8-2012

Enhancement of virtual colonoscopy system.

Rosario J. Pusateri
University of Louisville

ENHANCEMENT OF VIRTUAL COLONOSCOPY SYSTEM

By

Rosario J. Pusateri
B.S. (ECE), University of Louisville, 2010
B.S. (Physics), University of Louisville, 2011

A Thesis
Submitted to the Faculty of the
University of Louisville
J. B. Speed School of Engineering
as Partial Fulfillment of the Requirements
for the Professional Degree

MASTER OF ENGINEERING

Department of Electrical and Computer Engineering

August 2012

ENHANCEMENT OF VIRTUAL COLONOSCOPY SYSTEM

Submitted by:

_____

Rosario J. Pusateri

A Thesis Approved on

_____

(Date)

by the Following Reading and Examination Committee:

_____

Dr. Aly A. Farag, Thesis Director

_____

Dean Thomas L. Starr

_____

Dr. James H. Graham

_____

Dr. Gerald W. Dryden

ACKNOWLEDGEMENTS

I would like to thank Dr. Farag for providing me the opportunity to conduct research at the Computer Vision and Image Processing (CVIP) Lab, Mike Miller for offering me support as a CVIP researcher, and Marwa Ismail for her intellectual guidance as a partner in the CVIP Lab colon project. I would also like to thank my committee members, Dr. James H. Graham and Dean Thomas L. Starr for their participation in my defense as well as Dr. Gerald Dryden and Dr. Rob Falk for their guidance as medical professionals in the CVIP Lab colon project. My thanks also goes to the many friends I have made in the CVIP Lab that have assisted in my growth as a student and a person. Finally, and with much love, I would like to thank my mother, Kim Courtney, and my girlfriend, Teresa McGeeney, for their infinite patience and love for me during my many trying years in study.

ABSTRACT

Rosario J. Pusateri

August 2012

Colorectal cancer is the fourth most common cancer, and the fourth leading cause of cancer related death in the United States. It also happens to be one of the most preventable cancers provided an individual performs a regular screening. For years colonoscopy via colonoscope was the only method for colorectal cancer screening. In the past decade, colonography or virtual colonoscopy (VC) has become an alternative (or supplement) to the traditional colonoscopy.

VC has become a much researched topic since its introduction in the mid-nineties. Various visualization methods have been introduced including: traditional flythrough, colon flattening, and unfolded-cube projection. In recent years, the CVIP Lab has introduced a patented visualization method for VC called flyover. This novel visualization method provides complete visualization of the large intestine without significant modification to the rendered three-dimensional model.

In this thesis, a CVIP Lab VC interface was developed using Lab software to segment, extract the centerline, split (for flyover), and visualize the large intestine. This system includes adaptive level sets software to perform large intestine segmentation, and CVIP Lab patented curve skeletons software to extract the large intestine centerline. This software suite has not been combined in this manner before so the system stands as a unique contribution to the CVIP Lab colon project. The system is also a novel VC pipeline when compared to other academic and commercial VC methods. The complete

system is capable of segmenting, finding the centerline, splitting, and visualizing a large intestine with a limited number of slices (~350 slices) for VC in approximately four and a half minutes. Complete CT scans were also validated with the centerline extraction external to the system (since the curve skeletons code used for centerline extraction cause memory exceptions because of high memory utilization).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1:

INTRODUCTION

1.1 Colorectal Cancer Screening Background

Colorectal cancer is a cancer that affects either the colon and/or rectum region of the large intestine [see figure 1]. In the United States it is the fourth most common cancer in both women and men as of 2012, and the fourth leading cause of death after lung, stomach, and liver cancer [1]. Colorectal cancer is also one of the most preventable cancers when controlling for lifestyle choice and environment as it can be thwarted with various medical diagnostic procedures. This is because the cancer usually begins as a localized benign colorectal polyp that takes nearly 10 to 15 years to develop into cancer.

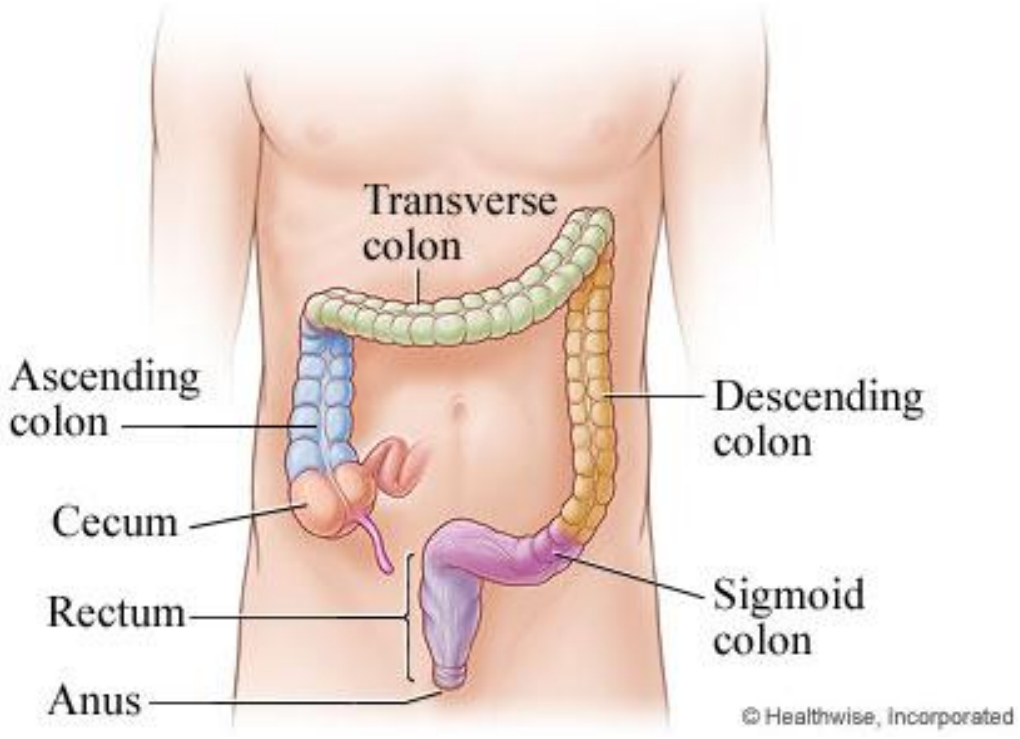

Figure 1: Anatomy of the large intestine [28].

The colonoscopy procedure is the most common method for diagnosing potential colorectal cancer cases. A traditional colonoscopy requires preparation with laxatives (typically magnesium citrate and/or sodium phosphate), and *bowel irrigation* via a combination of polyethylene glycol and electrolytes (sports drinks or similar fluid). This process is typically conducted 24 hours prior to the procedure and is usually unpleasant for the patient. The procedure itself requires the patient to be given mild-dose general anesthesia (*twilight anesthesia*). Following this, a CCD camera fitted colonoscope is inserted into the anus of the patient and progressed through the colon [see figure 2 & 3]. Potential polyps within a certain size range can be removed and biopsied for malignancy.

Figure 2: A colonoscope used for colonoscopy (without polyp extraction component), and a device for removing polyps [29].

Although, this procedure is the gold standard among most gastrointestinal doctors its invasiveness deter many patients from having it done as recommended (every 10 years after age 50) [2]. In addition the colonoscopy procedure may miss potential polyps in the

ascending colon near the cecum (proximal from the anus), and polyps hidden behind haustra. Computed tomography colonography (CT colonography) or virtual colonoscopy is an alternative approach to the traditional colonoscopy. The procedure is non-invasive (or nearly non-invasive) and provides complete visualization of the entire colon. This procedure is fast becoming an accepted method for colorectal cancer screening.

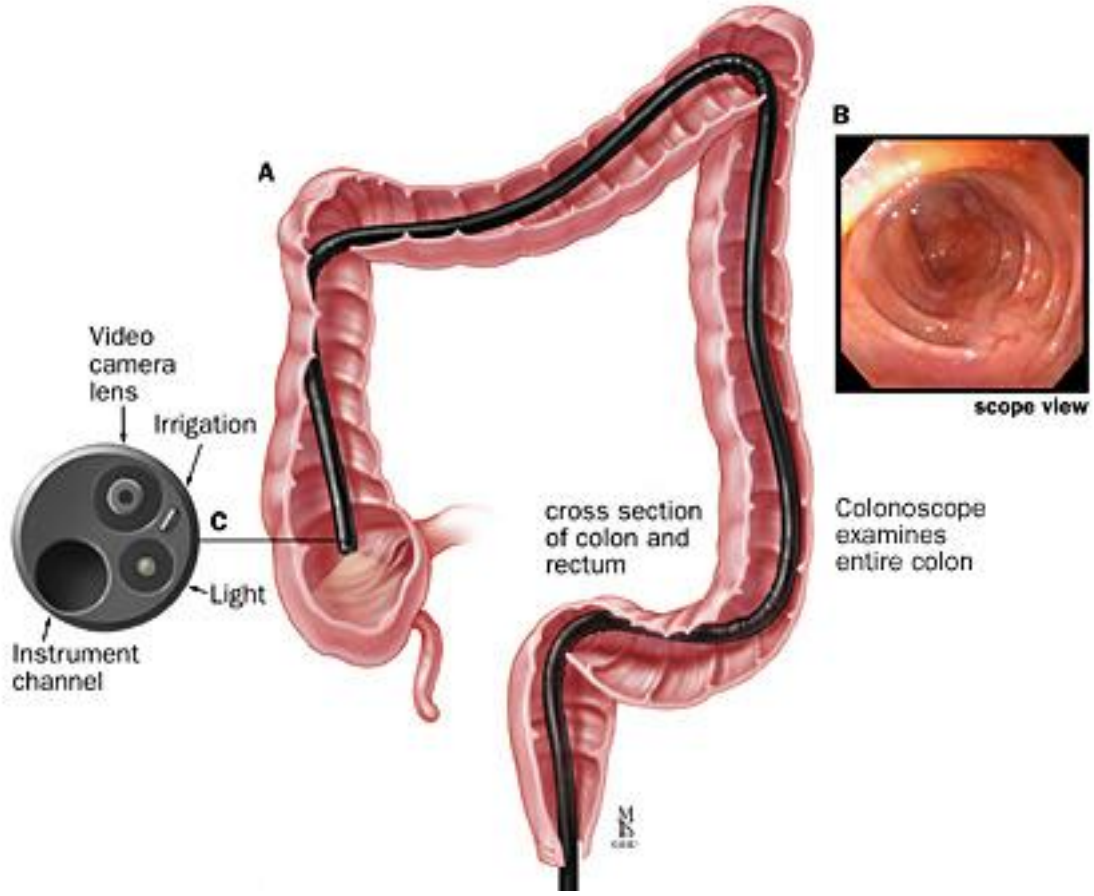

Figure 3: The colonscope in the colon (A), the view from the scope head inside the large intestine (B), and the scope head components (C) [30].

1.2 Virtual Colonoscopy (VC)

Dr.Vining and Dr. Gelfand at the Society of Gastrointestinal Radiologist first presented virtual colonoscopy to the medical community in 1994 [3]. Although it was only a non-interactive video of a virtual colonoscopy *flythrough* (explained below) it sparked an explosion of interest in the academic and medical community. Since then, VC has become one of the accepted methods for colorectal cancer screening.

The VC method for colorectal cancer screening is minimally invasive and requires less time than a traditional colonoscopy. The patient follows a colon-cleansing regimen similar to that done for the traditional colonoscopy screening. For the procedure, the patient lies in the supine position and a small tube is inserted into the anus to inflate the colon with a small amount of air (this is to create proper large intestine distension for better viewing). Then the patient is scanned with either CT or MRI (the conducted work was with CT images). The patient then moves to a prone position to be scanned again. Two scans in different positions are needed to ensure that no polyps are missed in pools of residual fluid or fecal matter. For recent VC procedures, the scans are typically low-dose, or rarely, ultra-low dose to minimize exposure to radiation. It has been concluded that low-dose CT resolution is adequate to reveal most significant polyps in colorectal scans [9].

After the procedure the scan images (DICOM for the work conducted) are processed by available software. Typically, the software renders the colon images in a manner that mimics a traditional colonoscopy. This is called the flythrough visualization method. Other means of visualization include: flattening [11], unfolded cube projection [10], and

flyover [12] (developed by the CVIP Lab and patented). These methods will be elaborated upon in the literature review.

1.3 CVIP Lab Virtual Colonoscopy Interface

The task assigned for this thesis is the consolidation of CVIP Lab segmentation, centerline extraction, and visualization-preparation/visualization software into one front-end system for VC [see figure 4]. Therefore, can these components be combined in a manner that is computationally efficient? The interface pipeline includes: a means of segmenting the colon images via adaptive level sets [23, 24, 17, 19], a method for extracting the colon centerline via curve skeletons using gradient vector flow [20, 21, 22], and visualization-preparation/visualization of the rendered colorectal images via the flyover method (using VTK and Qt). This pipeline will be surveyed part by part after a thorough literature review outlining research in the field of VC.

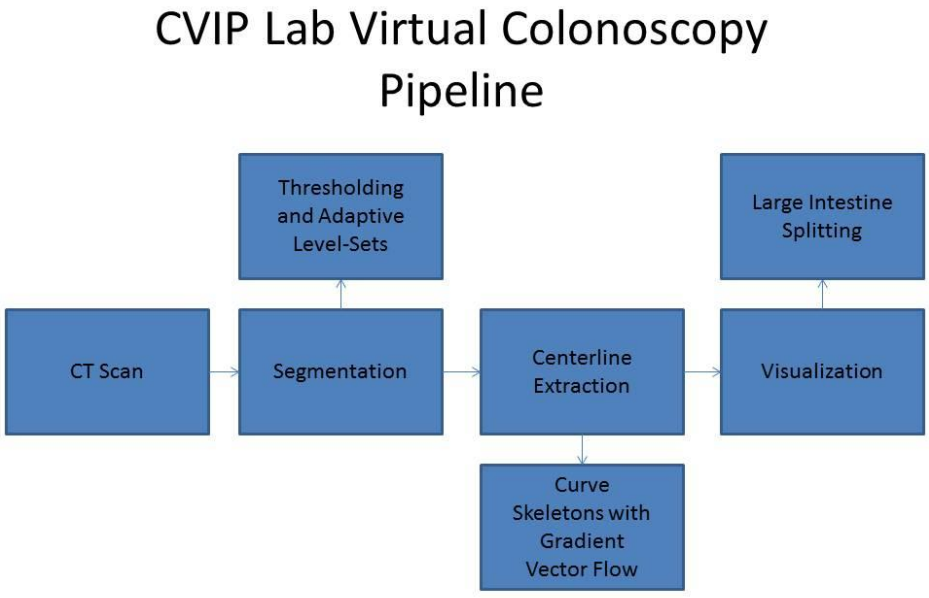

Figure 4: The proposed CVIP Lab VC interface pipeline.

CHAPTER 2:

LITERATURE REVIEW

2.1 VC Using the Flythrough Visualization Method

The flythrough method is the traditional model for virtual colorectal visualization. It aims to simulate a colonoscopy procedure by starting the camera at the anus and progressing it through the virtual large intestine toward the cecum; this is an antegrade VC (note that VC may also be retrograde, or from the cecum to the anus, as this allows more complete viewing of the colon). This was first proposed academically in the work by Lichan Hong, *3D Virtual Colonoscopy* (1995) one year after the first virtual colonoscopy demonstration [13]. In this foundation publication the patient was prepped and scanned with CT to obtain images of the large intestine. The visualization was obtained through the use of third-party volume visualization software, *VolVis*. The produced visualization was a flythrough movie that was non-interactive and a basic proof-of-concept.

Since 1994 numerous commercial VC products have been produced that utilize the flythrough visualization method. Although, current visualizations vary somewhat the flythrough remains the gold standard for medical doctors using VC because of its close similarity to traditional colonoscopy.

Figure 5: An example of a flythrough visualization (the red circle is a tagged polyp).

2.2 VC Using the Flattening Visualization Method

As with traditional colonoscopy, flythrough VC can miss potential polyps that are hidden behind haustra. This is because the antegrade flythrough VC can only make visible colorectal tissue exposed to the normal of the viewing camera. Therefore, polyps hidden behind haustra viewed from the antegrade direction will be missed. There are only two means of remedying this problem with VC flythrough: stopping the visualization camera at regular intervals and panning it about the colorectal interior or, conducting a retrograde flythrough in addition to the antegrade flythrough. Stopping at regular intervals is time consuming and unrealistic for efficient patient diagnosis. Antegrade in combination with retrograde visualization, while less time consuming then the former method, is still an involved process that requires redundancy in procedure. In addition,

antegrade and retrograde flythrough are both prone to having colorectal regions that are not visible from either direction [see figure 6].

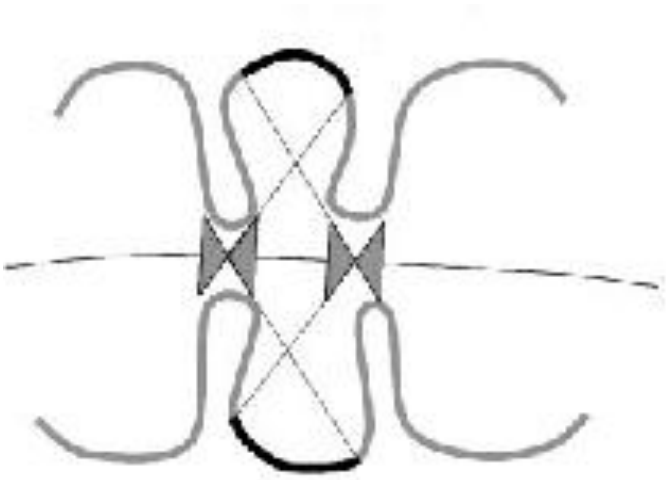

Figure 6: Cross cut of large intestine showing the camera field of view for antegrade and retrograde colonoscopy flythrough (the hidden regions are traced in black) [10].

For these reasons, it has become desirable to expose the entire virtual large intestine surface as the camera passes without redundant visualization or regular camera panning. Colorectal flattening had been proposed in *Nondistorting Flattening in Virtual Colonoscopy* and *Nondistorting Flattening Maps and the 3D Visualization of Colon CT Images* in 2000 [11, 14]. In this research, colon flatting is achieved by mapping from an assumed open cylindrical surface to a rectangular surface [see figure 7].

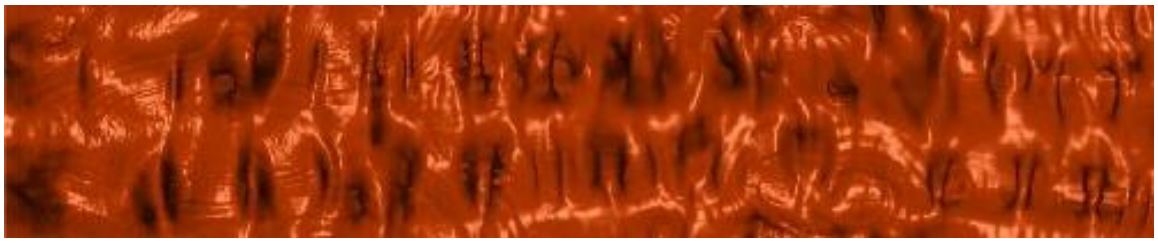

Figure 7: An example image of colorectal flattening [15].

The removal of distortion from the projection is achieved along a defined curve using geographic projection techniques. In *Conformal Virtual Colon Flattening* a similar result is achieved through more intensive application of topological methods [15]. Although in this approach, the colon is mapped to completely fill a proper geometric rectangle, whereas in the former approaches the colon did not map completely but included some offset.

2.3 VC Using the Unfolded Cube Projection Visualization Method

Colon flattening techniques reduce the need for redundant visualizations by revealing the entire surface of the organ, but a flattened colon surface is not visually intuitive to examine by a medical professional (flattening maps to two dimensions, while traditional visualizations of the colon interior are in three dimensions). It is also true that discerning three-dimensional forms from two-dimensional mappings can be difficult without proper training.

Unfolded cube projection is another researched method for colon visualization proposed in the paper *Three-Dimensional Display Modes for CT Colonography: Conventional 3D Virtual Colonoscopy versus Unfolded Cube Projection* in 2003 [10]. For this method a point on the centerline is defined with six camera normals. These normals are all orthogonal to one another and one normal is oriented in the antegrade centerline direction while another normal is oriented in the retrograde centerline direction. As a result, the views of the colon are projected to the inside of a cube. This imaging cube is unfolded and flattened to show all the potential views of the colon interior at once for every centerline position [see figure 8].

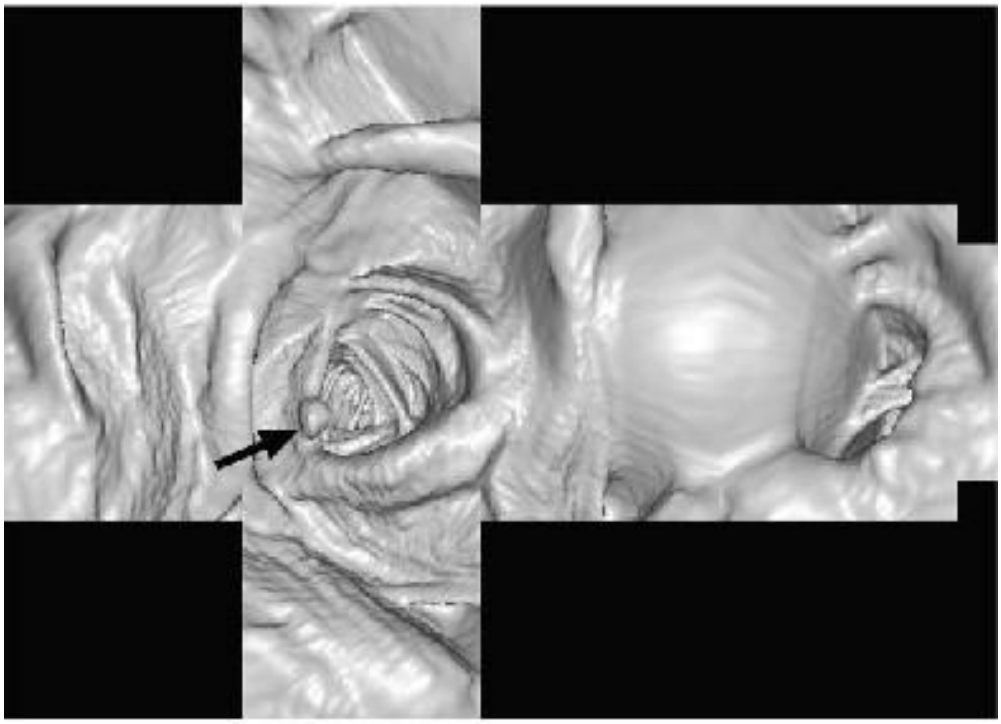

Figure 8: The unfolded cube projection with a polyp indicated with the black arrow (note how the projection shows both antegrade and retrograde perspectives) [10].

Though this method provides a more intuitive visualization its layout is an object of criticism. The projections are situated in a manner that is time consuming to diagnose as the reader is required to scan each side of the unfolded cube [see figure 8]. Therefore, unfolded cube projection solves the 'hidden polyps' problem of flythrough VC, but still requires considerable time to diagnose properly.

2.4 Summary

VC flythrough visualization is still considered a gold standard by many radiologists and medical professionals when considering the various VC methods. In recent years the other flattening and projection methods have become viable in some commercial applications. Viatronix, General Electric (GE), and Philips all offer

commercial VC products. All of these companies use the flythrough visualization while GE and Philips use some variation of colon flattening as well. Research in efficient visualization methods is still a relevant pursuit, as VC becomes a more accepted method for colorectal cancer screening.

CHAPTER 3:

CVIP LAB VC INTERFACE: DATA ACQUISITION

3.1 Pre-CT Scan Prep and CT Scan Protocol

The CT scans used to test and validate the CVIP Lab VC interface were obtained from the Virtual Colonoscopy Center at Walter Reed Army Medical Center in Washington D.C. and 3DR Incorporated in Louisville, KY. Patients involved in the scans underwent a twenty-four hour large intestine prep protocol that required: oral intake of 90 milliliters of sodium phosphate and 10 milligrams of bisacodyl (these are laxatives required to flush the large intestine of contents), as well as 500 milliliters of barium (a radiocontrast agent) and 120 milliliters of Gastrografin (known as diatrizoic acid) which contains iodine (also a radiocontrast agent). When ingested, the barium and Gastrografin cause the remaining luminal fluid (i.e. fecal matter and water) to have a high X-Ray absorption capacity (similar to bone), and objects with high absorption display brightly on CT scans. Barium and iodine are ideal radiocontrast agents since they have high atomic density without being radioactive.

The scans were done using either the GE LightSpeed four-channel scanner or the GE LightSpeed Ultra eight-channel scanner. The scanners were set to 100 mAs (current exposure time product) and 120 kVp (peak kilovoltage) with 1.25 millimeter to 2.5 millimeter collimation and 15 millimeter per second scan table speed. The spatial resolution for each scan is one cubic millimeter while the number of slices varies from 400 to approximately 550 slices. A single transverse scan typically contains: bone (high

contrast), luminal fluid (high contrast), tissue (mid-level contrast), and air (low-contrast) [see figure 9].

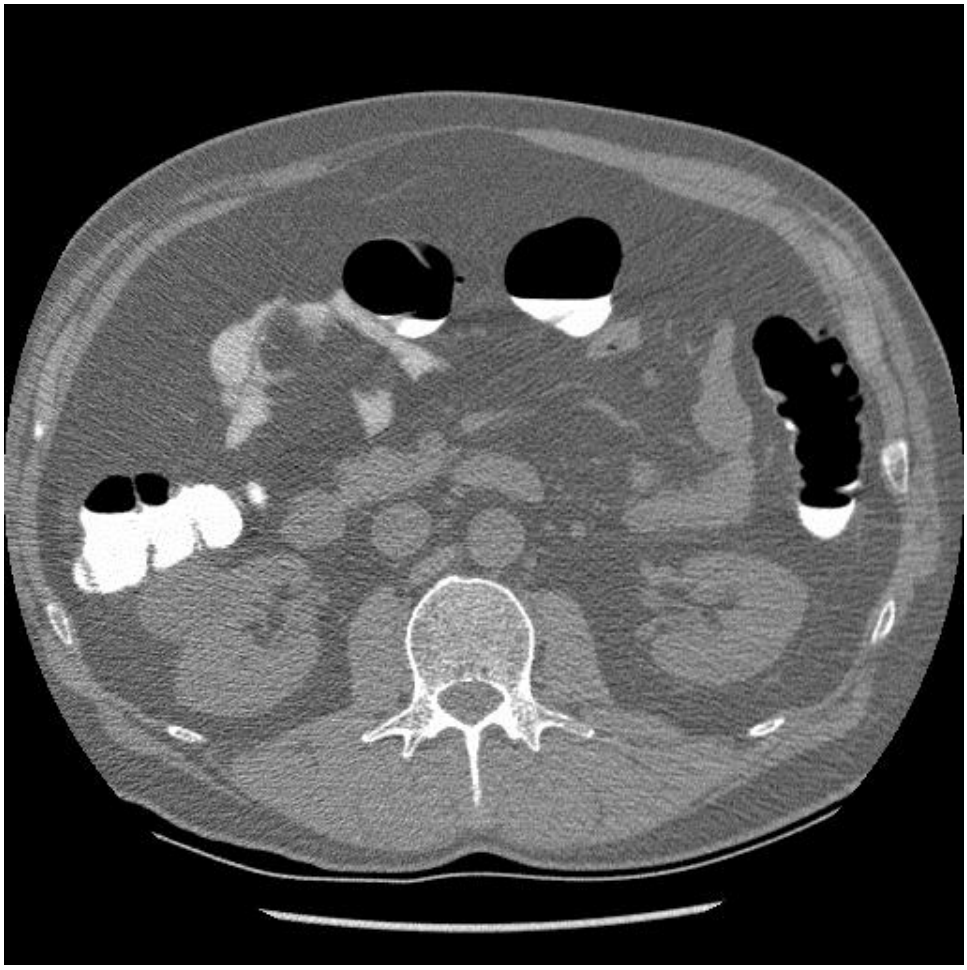

Figure 9: A CT scan slice with luminal fluid (white) in the large intestine regions.

3.2 DICOM Sorting and Conversion for the VC Interface

The input to the segmentation system used in the VC interface requires the images to be in portable graymap (PGM) format. Therefore, the DICOM scans need to be sorted, scaled to a PGM (0-255) scale, and converted to the PGM image format. The Matlab functions *dicominfo*, *dicomread*, *dicomwrite*, and *imwrite* allow the user to do this. The command *dicominfo* reads relevant information about a particular scan and can be used to

order a patient's scan slices according to slice order. Following sorting, *dicomread* and *dicomwrite* can be used to assign a DICOM to a variable and write it to an array respectively. The final step is to convert the array's Hounsfield scale (-1000 to 1000) to a typical grayscale (0 to 255) and use *imwrite* to make a PGM image of the rescaled array. The above protocol is done iteratively until all DICOMs have been sorted and converted.

3.3 Summary

Data acquisition for the CVIP Lab VC interface follows a similar prep procedure to other commercial VC methods and requires some manipulation of the raw DICOMs in order for the system to operate as desired. Concerning the colon prep procedure, laxatives are required for the current system to function properly, but luminal fluid tagging may not be necessary as described in section 3.1. Either a barium swallow or Gastrografin may be used, but one should suffice. Although Matlab is currently required for the sorting and conversion of the DICOMs, the conversion program can be put into a dynamic linked library (DLL) and made into an executable so that it can be utilized without a Matlab installation. Future work could be applied toward a completely self-contained C/C++ DICOM sorter and converter through the use of DCMTK (an open-source DICOM toolkit).

CHAPTER 4:

CVIP LAB VC INTERFACE: SEGMENTATION AND CENTERLINE EXTRACTION

4.1 Adaptive Level Sets Segmentation

4.1.1 Introduction

Proper visualization of the large intestine requires the colorectal tissue to be isolated from abdominal tissue and organs (i.e. small intestine, lungs, and liver). The segmentation of the large intestine is bimodal. By this it is meant that the colorectal tissue is considered foreground (logical true) while all other artifacts are considered background (logical false). In the CVIP Lab VC interface the segmentation of the large intestine is carried out via adaptive level sets after basic pixel threshold preprocessing.

The pixel threshold preprocessing is required to remove the remaining opacified luminal fluid in the large intestine [see figure 9]. The opacified fluid is typically brightly displayed in the scans (similar to bone) because of the barium and iodine ingested by the patient during the pre-scan prep period. Since the air in the colon is typically black and the surrounding tissue is mid-intensity brightness this fluid is removed via thresholding without difficulty [see figure 9].

4.1.2 Basic Level Sets

The level set method describes the shape of a curve in two-dimensional or three-dimensional space in terms of time. Level sets method is desirable for shape and curve analysis in two respects: it allows for the description of curves and shapes without the need to parameterize them, and it is ideal when describing shapes that have dynamic

topological structure (i.e. shapes that split into parts or develop holes). Basic level sets as it is applied in the VC interface can be described as follows.

Assume a curve, defined by Γ, that is set in the function Φ, which represents some higher dimensional space (three dimensions in this research). Given this, the curve may be defined by $\Gamma = \{X: \Phi(X) = 0\}$, which implies that the curve is the zero level of the function Φ ($X = \{x, y, z\}$ in this research). In order to describe the shape, the zero-level curve must be evolved with time through the higher dimensional space. Therefore, a time component must be added to Φ which describes the changing curve front. The curve front evolution function can now be described as $\phi = \phi(X, t)$ (note that a lower case Phi is used to denote the time component). Since the curve boundary (front) represents the zero level of the function ϕ it may be written as:

$$\phi(X, t) = 0. \tag{1}$$

The basic level sets function, as described by Osher and Sethian, is defined by the following [23].

$$\Phi_t + F \left| \nabla \Phi \right| = 0 \tag{2}$$

To describe Φ dynamically equation 2 can be written as the partial differential equation (PDE) given below.

$$\frac{\partial \phi(x,t)}{\partial t} + F\left|\nabla \phi(x,t)\right| = 0 \tag{3}$$

In discrete applications the evolving level set can be described by the following.

$$\phi(t + \Delta t) = \phi(t) - F\Delta t \left| \nabla \phi \right| \tag{4}$$

where, F describes the velocity field and $|\nabla\phi|$ is the Euclidean norm of the curve gradient. The velocity field, F, determines whether the curve shrinks ($F > 0$), grows ($F < 0$), or remains static ($F = 0$). In this application F is defined by the following.

$$F = \pm 1 - \varepsilon\kappa \tag{5}$$

where κ describes the curvature of the front and ε describes the bending of the front. The above formulation provides a foundation for adaptive level sets, which uses statistical methods to improve the accuracy of the evolving level sets front [17, 19, 23].

4.1.3 Adaptive Level Sets

Adaptive level sets supplements statistical parameters to the basic level sets formulation described above. In essence, it places statistical models in regions of the object being modeled via level sets. These statistical models are reevaluated after each evolution of the level set in order to more accurately represent the respective regions. Adaptive level sets as it is applied in the VC interface is given as follows.

As stated above, the model followed for segmentation of the large intestine is bimodal where large intestine is foreground and non-large intestine is background. Gaussian distributions are assumed for each region. In terms of probability this is a two-class problem where the class is defined as i, therefore $i = 1, 2$. The mean, variance, and prior probability are defined as μ_I, σ_i^2, and π_i respectively and given by the below equations.

$$\mu_i = \frac{\int H_\alpha(\phi_i) I(\mathrm{X}) dx}{\int H_\alpha(\phi_i) dx} \tag{6}$$

$$\sigma_i^2 = \frac{\int H_\alpha(\phi_i)(\mu_i - I(x))^2 dx}{\int H_\alpha(\phi_i) dx} \tag{7}$$

$$\pi_i = \frac{\int H_\alpha(\phi_i) dx}{\sum_{i=1}^{2} \int H_\alpha(\phi_i) dx} \tag{8}$$

where, $x = \{x, y, z\}$ or a voxel in the CT scan image space, $H_\alpha(\bullet)$ denotes the Heaviside step function used as a differential representation of the unit step function, and the statistical priors satisfy the condition $\sum_{i=1}^{2} \pi_i = 1$. These parameters are updated after each evolution of the level set front and used in the classification model given below.

$$i^*(x) = \arg\left(\max_{i=1,2}(\pi_i p_i(I(x)))\right) \tag{9}$$

where $I(x)$ is the image data in the $x = \{x, y, z\}$ domain and $p_i(\bullet)$ is the probability density function. Equation 9 uses Bayesian criteria to determine if a particular voxel x is included in the foreground (large intestine) or background (non-large intestine). Specifically, if the voxel x belongs to the class $i = i^*(x)$ then the front will propagate.

Using the classification criteria described above the final level sets function can be defined by putting the level sets PDE (equation 3) in general form with the derivative of the Heaviside step function, given by $\delta_\alpha(z)$. This yields the following equation defined for a particular class i.

$$\frac{\partial \phi_i(x,t)}{\partial t} = \partial_\alpha(\phi_i(x,t))(\varepsilon \cdot \kappa \pm 1)|\nabla \phi_i(x)| \tag{10}$$

In this equation, the Heaviside step function derivative, $\delta_\alpha(z)$ is used to select a narrow band of points about the curve front for criteria checking. This narrow banding restricts

the number of iterations required to propagate the front since fewer points are considered. This method is acceptable since only points at the front need to be considered for curve propagation [24].

4.1.4 VC Interface Segmentation Procedure

The CVIP Lab VC segmentation interface utilizes the adaptive level sets process described above along with user input to segment the large intestine. The interface requires the user to seed within the large intestine regions at various locations in the CT scan. More seeds provide accurate segmentation but typically no more than 40 seeds are required for an entire scan (~450-550). Ideally, seeds should be placed every 30 to 40 slices, particularly in scan regions where there are multiple large intestine artifacts on a single transverse CT slice. In conjunction with the seed points the user can define the radius of the seeds, the curvature coefficient κ (used in equation 5), and the number of iterations (these are all saved to a text file excepting the number of iterations which is passed directly). The seed radius and curvature coefficient are determined for every seed selected. The radius is three and the curvature coefficient is zero by default as these were the most ideal values for accurate segmentation. The number of iterations should be 100+ for ideal segmentation although as little as 75 iterations may be acceptable [17].

After seeding the segmentation button can be selected. Upon execution the program reads the text file where the seed data was written. This text file contains the x, y, and z position as well as the radius and curvature coefficient for every seed selected. These seed initializations as well as the number of iterations determined by the user are passed to the adaptive level sets segmentation software. Each seed is propagated as an

independent evolving front in three-dimensional space. As stated in 4.1.3, for each iterative step equations 6, 7, and 8 are evaluated to determine the extent of expansion for each seed point via the Bayesian classifier of equation 9 and the general adaptive level sets formulation of equation 10. Figure 10 displays the original CT slice (after thresholding) and the segmented result after the segmentation procedure.

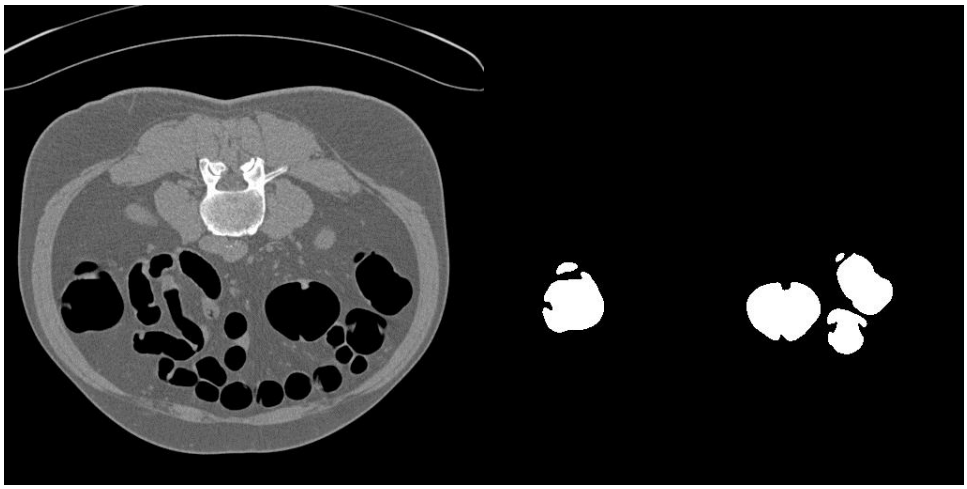

Figure 10: The original CT slice after thresholding (left), and the CT slice after the adaptive level sets segmentation procedure (right).

4.2 Centerline Extraction using Curve Skeletons Method

4.2.1 Introduction

Although it is possible to model the large intestine three dimensionally without the extraction of the centerline, it would serve little use to a medical practitioner. The centerline provides a graphically viable path for the virtual camera to follow within the large intestine. An object centerline can mimic the path of a colonoscope traveling through the large intestine much as it would in a traditional colonoscopy. In the CVIP Lab VC interface this task is accomplished using the curve skeleton extraction software

developed by M. Sabry Hassouna and described in his publication *On the Extraction of Curve Skeletons using Gradient Vector Flow* [21].

4.2.2 Curve Skeletons Defined

A curve skeleton for a three-dimensional object (such as the large intestine) can be defined as a sequence of paths between two voxels. Therefore, centerline extraction can be defined, in the simplest case, as a minimum-cost path problem between two points in three-dimensional space (for this circumstance). Assume that the path can be defined by $\gamma(s) : [0,\infty) \rightarrow R^3$ and it must be minimized between the points *A* and *B*. Then the minimum cost function can be defined by the following equation.

$$T(x) = \min_{\gamma Ax} \int_A^x u(\gamma(s))ds \tag{11}$$

where *x* is some point in the three-dimensional space R^3, $u(\bullet)$ is the cost function, and the set γAx is all the paths connecting *A* to the point *x* [21]. It can be seen that the minimum integral of the cost function will yield the path with the lowest cost. It is known from geometrical optics that the Eikonal equation is satisfied by equation 11 with the following formulation.

$$|\nabla T(x)|F(x) = 1 \tag{12}$$

where $F(x)$ is the (optical/energy) wave speed. The wave speed can be derived by the inverse of the cost function given by.

$$F(x) = \frac{1}{u(x)} \tag{13}$$

For the path to be shortest then this wave speed (*F(x)*) must be maximized. A thorough proof of the above assertion is beyond the scope of this paper as the VC interface only utilizes its implementation. A complete proof of the curve skeleton theorem can be found in M. Sabry Hassouna's publication given above in section 2.1 [21].

4.2.3 Determining Object Medialness via Gradient Vector Flow

The medialness of an object can be understood as its centeredness. Recall that the curve skeleton should graphically describe an object in simplest terms (one dimensional lines). Therefore, approximating the centeredness of an object can facilitate the extraction of the curve skeleton. In the VC interface, the medialness of the large intestine is approximated by gradient vector flow (GVF) in conjunction with a medialness parameter defined in M. Sabry Hassouna's publication [21].

GVF uses a vector field, defined by *V(x)*, to minimize the functional defined as follows.

$$E(V) = \iiint \mu|\nabla V|^2 + |\nabla f|^2 |V - \nabla f|^2 \, dx \tag{14}$$

where $x = (x, y, z)$ (a voxel), μ is a regularization parameter, and *f(x)* is an edge map extracted from the image space *I(x)* (a CT scan volume in the case of the large intestine). The vector field flows from the boundary of the object towards its center. Near the boundary the field flows slowly while toward the object center it flows more quickly. As a result, the magnitude of *V* is large near the object's center and small near the object boundary. The following medialness parameter constrains the GVF to a narrow region near the center of the object [21].

$$\lambda(x) = 1.0 - \left(\frac{|V(x) - \min|V||}{\max|V| - \min|V|} \right)^{q}, 0 < q < 1 \tag{15}$$

For a thorough definition of the medialness parameter, λ, as it relates to curve skeletons please refer to M. Sabry Hassouna's publication [21]. As stated above, by using equation 15 the computation of the GVF will be constrained to the medial regions of the object thereby reducing the time needed to compute the result.

4.2.4 Extraction of Centerline

The final extraction of the object's curve skeleton(s) can be defined for two cases: one in which there is only one curve skeleton (ideally the case for the large intestine), and a case where there are multiple curve skeletons that need to be related to each other (a general object). The multiple curve skeleton case will not be reviewed in this paper as it does not relate to the extraction of the colon centerline, but it can be found in the references [21]. For the single curve skeleton case (a large intestine centerline), the set of voxels (nodes) approximated about the convergence of the vector field (through GVF) can be used in conjunction with the following ordinary differential equation (ODE) to find the centerline.

$$\frac{dP(s)}{ds} = -\frac{\nabla T}{|\nabla T|}, P(0) = P_0 \tag{16}$$

where P represents the voxels (or nodes) in question. This equation expresses the fast marching method for determining which voxel is nearest, in terms of travel time, to the propagating wave front, F. The gradient of the minimum cost function, ∇T, represents the normal to the wave front, which is the direction yielding the fastest travel time.

Therefore, any voxel in the direction of this unit normal will be the next candidate in the object centerline.

4.2.5 VC Interface Centerline Extraction Procedure

The curve skeletons code is entirely automated within the VC interface system. The input to the system is the segmented images from the segmentation procedure. These input images are in PGM format. The output from the curve skeletons code is a series of text files that include run times and most importantly the location of the large intestine centerline. This centerline text file is essential for the visualization routine, which will be discussed in the following chapter. The centerline is required for the colon splitting procedure that creates the flyover view as well as the camera path which allows for internal visualization of the colon [see figure 11].

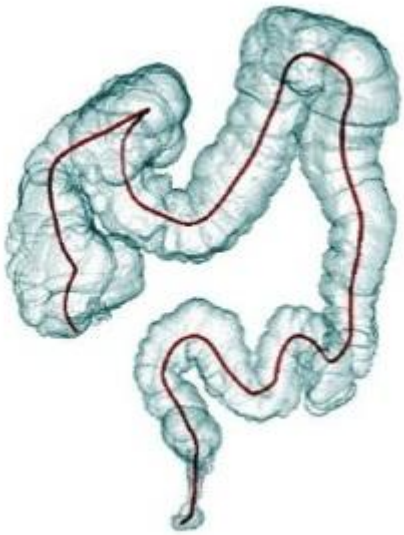

Figure 11: A solid model of the colon with the centerline shown in red.

4.3 Summary

The accurate segmentation and centerline extraction of the large intestine is essential for the VC interface to visualize the colorectal interior. The methods described above are both robust and accurate, but the segmentation/centerline extraction process does have some limitations. The segmentation method, though accurate, is restricted by the manual seeding requirement. The user must be able to differentiate between large intestine and small intestine (both of which are filled with air and therefore are dark on the scans). Additionally, the manual seeding process can be time consuming (times will be provided in Chapter 6). The GVF curve skeletons software is extremely well suited to determine the skeleton graph of a complex object but it is hindered in two ways: inability to extract the centerline for a large intestine with collapsed portions (i.e. non-distended), and potential for run-time failures because of high memory consumption. The CVIP Lab has been pursuing various methods for automatic colorectal segmentation therefore such an automatic system may be a future addition to the interface. As for centerline extraction, it may be more efficient to extract the centroids of the segmented large intestine slice-by-slice and connect them with some tensor linking technique. Ultimately, the interface is structured in such a way that future modifications can be added as desired to test functionality and efficiency.

CHAPTER 5:

CVIP LAB VC INTERFACE: VISUALIZATION

5.1 Introduction

The Visualization Toolkit (VTK) is an open-source visualization, graphics, and image-processing library. It is used in the fields of mechanical design and drafting, testing and simulation, and medicine (as with this research). In the case of the CVIP VC interface it is used for the visualization of a large intestine solid model, traditional flythrough visualization, and the CVIP Lab patented Flyover visualization. In addition to these mapped and rendered three-dimensional models the interface also displays the transverse, coronal, and sagittal views of a particular CT scan slice (this is also done with the VTK library) [see figure 12]. The VTK library is the hub of the CVIP Lab VC interface's visualization capabilities.

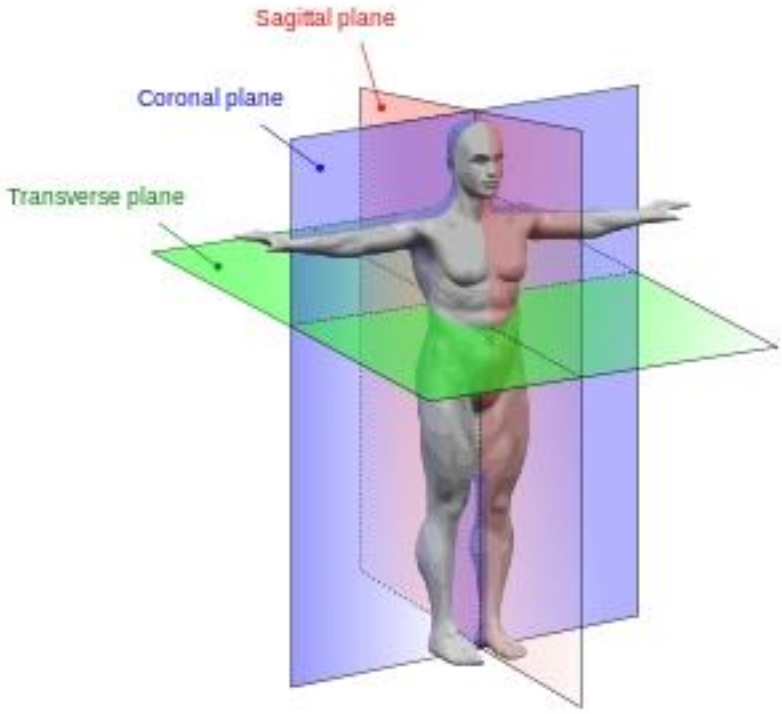

Figure 12: The anatomical planes for a human body (these also correspond to the VC interface's visualized CT planes) [27].

5.2 General Visualization Protocol Using VTK

VTK works in a pipeline fashion (i.e. the output of one process is the input of another). The general makeup of a VTK pipeline is as follows: *i*) input data sources (PGM, PPM, generated files, etc.), *ii*) filter input data (step is optional, but it usually involves data reduction, smoothing, conversion, etc.), *iii*) map data (converts the data input into a VTK object that can be visualized), *iv*) convert mapped data to a VTK actor (allows for the adjustment of object color, opacity, etc.), *v*) set actors to renderers and render windows (displays objects in windows and allows for user interactions) [see figure 13]. As stated in step *v* above, controls and interface options can be added to render

windows to allow the user to make selections with a mouse or use hotkeys. This is used extensively in the CVIP Lab's VC interface.

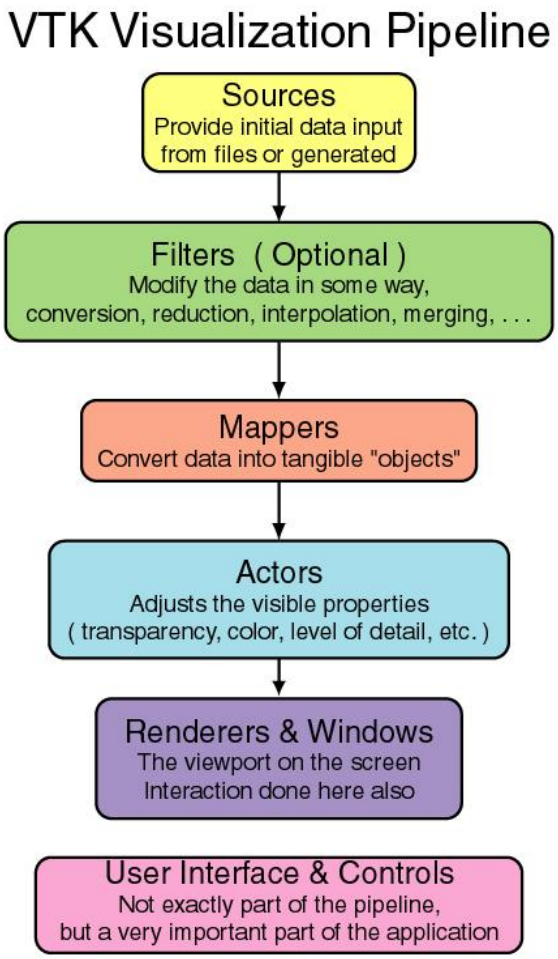

Figure 13: The VTK pipeline [25].

5.3 Transverse, Coronal, and Sagittal CT Slice Visualization

The VC interface displays the three planes of a CT scan in order to enhance the user's ability to diagnose an anomaly detected in the rendered visualizations. Radiologists are all familiar with colorectal CT scans but the same cannot be said of rendered colorectal visualizations including: flythrough, flattening, or flyover. Therefore, the original CT images provide a benchmark for comparison. Fortunately, VTK includes

libraries that can take a CT scan input (in DICOM, BMP, or PGM format) and display it in the three anatomical planes (transverse, coronal, and sagittal) [see figure 12].

In the VC interface the original images are PGM formats of the original CT scan images (as stated in section 3.2). These original PGMs for a single scan are read sequentially using the VTK library *vtkPNMReader*. Within this library the x, y, and z extent of the scan volume is defined. The CT image sets used to validate the CVIP Lab VC interface are 512 by 512 on the x-y plane, and of varying z dimensions (400~550) depending on patient height. In addition to image dimensions the library is also used to define the range of values that are the intensities of the PGM images. After reading the scan volume the images are filtered via *vtkImageCast* and *vtkImageMapToWindowLevelColors* respectively. These filtering libraries format the images for proper viewing in a VTK render window by scaling intensity values (see VTK pipeline [figure 13]).

Prior to converting the scan into a working VTK actor it must be formatted to display the scan in terms of the three anatomical planes. This is accomplished through the use of the libraries *vtkImageStencil* and *vtkImagePlaneWidget*. The latter library (*vtkImageStencil*) combines the various images (slice-by-slice) in a relatable fashion by a "cookie-cutter operation" as defined in VTK documentation, while the former library (*vtkImagePlaneWidget*) allows for the access of a particular slice on the remapped scan (i.e. remapped to the transverse, coronal, and sagittal planes) [26]. Following the above formatting steps the newly defined PGM image planes can be made into two-dimensional image actors using *vtkImageActor* and the actors can be set to a VTK renderer and

rendering window using *vtkRenderer* and *vtkRenderWindow* respectively. The windows made by *vtkRenderWindow* are associated with window controlling and interacting tools as well, but these will be covered in section 5.6.

5.4 Colorectal Solid Model Visualization

Because the local topologies of the large intestine vary greatly from patient to patient a rendered solid model of the organ can provide valuable information to a medical professional performing a VC. For instance, a thick tissue wall in the colon or rectum can indicate cancer without the existence of a polyp (this is called non-polypoid colorectal neoplasm). A colorectal solid model, in conjunction with the CT slice visualizations, can assist a medical professional in determining if a colorectal wall thickening is the result of abnormality or part of a patient's unique anatomy. Using VTK along with the segmented colon images a solid model of the colon can be visualized and viewed with little data manipulation.

As with the CT slice visualizations above the images must be read using *vtkPNMReader*, except these images are the binary segmentations of the large intestine, and formatted for size (x and y) and depth (number of slices or z). Following reading in the image data the images need to be filtered (smoothed) so that following steps can create an isosurface and a triangular mesh for visualization. Smoothing the image data is done using the *vtkImageGaussianSmooth* class, which, as its name implies, uses Gaussian smoothing to create ideal edges for the creation of an isosurface. After the filtering procedure the images are ideal for creating the isosurface. An image isosurface is simply a surface of constant values that are established in three-dimensional space. The

isosurface is created using *vtkMarchingCubes*. This class uses the marching cubes method first defined by William E. Lorensen and Harvey E. Cline in *Marching Cubes: A high resolution 3D surface construction algorithm* published in *Computer Graphics* (1987).

The output from the above process is points defining the large intestine surface along with any properly formed triangulations between these points [26]. Three filtering steps are required after the above isosurface creation, one to remove excess or poorly placed isosurface points, another to reduce the number of surface triangles (as there are often more than necessary to render the large intestine surface), and a final step to adjust the remaining points using interpolation. Junk isosurface points are merged or removed by way of the *vtkCleanPolyData* class, and redundant surface triangles are removed via *vtkDecimatePro* [26]. The final step is carried out using the *vtkWindowedSincPolyData* class. This class uses a windowed sinc function interpolation kernel to adjust points in an ideal fashion so that the surface triangles provide the best visualization (see VTK documentation for greater detail regarding the *vtkWindowedSincPolyData* class [26]).

The final three steps in the visualization pipeline for the large intestine model are: the generation of the normals for the large intestine polygon mesh, the mapping of the normals to create a VTK object, and the creating of a VTK actor using the mapped VTK object. The creation of the large intestine object normals is done with the class *vtkPolyDataNormals*. These normals are defined for all of the triangular vertices of the isosurface. Mapping of these normals into a VTK object is done using *vtkDataSetMapper*. This mapper operates in a similar manner as the mapper used for the

CT scan slices except its input is computed data rather than pixel maps. Finally, the mapped VTK object is made into a VTK actor using *vtkActor* and set to a renderer and render window as described in section 5.2 using *vtkRenderer* and *vtkRenderWindow*.

5.5 Colorectal Flythrough Visualization

The process for creating the VC flythrough visualization is very similar to that done for the large intestine solid model visualization described in section 5.3. For the flythrough visualization the output from *vtkPolyDataNormals* is used in the VTK class *vtkPolyDataMapper*. The class *vtkPolyDataMapper* is similar to the class *vtkDataSetMapper* used for solid model visualization except for the way in which the data is handled. In *vtkPolyDataMapper* the data are known to be defined vertices, lines, or polygons, while in *vtkDataSetMapper* the data are represented as raw data values. Following data mapping a visuals lookup table is applied to the mapped surface. This is done through the class *vtkLookupTable*. A lookup table simply applies a red, green, blue, and alpha transparency (RGBA) to the object to be visualized. Using this class the user can define the appearance of the flythrough visualization and set it to resemble the interior of the large intestine. The mapped flythrough object is turned into a VTK actor, set to a VTK renderer, and placed in a render window as with all VTK visualizations.

5.6 Colorectal Flyover Visualization

The flyover visualization is a CVIP Lab patented method for viewing the large intestine. Essentially, it splits the colon into halves which are viewed from overhead [see figure 14].

Figure 14: The colorectal flyover views in which one half is colored blue and the other half is colored red.

This visualization lacks the hidden regions of the flythrough, while maintaining a topology that is intuitively interpreted by a medical professional. Though VTK is essential in the final visualization of the flyover it does not provide all of the tools necessary to split a solid object into corresponding halves. The flyover visualization is created by the following protocol:

1) The normals for the large intestine solid model are generated as they were in section 5.3.
2) The centerline points as well as distance from centerline boundary (DFB) values (added to the centerline point data in a small subroutine) are read into the classes *vtkPoints* (holds three-dimensional points for manipulation and visualization) and *vtkKochanekSpline* (used to compute interpolating splines using Kochanek basis [26].
3) The x, y, and z centerline points are interpolated using *vtkKochanekSpline* and written to a text file.
4) The text file of interpolated centerline points are read and used to determine normals for each centerline position.
5) Another centerline is created for the other large intestine half using rigid translations.
6) The normals are determined for the new centerline.
7) The large intestine is broken into segments (each segment corresponding to a centerline(s) point) so that it can be split into halves part-by-part, and the polygon data (determined in step 1) for each segment is assigned to a C array of *vtkCellArray*s (this class stores vertex, line, or polygon data).
8) The polygons stored in the array of *vtkCellArray*s are matched to their corresponding (i.e. closest) centerlines via Euclidean distance.
9) The colon is split using the class *vtkPolyDataClipper* for each large intestine segment with a finite implicit plane (created through the class *vtkPlane*).

10) The polygons for each corresponding half (left and right) are stored in their own containers via *vtkAppendPolyData*.
11) Finally, the polydata is filtered using *vtkCleanPolyData*, the normals are computed with *vtkPolyDataNormals*, and the normals are written to VTK visualization files with *vtkPolyDataWriter*.

The above procedure creates two files (Left.vtk and Right.vtk) that are later read by the QT/VTK main interface thread and displayed on the QVTK window.

5.7 Window Controlling and Interacting

Window controlling is a feature in VTK that allows a user to interact and control a VTK window. VTK window controllers and interacting tools can allow the user to: manipulate object size through a mouse wheel, move an object with the mouse, annotate points on an object/image, use hot keys, as well as various other interactive features. VTK window interactions can be done through *vtkRenderWindowInteractor*, *vtkGenericRenderWindowInteractor*, and *vtkStyleInteractorImage*. The first class (*vtkRenderWindowInteractor*) is the general window interactor used for most VTK objects. It allows for object annotation, movement, and other basic object manipulations. The second class (*vtkGenericRenderWindowInteractor*) is more versatile than the first class in that the user can program interaction events such as special mouse usage or hotkeys. The final class (*vtkStyleInteractorImage*) is used in conjunction with *vtkGenericRenderWindowInteractor* to associate user events with uniquely programmed observers. These observers are typically classes or functions (as is the case in the CVIP VC interface). All of the above VTK window interaction classes are used in the CVIP VC

interface. VTK interactors offer a wide array of tools beyond the scope of this thesis' analysis, but further information can be found in VTK web documentation [26]. Section 6.1 (system design) will go into further detail as to the use of interactors in the CVIP VC interface.

5.8 Summary

The preceding sections describe all of the components of the CVIP Lab VC interface. It has been shown that VTK provides a robust open-source library for visualization, and its application to the VC interface broadened the system's capabilities. In the following section (6.1) on overview of the system design will be given including functionality, and layout.

CHAPTER 6:

CVIP LAB VC INTERFACE: SYSTEM DESIGN AND VALIDATION

6.1 System Design

6.1.1 Introduction

The CVIP VC interface was designed using Qt (specifically Qt Designer), a cross platform user interface (UI) framework, in conjunction with VTK. The Qt platform contains a wide array of tools that can be applied to any UI design. Qt Designer provides a platform to design an UI visually (i.e. without the need to program it from scratch). The file created by Qt Designer is a UI file. This must be compiled in CMake with a corresponding header file to generate a UIC (user interface compiler) file, and an MOC (meta-object compiler) file. The UIC file is a compiled combination of the user programmed parameters (such as signals and slots) in the header file and the codified geometries and parameters that were created in Qt Designer. This UIC file contains the class declaration and methods/variables for the entire UI system.

The CVIP VC interface was designed as a tabbed window with four tabs named as follows: Flyover (displays the flyover visualization and contains primary push buttons, and driver controls), CT (displays the sagittal, transverse, and coronal CT slices from left to right accordingly), Extra (displays the large intestine solid model with centerline (blue dots), and camera position (red dot) as well as the flythrough for comparison, and Light/Camera (light and camera control tools). A tabbed interface was selected because of its high mobility between platforms. Although a multi-window interface may be desirable with multiple monitors it would be cluttered and unusable in a single monitor

setting. This interface design provides the flexibility to quickly cycle between CT visualization and flyover/flythrough visualization without the need for multiple screens. Each tab of the interface as well as the segmentation pop-up window will be described in the sections below.

6.1.2 Pop-Up Window: Segmentation Interface

The segmentation interface is a separate pop-up window that displays the CT scan slice-by-slice, and allows the user to: seed the scan for segmentation, cycle through the slices via pushbuttons (*Next* and *Previous*), change position (text edits *x*, *y*, and *z*), seed radius (text edit *r*), or seed curvature parameters (text edit *Smoothness Coefficient*), determine the number of segmentation iterations (text edit *Number of Iterations*), segment after seeding (*Segment*), and close the segmentation window (*Close*). During seeding the seed points are written to a text file. When the segmentation pushbutton is selected the seed point text file is read into a queue and passed into the segmentation protocol. When completed, the segmentation protocol saves the segmented binary images as PGMs and returns function to the segmentation interface. Upon closure of the segmentation window the centerline extraction process begins automatically using the segmented PGMs as input.

Figure 15: A screenshot of the segmentation interface.

6.1.3 Main Interface: Flyover Tab

The flyover tab contains all of the primary pushbuttons and driver controls (used to move through the large intestine). A radio button (*Preprocessed CT Scan*) gives the user the option to either run a set through the entire CVIP VC interface protocol (i.e. segmentation, centerline extraction, and visualization), or visualize a preprocessed scan directly (i.e. segmentation and centerline extraction are already completed). If the radio button is selected the user must load the original PGM format CT scan with the *Load Dataset* pushbutton, and the segmented images as well as the centerline data is assumed to be in the working directory (as it would be under normal operation). With the radio

button selected the *Segment* pushbutton will remain disabled. If the radio button is not selected the user must select the original PGM scan and this will enable the *Segment* button. Pressing the *Segment* button opens the segmentation pop-up window described in section 6.1.2. Upon the completion of segmentation and centerline extraction the *Visualize* button will be enabled. Pressing this button begins the entire visualization routine (described in chapter 5). Upon the completion of the visualization routine all of the driver, light, and camera controls will be enabled (this includes the driver controls *Stop*, *Reverse*, *Forward*, and the speed slide-bar between the *Reverse* and *Forward* pushbutton). The other interface options available for this tab, such as hot-keys or window interactions, will be described in section 6.1.7.

6.1.4 Main Interface: CT Tab

This tab displays the CT slices corresponding to the position of the camera on the large intestine centerline. The CT slices are displayed (from left to right) sagittal, transverse, and coronal. This tab is made available so that a medical professional may have another reference beside the flyover visualization, flythrough visualization, or large intestine solid model. The other interface options available for this tab, such as hot-keys or window interactions, will be described in section 6.1.7.

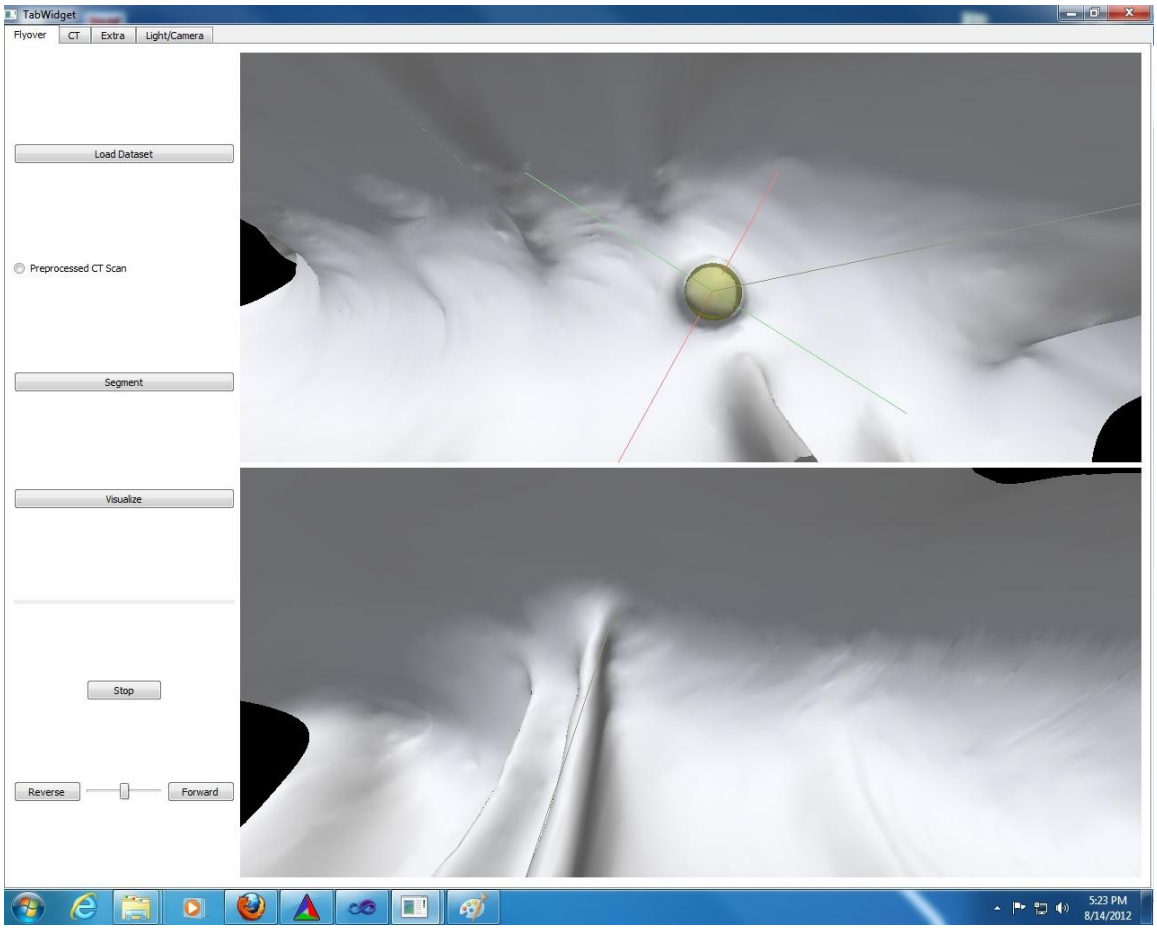

Figure 16: A screenshot of the flyover interface (note the marker glyph, colored yellow, on the top flyover view).

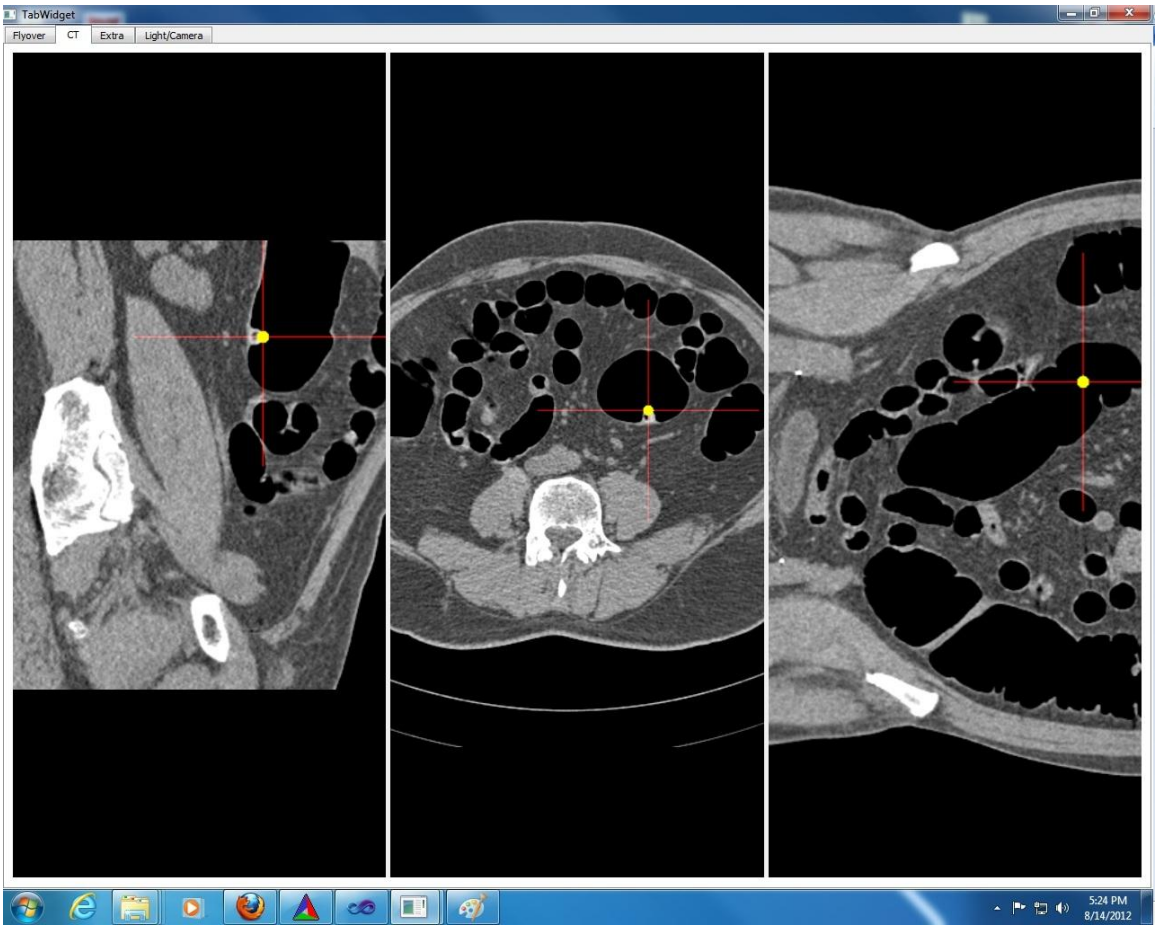

Figure 17: A screenshot of the CT interface (note the marker glyph, colored yellow, from the flyover view shows up on the corresponding anatomical area in the CT slices).

6.1.5 Main Interface: Extra Tab

This tab displays the large intestine solid model with the centerline (as blue dots) and the camera position (as red dots), and the flythrough visualization. As with the CT display this tab is for additional reference and comparison. The other interface options available for this tab, such as hot-keys or window interactions, will be described in section 6.1.7.

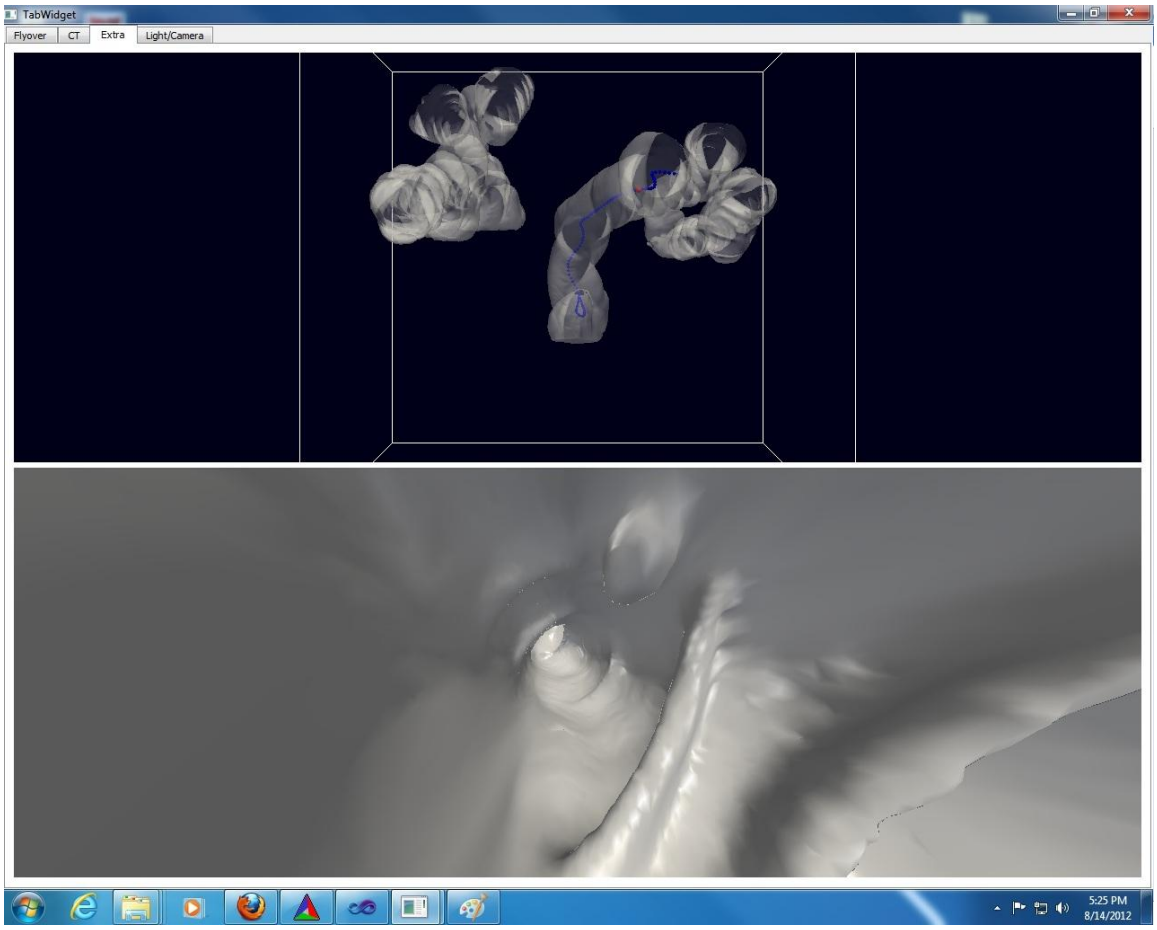

Figure 18: A screenshot of the solid model of the large intestine (top) with the centerline as blue dots and the camera position as a red dot, and the flythrough (bottom).

6.1.6 Main Interface: Light/Camera Tab

This tab contains all of the camera and light controls available in VTK, and these controls only apply to the flyover visualization. Though not an essential tool in the interface, it provides additional flexibility and control not available in other VC systems. In addition to the camera and lighting adjustment slide bars there are also options to: reverse the camera orientation through the radio button *Forward/Backward*, turn the camera stereo on and off with the pushbutton *Stereo On/Off*, and reset the light values

with *Reset To Defaults*. The camera's x, y, z, roll, elevation, and azimuth are also displayed for reference.

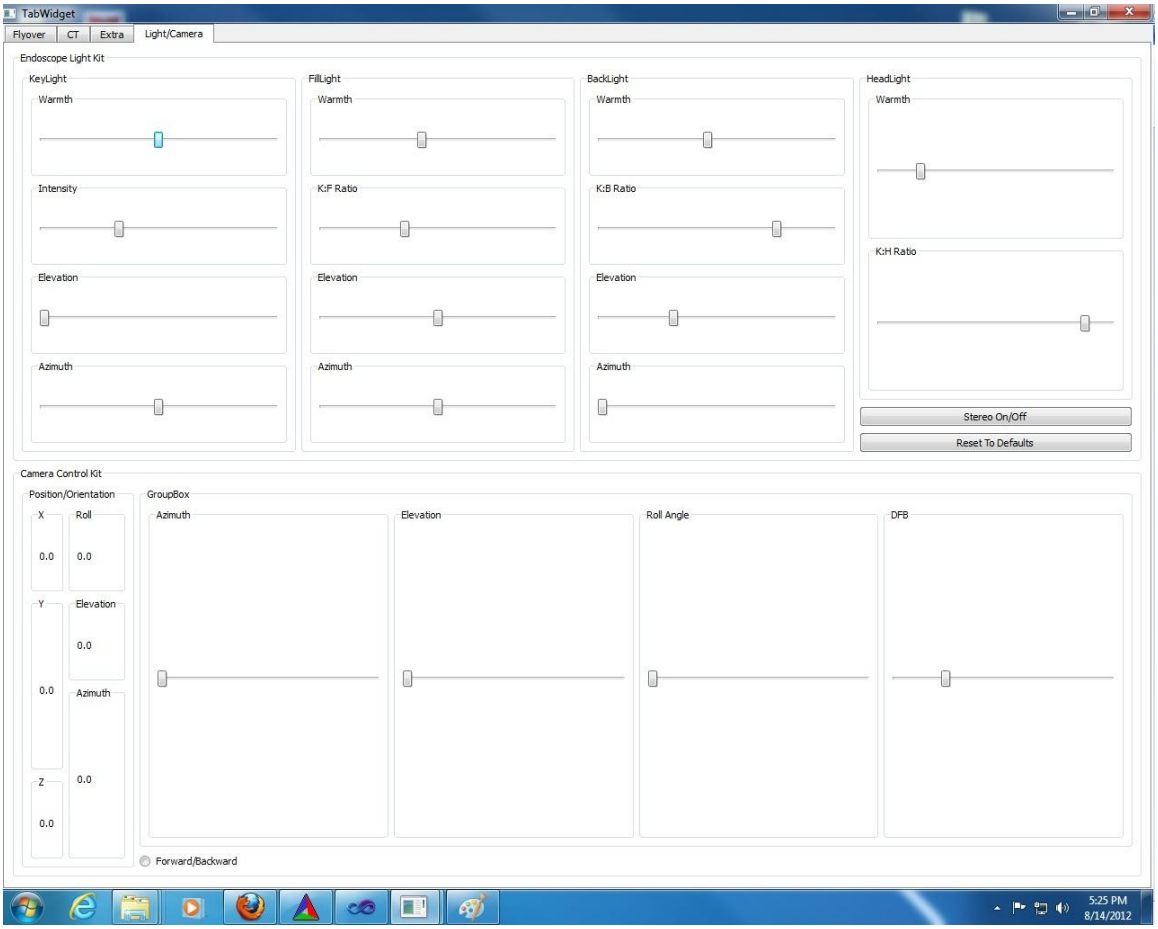

Figure 19: A screenshot of the camera and lighting controls.

6.1.7 Main Interface: User Interactions and Window Controls

There are various hot-keys and mouse controls available to the user when operating the CVIP Lab VC interface. For the Flyover tab the user is able to place a glyph (yellow sphere) on an object of interest by moving the mouse pointer over the object and pressing the "P" key. The glyph will remain until the user presses the "U" key. Glyphs can be added and removed as desired using the above procedure and more than one glyph can exist at a time. The user may also cycle through existing glyphs by pressing the "+"

or "-" key. This advances through glyphs in a forward and backward manner respectively. Since the glyphs appear on the CT tab as well as the Flyover tab these controls work the same for both. In the Flyover tab, the user may also manipulate the distance from centerline boundary (DFB) with the mouse wheel and reorient the colon halves with mouse dragging. Correspondingly, the user can direct red crosshairs with a left mouse click on the CT tab. In the Extra tab the user can manipulate the orientation of the large intestine solid model with mouse drags and zoom in and out of the model with the mouse wheel. In addition, the user can direct the camera view in the flythrough view with mouse drags and move forward and backward along the camera path with the mouse wheel. These window controllers are intended to assist in any diagnostics using the CVIP Lab VC interface. Future additions to the system can include other hot-keys and view controllers.

6.2 Validation

Validation of the system began with the assumption of a continuous pipeline as described in section 1.3. The computer used for initial compilation and testing contained an Intel® Xeon® Processor (W3565 quad-core 8MB Cache, 3.20 GHz, 4.80 GT/s Intel® QPI) and 24 GB RAM. A CT scan converted to PGMs is input to the system. The PGM inputs are segmented, the centerline is extracted, the large intestine is split, and the result is visualized. The largest CT scan (541 slices) was the first used to test the system's capabilities. The segmentation procedure completed successfully, but the centerline extraction experienced a run-time failure because of the inability to create new memory on the free store, specifically, during the creation of an object array of size 512x512x541

(this is 141819904 x (4 bytes-per-integer) x (32 bytes-per-object) = ~18 GB). The curve skeletons code used to extract the centerline of the large intestine was originally intended to run as a standalone executable. Even in this form it has high memory consumption as multiple variables are created on the free store in addition to the one that throws an exception in the VC interface. When the curve skeletons code was included with the VC interface it also had to compete with the memory consumed by the creation of the UI and VTK variables, which both contain large memory volume classes and variables. To test the threshold of the system the slice count was gradually reduced. At 400 slices the system pipeline worked successfully but the isosurface for the large intestine solid model did not visualize successfully (i.e. the object mesh could not be generated because of polygon discontinuities). Therefore, the slice number was reduced incrementally to 350 which led to a successful solid model. The results are provided in table one, and segmentation results can be seen in figure 20.

Table 1: Run times for the successful run through.

Set 1 (350 slices)	Run Time (seconds)
Segmentation	232
Centerline Extraction	23
Visualization	18
Total Time	273 (4.55 minutes)

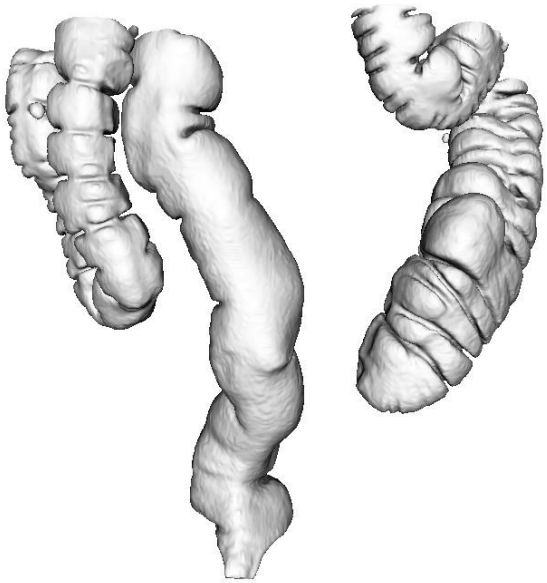

Figure 20: The segmentation results from the successful full run through.

Although the results of the initial trials were disappointing some validation of whole CT scans was still desired. In order to accomplish proper validation, the centerline extraction step was done externally from the VC interface using an executable so that there would be no competition for memory resources. The time of execution for external centerline extraction was treated as if it were within the system (as in-system run-times were similar for fewer CT slices). Before beginning this process a segmentation standard had to be used for all available sets. In order to determine that standard six trials were conducted. For the first trial a single seed was placed in each large intestine object every

50 slices. This trial failed to segment, therefore it was repeated (trial two) except seeding was done every 30 slices, but this failed as well. For trials three and four the number of seeds per large intestine object was increased to three, and they were placed every 50 (trial three) and 30 (trial four) slices. These trials resulted in successful segmentation. The final two trials used heavy seeding (greater than 10) per large intestine object every 50 (trial five) and 30 (trial six) slices. These two trials were done to compare the segmentation results to trials three and four. There was no discernible difference between the two sets. Therefore, it was assumed that three seeds per large intestine object every 50 slices (trial three) would be an adequate segmentation method. Refer to the table and figures below for further elaboration.

Table 2: Runtimes for the segmentation trials.

Segmentation Method Trials	Run Time (seconds)
Trial Three	515 (8.58 minutes)
Trial Four	605 (10.08 minutes)
Trial Five	993 (16.55 minutes)
Trial Six	1088 (18.13 minutes)

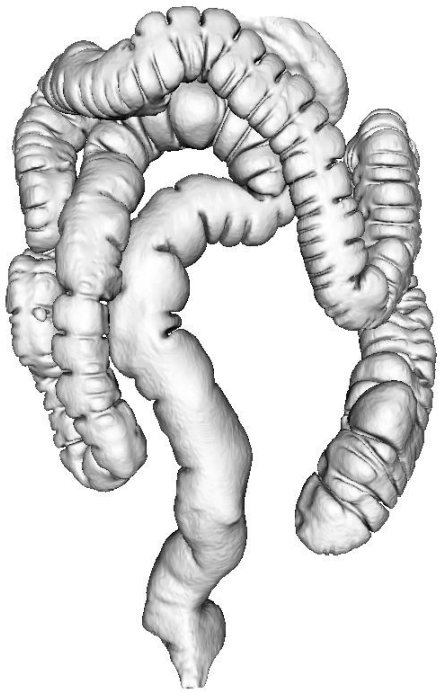

Figure 21: Segmentation trial six result from heavy seeding for every large intestine object every 30 slices (~1088 seconds or ~18.13 minutes for entire segmentation process).

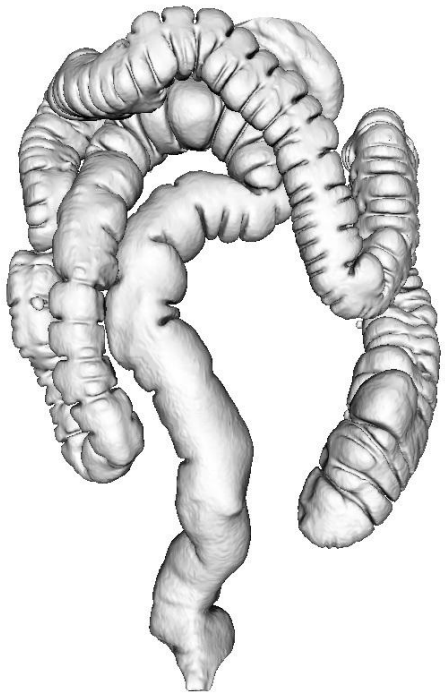

Figure 22: Segmentation trial five result from heavy seeding for every large intestine object every 50 slices (~993 seconds or ~16.55 minutes for entire segmentation process).

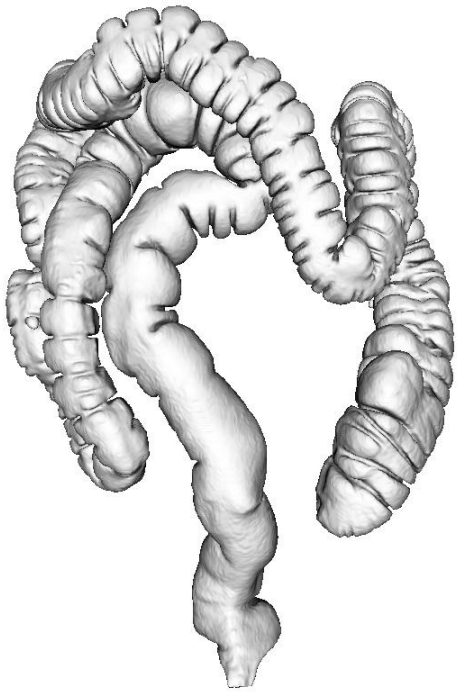

Figure 23: Segmentation trial four result from three seeds for every large intestine object every 30 slices (~605 seconds or ~10.08 minutes for entire segmentation process).

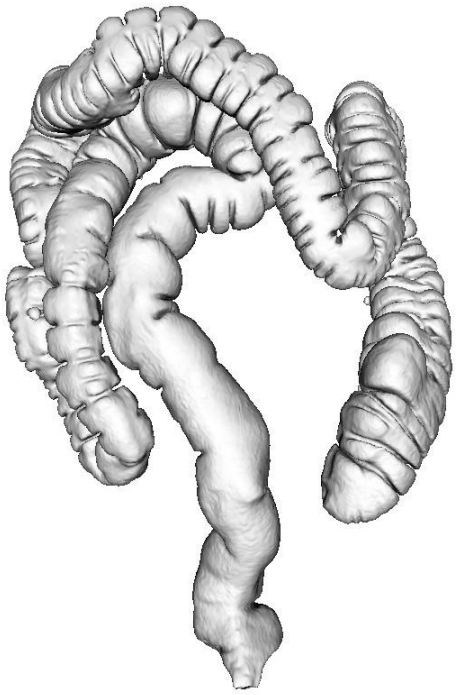

Figure 24: Segmentation trial three result from three seeds for every large intestine object every 50 slices (~515 seconds or ~8.58 minutes for entire segmentation process).

The segmentation procedure was carried out for each CT set as if it were going through the entire CVIP Lab VC pipeline. Upon completion, the sets were saved in the interface's working directory so that they could be read by the centerline extraction executable. The run time and number of seed for the segmentation of each CT scan is given in the table below.

Table 3: Runtime for segmentation along with the total seed count.

Segmentation	Run Time (seconds)	Number of Seeds
Set 1 (541 slices)	525	127
Set 2 (428 slices)	421	122
Set 3 (470 slices)	420	130
Set 4 (484 slices)	448	118
Set 5 (438 slices)	432	110
Set 6 (462 slices)	439	107
Set 7 (387 slices)	391	94
Set 8 (432 slices)	427	116
Set 9 (432 slices)	431	129
Set 10 (425 slices)	422	126

The large intestine centerline was extracted using an executable external to the VC system. Again, this was justified because the run-times for the external centerline extraction were identical to the run-times of the centerline extraction within the VC system for CT scans with fewer slices. Centerline extraction was done twice to compare the results. Refer to the table below for run-time results.

Table 4: Runtime for centerline extractions (two runs were done for comparison).

Centerline Extraction	Run 1 (seconds)	Run 2 (seconds)	Average Time (seconds)
Set 1 (541 slices)	55	51	53
Set 2 (428 slices)	28	28	28
Set 3 (470 slices)	31	31	31
Set 4 (484 slices)	36	36	36
Set 5 (438 slices)	20	19	19.5
Set 6 (462 slices)	49	49	49
Set 7 (387 slices)	48	48	48
Set 8 (432 slices)	28	28	28
Set 9 (432 slices)	30	30	30
Set 10 (425 slices)	30	30	30

Following the centerline extraction trials both centerlines for each set were used along with the segmentation results in the large intestine splitting and visualization routine (included in one step called visualization). The results are given in table five.

Table 5: Runtime for both centerlines extracted along with the total number of centerline points for each.

Visualization	Centerline 1	Centerline 2	Average Time (seconds)
Set 1 (541 slices)	84 seconds/5478 points	39 seconds/5478 points	61.5
Set 2 (428 slices)	58 seconds/3999 points	26 seconds/3998 points	42
Set 3 (470 slices)	56 seconds/3597 points	26 seconds/3597 points	41
Set 4 (484 slices)	79 seconds/5337 points	36 seconds/5337 points	57.5
Set 5 (438 slices)	58 seconds/3495 points	25 seconds/3495 points	41.5
Set 6 (462 slices)	57 seconds/4369 points	36 seconds/4369 points	46.5
Set 7 (387 slices)	64 seconds/4071 points	30 seconds/4071 points	47
Set 8 (432 slices)	49 seconds/3503 points	25 seconds/3503 points	37
Set 9 (432 slices)	59 seconds/4824 points	31 seconds/4824 points	45
Set 10 (425 slices)	56 seconds/4300 points	29 seconds/4300 points	42.5

The average total run-time as well as the average of the total run-times for all CT scans is given by table six.

Table 6: Average total run time for each set and the average total run time for all sets.

Set	Average Total Run Time (seconds)
1	639.5 (10.66 minutes)
2	491 (8.18 minutes)
3	492 (8.2 minutes)
4	541.5 (9.02 minutes)
5	493 (8.22 minutes)
6	534.5 (8.9 minutes)
7	486 (8.1 minutes)
8	492 (8.2 minutes)
9	506 (8.4 minutes)
10	494.5 (8.24 minutes)
Average Total Run Time For All Sets	516.72 (8.61 minutes)

6.3 Summary

The CVIP Lab VC system was designed with the intention of being user-friendly and portable. The system was also compiled and run on a Macbook Pro (Mac OS X 10.6.8 with Intel Core 2 Duo 2.8 GHz and 4 GB RAM) with a Windows 7 emulator and it operated with similar result. As a proof-of-concept it is a valuable tool and opens many doors to future development in showcasing the flyover colorectal visualization method. However, significant strides must be taken to automate segmentation and reduce high memory consumption.

CHAPTER 7:

CONCLUSION AND FUTURE WORK

In section 1.3 it was asked if the CVIP Lab's adaptive level sets, curve skeletons, and large intestine visualization software can be combined in a manner that is computationally efficient. The response to this question is inconclusive, as they can be combined (as is the case in the current system), but they can only be used in series with certain constraints. That said, the primary contribution of this thesis work is the development of a front end VC system that includes a method for large intestine segmentation, centerline extraction, splitting (for flyover), and visualization. The particular system developed is novel when compared to existing VC systems (either academic or commercial). The CVIP Lab VC interface represents a proof-of-concept for VC using computer vision and image processing tools developed in the CVIP lab.

The CVIP Lab VC interface requires future improvements in order to operate as a complete (and commercially viable) front-end system. These improvements may/must include:

1) Automation of large intestine segmentation.
2) Use of less memory dependent centerline extraction method.
3) Or, advanced memory management programming.

Automatic segmentation would remove the need for a medical professional to seed the large intestine objects in the CT scan. Ideally, an automatic segmentation protocol would speed up processing time for the CVIP Lab VC system making it more desirable as a tool for medical diagnostics. Centerline extraction, though fast, requires extensive use of

computer memory. As stated earlier, this high memory usage presents itself as potential run-time failures with large CT scans. Potential remedies to this may include an object centroid based approach to centerline extraction (where the centroids of the large intestine objects are found in each CT slice and approximated as the centerline), or an advanced memory management scheme that involves CT scan data compression or some similar approach. Ultimately, any substitute centerline extraction method should aim to maintain or surpass the run-time efficiency of the CVIP curve skeletons software while constraining the accuracy of the centerline to reasonable tolerances.

As stated in section 6.3, the VC interface works as a proof of concept, and offers future students and researchers the opportunity to expand its capabilities since it was developed in an evolvable, modular fashion. It is the hope of the author that this VC system will provide a platform for expanding CVIP medical imaging projects related to colorectal diagnostic imaging into new fields.

REFERENCES

[1] "What You Need to Know About™: Cancer of the Colon and Rectum." National Cancer Institute at the National Institutes of Health. Web. 24 July. 2012. <http://www.cancer.gov/cancertopics/wyntk/colon-and-rectal>

[2] "American Cancer Society Guidelines for the Early Detection of Cancer: Colorectal cancer and polyps." American Cancer Society. Web. 24 July. 2012. <http://www.cancer.org/Healthy/FindCancerEarly/CancerScreeningGuidelines/american-cancer-society-guidelines-for-the-early-detection-of-cancer>

[3] Dachman, Abraham H., and Hiro Yoshida. "Virtual Colonoscopy: Past, Present, and Future." Radiologic Clinics of North America (2003): 377-393.

[4] Aref, Wael M., Ahmed El-Mazny, and Farid G. Amin. "A Comparative Study between Virtual Colonoscopy (CT Colonoscopy) and Conventional Colonoscopy in Different Presentations of Suspected Colonic Disorders." Life Science Journal, 9(3): 561-567, 2012.

[5] Royster, Andrew P., Helen M. Fenlon, Peter D. Clarke, David P. Nunes, and Joseph T. Ferrucci. "CT Colonoscopy of Colorectal Neoplasms: Two-Dimensional and Three-Dimensional Virtual-Reality Techniques with Colonoscopic Correlation." American Journal of Roentgenology, 167: 1237-1242, 1997.

[6] Pickhardt, Perry J., J. Richard Choi, Inku Hwang, James A. Butler, Michael L. Puckett, Hans A. Hildebrandt, Roy K. Wong, Pamela A. Nugent, Pauline A. Mysliwiec, and William R. Schindler. "Computed Tomographic Virtual Colonoscopy to Screen for Colorectal Neoplasia in Asymptomatic Adults." The New England Journal of Medicine, 349(23): 2191-2200, 2003.

[7] Fenlon, Helen M., David P. Nunes, Paul C. Schroy III, Matthew A. Barish, Peter D. Clarke, and Joseph T. Ferrucci. "Comparison of Virtual and Conventional Colonoscopy for the Detection of Colorectal Polyps." The New England Journal of Medicine, 341(20): 1496-1503, 1999.

[8] Cotton, Peter B. Valerie L. Durkalski, Benoit C. Pineau, Yudo Y. Palesch, Patrick D. Mauldin, Brenda Hoffman, David J. Vining, William C. Small, John Affronti, Douglas Rex, Kenyon K. Kopecky, Susan Ackerman, J. Steven Burdick, Cecilia Brewington, Mary A. Turner, Alvin Zfass, Andrew R. Wright, Revathy B. Iyer, Patrick Lynch, Michael V. Sivak, and Harold Butler. "Computed Tomographic Colonography (Virtual Colonoscopy): A Multicenter Comparison with Standard Colonoscopy for Detection of Colorectal Neoplasia." Journal of the American Medical Association, 291(14): 1713-1719, 2004.

[9] van Gelder, Rogier E., Henk W. Venema, Jasper Florie, C. Yung Nio, Iwo W. O. Serlie, Michiel P. Schutter, Jeroen C. Van Rijn, Frans M. Vos, Afina S. Glas, Patrick M. M. Bossuyt, Joep F. W. Bartelsman, Johan S. Lameris, and Jaap Stoker. "CT Colonography: Feasibility of Substantial Dose Reduction – Comparison of Medium to Very Low Doses in Identical Patients." Radiological Society of North America, 232(2): 611-620, 2004.

[10] Vos, Frans M., Rogier E. van Gelder, Iwo W. O. Serlie, Jasper Florie, C. Yung Nio, Afina S. Glas, Frits H. Post, Roel Truyen, Frans A. Gerritsen, and Jaap Stoker. "Three-Dimensional Display Modes for CT Colonography: Conventional 3D Virtual Colonoscopy versus Unfolded Cube Projection." Radiological Society of North America, 228(3): 878-885, 2003.

[11] Haker, Steven, Sigurd Angenent, Allen Tannenbaum and Ron Kikinis. "Non-distorting Flattening for Virtual Colonoscopy." MICCAI, 3: 358-366, 2000.

[12] Hassouna, Sabry M., A. A. Farag and Robert Falk "Virtual Fly-Over: A New Visualization Technique for Virtual Colonoscopy." MICCAI, 9: 381-388, 2006.

[13] Hong, Lichan, Arie Kaufman, Yi-Chih Wei, Ajay Viswambharan, Mark Wax, and Zhengrong Liang. "3D Virtual Colonoscopy." Biomedical Visualization: 26-32, 83, 30. 1995.

[14] Haker, Steven, Angenent, Sigurd, Tannenbaum, Allen, and Kikinis, Ron. "Nondistorting Flattening Maps and the 3D Visualization of Colon CT Images." IEEE Transactions on Medical Imaging, 19: 665-670, 2000.

[15] Hong, Wei, Xianfeng Gu, Feng Qiu, Miao Jin, and Arie Kaufman. "Conformal Virtual Colon Flattening." Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling: 85-92, 2006.

[16] Chen, Dongqing, Aly A. Farag, Robert L. Falk, and Gerald W. Dryden. "A Variational Framework for 3D Colonic Polyp Visualizations in Virtual Colonoscopy." ICIP: 2617-2620, 2009.

[17] Chen, Dongqing, Rachid Fahmi, Aly A. Farag, Robert L. Falk, and Gerald W. Dryden. "Accurate and Fast 3D Colon Segmentation in CT Colonography." ISBI: 490-493, 2009.

[18] Chen, Dongqing, Aly A. Farag, Robert L. Falk, and Gerald W. Dryden. "On Clinical Validation of Fly-Over Visualization Technique for Virtual Colonoscopy." ICIP: 2593-2596, 2009.

[19] Chen, Dongqing, Hossam Abdelmunim, Aly A. Farag, Robert Falk, and Gerald Dryden. "Segmentation of Colon Tissue in CT Colonography Using Adaptive Level Sets Method." MICCAI Workshop: 108-115, 2008.

[20] Hassouna, M. Sabry, and Aly A. Farag. "Multistencils Fast Marching Methods: A Highly Accurate Solution to the Eikonal Equation on Cartesian Domains." IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(9): 1563-1574, 2007.

[21] Hassouna, M. Sabry, and Aly A. Farag. "On the Extraction of Curve Skeletons using Gradient Vector Flow." ICCV: 1-8, 2007.

[22] Hassouna, M. Sabry, and Aly A. Farag. “Accurate Tracking of Monotonically Advancing Fronts.” CVPR, 1: 355-362, 2006.
[23] Osher, S., and J. A. Sethian. “Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations” Journal of Computational Physics, (79): 12 – 49, 1988.
[24] H. Abdelmunim, and Aly A. Farag. “Adaptive Segmentation of Multimodal 3D Data using Robust Level Set Technique.” MICCAI: 143-150, 2004.
[25] Bell, John T. “Visualization Toolkit (VTK) Tutorial.” University of Illinois, Chicago. Web. 8 Aug. 2012. <http://www.cs.uic.edu/~jbell/CS526/Tutorial/Tutorial.html>
[26] “VTK 4.0.2 Documentation.” VTK. Web. 10 Aug. 2012. <http://www.vtk.org/doc/release/4.0/html/index.html>
[27] “Anatomical terms of location” Wikipedia Web. 12 Aug. 2012. <http://en.wikipedia.org/wiki/File:Human_anatomy_planes.svg>
[28] “Large Intestine” WebMD. Web. 11 Aug. 2012. <http://www.webmd.com/digestive-disorders/large-intestine>
[29] Crump, David B. “Colonoscopy: The colonoscope” Web. 11 Aug. 2012 <http://www.drcrump.com/id27.htm>
[30] “Colonoscopy Screening Test” John Hopkins Medicine. Web. 11 Aug. 2012. <http://www.hopkinscoloncancercenter.org/CMS/CMS_Page.aspx?CurrentUDV=59&CMS_Page_ID=33CD25B0-CCC6-4F55-A226-3C202E67D0B1>