gelfand, R. Bechtold, E. Scharling, E. Grishaw, and R. Shifrin. Technical Feasibility of Colon Imaging with Helical CT and Virtual Reality. *Annual Meeting of American Roentgen Ray Society*, 1994, pp. 104.
- [21] C. Ware and S. Osborne. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. *ACM Symposium on Interactive 3D Graphics*, pp. 175-183. ACM, March 1990.