COMPUTER-AIDED VISUALIZATION OF COLONOSCOPY

Ruibin Ma

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

Chapel Hill
2020

Approved by:

Stephen M. Pizer

Jan-Michael Frahm

Sarah K. McGill

James N. Damon

Ron Alterovitz

ABSTRACT

Ruibin Ma: Computer-aided Visualization of Colonoscopy
(Under the direction of Stephen M. Pizer and Jan-Michael Frahm)

Colonoscopy is the most widely used medical technique to examine the human large intestine (colon) and eliminate precancerous or malignant lesions, i.e., polyps. It uses a high-definition camera to examine the inner surface of the colon. Very often, a portion of the colon surface is not visualized during the procedure. Unsurveyed portions of the colon can harbor polyps that then progress to colorectal cancer. Unfortunately, it is hard for the endoscopist to realize there is unsurveyed surface from the video as it is formed. A system to alert endoscopists to missed surface area could thus more fully protect patients from colorectal cancer following colonoscopy. In this dissertation computer-aided visualization techniques were developed in order to solve this problem:

1. A novel Simultaneous Localization and Mapping (SLAM) algorithm called RNNSLAM was proposed to address the difficulties of applying a traditional SLAM system on colonic images. I improved a standard SLAM system with a previously proposed Recurrent Neural Network for Depth and Pose Estimation (RNN-DP). The combination of SLAM's optimization mechanism and RNN-DP's prior knowledge achieved state-of-the-art performance on colonoscopy, especially addressing the drift problem in both SLAM and RNN-DP. A fusion module was added to this system to generate a dense 3D surface.

2. I conducted exploration research on recognizing colonic places that have been visited based on video frames. This technique called image relocalization or retrieval is needed for helping the endoscopist to fully survey the previously unsurveyed regions. A benchmark testing dataset was created for colon image retrieval. Deep neural networks were successfully trained using Structure from Motion results on colonoscopy and achieved promising results.

3. To visualize highly-curved portions of a colon or the whole colon, a generalized cylinder deformation algorithm was proposed to semi-flatten the geometry of the colon model for more

succinct and global visualization.

ACKNOWLEDGEMENTS

First I would like to thank my advisor, Dr. Stephen M. Pizer, for his support and advising during my PhD. I have learned a lot from him in both study and life. I would also thank my co-advisor Dr. Jan-Michael Frahm and my other committee members, Dr. Sarah K. McGill, Dr. Ron Alterovitz, and Dr. James Damon for their great advice and generous help provided during my PhD.

Second I would like to thank my teammates, especially Dr. Julian Rosenman and Dr. Rui Wang. I really enjoyed working in my team. The achievement today is a product of the smart ideas and the hard work of us together.

Finally I would like to thank my family. To my wife Jiayu, you accompanied me throughout my PhD, and your encouragement is the best driving force for me to have moved forward. To my parents, I am happy to make you proud of your son today.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

APE Absolute Pose Error. xiii, 66–69, 85, 86

BoW Bag-of-Words. 45, 76

CNN Convolutional Neural Networks. 5, 6, 8, 45, 71, 76, 78, 81, 83–86, 108, 113–115

ConvLSTM Convolutional Long-Short-Term-Memory. 38

DGM Differentiable Geometric Module. 38, 39

DSO Direct Sparse Odometry. viii, x, xi, 3, 6, 20, 24, 25, 27, 30, 33, 34, 37, 52–56, 60–64, 66, 67, 107, 112

LSD-SLAM Large-Scale Direct Monocular SLAM. 3, 6, 24, 25, 56

RANSAC Random Sample Consensus. 4, 23, 45

RCC Relative Curvature Condition. 89–91, 95, 96

RMF Rotation Minimizing Frame. 46, 47, 89, 93–96

RNN-DP Recurrent Neural Network for Depth and Pose Estimation. iii, vii, viii, x, xi, 5, 6, 12, 30, 37–41, 52–56, 60, 62–64, 66, 67, 83–85, 107, 110, 112, 114

SfM Structure from Motion. 8, 13, 26, 40, 41, 45, 53, 66, 71, 73, 76, 81, 108–110, 114

SIFT Scale-Invariant Feature Transform. 3, 4, 8, 25, 113, 114

SLAM Simultaneous Localization and Mapping. iii, 3, 4, 6, 12, 23, 24, 26, 28, 33, 37, 54, 81, 107, 111, 114

CHAPTER 1

Introduction

Colorectal cancer is the second most common cancer worldwide [1]. Colonoscopy is the most commonly used technique for lesion (typically polyps) screening inside the large intestine (colon). During a colonoscopy, a physician examines human colons by directly viewing video frames produced by a camera and a light source installed at the tip of an endoscope. If a polyp is detected, it can be immediately excised by the colonoscope's built-in tools. A colonoscopy is a "pulling" procedure: an endoscopist first moves the colonoscope to the far end of the large intestine and then gradually pulls the endoscope back while successively insufflating, *i.e.*, ballooning the colon with CO_2. The examination and operations are conducted during the pulling procedure. However, colonoscopy is not perfect. In Sections 1.1, 1.2, 1.3, I will introduce the problem of unsurveyed regions, the difficulty of place recognition in colon, and the difficulty of succinctly visualizing a curved 3D colon model respectively, and I will introduce my proposed solutions.

1.1 Unsurveyed Regions Lead to Unsurveyed Polyps

During a colonoscopy, one important reason for unsurveyed polyps is that they have not been inside the field of view of any video frame, at least not with adequate quality; that is, the colonic surface was not fully surveyed. The accuracy of an examination is thus highly affected by the percentage of coverage. There are at least three reasons for unsurveyed regions: 1) lack of orientations of the camera to view the full circumference of the colon; 2) occlusion by the colon structure itself, especially by the narrow rings in the colon called haustral ridges; 3) poor ability of a human to notice the unsurveyed regions from the first-person perspective, especially since the endoscopist is focusing on finding polyps. One such example is in Fig. 1.1. Hong et al. [2] evaluated the unsurveyed region of a procedure by virtual colonoscopy. They found 23% of the surface was missed in a simulation.

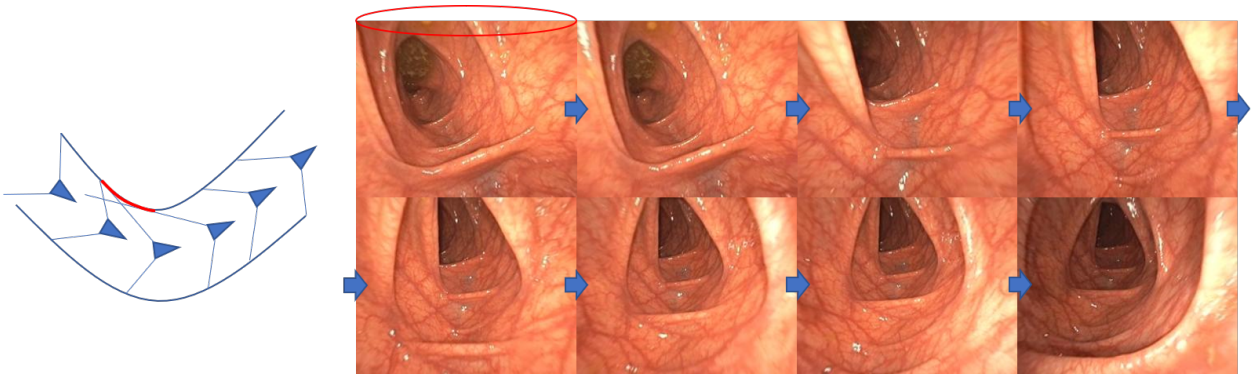


Figure 1.1: One example of a unsurveyed region in colonoscopy. The camera is quickly moving through a high-curvature region (flexure). The left image is an illustration of the camera path (the blue triangles), and their orientations and field of view (the thin straight lines). The two boundary blue curves represent the colon surface. The red region is not inside the field of view of any of the camera poses. The respective region is marked by a red circle in the upper left snapshot.

1.1.1 Solution: Reconstruct 3D Colon by SLAM.

The team of which I am a member [1] has built an automatic system that detects large [2] unsurveyed regions and alerts the endoscopist in real time so that the colonoscopist can immediately survey the previously unsurveyed regions. We proposed a paradigm that reconstructs a video into a 3D surface while leaving the unsurveyed regions blank (unreconstructed). Such blank regions can explicitly visualize which part is unsurveyed and can give the material for computing the unsurveyed surface area. There are two requirements for such a system: 1) dense reconstruction, because the surface is used to compute unsurveyed area; 2) low-latency, because the endoscopist needs to be alerted before the camera is too far away from the unsurveyed region. In computer vision and robotics, the essential part of this task is called Simultaneous Localization and Mapping (SLAM).

1.1.2 Difficulties of Traditional SLAM for Colonoscopy.

The two major components of a typical SLAM system are tracking and mapping [3]. The tracking component uses visual clues to predict camera poses (6-DoF relative rigid transform, detailed in 2.2) for each incoming frame (image) and to create keyframes. Keyframes are the selected frames to represent 3D reconstruction. They are chosen to reduce redundancy. The mapping component

[1] Ruibin Ma, Rui Wang, Stephen Pizer, Julian Rosenman, Sarah McGill, Jan-Michael Frahm

[2] We only want to alert colonoscopists if there are significant unsurveyed regions because small unsurveyed regions can easily be false positives. Whether a hole is large is judged by an area threshold detailed in Chapter 3.

predicts a 3D scene, e.g., 3D point locations. The mapping component can be divided into local mapping and global mapping. Local mapping computes 3D scene based on local information such as the latest keyframe or several recent keyframes. Typically, it manages several recent keyframes and jointly optimizes their poses and visible 3D point positions (local bundle adjustment). Global mapping typically does loop closure: when the camera moves to a historical location and similar scenes are observed between the current frame and a historical frame, a relative camera pose can be estimated between them; this estimation can be added as an extra constraint to the 3D reconstruction to reduce accumulated error. This is typically done by a global bundle adjustment for all the keyframes to make the predictions more consistent. Global mapping is not necessary for a SLAM system. For example, DSO [4] only contains tracking and local mapping.

The tracking and mapping components are both optimization problems. They try to minimize the inconsistency between warped images: when the content of one frame is warped to another frame based on the predicted camera poses and 3D scene, there can be an inconsistency between either their intensities or 2D point locations. This is quantified by an error function. Depending on whether the error functions in tracking and mapping directly use photometric error (intensity difference) or not, SLAM methods can be divided into "direct" methods [4, 5] and "indirect" methods [6, 7]. Indirect methods first extract some salient points (feature points) to represent the image content and use them to find correspondences and compute error functions. One example is ORB-SLAM. ORB-SLAM is based on ORB (Oriented FAST[3] and Rotated BRIEF[4]) feature points and reprojection error. However, it cannot handle a low-feature environment, like blank walls, or in our case colon images because feature points that can establish accurate matchings between two images can hardly be detected. Such an example is shown in Fig. 1.2. Despite using Scale-Invariant Feature Transform (SIFT) [10, 11] feature points, there are still many outlier points. In contrast, the direct methods, Large-Scale Direct Monocular SLAM (LSD-SLAM) [5] and Direct Sparse Odometry (DSO) [4], use depth maps (an image of depth values at each pixel) instead of feature points and directly use photometric error to estimate camera poses. In DSO, sparsity enables joint optimization of depth values and camera poses. The advantage of direct methods is that they do not depend on

[3]Features from Accelerated Segment Test [8]
[4]Binary Robust Independent Elementary Features [9]

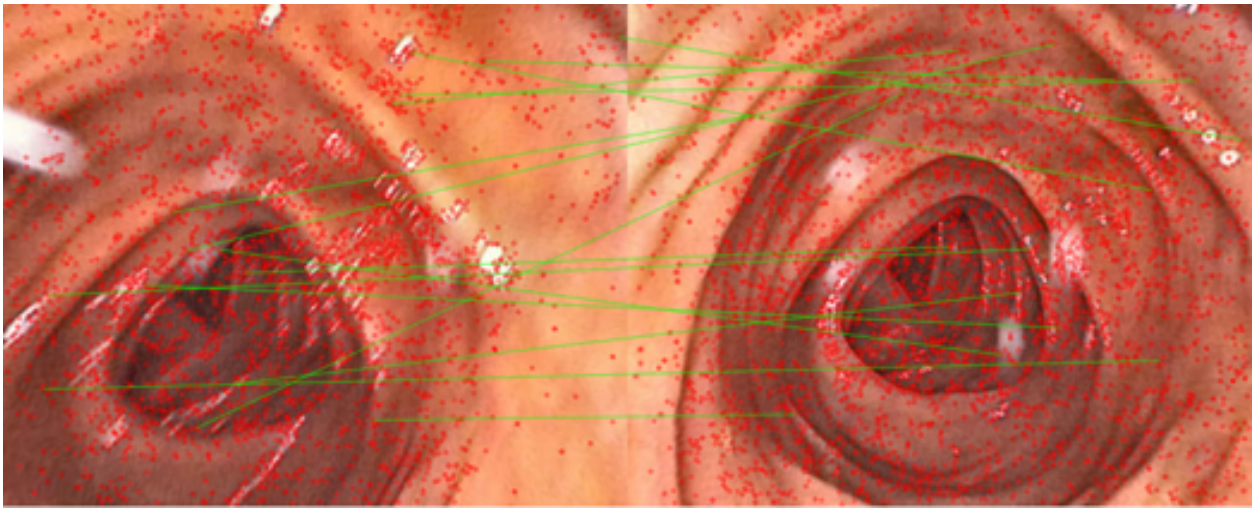


Figure 1.2: The red dots in the images are extracted salient points by the SIFT algorithm. They are called "feature points". SIFT feature point matchings between two colonoscopic images are shown as green lines. These matchings are already filtered by the RANSAC algorithm to fit one reasonable camera transform between the two images. The quantity and accuracy of the matchings are both very low.

the successful extraction of feature points and can support a fine-grained and visually appealing reconstruction. The disadvantage is that they are more sensitive to intensity change and geometric distortion of the 3D scene they are viewing.

I use the term "textureness" to refer to the degree of abundance of high-gradient image regions that can be leveraged by the aforementioned direct or indirect methods. Unfortunately, colon images are very "low-textured" because the colon surface is very smooth. This makes the existing SLAM algorithms work poorly. More importantly, SLAM has a fundamental assumption of brightness/appearance constancy of the scene: 1) the algorithms assume two images are viewing the same 3D scene and the relative camera locations are estimated based on rigid geometric relation; 2) direct methods also assume the intensities of the same 3D point are close in two images and compute error functions by taking intensity differences. More details about the algorithms are introduced in Section 2.5. In colonoscopy these assumptions are principally broken given that the camera carries the light source and the colon deforms. That means we need to also deal with intensity variation and geometric distortion, a significant challenge. The 3D reconstruction output of our system is thus an approximated estimation of the colon structure over a short time interval.

1.1.3 Difficulties of End-to-end Deep Learning Methods

Because Convolutional Neural Networks (CNN) can learn features automatically, it is conceivable that deep learning could help to solve the problem in colonoscopic reconstruction. Recently, CNNs have begun to produce results of comparable quality to traditional geometric computer vision methods for depth estimation in well-textured parts of the image [12, 13, 14, 15, 16]. They achieve significantly more complete results for poorly textured areas through a learned prior. In principle, this approach can be used in the poorly textured colon. Commonly a CNN requires thousands or even millions of images with groundtruth labels for training. In our problem we need groundtruth of depth values at image pixels. However, there is no accurate groundtruth depth available for endoscopic videos. Wang et al. proposed a deep neural network architecture named Recurrent Neural Network for Depth and Pose Estimation (RNN-DP) [17] that can be trained in an unsupervised fashion for depth and camera pose estimation for common outdoor and indoor scenes. However, surface deformation, large illumination changes, and lack of texture in colonoscopic videos leads to unsatisfactory performance of the RNN-DP when trained without supervision on colonoscopy.

1.1.4 The Drift Problem

One common problem for both the SLAM systems and the end-to-end depth and pose prediction networks is the "drift" problem. That problem is that the prediction error is accumulated or even amplified through a sequential prediction, in which a later input is based on previous results. If one result is erroneous, the error will be propagated into the prediction of a later one. Two typical types of drift problem are scale drift and camera pose drift. In scale drift the size of the scene being reconstructed is continuously changing; in camera pose drift the camera positions deviate from the true path more and more as the camera is moving. There are two reasons for these drift problems: 1) non-accurate tracking leads to large errors between successive frames; 2) no inter-frame joint optimization is used or the optimization is only local. The former reason usually applies to common SLAM systems because the tracking is purely computed based on color or keypoint matchings and no prior knowledge is used. The latter reason usually applies to CNNs because an end-to-end CNN only performs forward prediction based on learned knowledge and it does not optimize on a specific case. Examples showing these drift problems are provided in Section 3.2.

1.1.5 Combining SLAM and Deep Learning May Improve the Result

There have been research efforts combining deep learning and a traditional SLAM system. CNN-SLAM [18] incorporates CNN-predicted depth maps into the LSD-SLAM framework. Depth maps provide a denser and more accurate uncertainty estimation (uncertainty score of depth values used as weights in inter-frame propagation). Yang et al. [19] also replaced the stereo measurements in Stereo DSO [20] by depth values predicted by CNN. The effectiveness of using a CNN comes from a robust depth prior and gives reasonable depth prediction to assist a SLAM system.

1.1.6 SLAM Has Been Applied in Colonoscopy

SLAM techniques have been applied to colonoscopy. Chen et al. [21] proposed a method that uses an adversarial depth network to predict the depth channel for an RGB-D sequence. The sequence was then fused into a dense 3D point cloud (surfels [22]) by ElasticFusion [23]. They showed their result on a phantom colon chunk. However, this method cannot achieve satisfactory result for real-time unsurveyed region detection by 3D reconstruction.

To achieve this aim, I have proposed a SLAM system (RNNSLAM [24]) by combining Dr. Rui Wang's RNN-DP [14] and a standard SLAM system (DSO) [4]. It achieved state-of-the-art performance on real time 3D colon reconstruction. This system has much less drift compared to either RNN-DP or DSO thanks to the combination of optimization and prior knowledge.

1.2 It Is Difficult to Fully Survey Previously Unsurveyed Colon Regions

After a system for unsurveyed region detection is developed, the endoscopist will want to return to survey the previously unsurveyed region. Because the endoscope has moved while the SLAM algorithm forms a 3D colon, the next problem is how to guide the endoscopist to navigate back to the detected unsurveyed region. However, it is often difficult for endoscopists to navigate back to the unsurveyed regions by simply looking at the reconstructed colon chunk or the video frames [25]. This is because the colon images are very similar to each other. There are not enough features or patterns that can be recognized easily by a human. It even takes a lot of time for experts to tell if two images are at the same position in the colon. Given that the unsurveyed region problem happens quite often, it would be tedious for an endoscopist to visually navigate to the region of interest at each time when an unsurveyed region is detected. The unsurveyed region detection system will not be fully useful unless the endoscopist has an efficient way to re-check certain regions. Therefore, an automated system is needed to help the endoscopists recognize colonic places that have been seen

before, that is, "relocalize" the current video frame.

1.2.1 Motivation of Relocalization in Colonoscopy

Relocalization in a colonoscopy means retrieving the current position of the camera with regard to existing video frames or finding a specific position or view-point that appeared in history. Relocalization in colonoscopy is often needed for the following reasons:

1. As recently stated, the endoscopist may need to recheck a certain region. Based on our existing unsurveyed region detection system mentioned in Section 1.1, this place recognition algorithm should save image features during the reconstruction and do image retrieval for each new frame once the endoscopist decides to go back to a unsurveyed region. The image retrieval algorithm will retrieve the current relative position in the reconstructed model. This positional information should be highlighted on the 3D model. The endoscopist can then be guided back to the surrounding locations of the unsurveyed region. The central part of this method is an efficient relocalization / image retrieval / place recognition algorithm. As mentioned previously, this task is non-trivial because colon images are highly homogeneous and take quite some effort to distinguish and associate.

2. There have been efforts [21, 24] to reconstruct a 3D colon for better visualization. However, a colonoscopic sequence is often broken by low-quality frames such as the frames polluted by fecal matter, bleeding, or cleansing water. The reconstruction cannot be continued at this point, and a new reconstruction must be started when normal frames becomes available. Place recognition can thus be used to recognize potential overlap between the old and new reconstructions and to connect them together to achieve a more complete model. Place recognition can also be used to establish long-range correspondence and thus support a loop closure module that can reduce drift in reconstruction. These two motivations can be regarded as follow-up research of the RNNSLAM system. The development of colon place recognition can make the unsurveyed region detection system more efficient, more accurate, and practically useful.

3. Place recognition is useful when a patient takes a second colonoscopy after several years. It is often needed to check the region where a polyp was removed in order to assure that there has been no re-growth of the polyp. Although this task is very important for patient revisits, it is practically very difficult for the same reasons given above. Some existing methods include

tattooing the colon mucosa to mark the place.

1.2.2 Image Retrieval Techniques

Automatic image retrieval is a widely studied area in computer vision. Traditional methods are often based on local features like SIFT [10]. However, colonoscopic images are very low-textured compared to common images in traditional computer vision tasks. There are not enough salient patterns to detect. Therefore, these traditional methods cannot guarantee most of the previously visited placed can be accurately retrieved.

Recently, the development of CNNs is showing promising results in the place recognition field [26, 27, 28, 29, 30]. Patterns inside the images are condensed into a global image descriptor for computing image similarities. The groundtruth for this task is typically the knowledge of image-to-image correspondences. One major challenge for colonoscopic place recognition is that there is no existing groundtruth, e.g., GPS locations, human annotations, etc. To acquire the image-to-image correspondences, I leveraged the Structure from Motion (SfM) results on more than 60 colonoscopic videos. Following the spirit of [30], I successfully trained working models for colonoscopic place recognition.

Evaluation is another major challenge in this area. First, I summarized the evaluation metrics, which were different across the literature. I proposed proper metrics for different applications. Second, since there were no benchmarks for colonoscopic place recognition, Sarah McGill and I created a dataset (Colon10K) [31] for this purpose. I conducted extensive evaluations of different CNN architectures. The results have shown our training strategy using SfM results was effective and had much better performance than traditional methods under all metrics.

1.3 Interior Surface Visualization of a Highly Curved Colon Model

Visualizing a colon surface is complicated because of the highly-curved geometry of portions of the colon as shown in Figure 1.3. Viewing the 3D surface in an endoscopic perspective may not be the optimal approach, because the endoscopist may not be aware of the lack of camera orientations and occlusions. In this part of the dissertation I focused on finding a more efficient way to visualize a highly curved colon model as a whole. This can be regarded as a geometrical postprocessing or enhancement stage of the previously discussed reconstruction pipeline. The central idea is to semi-flatten the highly curved geometry to visualize its interior surface more succinctly.

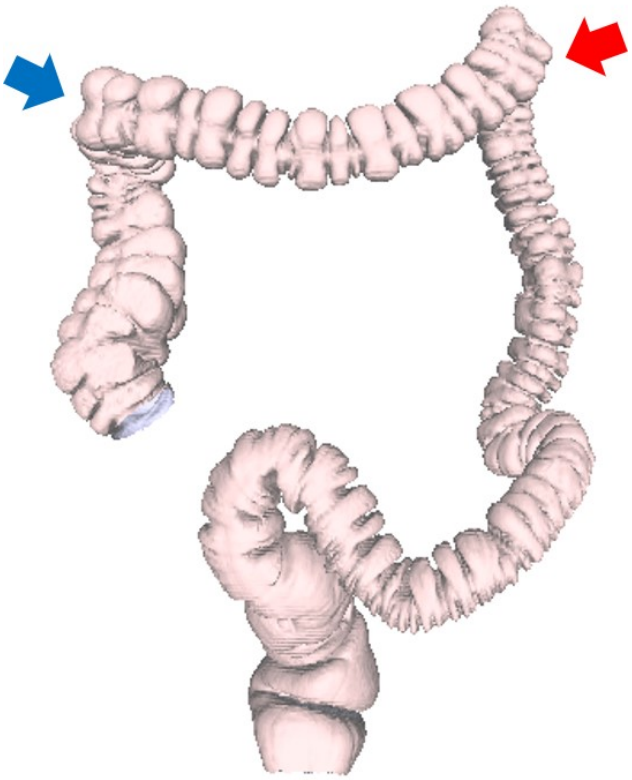


Figure 1.3: A 3D model of a real colon. The surface has high curvatures, especially at the flexures. The blue arrow points to the hepatic flexure. The red arrow points to the splenic flexure.

1.3.1 Colon (Semi-)Flattening

Visualization of medical images of the human colon has long been a research topic. For example, virtual colonoscopy (VC) [32] processes computed tomography (CT) images of the colon in order to sense and visualize the interior and target polyps. Surface and volume rendering methods have been proposed [33]. This virtually endoscopic approach simulates real colonoscopy by flying through the generalized cylindrical surface with a camera. However, just as with real colonoscopy, this approach has the drawback that the operator may miss visualizing some portion of the surface. McGill et al. in [34] show that there is noticeable unsurveyed surface on the reconstruction from endoscopic videos (endoscopogram) [35, 36, 37] due to limited field of view (FOV) and occlusion of haustral ridges (narrow rings inside the colon). Endoscopists do not even realize there are parts of the surface still not seen when they finish the procedure. Paik, D. et al. [38] tried cylindrical and planar map projections in virtual colonoscopy to enlarge the FOV but could not promise to present the whole surface. Therefore, I have sought a viewing approach that can present the whole interior

to physicians, given the interior surface of the colon geometrically and ideally also with texture.

Unfolding approaches have been proposed to visualize the highly curved interior surface at-a-glance. Bartrolí, A. et al. [39] proposed a method using nonlinear ray-casting in virtual colonoscopy to provide a 2D surface coded by a height map with shading information. Conformal mappings [40, 41, 42] of surfaces segmented from colonography have also been done to map tubular surfaces to a plane while keeping the curvature information as intensities. However, the 3D geometry is eliminated by these methods, and asking physicians to visualize the 3D model and positions in their mind can decrease the accuracy of diagnosis. These mapping methods also introduce great area distortion.

An alternative method is to straighten the quasi-tubular surface while presenting the tubular structure and local curvature patterns.

1.3.2 Generalized Cylinder Deformation Problem

The generalized cylinder, or quasi-tube, is a common and useful kind of model in computer graphics [43]. It is widely accepted that the mathematical definition of a generalized cylinder consists of a centerline (or medial axis, skeletal curve, etc.) and its orthogonal cross sections [44]. Accordingly, some 3D models of generalized cylinders can be constructed by spline-based methods [45, 46, 47, 48]. In this dissertation, however, I consider a discretized representation of generalized cylinders, *i.e.*, a triangle mesh (usually of high resolution), which is a more general representation in computer graphics.

Deformation of generalized cylinders is a common and important problem in computer graphics, as shown in Figure 1.4. Compared with the free-form deformation [49] that uses enveloping control points, direct manipulation [50] can be more intuitive by moving control vertices on the surface. As such, a number of large-mesh deformation methods [51, 52, 53, 54, 55, 56, 57, 58, 59] using direct manipulation have been proposed. Further, skeleton-based methods, *i.e.*, the skinning methods [60, 61], have the advantage of imposing prior knowledge or efficient control of the object shape/skeleton. Joints and edges are commonly adopted to deform 3D character meshes [62]. Such skeletons are either manually given [63] or computed by methods like the Voronoi diagram [64] or harmonic skeleton [65].

In this part of the dissertation, however, I focused on the common practice of driving generalized cylinder deformations by the shape of centerlines. I dealt with generalized cylinders with a notably

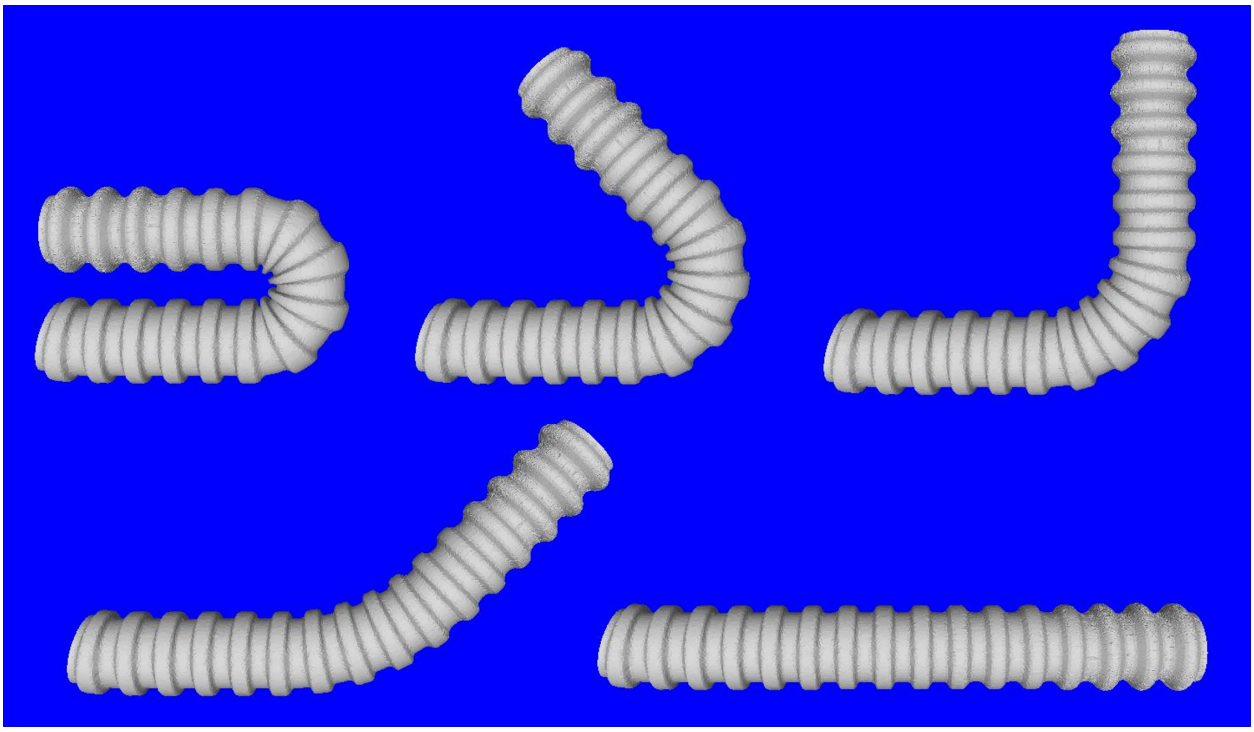


Figure 1.4: Centerline-guided generalized cylinder deformation. A generalized cylinder is deformed at large-scale while maintaining its local curvature patterns.

tubular structure (one outstanding dimension). The centerline of a generalized cylinder was densely sampled and represented as a parametric spatial curve which could be deformed into a parametric target shape. In this situation, deforming generalized cylinders via direct manipulation becomes challenging, especially for highly curved centerlines because it is difficult to manually impose accurate control of the centerline curvature. This difficulty is also present for cage-based free-form deformations or skeleton-driven approaches that use very sparse skeletal control points.

Closely following the mathematical definition of a generalized cylinder [44], I used a centerline and discrete profile cross sections as the skeleton. Skeletonization methods have been proposed to extract centerlines [66], cross sections [67], or both [68]. However, none of these methods considered preventing surface foldings resulting from intersections between extracted cross sections. To address this problem, our proposed approach extracts cross sections and tries to minimize their intersections before deformation. A tentative centerline is extracted [66] and modified based on the relative curvature condition [69] that detects intersection. The modified centerline and its respective cross sections are then used as the canonical skeleton for the followup deformation. Given the target centerline shape as a parametric 3D curve, all the orthogonal cross sections are mapped using

rotation minimizing frames [70]. Next, because our densely sampled centerline and cross sections divide the mesh into small regions, the displacements in each region can be solved efficiently by minimizing a quadratic thin-shell bending energy [71, 72] with the displacements on the cross sections as constraints. The output of my algorithm is a generalized cylinder whose geometry follows the target centerline shape while largely keeping local curvature patterns. The deformation is then followed by a slit-open operation along a longitudinal line on the surface to allow direct visualization of the interior.

The applications of my deformation method for quasi-tubes are not limited to straightening the colon and visualizing its interior. It can be extended to the simulation of tube deformation into any designed posture. It can also be a new way for producing animations from a high-resolution mesh data structure.

1.4 Thesis and Contributions

Our team's colonoscopic visualization system makes colonoscopy more accurate while maintaining efficiency by addressing the unsurveyed surface problem and relocalization difficulty. My contributions in this system of improved simultaneous localization and mapping in colon videos, colonic place recognition, and surface deformation of the colon model are critical to the system's success.

The contributions of this dissertation are as follows:

1. A SLAM system that works for colonoscopic images has been designed. The pipeline called RNNSLAM combines the depth and pose predictions of RNN-DP [17] and a standard SLAM system [4]. It takes advantage of both RNN-DP's prior knowledge and SLAM's optimization mechanism (direct image alignment and local bundle adjustment). It can reduce both scale drift and camera pose drift that exist in common SLAM pipelines and end-to-end deep neural networks. This is an improvement to generic SLAM tasks.

2. A pipeline for generating dense 3D colon model for a short period of colon video to detect unsurveyed regions has been designed. A fusion module takes the RNNSLAM's output and generates a unified 3D mesh. A windowed depth map smoothing algorithm was designed to ensure the consistency needed for the fusion.

3. A benchmark for colonoscopic image retrieval / place recognition has been contributed. This includes a testing dataset with manually labeled groundtruth and working deep neural networks

trained novelly using the SfM results of a large collection of unlabeled colonoscopic videos. High performance has been achieved under multiple metrics.

4. A centerline-driven generalized cylinder deformation algorithm has been designed. This algorithm includes 1) a skeletonization method based on the relative curvature condition that minimizes intersection between cross sections and enables large-scale geometrical deformation; 2) an efficient solution of fine-scale (vertex-wise) deformations based on the Thin Shell model that uses the displacements on the cross sections (result of geometrical deformation) as constraints; 3) a strategy to control twisting using rotation minimizing frames. This algorithm provides an efficient way to deform a generalized cylinder by a densely sampled parametric centerline. The target shape can be easily controlled by the form of the target parametric expression.

5. The above centerline deformation algorithm was novelly applied to the colon semi-flattening problem. The colon surface is first straightened while keeping the local curvature patterns and then slit open along a longitudinal line on the surface. The result is a semi-flattened surface showing the interior surface succinctly without losing the local geometry. This algorithm has less area distortion and can handle colonic regions with very large curvature.

1.5 Overview of Chapters

The remainder of this dissertation is organized into the following chapters: Chapter 2 introduces the medical and methodological background of the proposed methods; Chapter 3 introduces the RNNSLAM system and the pipeline for real-time colon reconstruction for unsurveyed region detection; Chapter 4 introduces my contributions in colonoscopic place recognition / image retrieval; Chapter 5 introduces the generalized cylinder deformation algorithm and its application on colon semi-flattening; Chapter 6 includes the conclusion, discussion and future work.

CHAPTER 2

Background

2.1 Medical Background

2.1.1 Colon Anatomy

The colon is also called the large intestine. "*The large intestine, also known as the large bowel, is the last part of the gastrointestinal tract and of the digestive system in vertebrates. Water is absorbed here and the remaining waste material is stored as feces before being removed by defecation*" [73]. The colon is a long tubular organ connected to the small intestine and the anus at its two ends.

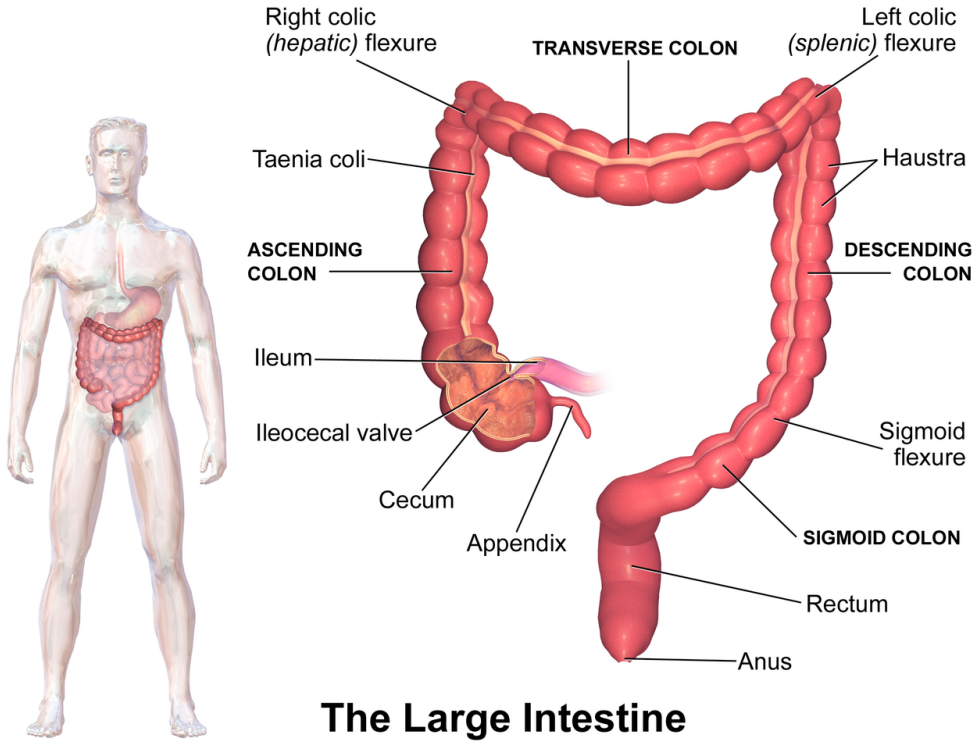


Figure 2.1: The anatomy of the human colon. Picture source: Wikipedia - large intestine.

Figure 2.1 is an illustration of the anatomy the human colon. The colon can be divided into five chunks: ascending colon, transverse colon, descending colon, sigmoid colon and rectum. Some

important landmarks include the hepatic flexure, the splenic flexure, the illeocecal valve between the small intestine and the cecum, the cecum and the appendiceal orifice (to the appendix).

The colon interior surface is highly curved: it contains many small pouches called "haustra" (singular: haustrum), which give the colon its segmented appearance. We call the narrow ring between two haustra a "haustral ridge". The ribbons outside of the colon are called taenia coli. Taenia coli exist along the ascending colon, the transverse colon, the descending colon and the sigmoid colon. The colon surface along the taneia coli is flatter and smoother compared to the other part of the surface, and the haustral ridges becomes flat when they come across the taenia coli.

2.1.2 Colonoscopy Procedure

Colonoscopy is a technique to check for abnormalities in human colons. As shown in Figure 2.2, during a colonoscopy, a physician puts an endoscope into the patient's colon. The endoscope is a long thin tube of fibers with a camera at the end of the tube. It has a channel for injecting water and CO_2 and removing in-colon materials. Moreover, through the channel can be passed tools for biopsy or lesion removal.

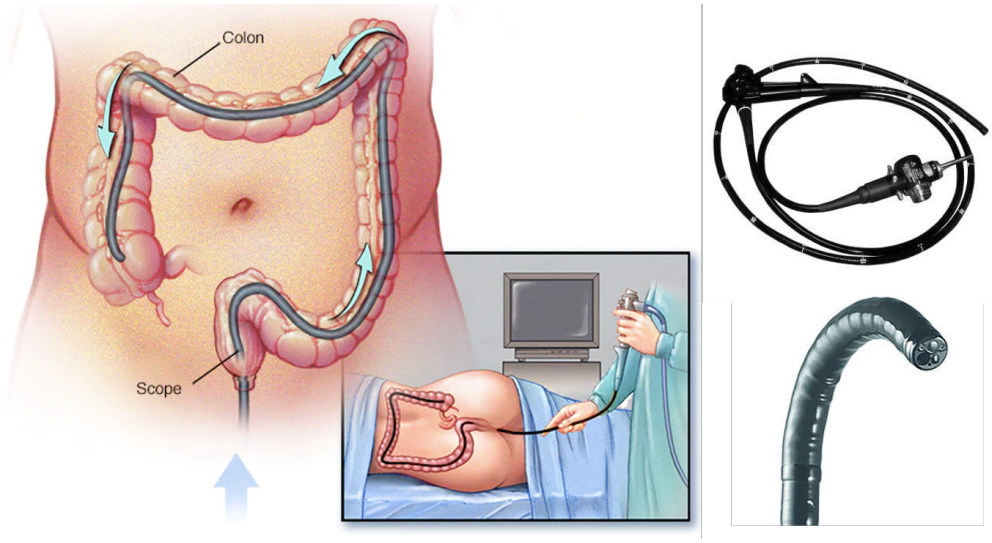


Figure 2.2: Illustration of colonoscopy and the colonoscope. Picture source: Mayo Clinic, Medical Expo.

During a colonoscopy the doctor first inserts the endoscope to the end of the colon until the landmarks (cecum, illeocecal valve, and appendiceal orifice) are seen. The screening happens while he/she gradually pulls the endoscope out of the colon. While the endoscope is being pulled, the colon is successively insufflated by CO_2. The purpose of this is to lower the colon's curvatures for

better visualization. The endoscopist constantly rotates the tip of the endoscope to check the colon.

One of the major purpose of colonoscopies is to check for "polyps". A polyp is an abnormal growth of cells inside the colon. One example is shown in Figure 2.3. Although a polyp is usually harmless, it can grow into colon cancer over time. Therefore, during a colonoscopy, if a polyp of adequate size is seen, it will be resected by the tools passed through the endoscope, as illustrated in Figure 2.3.

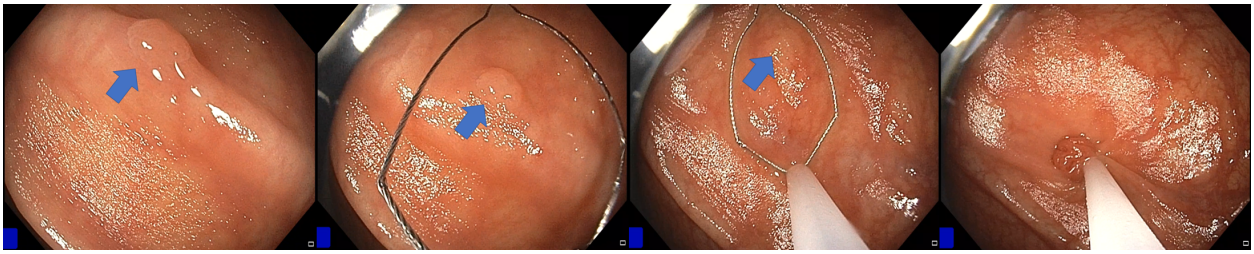


Figure 2.3: A polyp and how it was resected in a colonoscopy.

2.2 Rigid Body Transforms

In the optimization process of SLAM and other 3D vision tasks, part of the goal is to find the camera poses that give minimum (photometric or reprojection) error:

$$\hat{G} = \text{argmin}(f(G)) \tag{2.1}$$

where G is represents a 4×4 rigid tranform matrix (to be detailed next). If we naively apply gradient descent, we have

$$G_{k+1} = \delta G + G_k \tag{2.2}$$

where k stands for iteration step. However, the resulting matrix G_{k+1} may not be a valid rigid transform matrix because the constraint of G can be violated. In other words, the space of G is not a linear space. This requires us to study the space of rotations and rigid transforms. We also need to study how to linearize the G space using a most parsimonious representation.

2.2.1 Rotations and the Special Orthogonal Group

The 3D rotation group SO(3) is the group of all rotations about the origin in $\mathbb{R}^3$. A 3D rotation matrix $\mathbf{R}$ is an element in SO(3), denoted by $\mathbf{R} \in \mathrm{SO}(3)$.

$$\mathrm{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}\mathbf{R}^T = I, det(\mathbf{R}) = +1\} \tag{2.3}$$

If the matrix $\mathbf{R}$ is time-dependent, we can take derivative of $\mathbf{R}^T(t)\mathbf{R}(t) = I$ to get

$$\dot{\mathbf{R}}(t)\mathbf{R}^T(t) + \mathbf{R}(t)\dot{\mathbf{R}^T}(t) = 0 \tag{2.4}$$

$$\dot{\mathbf{R}}(t)\mathbf{R}^T(t) = -(\dot{\mathbf{R}}(t)\mathbf{R}^T(t))^T \tag{2.5}$$

Therefore, the matrix $\dot{\mathbf{R}}(t)\mathbf{R}^T(t)$ is skew-symmetric; there must be a 3D vector ω such that $[\omega]_\times = \dot{\mathbf{R}}(t)\mathbf{R}^T(t)$ where $[\omega]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$. This is equivalent to

$$\dot{\mathbf{R}}(t) = [\omega]_\times \mathbf{R}(t) \tag{2.6}$$

If $\mathbf{R}(t_0) = I$, the solution of this linear differential equation will be

$$\mathbf{R}(t) = exp([\omega]_\times t) \tag{2.7}$$

where *exp* is the matrix exponential

$$exp(A) = I + A + \frac{(A)^2}{2!} + \frac{(A)^3}{3!} + ... \tag{2.8}$$

Therefore we have a mapping from any skew-symmetric matrix $[\omega]_\times$ to a rotation matrix $R = exp([\omega]_\times)$. We can also say all the possible $[\omega]_\times$ form the tangent space at identity of the Lie group SO(3). This tangent space is the Lie algebra so(3):

$$\mathrm{so}(3) = \{[\omega]_\times \in \mathbb{R}^{3\times 3} | \omega \in \mathbb{R}^3\} \tag{2.9}$$

so(3) is linear with 3 degrees of freedom. so(3) and SO(3) are connected by exponential map:

$$exp : \mathrm{so}(3) \to \mathrm{SO}(3); [\omega]_\times \to \mathbf{R} \tag{2.10}$$

Geometrically, ω is the axis-angle parameterization of the rotation as shown in Figure 2.4. The angle of the rotation is $\|\omega\|$, where $\|...\|$ represents the L^2 norm. The axis of the rotation is $\frac{\omega}{\|\omega\|}$. Further derivation of Equation 2.8 results in the following closed-form relation:

$$\mathbf{R} = \exp([\omega]_\times) = \begin{cases} I & \|\omega\| = \mathbf{0} \\ I + \frac{\sin\|\omega\|}{\|\omega\|}[\omega]_\times + \frac{1-\cos\|\omega\|}{\|\omega\|^2}[\omega]_\times{}^2 & \text{otherwise} \end{cases} \tag{2.11}$$

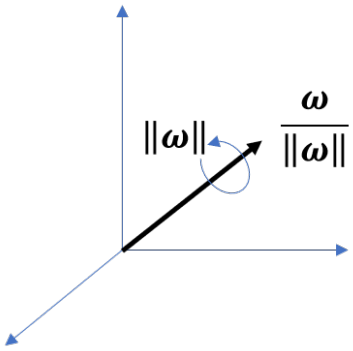


Figure 2.4: The axis-angle parameterization of a 3D rotation. The 3D rotation is parameterized by a $\omega \in \mathbb{R}^3$. The axis of the rotation is $\frac{\omega}{\|\omega\|}$. The angle of the rotation is $\|\omega\|$

Figure 2.5 visualizes the exponential map. A tangent plane is fit to the identity point. Tangent vectors in the same direction are mapped to the same geodesic path in the non-Euclidean space. $I \cdot exp([\omega]_x)$ means moving along the geodesic path in the direction of $\frac{\omega}{\|\omega\|}$ for an amount of $\|\omega\|$.

2.2.2 Rigid Transforms and the Special Euclidean Group

The special Euclidean group SE(3) is the group of all rigid transforms in $\mathbb{R}^3$. A 3D rigid transform $\mathbf{G}$ in homogeneous coordinates is composed of a rotation and a translation:

$$\mathbf{G} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \text{ with } \mathbf{R} \in \mathrm{SO}(3) \text{ and } \mathbf{t} \in \mathbb{R}^3 \tag{2.12}$$

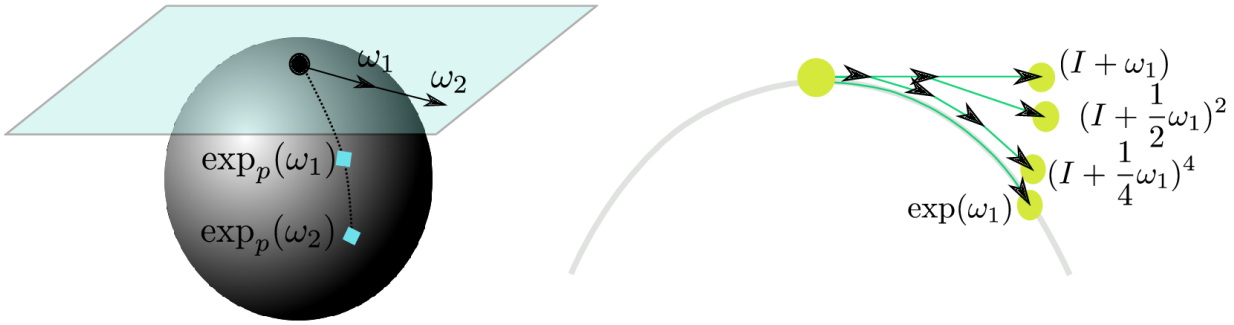


Figure 2.5: Illustration of the exponential map. The exponential map fits a linear tangent plane (actually it is a 3D plane for so(3)) to the identity point of the non-Euclidean space. Vectors in the same direction correspond to a geodesic path in the non-Euclidean space. This allows us to compute derivatives in the tangent space and move the point along a geodesic flow in the non-Euclidean space. This figure is from [74].

$\mathbf{G}$ is a rigid transformation matrix between two 3D points. It has 6 degrees of freedom (DoF); three come from rotation, and three come from translation. Analagous to SO(3), SE(3) has an associated Lie algebra se(3):

$$\mathbf{se}(3) = \left\{ \mathbf{g} = \begin{pmatrix} [\omega]_\times & v \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{4\times4} | [\omega]_\times \in \mathrm{so}(3), v \in \mathbb{R}^3 \right\} \tag{2.13}$$

The so called twist matrix $\mathbf{g}$ and the transform matrix $\mathbf{G}$ are also connected by the exponential map $\mathbf{G} = exp(\mathbf{g})$. There exists a compact representation $\xi \in \mathbb{R}^6$ of each $\mathbf{g}$ and $\mathbf{G}$:

$$\xi = \begin{pmatrix} v \\ \omega \end{pmatrix} \tag{2.14}$$

2.2.3 Optimization on se(3)

As mentioned in the beginning of this section, if we directly apply $G_{k+1} = \delta G + G_k$ to solve $\hat{G} = \mathrm{argmin}(f(G))$, G_{k+1} has no reason to be still inside SE(3) because the constraints of SE(3) can be violated; in particular, it should contain a rotation matrix in the upper left corner. In contrast, the Lie algebra se(3) does not have any constraint on the 6 DoF space. Linearized pose-increments will be expressed as the coefficients ($\delta\xi$) of Lie-algebra elements $\mathbf{g}_{\delta\xi} \in \mathrm{se}(3)$. In other words, with the exponential map, rotations can be linearized for the iterative linearize-solve-update strategy.

The new update step will be

$$G_{k+1} = exp(\mathbf{g}_{\delta\xi}) \cdot G_k \tag{2.15}$$

which is based on the right invariant vector field [1]. The resulting G_{k+1} is guaranteed to be inside SE(3). Sometimes, with a slight abuse of notation, the above equation is simplified as

$$G_{k+1} = exp(\delta\xi) \cdot G_k \tag{2.16}$$

During optimization, the derivatives (Jacobian) are computed using the chain rule:

$$\frac{\partial f}{\partial \xi} = \frac{\partial f}{\partial \mathbf{G}} \frac{\partial \mathbf{G}}{\partial \xi} \tag{2.17}$$

The state $\mathbf{G}$ is finally updated by Equation 2.16.

2.3 Epipolar Geometry

This section aims to introduce some basic concepts in two view geometry.

When two cameras are looking at the same set of 3D points, the 2D projection locations of those points in each camera follows the epipolar constraint, which is illustrated in Figure 2.6. In this figure the camera center of the two cameras are represented by C and C'. Each of them is a 3D vector. Without a loss of generality, we assume that $C = [0, 0, 0]^T$ and that the camera coordinate system of C is the same as the world coordinate system. The two planes represents the image plane of the two cameras; they are orthogonal to the cameras' viewing directions. To simplify the problem, we assume that the distances between the image planes and the camera centers equal 1.0. A 3D point is represented by $X = [x_1, x_2, x_3]^T$. X is the 3D coordinates of that point in the world coordinate system, and because C shares the same coordinate system as the world coordinate system, X is also the coordinates of that point in C. The projection of X onto C's image plane is represented by

[1] To be consistent with DSO [4] which is introduced in Section 2.5.3

$x = [x_1, x_2]^T$. X and x have the following relation:

$$\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \frac{1}{X_3} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \tag{2.18}$$

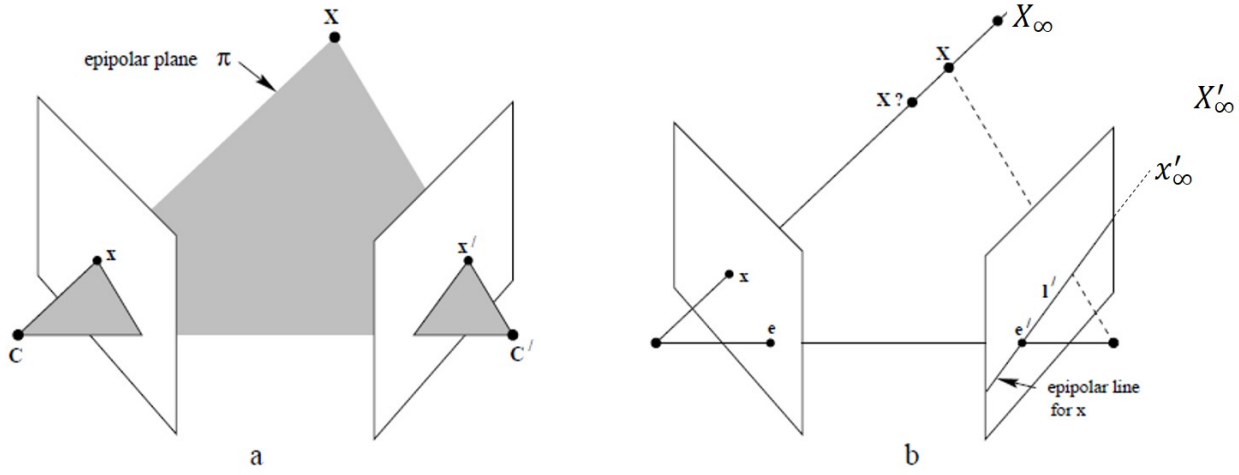


Figure 2.6: Epipolar geometry illustration. Left: basic epipolar geometry illustration. The two camera centers are represented by C and C'. Their image formation planes are drawn in the picture. X is a 3D point. Its projection onto the two images are x and x'. Right: point-to-line correspondence. When X is unknown, the possible locations of x' are on a line called an epipolar line. The projections of the camera centers onto the other camera are called epipoles (e and e').

Similarly, $x' = [x'_1, x'_2]$ represents X's projection onto the other image. If we write X's coordinates in the coordinate system of C' as X', we have

$$\begin{bmatrix} x'_1 \\ x'_2 \\ 1 \end{bmatrix} = \frac{1}{X_3} \begin{bmatrix} X'_1 \\ X'_2 \\ X'_3 \end{bmatrix}, \text{ where } \begin{bmatrix} X'_1 \\ X'_2 \\ X'_3 \end{bmatrix} = R \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + t \tag{2.19}$$

From Equation 2.18 and 2.19, we have

$$\frac{X'_3}{X_3} \begin{bmatrix} x'_1 \\ x'_2 \\ 1 \end{bmatrix} = R \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} + \frac{t}{X_3} \tag{2.20}$$

When the depth of the 3D point in C (X_3) is known, a unique point-to-point correspondence can be establish as in Equation 2.20. However, in most cases we only have the two images (x and x') and the depth is unknown. The location of the 3D point can vary along a ray defined by C and $[x_1, y_1, 1]^T$. The projection of that ray in the other image is a straight line. This line is called an **epipolar line**. It is only possible for x' to lie on this line. The projection of C itself in the coordinate system of C' is called an **epipole** (e'). It can be observed that all epipolar lines must pass through the epipole. The plane defined by C, C' and X is called an **epipolar plane**. The distance between C and C' is called the **baseline**. For stereo depth estimation algorithms, the epipolar constraint is an extremely useful tool to reduce the search space (finding point correspondence for each pixel) from a 2D image to a line.

Geometrically, we found a point-to-line correspondence that is independent of the location of X. Algebraically we can also find such relationship by setting X_3 to infinity; Equation 2.20 will become

$$\begin{bmatrix} x'_{\infty 1} \\ x'_{\infty 2} \\ 1 \end{bmatrix} = H_\infty \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, H_\infty = \frac{X_3}{X'_3} R \tag{2.21}$$

where x'_∞ represents the projection of the infinite 3D point in C'. The epipolar line in the image of C' can be expressed as $e' \times H'_\infty = [e']_\times H_\infty x$. The fact that x' is on that line can be written as $x'^T [e']_\times H_\infty x = 0$, which is more generally written as

$$x' F x = 0 \tag{2.22}$$

This F matrix is called the **fundamental matrix**. In the above derivation we assumed that the cameras are simply projecting onto the planes of depth 1 (canonical cameras). In such a setting F is also called the **essential matrix** (E). E is determined by the relative camera rotation and translation and thus has at most 6 DoF, but because $x' E x = 0$ even after we multiply E by any scalar, E is scale ambiguous and has only 5 DoF. Intrinsically, this loss of one DoF is because the projection process is scale ambiguous (absolute depth is unknown).

Compared to E, the fundamental matrix is a much more general matrix describing the epipolar constraint between two images. Even if the points in 2D images are subject to a linear transformation,

the point-to-line constraint still holds. That means the F matrix not only involves extrinsic camera motion but also encodes intrinsic image formation information. F thus has higher DoF. Because the maximum DoF of a 3×3 matrix is 9 and F is scale ambiguous (for the same reason as E), it has at most 8 DoF. Further, because all epipolar lines pass through an epipole, F has a non-zero null space (F has rank 2). That means that the determinant of F equals zero and this extra constraint causes F to lose another DoF. F **has 7 DoF**. Another property of F is that its transpose F^T is also a fundamental matrix. F maps x to a line in C', and F^T maps x' to a line in C.

In theory the fundamental matrix can be estimated (up to a scale factor) given 7 pairs of matching keypoints (x and x'). The normalized eight point algorithm [75] is a widely used direct linear transform algorithm to estimate F based on 8 pairs of matchings. When there are a large number of matchings, RANSAC algorithms are often used to deal with outlier matchings. The fundamental matrix is a very useful matrix to estimate the relative camera poses between two images. This topic is out of the scope of this dissertation. Read Hartley and Zisserman's book [75] for details.

2.4 Simultaneous Localization and Mapping (SLAM)

SLAM is a problem in robotics and computer vision. It takes a video as input, which can be either monocular or binocular, or even some other modalities. As the name indicates, the expected output is composed of the "localization" part and the "mapping" part.

The localization part computes the camera trajectory, *i.e.*, to localize the camera. The trajectory is composed of a sequence of camera poses. Each camera pose is a 6-DoF (degree of freedom) tuple as introduced in Section 2.2. Localization is the essential goal of SLAM in robotics.

An important concept in SLAM is called the *keyframe*. Because of the redundancy in video frames, it is not efficient or necessary to record every frame in the video. Therefore, keyframes are selected dynamically during SLAM to represent the camera trajectory. The latest keyframe is used as the reference frame for any incoming frames. The relative camera poses of the incoming frames are first computed with regard to the latest keyframe. A new frame will become a new keyframe if it meets certain criteria, e.g., whether the camera transformation is large enough or whether difference of the image contents is large enough, etc.

The mapping part of SLAM computes a 3D scene. Compared to the localization, this part of the problem definition has more freedom: the 3D scene representation is not uniform across different methods. For example, the ORB-SLAM [6, 7] computes 3D point locations of the matching feature

points and use those sparse 3D points as the 3D scene representation. The LSD-SLAM [5] and DSO [4] compute depth values for some pixels in each keyframe. Because the camera poses of the keyframes are computed in the localization part, the absolute 3D position of the pixels with depth is thus known.

In standard SLAM systems the camera pose and the 3D scene associated to each keyframe are usually solved together in an optimization problem. That is why the problem is called Simultaneous Localization and Mapping.

With this basic overview of the mapping, we can see that keyframes also play an important role in the mapping. First, the 3D scene information is stored on the basis of keyframes as mentioned. More importantly, the optimization mechanism in the mapping module usually jointly optimizes the 3D information and the camera poses of several keyframes. Choosing keyframes can increase the disparity between the frames being optimized and thus reduce the ambiguity in the optimization.

Typically a SLAM system uses a maximum-likelihood approach to estimate the camera poses and the 3D scene. Depending on whether a SLAM system directly uses image intensities to compute the loss or uses some intermediate representation of the image to compute the loss, it can be classified as a "direct" or "indirect" method.

2.4.1 Direct Methods

The fundamental formulation of a direct method's error function is to compare the intensity of a point (p) in one image with the intensity of the point's projection point (p') in another image. Suppose that p is associated with a depth value d_p and that the relative camera pose transformation matrix between p and p' is represented by T. Then p and p' are connected by the following equation:

$$\boldsymbol{p'} = \Pi(T\Pi^{-1}(\boldsymbol{p}, d_{\boldsymbol{p}})) \tag{2.23}$$

where Π is the projection function from 3D to 2D image coordinates including camera calibration. With a depth value $d_{\boldsymbol{p}}$, Π^{-1} can retrieve the 3D position of a 2D point. Then a basic *photometric error* can be computed as

$$E = \|I'[\boldsymbol{p'}] - I[\boldsymbol{p}]\|_2 \tag{2.24}$$

where I and I' stand for the two images. $I[p]$ stands for the intensity at p in I. Here I am using the 2-norm ($\|...\|_2$) as an example, but different norms can be used. Moreover, this is just one single term the photometric error computed from a single pair of points. In real applications the optimization system takes multiple points and cameras at the same time. In this equation we are optimizing over the parsimonious representation (ξ) of T and d_p.

Examples of direct methods include the DSO [4], the LSD-SLAM [5], and others [76, 77, 78].

2.4.2 Indirect Methods

In indirect methods, there is an intermediate step between the image intensities and the loss function. A representation of the image is computed and used to compute the loss. The most widely used representation is the so-called keypoints. Keypoints are extracted from the regions of salient points by a feature detection algorithm. For example, the SIFT algorithm extracts keypoints from multiscales and represents each keypoint by $[x, y, s, \theta]$ where x and y are the pixel location of the keypoint, s is the scale factor and θ is the orientation.

After the feature detection step, a local descriptor is extracted at each keypoint to represent the pattern in the local region. For example, a SIFT descriptor is a 128-D vector of concatenated gradient histograms of the local patch around $[x, y]$ and at appropriate scale and orientation. After this step, an image is represented by a large number of keypoints and their descriptors. The keypoints in two images are then matched according to the distance between their descriptors. Two points are connected if they are mutually closest to each other in terms of inter-descriptor distance. For two images, a group of keypoint-to-keypoint correspondences are established. For more than two images, a chain of keypoint correspondences can contain more than two 2D points. The keypoints in a chain all correspond to the same 3D point.

Indirect methods usually represent the 3D scene by the absolute 3D locations of the keypoints. A pair (or a group) of matching keypoints should share the same 3D point. The loss is the distance between the reprojected points and their actual measurements (matching keypoints). For example, a basic formulation of this "*reprojection error*" is as follows:

$$E = \sum_{i \in N(P)} \|\Pi T_i P - p_i\| \tag{2.25}$$

where P is the 3D position, $N(P)$ is the set of image indices that contain matching 2D points of P,

p_i is the respective 2D point in the i-th image, and T_i is the absolute camera pose matrix of the i-th image. The noise in the measurements (p_i) combined with the system being over-determined means the projected point ($\Pi T_i P$) may not be exactly the same as the keypoints. The difference between them is what we want the system to minimize. The arguments being optimized over are the camera pose representations of T_i and the 3D locations (P).

ORB-SLAM [6, 7] is an example of an indirect method. ORB features are used to establish point correspondences.

2.4.3 Direct vs Indirect

A major advantage of indirect methods is the well-established bundle adjustment algorithm. A standard bundle adjustment optimizes the reprojection error over a group of camera poses and a group of 3D keypoint locations. Many standard libraries [79] have been developed for this setup. Bundle adjustment, because of its computational cost, is not always used in SLAM systems and is more used in SfM. A successful application of bundle adjustment is the windowed bundle adjustment in ORB-SLAM [6].

Another advantage of indirect methods lies in the consistent representation of the image for both tracking and loop closure. In tracking, keypoints are matched according to the distances between descriptors. Loop closure is the process to retrieve historical images that share overlapping visual content with the current image. Images represented by keypoints and descriptors can be efficiently retrieved by an inverted list like the visual vocabulary tree. There is no need to develop special representations of images to do image retrieval. Also, point-to-point correspondences can be easily established for the current image and the retrieved image in exactly the same way as the ordinary tracking. A global bundle adjustment to "close the loop" (reduce drift) can also be applied in the same way as the local bundle adjustment thread. This can make the whole system more consistent, easier to implement and more robust.

Basically, an indirect method will work well if the keypoints can be matched well, that is, if the feature extraction algorithm can give enough inlier (correct) point-to-point matchings between images. The performance of the feature extraction algorithms is affected by the quality of the images. If the quality of the images is high, the algorithms can usually tolerate high image variance, e.g., scale change, global brightness change, rotation, etc. Unfortunately, for colonoscopic images, there are not enough salient points to detect, and the tracking function of indirect SLAM methods easily

crashes.

Comparatively, direct methods do not rely on successful extraction of keypoints. As claimed in [4], direct methods offer more photometric and geometric robustness for low quality images. By directly matching the intensities, direct methods can potentially match image regions that cannot be recognized by features. Another benefit of direct method is that the 3D scene representation no longer relies on the feature points. In other words, the density of the 3D scene is not limited by the number of extracted feature points. More fine-grained representation of 3D scenes can be used.

A challenge of indirect methods is the need to design a customized bundle adjustment algorithm because direct methods usually store 3D information as the depth values of pixels. Direct methods also do not provide a consistent way for loop closure. A special image retrieval algorithm has to be designed if loop closure is desired.

The RNNSLAM part of this dissertation is based on a direct method called Direct Sparse Odometry (DSO), which is reviewed in Section 2.5.

2.5 Direct Sparse Odometry

The two major components of DSO are tracking and local windowed optimization. Tracking is the process of computing a tentative camera pose of each incoming frame. Local windowed optimization is the process of jointly optimizing the camera poses and certain depth values of the latest n_w keyframes. This step is also called local bundle adjustment. One important background area of local windowed optimization is marginalization by Schur complement, so that will be introduced specifically. I will review tracking in Section 2.5.1, review marginalization in Section 2.5.2 and review local windowed optimization in Section 2.5.3

2.5.1 Tracking

In DSO the tracking module used a conventional method called *direct image alignment* [77]. It computes a camera pose by minimizing a photometric error E between the current frame I and its reference keyframe K. If $\boldsymbol{p}'$ is the corresponding point of $\boldsymbol{p}$ in the current image I,

$$E = \sum_{\boldsymbol{p} \in \mathcal{P}_K} \alpha_{\boldsymbol{p}} \| I[\boldsymbol{p}'] - K[\boldsymbol{p}] \|_\gamma \tag{2.26}$$

$\boldsymbol{p}$ projects to $\boldsymbol{p}'$ through Equation 2.23. $\mathcal{P}_K$ is a set of points in image K that are sparsely sampled in regions with depth values and that have sufficient gradient magnitude. $\|...\|_\gamma$ is the Huber norm[2]. $\alpha_{\boldsymbol{p}}$ is a weighting parameter depending on the gradient magnitude at $\boldsymbol{p}$:

$$\alpha_{\boldsymbol{p}} = \frac{c^2}{c^2 + \|\nabla I[\boldsymbol{p}]\|_2^2} \tag{2.27}$$

where c is a constant. To well-constrain the problem, the residual is taken over a small neighborhood $\mathcal{N}_p$ (8 pixels) around $\boldsymbol{p}$:

$$E = \sum_{\boldsymbol{p}\in\mathcal{P}_K} \sum_{\hat{\boldsymbol{p}}\in\boldsymbol{\mathcal{N}_p}} \alpha_{\hat{\boldsymbol{p}}} \|I[\hat{\boldsymbol{p}}'] - K[\hat{\boldsymbol{p}}]\|_\gamma \tag{2.28}$$

Finally, to make the process more robust to global brightness change, two affine transformations (by a_K, a_I, b_K, b_I) of the intensities are applied to $I[\hat{\boldsymbol{p}}']$ and $K[\hat{\boldsymbol{p}}]$ respectively:

$$E = \sum_{\boldsymbol{p}\in\mathcal{P}_K} \sum_{\hat{\boldsymbol{p}}\in\boldsymbol{\mathcal{N}_p}} \alpha_{\hat{\boldsymbol{p}}} \|I[(\hat{\boldsymbol{p}}'] - b_I) - \frac{e^{a_I}}{e^{a_K}}(K[\hat{\boldsymbol{p}}] - b_K)\|_\gamma; \tag{2.29}$$

a_K, a_I, b_K, b_I are also jointly optimized. This energy is minimized on an image pyramid from coarse level to fine by the Gauss-Newton method. Read [77, 4, 20] for details.

2.5.2 Marginalization and Schur Complement

Tracking provides a good initial pose of each incoming image. If an image is established as a keyframe (the strategy will be introduced in Section 2.5.3), it will be included into local windowed optimization, which is critical to SLAM's accuracy. Local windowed optimization uses a non-linear least squares approach to jointly optimize the camera poses, certain sparse depth values and other parameters of several keyframes. It will be introduced in detail in the next section. It is first necessary to understand marginalization, an essential step of local windowed optimization. Because a "window" is maintained to limit the amount of computation, some old variables need to be marginalized (removed/finalized) from the current optimization system. This section introduces the mathematical basics of marginalization.

[2] $\|a\|_\gamma = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| < \gamma \\ \gamma(a - \frac{\gamma}{2}) & \text{otherwise} \end{cases}$

In a generic nonlinear least squares problem minimizing $E(\mathbf{x}) = \sum_{i=1}^{n} r_i^2(\mathbf{x})$ where $\mathbf{x}$ is the set of parameters and r_i is a residual, suppose $\mathbf{x_c}$ is the current value of $\mathbf{x}$. We can perform the following Gauss-Newton approximation about the linearization point $\mathbf{x_c}$:

$$E(\mathbf{x}) \approx -2(\mathbf{x} - \mathbf{x_c})^T b + (\mathbf{x} - \mathbf{x_c})^T H(\mathbf{x} - \mathbf{x_c}) + E(\mathbf{x_c}), \text{ where } b = -J^T \mathbf{r} \text{ and } H = J^T J; \quad (2.30)$$

$\mathbf{r}$ is the residual vector estimated at $\mathbf{x_c}$, and J is an approximation of the Jacobian of $\mathbf{r}$; for the time being J is estimated at $\mathbf{x_c}$. To minimize such an energy, we use the Gauss-Newton method to compute a change $\delta\mathbf{x}$ in $\mathbf{x}$:

$$H(\mathbf{x} - \mathbf{x_c}) = H\delta\mathbf{x} = b \quad (2.31)$$

When the optimization system gets too large, we need to reduce the its size by marginalizing some variables. Suppose $\mathbf{x}$ is composed of the portion we want to marginalize ($\mathbf{x_m}$) and the portion we want to keep ($\mathbf{x_k}$). Then the Equation 2.31 can be written as

$$\begin{bmatrix} H_{mm} & H_{mk} \\ H_{km} & H_{kk} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x_m} \\ \delta\mathbf{x_k} \end{bmatrix} = \begin{bmatrix} b_m \\ b_k \end{bmatrix} \quad (2.32)$$

Our goal is to get a new linear system that involves only $\mathbf{x_k}$ ($H'_{kk}\delta\mathbf{x_k} = b'_k$). This is obviously not equivalent to $H_{kk}\delta\mathbf{x_k} = b_k$. In other words, simply extracting a matrix block from the current Hessian matrix (H) does not work. The new linear system can be formed using the Schur complement:

$$H'_{kk} = H_{kk} - H_{km}H_{mm}^{-1}H_{mk},\ b'_k = b_k - H_{km}H_{mm}^{-1}b_m \quad (2.33)$$

Then we solve

$$H'_{kk}\delta\mathbf{x_k} = b'_k \quad (2.34)$$

The information contained in $\mathbf{x_m}$ has not been trivially thrown away; it has been injected as prior information to the new optimization system.

This new linear system is minimizing the following energy:

$$E'(\mathbf{x_k}) = -2(\mathbf{x_k} - \mathbf{x_{kc}})^T b'_k + (\mathbf{x_k} - \mathbf{x_{kc}})^T H'_{kk}(\mathbf{x_k} - \mathbf{x_{kc}}) + E'(\mathbf{x_{kc}}) \quad (2.35)$$

This energy is the remaining energy after margianlization. It is the remaining error from present keyframes. When a new keyframe comes in, new error terms will be added to E'. This combination will be used in the next local windowed optimization to solve for new increments. All constant terms can be ignored.

In practice, the H and b of current optimization system are constantly being maintained to solve for new update increments. Adding new energy terms will enlarge (concatenate rows and columns to) the propagated H'_{kk} and b'_k.

Since we already performed linearization in this marginalization, we should use the same approximation to J in all the following local windowed optimizations and marginalizations. In other words, $\mathbf{x_c}$ will move, but J will stay for linearization consistency. For example, if there is a new residual r^{new} involving an old parameter x^{old}, the derivative $\frac{\partial r^{new}}{\partial x^{old}}$ should be computed at the linearization point that x^{old} used in its first-ever marginalization. This is because we are adding the propagated error terms with the new terms. If they use different J values for the same parameters, the null space of the optimization problem will collapse. For SLAM problems the null space is an absolute rigid body transform plus an absolute scale (7 DoF). This is the so-called similarity ambiguity. Such an intrinsic property will be destroyed if different J values are used for the same parameter in one error. The prediction would be biased to some "nonexistent" information.

Summarizing this section, marginalization by Schur complement helps us to reduce the size of one optimization problem and to propagate error terms to the next optimization. It serves as a bridge between two local windowed optimizations when there is an overlap between their parameters.[3] In the next section I will go into the local windowed optimization.

2.5.3 Local Windowed Optimization

Local windowed optimization / local bundle adjustment jointly optimizes the camera poses and some sparse[4] depth values for the latest n_w keyframes ($n_w = 7$ in the implementation of [4]). To fully understand it, we need to consider two aspects: 1) back-end optimization, the mathematical

[3]Schur complement is also used to solve $H\delta\mathbf{x} = b$ efficiently. This will be discussed in the front-end part of Section 2.5.3.

[4]The reason that DSO uses sparse depth values is that sparse points do not require point-to-point constraints (like smoothness) and can maintain the sparsity of the Hessian matrix, which is critical to solving $H\delta\mathbf{x} = b$ efficiently. This will be detailed at the end of this section. In our colonoscopic application, we use DSO-optimized camera poses to fuse the dense depth maps produced by RNN-DP.

basics for solving an optimization problem for poses and depths; 2) front-end management, the strategy of forming the optimization problem, choosing which parameters to solve for and managing the keyframes.

Back-end: In Section 2.5.1 we studied the photometric error (Equation 2.29) used in tracking. It optimizes over a relative camera pose between I and K and the affine intensity parameters. For a windowed optimization, the formulation is similar but involves a window ($\mathcal{W}$). That is, it is extended to the following format:

$$E_{total} = \sum_{K\in\mathcal{W}} \sum_{\boldsymbol{p}\in\mathcal{P}_K} \sum_{I\in obs(\boldsymbol{p})} \sum_{\hat{\boldsymbol{p}}\in\mathcal{N}_{\boldsymbol{p}}} \alpha_{\hat{\boldsymbol{p}}} \| I[(\hat{\boldsymbol{p}}'] - b_I) - \frac{e^{a_I}}{e^{a_K}}(K[\hat{\boldsymbol{p}}] - b_K) \|_\gamma \tag{2.36}$$

$\mathcal{W}$ contains the latest n_w keyframes. $\mathcal{P}_K$ is the set of 2D points hosted by keyframe K. The sampling strategy of these points will be introduced in detail in the front-end section. $obs(\boldsymbol{p})$ is the set of other keyframes in $\mathcal{W}$ that observe $\boldsymbol{p}$.

The variables being optimized over are the camera poses of the keyframes in $\mathcal{W}$, the depth values (actually inverse depth) of the points in all $\mathcal{P}_K$'s and the affine intensity parameters. Let's denote the variables by a vector $\mathbf{x} \in \mathrm{se}(3)^{n_w} \times \mathbb{R}^m$ where m is the number of variables including the depth values and the affine intensity parameters of each keyframe. Therefore, $\mathbf{x}$ is vector of dimension $d = 6n_w + m$.

The Gauss-Newton method is used to minimize the energy in Equation 2.36:

$$\delta\mathbf{x} = H^{-1}b, \text{ where } H = J^T A J,\ b = -J^T A \mathbf{r} \tag{2.37}$$

where $\mathbf{r} \in \mathbb{R}^n$ is the (weighted) residual vector. The dimension n of $\mathbf{r}$ equals the total number of elements in Equation 2.36. $A \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing the weights ($\alpha_{\hat{\boldsymbol{p}}}$) of all the terms. $J \in \mathbb{R}^{n\times d}$ is the Jacobian matrix of $\mathbf{r}$. $J^T A J \in \mathbb{R}^{d\times d}$ is the Gauss-Newton approximation of the Hessian of E_{total}.

As in Equation 2.16, the camera pose variables in $\mathbf{x}$, *i.e.*, part of $\mathrm{se}(3)^{n_w}$, update their non-Euclidean rigid transform matrices ($\mathrm{SE}(3)^{n_w}$) by the exponential map and multiplicative increments. Taking this into consideration, we denote an update of the non-Euclidean state $\mathbf{X} \in \mathrm{SE}(3)^{n_w} \times \mathbb{R}^m$ by $\mathbf{X}_{\text{new}} = \mathbf{x} \circ \mathbf{X}$. $\mathbf{X}$ includes the rigid transform matrices, depth values, and affine intensity parameters.

The Euclidean portion of $\mathbf{X}$ is still updated by ordinary additive increments. If the linearization point is $\mathbf{X_0}$ and the accumulated update is $\mathbf{x}$, the current state will be $\mathbf{X} = \mathbf{x} \circ \mathbf{X_0}$.

For each element r_i in $\mathbf{r}$,

$$r_i = (I[\hat{\boldsymbol{p}}'] - b_I) - \frac{e^{a_I}}{e^{a_K}}(K[\hat{\boldsymbol{p}}] - b_K). \tag{2.38}$$

The state involved in this term includes the camera pose matrices G_I and G_K, the depth value d_p and a_I,b_I,a_K,b_K. The respective row in J is computed with respect to an additive increment of $\mathbf{x}$:

$$J_i = \frac{\partial r_i((\delta + \mathbf{x}) \circ \mathbf{X_0})}{\partial \delta} \mid_{\delta=0} \tag{2.39}$$

An expanded version of this equation is

$$J_i = \left[\frac{\partial I[\hat{\boldsymbol{p}}']}{\partial \hat{\boldsymbol{p}}'} \frac{\partial \hat{\boldsymbol{p}}'}{\partial \delta_{geo}}, \frac{\partial r_i}{\delta_{photo}} \right] \tag{2.40}$$

where δ_{geo} includes the variables of two camera poses and a depth value. They influence the projected point location. δ_{photo} is a_I,b_I,a_K,b_K. This formulation regards $\hat{\boldsymbol{p}}'$ as a function of δ_{geo} and regards $K[\hat{\boldsymbol{p}}]$ as a constant. A naive idea is to estimate this Jacobian at the current state $\mathbf{X}$. However, because E_{total} is a composition of both new terms contributed by the latest keyframe and the old terms from the marginalization of the previous bundle adjustment, we have to pay attention to use a consistent J approximation between old terms and new terms. As a reminder, J is used to compute H and b in Equation 2.37 and will thus influence the computation of $\delta\mathbf{x}$

As mentioned in Section 2.5.2, after marginalization the linearization point of remaining variables ($\mathbf{x_k}$) should no longer change because some remaining error terms involving them will be added to the next bundle adjustment. In Equation 2.36, all the variables belonging to the old keyframes (depth values, poses and affine intensity parameters) should use the same linearization point as the one propagated from previous marginalization. In fact, it is the first-ever marginalization involving them. Some variables may have been through multiple rounds of marginalization and local bundle adjustment, but their portions of $\mathbf{X_0}$ should never change. Only their portions of $\mathbf{x}$ get accumulated. Because the Jacobian is also a linearization, it should also stay the same after the first marginalization for the old variables. In detail, in Equation 2.40 $\frac{\partial \hat{\boldsymbol{p}}'}{\partial \delta_{geo}}$ and $\frac{\partial r_i}{\partial \delta_{photo}}$ are always

estimated at $\mathbf{X_0}$ ($\mathbf{x} = 0$) rather than the current $\mathbf{X}$ for the old variables. This is the so-called First-Estimate-Jacobian [80] technique. Although there can be an error between the true Jacobian and the First-Estimate-Jacobian, this is in fact a good approximation because $\frac{\partial \hat{\boldsymbol{p}}'}{\partial \delta_{geo}}$ and $\frac{\partial r_i}{\partial \delta_{photo}}$ are quite smooth in SLAM. It turns out that not only the efficiency but also the accuracy of SLAM can be improved by this technique [80] despite the approximation, because the null space corruption is prevented. $\frac{\partial I[\hat{\boldsymbol{p}}']}{\partial \hat{\boldsymbol{p}}'}$ only involves the image gradient and does not affect the null space of the system, so this term is always computed at the current state $\mathbf{X}$.

In contrast to the old variables, the new variables of the latest keyframes are not constrained because they haven't been through marginalization yet. Therefore, after each Gauss-Newton iteration in the form of

$$\mathbf{x}_{new} = \mathbf{x} + \delta = x + H^{-1}b, \tag{2.41}$$

the linearization point $\mathbf{X_0}$ of the new variables will be updated ($\mathbf{X_0} = \mathbf{x} \circ \mathbf{X_0}$). The respective part of $\mathbf{x}$ has been set to 0. (The part of old variables is accumulated instead.) The linearization point will be fixed after the first marginalization involving this keyframe.

Front-end In the back-end part I reviewed the mathematical basics for solving a local bundle adjustment problem. In this part I will review how DSO forms successive local bundle adjustment problems, i.e., how to determine $\mathcal{W}$ and $\mathcal{P}_K$ of Equation 2.36. I will also review the strategy of choosing variables to marginalize. Specifically, as mentioned before, the camera trajectory of SLAM is represented by the camera poses of a set of keyframes chosen from video frames to reduce redundancy. The 3D scene of DSO is represented by the inverse depth values of some sparse points in the keyframes. A point, as well as its inverse depth value, is thus "hosted" by a keyframe. The set of sparse points hosted by a keyframe K was denoted as $\mathcal{P}_K$ in both Equation 2.29 and Equation 2.36. Next, I will review a) how a keyframe is created, b) how to select points from a keyframe, and c) how to marginalize a keyframe.

a) Keyframe Creation: When a new frame comes in, the tracking module is first used to estimate a tentative camera pose. As mentioned in Section 2.5.1, the tracking module optimizes a photometric error $E = \sum_{\boldsymbol{p} \in \mathcal{P}_K} \sum_{\hat{\boldsymbol{p}} \in \mathcal{N}_{\boldsymbol{p}}} \alpha_{\hat{\boldsymbol{p}}} \| I[(\hat{\boldsymbol{p}}'] - b_I) - \frac{e^{a_I}}{e^{a_K}}(K[\hat{\boldsymbol{p}}] - b_K) \|_\gamma$. This error uses the latest keyframe K as the reference; all the points ($p \in \mathcal{P}_K$) in K that have depth values are projected to the current

frame I. Here we need to pay special attention to the set $\mathcal{P}_K$: In DSO all the points hosted by other keyframes inside the local window are first projected to K. The projected points and the points hosted by K itself together form a semi-dense depth map. $\mathcal{P}_K$ includes all the points in this semi-dense depth map. Given the tracking result of a new frame, it will be established as a new keyframe if it meets one of the following three criteria:

1. There is a large change in the field of view between I and K. This is estimated by the mean squared point displacement $(\frac{1}{|\mathcal{P}_K|}\sum_{\boldsymbol{p}\in\mathcal{P}_K}\|\boldsymbol{p}-\boldsymbol{p}'\|_2^2)^{\frac{1}{2}}$.

2. The magnitude of the translation between I and K is above a threshold.

3. The relative brightness factor between I and K ($|a_K - a_I|$, as defined in Equation 2.29) is above a threshold.

b) Point Management: This part reviews how to sample the points hosted by each keyframe ($\mathcal{P}_K$ in Equation 2.36) in local windowed optimization. We first define two kinds of points: candidate points and active points. Candidate points are the points initially sampled from each new keyframe. Active points are the points chosen from the candidate points and finally used in the local windowed optimization. $\mathcal{P}_K$ actually refers to the active points.

In order to limit the size of the optimization problem, the total number of active points is restricted to n_p (2000 points[5]). These n_p points are hosted by different keyframes in the local window. Next I will review 1) how the candidate points are selected, 2) how their initial depth values are estimated, and 3) how the active points are chosen.

1. Candidate points: When a keyframe is created, n_p candidate points are sampled from the image. Ideally, the points should be sampled at regions with sufficiently high gradients in reasonably distinct directions. They should also be well distributed across the whole image. An adaptive gradient threshold is computed for each 32×32 block in the image. Pixels whose gradient magnitudes are maximal in local windows and are also above the respective thresholds will be selected as candidate points. If the number of candidate points is not enough, the

[5]This is an empirical value used in DSO[4], used to sparsely represent the scene.

threshold will be decreased and the window size will be increased. The selection process will be repeated until enough candidate points are selected.

2. Initial depth: the initial depth value of each candidate point is estimated by searching on the epipolar line (detailed in Section 2.3).

3. Active points: Because some active points are marginalized together with some old keyframes (detailed in the upcoming section c: marginalization strategy), the total number of active points will be smaller than n_p. Some candidate points of the new keyframe will be chosen to maintain enough active points. First, all the existing active points are projected to the reference keyframe K. All the candidate points are also projected to K. The candidate points that maximize the distances to existing active points will be activated.

c) Marginalization Strategy: Marginalization is needed when we need to remove a keyframe and its associated active points from the optimization window. Keyframes are marginalized based on the following rules:

1. The latest two keyframes are always maintained.

2. Those keyframes with less than 5% of points visible in the latest keyframe will be marginalized.

3. If there are more than n_w keyframes (n_w is the window size), the one whose camera is the farthest (in 3D space) from the latest keyframe's camera will be marginalized.

The marginalization strategy for points is as follows:

1. The active points hosted by a marginalized keyframe are marginalized.

2. The active points that are not observed by the latest two keyframes are marginalized.

3. For all other active points that still exist, their residuals related to the marginalized keyframe will be dropped from the system. This corresponds to removing some rows in the Jacobian J and residual vector $\mathbf{r}$.

To understand the rationale behind the point marginalization strategy, we need to study the

factor graph[6] of the optimization system and the sparsity of the Hessian matrix (as detailed in Equation 2.36). Those are illustrated in Figure 2.7. The left image is the factor graph. The ellipses represent the parameters (variables/unknowns). For example, a keyframe (KF) has a camera pose and the affine intensity parameters. A point has an associated depth value. The rectangles represent error terms (factors). An error term is connected to two keyframes and one point according to the definition in Equation 2.36. From $H = J^T AJ$ it can be derived that $H_{ij} = 0$ if parameters i and j do not share any common factor. On the other hand, the parameters connected to the same error term are correlated ($H_{ij} \neq 0$). This is reflected in the Hessian matrix structure on the right of Figure 2.7. For this Hessian matrix example, the parameter vector is formed as [camera poses, point depth values]. Colored blocks show non-zero values in the Hessian matrix, and that means those variables are correlated. For example, the green blocks show the correlation between the camera poses and point depths. There is no common factor between points, so the block corresponding to the points is diagonal.

This diagonal block in the Hessian matrix is essential for solving $H\delta\mathbf{x} = b$ efficiently. Suppose $\delta\mathbf{x}$ is compose of the increment in camera poses ($\delta\mathbf{x}_1$) and the increment in depth values ($\delta\mathbf{x}_2$). $H\delta\mathbf{x} = b$ can be written as

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x_1} \\ \delta\mathbf{x_2} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{2.42}$$

Just as described in Section 2.5.2, we can use a Schur complement to eliminate the portion of depth values first:

$$H'_{11}\delta\mathbf{x_1} = b'_{11}, \text{ where } H'_{11} = H_{11} - H_{12}H_{22}^{-1}H_{21},\ b'_1 = b_1 - H_{12}H_{22}^{-1}b_1 \tag{2.43}$$

Because H_{22} is diagonal, we can get its inverse trivially. H'_{11} is a small matrix, so the resulting linear system for camera poses can be solved very fast. With $\delta\mathbf{x_1}$, $\delta\mathbf{x_2}$ can be solved efficiently:

$$\delta\mathbf{x_2} = H_{22}^{-1}(b_2 - H_{21}\delta\mathbf{x_1}) \tag{2.44}$$

[6] A factor graph is a graphical model representing the variable-factor relationships inside a problem. Variables are the unknowns and factors are the (loss) functions. A factor is connected to a variable if it is dependent on the variable. One example is in Figure 2.7. A factor graph is built based on locality nature of the factors.

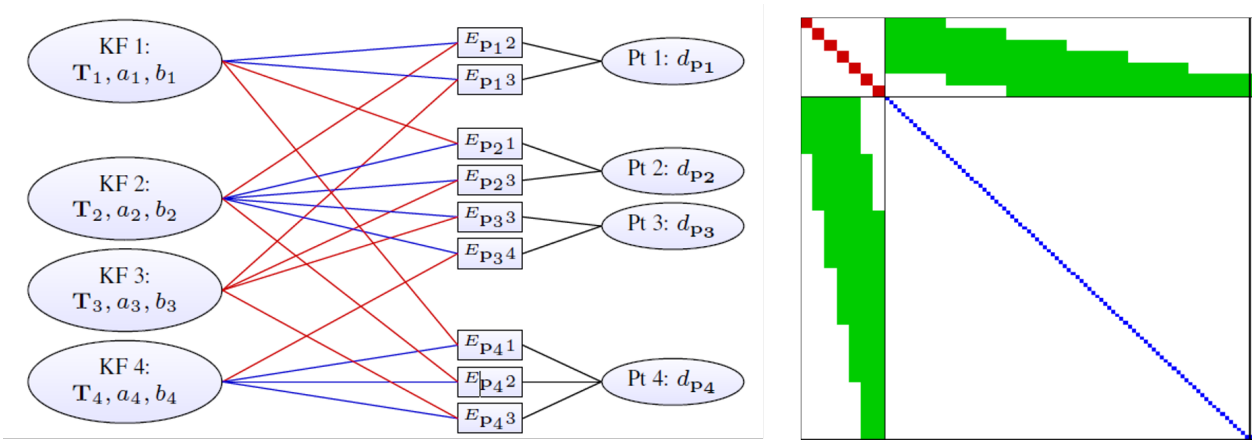


Figure 2.7: These two figures are from DSO [4]. Left: an example factor graph of local windowed optimization in DSO. Ellipses stand for parameters. KF represents a keyframe. It holds the camera pose (T) and two affine intensity parameters (a, b). Pt represents a point. It has a depth value (d). Rectangles stand for error terms. An error term $E_{\mathbf{p_i}j}$ is connected to three groups of parameters including the depth value of $\mathbf{p_i}$, the camera pose of its host keyframe (blue line), and the camera pose of another keyframe (red line, the j index in $E_{\mathbf{p_i}j}$). Right: a simplified Hessian matrix structure of a sparse local windowed optimization. The $\mathbf{x}$ vector of this Hessian is composed of 7 camera poses and a number of point depths. The colored regions indicate the corresponding parameters are correlated. For example, the red blocks show pose-to-pose correlation. The blue dots show point-to-point correlation. Note that the bottom-right block is diagonal because there is no point-to-point connection on the factor graph.

In this process, the sparsity of the Hessian matrix is critical. This sparsity should be maintained during marginalization. Therefore, when we marginalize a keyframe, we should drop all the residual terms involving this keyframe and remaining active points. Otherwise, during the Schur complement marginalization as described in Section 2.5.2, some point-to-point prior information will be injected to the new Hessian matrix, leading to non-zero off-diagonal values (destroying sparsity) in the bottom-right block. This will be reflected as a point-to-point connection in the factor graph.

This is also why DSO uses sparse points rather than dense depth maps. Dense depth maps often require geometrical regularization terms like smoothness. However, that geometry-to-geometry constraint will also affect the sparsity of the Hessian matrix and make local windowed optimization infeasible to solve.

2.6 Recurrent Neural Network for Depth and Pose Prediction (RNN-DP)

Monocular depth map prediction is a very popular area in computer vision. It has been used in 3D model reconstruction, SLAM, or even recognition. One important component in our colonoscopic reconstruction system is a recurrent neural network for depth and pose prediction (RNN-DP) [17].

As its name suggested, it is an end-to-end network that predicts a depth map and a relative camera pose (to the previous frame) for each image in a video. This network was originally proposed for generic depth map prediction problems and then was adapted for colon reconstruction. In this section I will first review its architecture and loss and then review how we trained it on colonoscopic images.

2.6.1 Architecture and Loss

Architecture RNN-DP is composed of a depth estimation network and a pose estimation network. They are illustrated in Figure 2.8.

The depth estimation network follows a U-Net [81] architecture. U-Net is a fully convolutional neural network (FCN) with skip-connections (feature maps from the encoding layers are concatenated with the feature maps from the decoding layers). It is a widely adopted generic image-to-image transformation network. In RNN-DP some of the encoding layers are replaced by Convolutional Long-Short-Term-Memory (ConvLSTM) layers [82] (the red blocks in Figure 2.8). ConvLSTM was developed for building recurrent neural networks with image processing capability. It uses convolutional operations in a traditional LSTM unit. At a high level, a ConvLSTM converts image feature maps to image feature maps with the same behavior as a common convolutional layer, but the hidden states inside it give it memory to do sequential jobs. With such layers, RNN-DP is capable of handling videos better than a vanilla U-Net; this is based on the assumption that the video frames are temporally correlated.

The pose estimation network uses a VGG architecture, e.g., VGG16 [83]. It regresses the concatenation of an input image and its depth map (predicted in depth estimation network) into a 6-DoF camera pose. The camera pose is relative to the previous frame. A depth map is concatenated into the input because it contains important geometrical information to infer camera transforms. Some of the convolutional layers are also replaced by ConvLSTM layers.

Loss Before getting into the loss function definition, we first need to understand two facts:
1) The training samples of RNN-DP are short video sequences 10 consecutive frames long. For data augmentation, every sample is reversed to generate another training sample. 2) The Differentiable Geometric Module (DGM) [13] allows us to warp an image and compute a related optical flow field using a depth map and a relative camera pose. This module is differentiable, so it can be used in

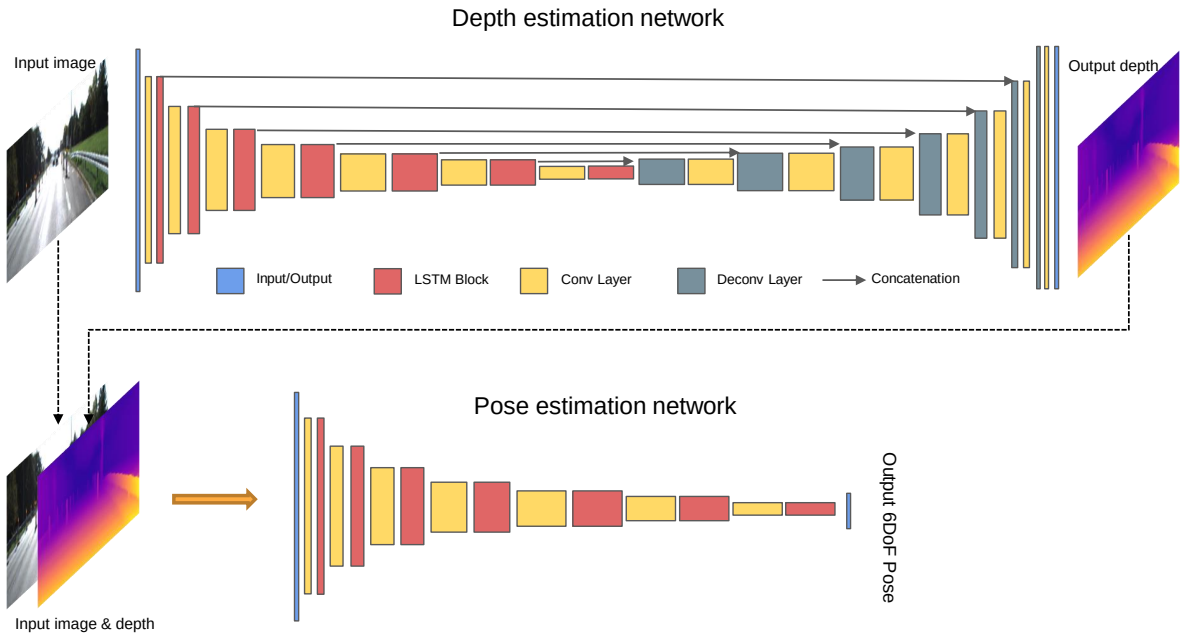


Figure 2.8: Architecture of RNN-DP. This image is from RNN-DP [14].

neural networks.

As shown in Figure 2.9, the loss function of RNN-DP is composed of the multiview reprojection loss, the forward-backward flow consistency loss, the smoothness loss (not drawn in Figure 2.9), and the absolute depth loss.

1. **Multiview reprojection loss**: Each image in a training sample is warpped to the other images by a DGM. A photometric error (RNN-DP used absolute intensity difference) can be computed between the target image the the warpped image. The regions without correspondence are ignored. The multiview reprojection loss is the weighted sum of all the photometric error terms. The term weight decays exponentially based on the time difference of the two frames.

2. **Forward-backward flow consistency loss**: Taking two consecutive frames A and B, a DGM can compute a forward flow $F_{A\to B}$ and a backward flow $F_{B\to A}$. A pseudo-forward flow $\hat{F}_{A\to B}$ can be computed using $-F_{B\to A}$, the depth map of A, and the relative camera pose from A to B. A pseudo-backward flow $\hat{F}_{B\to A}$ can be computed in a similar way. A forward-backward flow consistency loss equals $w_{A\to B}|F_{A\to B} - \hat{F}_{A\to B}| + w_{B\to A}|F_{B\to A} - \hat{F}_{B\to A}|$, where $w_{A\to B}$ and $w_{B\to A}$ are indicators of whether correspondences exist between two flows. The total flow consistency loss for a 10-frames subsequence is the sum of such terms for every

two consecutive frames in that subsequence.

3. **Smoothness loss**: This loss is an edge-aware smoothness term $\sum_{\Omega} |\nabla d| e^{-\nabla I}$, where Ω is the image domain, d is the inverse depth, and I is the image intensity. The total smoothness loss is the sum of all such terms of all images.

4. **Absolute depth loss**: If ground truth depth is available, an absolute depth loss can be included in training: $|d - \hat{d}|$. This term is computed and summed over all the images and over all the pixels where ground truth (inverse) depths $\hat{d}$ are available.

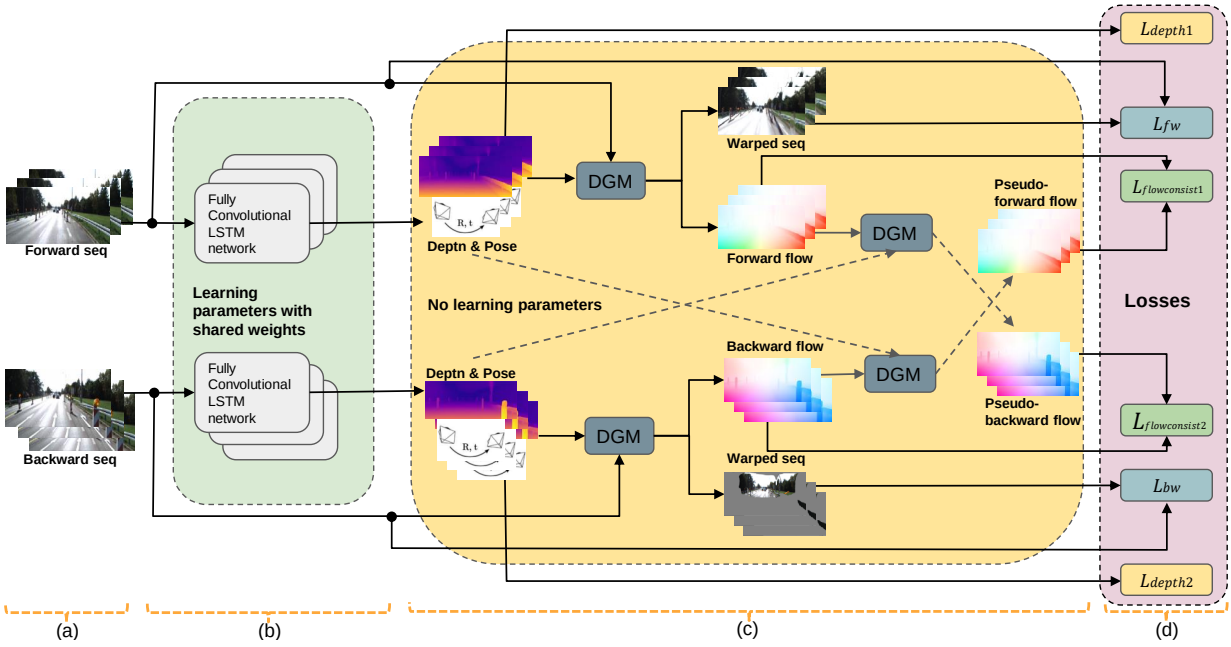


Figure 2.9: Loss function definition of RNN-DP. This image is from RNN-DP [14].

RNN-DP achieved high performance on common computer vision datasets like KITTI [84]. It can be trained without the absolute depth loss and can still achieve a result comparable to some other supervised networks. For details, read [14].

2.6.2 RNN-DP Trained on Colonoscopic Images

Although one advantage of RNN-DP is its self-supervised training capability, it does not converge on colonoscopic images because of the weak texture in colonoscopic images and other challenges that I mentioned in the Introduction chapter. Therefore, Dr. Rui Wang and I trained RNN-DP with ground truth depth generated by SfM [85]. Specifically, we used SfM to produce a sparse depth map

for each individual colonoscopy video frame. These sparse depth maps are then used as groundtruth for RNN-DP training. We collected 60 colonoscopy videos, each containing about 20K frames. Then we grouped every 200 consecutive frames into a subsequence with an overlap of 100 frames with the previous subsequence. Thereby we generated about 12K subsequences from 60 colonoscopy videos. Then we ran SfM on all the subsequences to generate sparse point clouds; they are converted to sparse depth maps for individual frames. Following the training pipeline in RNN-DP, these sparse depth maps are used as ground truth for training. Because individual reconstructions have different scales, as an approximation, we normalized each depth map by dividing by the median depth value.

To avoid the error from specularity (saturation), a specularity mask for each frame was computed based on an intensity threshold. Image reprojection errors in saturated regions were explicitly masked out during training.

Colonoscopy images also contain severe occlusions by haustral ridges, so a point in one image may not have any matching point in other images. The original RNN-DP did not handle occlusion explicitly. In order to properly train it on colonoscopy video, an occlusion mask was computed to explicitly mask out image reprojection error at occluded regions. The occlusion mask is determined by a forward-backward geometric consistency check, which was introduced in RNN-DP [14].

Trained with the aforementioned ground truth and improvements, the network was able to generate reasonable depth maps and camera poses. Some depth map examples are shown in Figure 2.10. These outputs are leveraged in RNNSLAM [24], which will be introduced in Section 3.

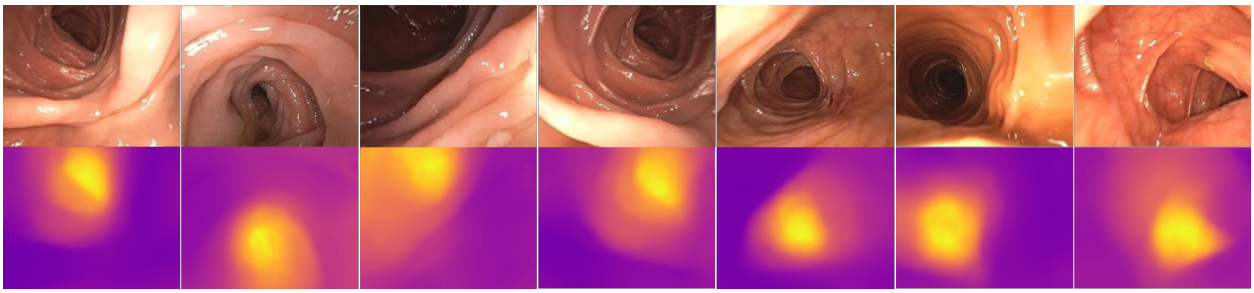


Figure 2.10: Colonoscopic depth map result examples predicted by our retrained RNN-DP.

2.7 Depth Map Fusion Algorithms

The depth maps generated by either algorithms or active sensors are based on individual frames. In order to generate a unified 3D model, e.g., a triangle mesh, depth map fusion algorithms are needed. If the RGB (red-green-blue) images are also available, it is also desired to fuse the color information

into the mesh as well. RGBD (RGB and depth) fusion is a widely studied area. Depending on the intermediate 3D representations during reconstruction, mainstream fusion algorithms can be divided into volumetric methods and point-based methods.

2.7.1 Volumetric Methods

A volumetric method represents the 3D scene by a 3D grid. The basic idea is that when an RGBD image comes in, every voxel in the 3D grid will be labeled as "empty" (space in front of the object) or "occluded" (space behind the object). After all the RGBD images are processed, the boundary between these two kinds of voxels will be extracted, which is the desired 3D model. The most famous method in this category is the truncated signed distance function (TSDF) [86].

In a TSDF method each voxel stores a signed distance value (D) and a weight (W). The distance value indicates whether (and how much) the voxel is in front of the surface or behind the surface. When a new frame comes in, the distance values in the whole volume are updated following this process: 1) The depth map is converted into a range surface (triangle mesh). 2) A ray is cast from the camera center to each voxel in the volume. A ray may intersect with the range surface at a point. The distance (d, negative in front of the surface, positive behind the surface) between the point and the voxel is computed and truncated by D_{max} and D_{min}. 3) A weight (w) of the new distance value is computed. It may depend on many factors. For example, it is smaller when the angle between the ray and the surface is small. Also, it is higher if the distance is close to zero; it decreases zero if it reaches D_{max} or D_{min}. 4) D and W are updated by $D := (WD + wd)/(W + w)$ and $W := W + w$.

The above process is conducted for all the voxels for each new frame. Finally, a Marching Cubes algorithm is performed to extract the zero crossings of the TSDF volume to form a mesh. A very successful application of TSDF is KinectFusion [87]. Beside TSDF, it takes depth maps and uses Iterative Closest Points to estimate camera poses.

Although volumetric methods are quite efficient, they have limited flexibility when the scale of the scene being reconstructed is unknown before reconstruction, as in our colonoscopic reconstruction. "Many systems assume a static scene, and cannot robustly handle scene motion or reconstructions that evolve to reflect scene changes" [22].

2.7.2 Point-based Methods

In contrast to volumetric methods, point-based methods generate 3D points dynamically without assuming a 3D scene (volume). A widely used point-based data structure is called a "surfel" (surface

elements) [88]. A surfel's attibutes comprise a 3D point position, a normal direction, color information, a radius, etc. [89]. Surfels, as a kind of generic 3D structure representation, can be directly rendered without connectivity using a method called Surface Splatting [90].

A widely used algorithm was proposed by Keller et al. [22]. This algorithm at a high-level is similar to KinectFusion [87]; it uses Iterative Closest Points (ICP) to estimate camera poses and uses camera poses to fuse the depth maps. The difference is that it uses surfels throughout the whole reconstruction process. Because in our application camera poses can be computed via better approaches (SLAM or neural network), we are mainly interested in the depth fusion part, i.e., how to generate surfels.

The 3D scene in [22] is a cloud of surfels. This surfel cloud is created and maintained by iterating the following two-step sequence:

1. Depth Map Preprocessing: When a new depth map comes in, it is converted into a set of surfels using the depth values and the associated camera pose and intrinsics. The normal directions are computed using central differences. The radii are computed based on the angles between the viewing directions and the normals.

2. Depth Maps Fusion: When new surfels are computed, they are either added into the surfel cloud or used to update some existing surfels. This requires associating existing model points with the new points. To do this, the model points are projected into the 4×4 upsampled incoming frame. All the model points projected to the 4×4 neighborhood of a pixel are potential candidates for association. Finally, based on some criteria like depth distance, confidence and distance to viewing ray (detailed in [22]), a best matching model point can be selected (if it exists). If a model point is successfully associated with an image point, that surfel's attributes will be updated using a weighted average, and its confidence will be increased.

This algorithm is able to reconstruct 3D scenes in real-time with comparable result to volumetric methods like [87], while offering great flexibility of not using a pre-fixed 3D grid. Based on this work, Schöps et al. [89] built SurfelMeshing.

SurfelMeshing [89] is an algorithm that reconstructs surfels and converts them into a triangle mesh. Its novel surfel denoising algorithm and meshing algorithm achieved state-of-the-art reconstruction quality. The meshing component also provides convenience for downstream geometric analysis. Its

surfel reconstruction algorithm is largely similar to [22]. Read [89] for other details. This program was used in our RNNSLAM [24] paper.

2.8 Instance-level Image Retrieval and Place Recognition

Image retrieval has a broad definition. Depending on applications, it includes two main areas: class-level (category-level) image retrieval or instance-level image retrieval. Class-level image retrieval aims at retrieving images belonging to the same class. For example, given an input image of a book, retrieve images of books. This is often used for collecting materials for certain topics. Instance-level image retrieval aims at retrieving images of the same instance or object. It assumes that the object is unique and that the retrieved images are subject to certain transforms to the query image, e.g., the perspective transform. For example, given an input image of the Eiffel Tower, retrieve images of the Eiffel Tower. Retrieving images of other towers will be considered as false positives. Therefore, retrieving an instance requires capturing unique features of an object. Those features should be distinctive even within a class.

Instance-level image retrieval is often called place recognition or geo-localization. The goal is typically to recognize a place or retrieve the location of a query image by retrieving similar images from a database that are well annotated. It is also useful for establishing correspondence between two frames in a group of images for downstream computer vision tasks like 3D reconstruction.

In my dissertation I focus on instance-level image retrieval in the colon because I am interested in recognizing colonic places that were revisited. Next I will briefly review the problem definition and related works on instance-level image retrieval.

2.8.1 Problem Definition

I denote a query image by q and a database of images by D ($\{d_k|k = 0...n\}$). Typically images in D are associated with information we want to retrieve, e.g., website source link, GPS pin or time stamp. The task it to find similar images of q in D that have overlap or shared content with q purely based on appearance. It should return an empty list if there are no such images.

Mathematically the task is to find an embedding function f and a similarity function sim that allows for fast retrieval: f converts an image x into a latent representation $f(x)$; sim measures the similarity of two representations and outputs a similarity score. sim is designed so that $sim(f(q), f(d_i)) > sim(f(q), f(d_j))$ if d_i shares some view of the same colonic structure as q and d_j does not.

2.8.2 Related Works

Automatic image retrieval is a widely studied area in computer vision. Traditional methods are often based on local features like SIFT [10]. From these they build global representations such as Bag-of-Words (BoW) [91], Vector of Locally Aggregated Descriptor (VLAD) [92], or Fisher Vector [93]. Spatial verification [94, 95] is often used to eliminate outliers and make the retrieval of the images generally reliable. The state-of-the-art algorithmic method was proposed by Schönberger et al. [95]. In [95] RootSIFT features [96] were extracted from Hessian affine feature corners [97]. These features were quantized using a vocabulary tree for BoW representation [91]. An inverted file was used for efficient retrieval. A "vote-and-verify" fast spatial verification was then conducted on the raw retrieved images. Because each pair of SIFT-like keypoint matching uniquely defines a 2D geometric transform called a similarity transform, the keypoint matchings between the query image and candidate images were used to vote in a 4D Hough space. The returned similarity transform hypotheses were then verified by a variant of the RANSAC algorithm. The raw images were finally re-ranked by the number of inliers.

Deep learning techniques have been applied on place recognition tasks [26, 27, 28, 29, 30]. It has been shown that the convolutional features or dense features learned by a CNN for a classification task, e.g., Imagenet [98], can be used for image retrieval tasks [26]. For convolutional features different pooling strategies have been proposed: max pooling [27], sum pooling [28], generalized mean pooling [30], etc. NetVLAD [29] used a generalized VLAD layer to compute a global descriptor from convolutional feature maps. Radenović et al. [99, 30] proposed a strategy that fine-tunes a CNN for retrieval tasks using correspondences established and filtered by SfM [100]; with this they achieved state-of-the-art results.

2.9 Generalized Cylinder Geometry Background

A *generalized cylinder* (GC), as defined in [44], is "a solid whose axis is a 3-D space curve and at any point on the axis a closed cross section is defined". Usually the cross sections are restricted to be normal to the axis, also called the "*centerline*" in this dissertation. Mathematically, the centerline is a space curve: $\boldsymbol{c}(s) = (x(s), y(s), z(s))$. In [66] it is defined as a curve inside the GC that maximizes the distance from the boundary. I define a *cross section* as the intersection between the surface and an orthogonal plane to a point on the centerline. If there are multiple connected components from the intersection, the profile of the one closest to the centerline point is taken as the cross section. Two

cross sections *intersect* when their planes intersect within the two connected components bounded by the two cross sections. The cross sections should not intersect if we want to achieve folding-free deformations. Sometimes in order to avoid intersections between cross sections, the orthogonality assumption can be relaxed; non-orthogonal cross sections can be substituted in some highly curved regions.

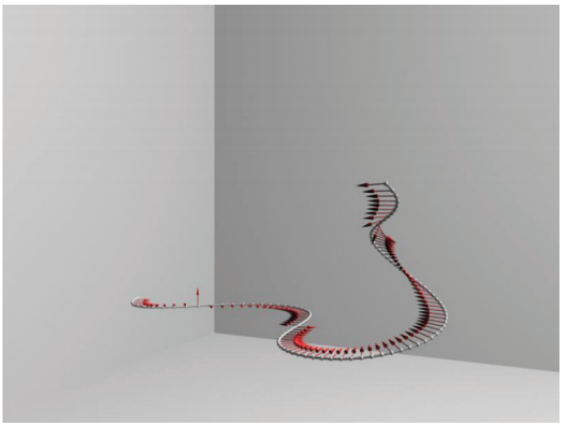
2.9.1 Frenet Frame and Rotation Minimizing Frame

At each point on the centerline a frame can be defined locally: ($\boldsymbol{t}$,$\boldsymbol{u}$,$\boldsymbol{v}$). For example, a well-known choice of the frame is the Frenet frame ($\boldsymbol{tangent}, \boldsymbol{normal}, \boldsymbol{binormal}$). With this choice, the ($\boldsymbol{u}$,$\boldsymbol{v}$) plane is orthogonal to the curve. The tangent is a unit vector tangential to the 3D curve. The normal is a unit vector in the direction of the derivative (w.r.t. the arc length parameter) of the tangent. The magnitude of that derivative is called curvature (κ). The binormal is the cross product of the tangent and the normal. The binormal's derivative is in the negative normal direction. The derivative equals -τ times the normal, where τ is the torsion.

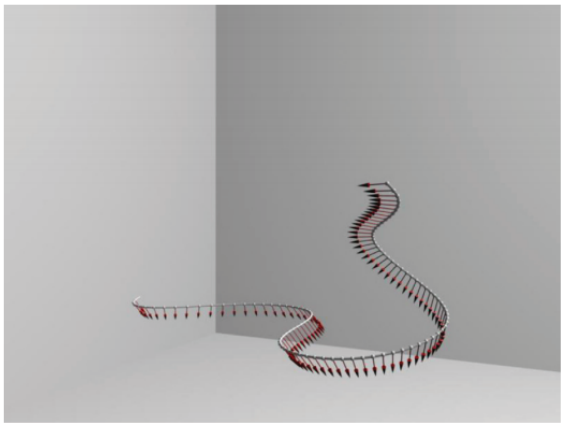
Although the Frenet frame is easy to compute, it is not an ideal frame system for surface deformation tasks, because its rotation about the tangent often leads to undesired twists [70, 48]. It is hard to maintain or adjust relative position (rotation) between neighboring cross sections during deformations.

Another frame system is the Rotation Minimizing Frame (RMF) [70]. An Rotation Minimizing Frame (RMF) is a moving frame on a curve that does not rotation about the tangent. Figure 2.11 shows a comparison between a Frenet frame and an RMF. It can be noticed that the Frenet frame has discontinuities and large twists on a general curve. These twists will be hard to track during deformation. On the other hand, the RMF is stable and continuous. Locally it does not rotate about the tangent. During deformation, the twist of a generalized cylinder can be easily handled by RMF. This will be detailed in Section 5.

Wang et al. [70] proposed a very efficient "double reflection" method to compute the RMF. It takes a discretized 3D curve (an ordered list of points) and a unit vector in the orthogonal plane at any point, and it computes an RMF at each point. See [70] for the details of the algorithm. RMF can be used to determine the transforms of the cross sections of a generalized cylinder. In the highly curved regions, interpolated non-orthogonal cross sections can be used to substitute orthogonal cross sections. This will be mentioned in Section 5.3.1.



(a) The Frenet frame of a spine curve. Only normal vectors are shown.



(b) A rotation minimizing frame (RMF) of the same curve in (a). Only reference vectors are shown.

Figure 2.11: A comparison between the Frenet frame and the RMF. On the left the normal directions are shown for Frenet frames. On the right (RMF), a "reference vector" is one of the two RMF vectors in an orthogonal plane. The shown reference vectors have minimum rotation about the tangents. Their orthogonal frame vectors are omitted. The twist and discontinuity of the Frenet frame is noticeable on the left. Comparatively, the RMF is stable and continuous along the curve. It has minimum rotation about the curve. This image is from [70].

2.9.2 Related Works of Skeleton-driven Surface Deformation

The motivation of studying the geometry of generalized cylinders is to deform (straighten) it in order to visualize the interior surface succinctly. The surface needs to be deformed globally. There has been extensive research on large-scale surface deformation for general human figure animation purposes [51, 52, 53, 54, 55, 56, 57, 58, 59]. These researchers achieved high performance even when animating some quite complicated meshes. While achieving a global deformation, local rigidity [56, 55] was preserved using either linear or non-linear methods. Widely used large mesh deformation methods like [56, 54] generally need the displacements of some vertices, cells, or control points as input.

However, the methods mentioned above do not consider the underlying structure/skeleton of the mesh. It is sometimes important to do shape modeling, for example by decomposing a mesh into the skeleton and the surrounding geometric details. In particular, the definition of of a generalized cylinder divides the shape into a centerline and cross sectional information. Since the global curvature properties of the cylinder is so strongly connected to those of the skeleton, when you want to manipulate globally, you will want to leverage the centerline instead of directly manipulating surface points. For example, one may want to compute the shape variations of a generalized cylinder

along a path in the shape space of the skeleton, as shown in Figure 1.4. The shape of the centerline is simply parameterized by a few parameters, but it is not straightforward to figure out the shape variations by directly manipulating some vertices or control points.

The underlying skeletons can also be valuable for statistical analysis in shape space as has been done for the s-rep [101]. Using a skeleton also allows us to add constraints to the structure of the shape during deformation in a more intuitive way.

In this section I will review some related works in skeletonization, skeleton-driven surface deformation, and especially methods related to centerlines of generalized cylinders.

Skeletonization: Verroust et al. [102] extracted skeletal curves from a point cloud by connecting centroids of level sets computed from a neighborhood graph. Mortara et al. [103] in their work first detected high curvature regions and computed topological rings by growing spherical bubbles from each region; the skeleton was computed by connecting the centroids of the rings. Chuang et al. [104] proposed the potential-field-based skeletonization method. These methods typically assume relatively smooth geometry and are unsuitable for a highly-curved colonic surfaces. They produce small branches for local high curvature regions, but for colon deformation, we hoped to model the colon by a single centerline.

The method proposed by Antiga et al. [105, 66, 106] finds a centerline on the Voronoi diagram of the triangle mesh by maximizing distance to the boundary. Their method matches better with our motivation and thus was used as an initialization of my algorithm (detailed in Chapter 5).

Skeleton-driven Deformation: Yoshizawa, et al. [64, 107] used a Voronoi-based medial surface as the skeleton and performed free-form deformation on it. Kho et al. [63] used a reference curve to drive mesh deformations. Shi et al. [108] proposed the mesh puppetry method that can use high-level user-defined constraints with sparse control-point-like skeletons. In [60], Kavan et al. used dual quaternions to eliminate artifacts from standard skeletal subspace deformations. Example-based deformations like the pose space deformation [109] and context-aware skeleton shape deformation [110] took advantage of known shape examples to make more reasonable interpolations.

The aforementioned algorithms were mostly proposed for computer graphics human figure deformations. The skeleton being used were generally sparse control points, for example, using three

points to represent an arm. However, a colon is a very long and curved object. Straightening a colon requires a densely sampled parametric curve as the skeleton. None of these method uses this kind of skeleton.

Centerline and Cross Sections: Wang et al. [111] proposed a soft-straightening method using a centerline and its cross sections. They prevented the intersection of cross sections by simulating electrical charges on the centerline and used the deflected electrical force lines to sample the cross sections. The cross sections are stacked to form the straightened surface. However, two problems remained unsolved in their work: 1) how to prevent intersections geometrically, *i.e.*, how to have a centerline and strictly planar cross sections while minimizing intersection; 2) based on the deformation of the skeleton, how to compute a point-wise displacement field without sampling or interpolation so as to maintain local curvature patterns on the surface.

The work of Zhang et al. [59] combined the skeleton (a centerline and cross sections) with the "as-rigid-as-possible" method [56]. They showed the semi-volume-preserving benefit of utilizing the skeleton, but their initial centerline shape is restricted to be straight and thus is not applicable for the long tubular meshes that we are dealing with. In [67] the level-sets computed by conformal mapping could also be an alternative for cross sections, but those contours were not planar and there was no centerline associated with them to perform skeletal manipulation.

CHAPTER 3

RNNSLAM: Real-time 3D Reconstruction System for Colonoscopy

This chapter introduces my work on reconstructing a 3D surface from colonoscopic videos for missing region visualization (RNNSLAM) [24]. The software of this project is in https://github.com/RuibinMa/RNNSLAM.git.

3.1 Method

3.1.1 Image Preprocessing

Informative Frame Selection The colonoscopy data I used in this dissertation was provided by Dr. Sarah McGill from UNC Hospital. In raw colonoscopy videos a large portion of frames are non-informative because the camera's view of the colon surface was notably occluded by either other parts of the colon surface or other materials like water. Such examples are shown in Figure 3.1. These frames are not helpful for training and are harmful for tracking. Therefore, I trained a neural network to identify those frames.

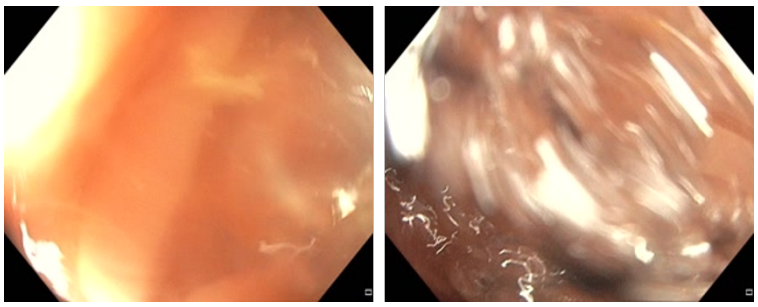


Figure 3.1: Two examples of non-informative frames.

The neural network is a Resnet-18 [112]. It does binary classification (informative or non-informative). It was trained by three full colonoscopy videos. I manually labeled each frame inside the videos. I only labeled the frames with no visible colon structure or texture as non-informative. About 50% of frames were determined as non-informative in this step.

The model I have trained can do the classification very well. Empirically, during my experiments most of the non-informative frames can be recognized and discarded. Moreover, the ones that are

remaining do communicate colon shape. However, the remaining images are optically distorted and need to be undistorted.

Image Undistortion The camera of the endoscope we used is a fisheye camera. Fisheye cameras have a large field of view but have severe distortion. Because the 3D vision techniques for common pinhole camera (perspective projection) have been well developed, I currently undistort the fisheye images into pinhole images. This comes with a sacrifice of the completeness of the images at the boundaries, which should be addressed as a future work.

I used the fisheye model presented in Scaramuzza et al. [113], which is implemented in MATLAB:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \\ a_0 + a_2\rho^2 + a_3\rho^3 + a_4\rho^4 \end{bmatrix} \tag{3.1}$$

where $[u, v]$ is the 2D point location ($[0, 0]$ at image center), $\rho = \sqrt{u^2 + v^2}$. a_0, a_2, a_3, a_4 are the distortion parameters. $[X, Y, Z]$ is the mapped 3D location in the camera's coordinate system (camera center is at $[0, 0, 0]$).

To get the fisheye distortion parameters, I captured 35 checkerboard patterns using the endoscope from different view-points. The checkerboard corners were then extracted from the images and an optimization algorithm was run to simultaneously get the camera poses and intrinsics (including distortion parameters). With the parameters the images were undistorted and cropped to the same size. An example is shown in Figure 3.2.

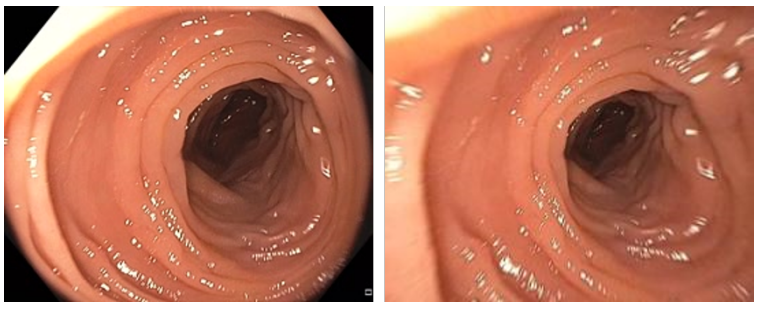


Figure 3.2: Fisheye colonoscopic image (left) and its undistorted version (right).

The frames input to the RNNSLAM system described in the following Section 3.1.2 have all been

preprocessed by the non-informative frame culling and the undistortion processing just described.

3.1.2 RNNSLAM Pipeline

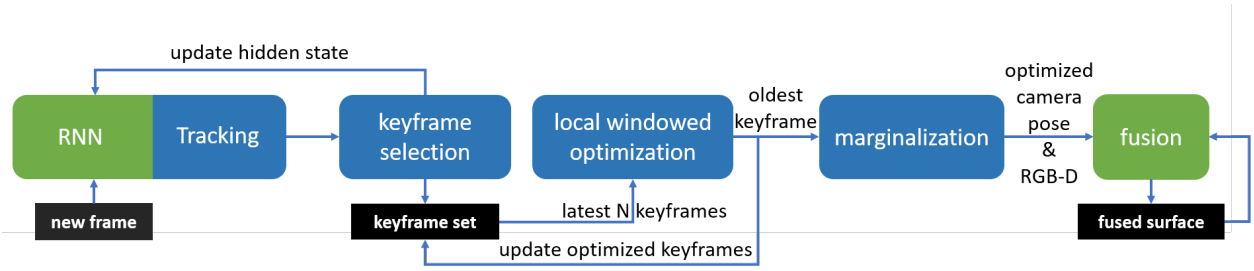


Figure 3.3: Full RNNSLAM pipeline.

The full pipeline of the RNNSLAM is described in Figure 3.3. This pipeline is adapted from DSO [4]. The original DSO includes the tracking, keyframe selection, local windowed optimization and marginalization modules. I developed a unified model by adding an RNN-DP to the tracking module and adding a fusion module to the end. In particular, I proposed a strategy to combine RNN-DP with the DSO framework in a way that each component informs the other. At a high level, the functions of these modules are

1. RNN (RNN-DP): predict depth and pose for each input frame.
2. Tracking: based on RNN-DP prediction, refine the camera pose based on intensity.
3. Keyframe selection: make decisions to establish new keyframes and to update RNN-DP's hidden states.
4. Local windowed optimization, i.e., bundle adjustment: jointly optimize some sparse depth values in each keyframe and their camera poses.
5. Marginalization: finalize output the oldest keyframe's camera pose and update the optimization system.
6. Fusion: take the RGB values, the RNN-DP-predicted depth map and optimized camera poses to fuse into a global mesh.

The details of the architecture and the losses of the RNN-DP have been introduced in Section 2.6. I also introduced how we trained RNN-DP for colonoscopic images. Basically, we collected 60

colonoscopic videos, divided them into 12K 200-frames short sequences, and ran SfM on them to generate sparse depth map ground truth. A specularity mask and an occlusion mask were added to the loss function to address the challenges in colonoscopic images. For details, please read Section 2.6 or [14, 24].

Our improved RNN-DP outputs frame-wise depth maps and tentative camera poses (relative to the previous keyframe). They are used to initialize the photoconsistency-based tracking [4] that refines the camera pose. The mathematical basis of the original tracking model of DSO was introduced in Section 2.5.1. I will describe how that original DSO tracking model is improved by using RNN-DP's output instead of the depth map generated by the tracking module.

3.1.3 Deep Learning Driven Tracking

In the original DSO the tracking module used a conventional method called direct image alignment [77, 4, 20]. It estimates a camera pose relative to the current keyframe (reference frame) by optimizing a photometric error as in Equation 2.29 ($E = \sum_{\boldsymbol{p} \in \mathcal{P}_K} \sum_{\hat{\boldsymbol{p}} \in \mathcal{N}_{\boldsymbol{p}}} \alpha_{\hat{\boldsymbol{p}}} \| I[\hat{\boldsymbol{p}}'] - K[\hat{\boldsymbol{p}}] \|_\gamma$). The input to this step is composed of three things:

1. A set of points in the reference frame whose depth values are known. These are the $\mathcal{P}_K$ and the depth values of $\mathcal{P}_K$ in Equation 2.29.

2. The camera pose of the current keyframe.

3. An initial relative camera pose of the new frame.

In the original DSO the initial value of the relative camera pose was set by a "constant motion" model. As my first improvement, I used the camera pose prediction from the RNN-DP for initialization.

My second improvement lies in the depth values of the points in keyframe K. In the original DSO each keyframe has a number of sparse points whose depth values are optimized in local bundle adjustment (detailed in Section 2.5.3). The depth map of the current reference keyframe K is generated in the original DSO by projecting all those sparse points of a local window's keyframes to keyframe K, with a slight dilation. The resulting depth map is thus a semi-dense depth map. This depth map can be very noisy. In our RNNSLAM, this depth map was replaced by our RNN-DP depth prediction, which is dense and more accurate.

There are three advantages of using the RNN-DP depth in keyframes: 1) The depth maps are less noisy because the RNN-DP is trained with a smoothness loss term. 2) The depth maps are scale-consistent. The resulting system is thus very stable against scale-drift, which has been a common problem of traditional SLAM methods. Strictly speaking, in training we used median-divided depth values and assumed that the training data is more or less scale-consistent, but it is not guaranteed that the training data are 100% scale consistent. Also, the depth values predicted by the RNN can also have small scale errors because of the latent parameters. However, according to our experiments, RNN-DP is much more scale-stable than SLAM. One reason is that it is not an optimization algorithm and does not explicitly use old predictions as input for new predictions.
3) The initial keyframe can directly use the RNN-DP prediction as its depth map rather than using an image of random noise. This makes the bootstrapping stage much faster and more stable. In fact, many SLAM systems cannot even bootstrap for challenging tasks such as colonoscopy.

3.1.4 Details of Integrating RNN-DP into DSO

Brief Review of Candidate and Active Point Management Strategy: As detailed in Section 2.5.3, in DSO's local windowed optimization, a local window is always maintained; it contains the most recent N_k keyframes and 3D points. The camera poses of the keyframes and the positions of the 3D points are subject to optimization. At any time point, the local window contains N_k keyframes. Each keyframe has N_p candidate points sampled from high gradient regions. However, to obtain the speed advantage of sparsity (detailed in Section 2.5.2), only N_p points in total will be activated for local bundle adjustment. Each active point is based on a host keyframe. Its 3D position is uniquely defined by a depth value in that keyframe. The local bundle adjustment jointly optimizes the depth values of those active points and the camera poses of the keyframes. The energy function is the sum of photometric errors of all the active points as in Equation 2.36 ($E_{total} = \sum_{K\in\mathcal{W}} \sum_{\boldsymbol{p}\in\mathcal{P}_K} \sum_{I\in obs(\boldsymbol{p})} \sum_{\hat{\boldsymbol{p}}\in\mathcal{N}_{\boldsymbol{p}}} \omega_{\hat{\boldsymbol{p}}} \left\| I[(\hat{\boldsymbol{p}}'] - b_I) - \frac{e^{a_I}}{e^{a_K}}(K[\hat{\boldsymbol{p}}] - b_K) \right\|_\gamma$). The Gauss-Newton method is used to minimize this energy function. It is performed each time when a new keyframe is inserted.

Integration Details: As illustrated in Figure 3.4, when a new frame comes in, it is first processed by the deep-learning-driven tracking module described in the previous section. With the predicted camera pose, the candidate points in the existing keyframes are projected to this incoming frame.

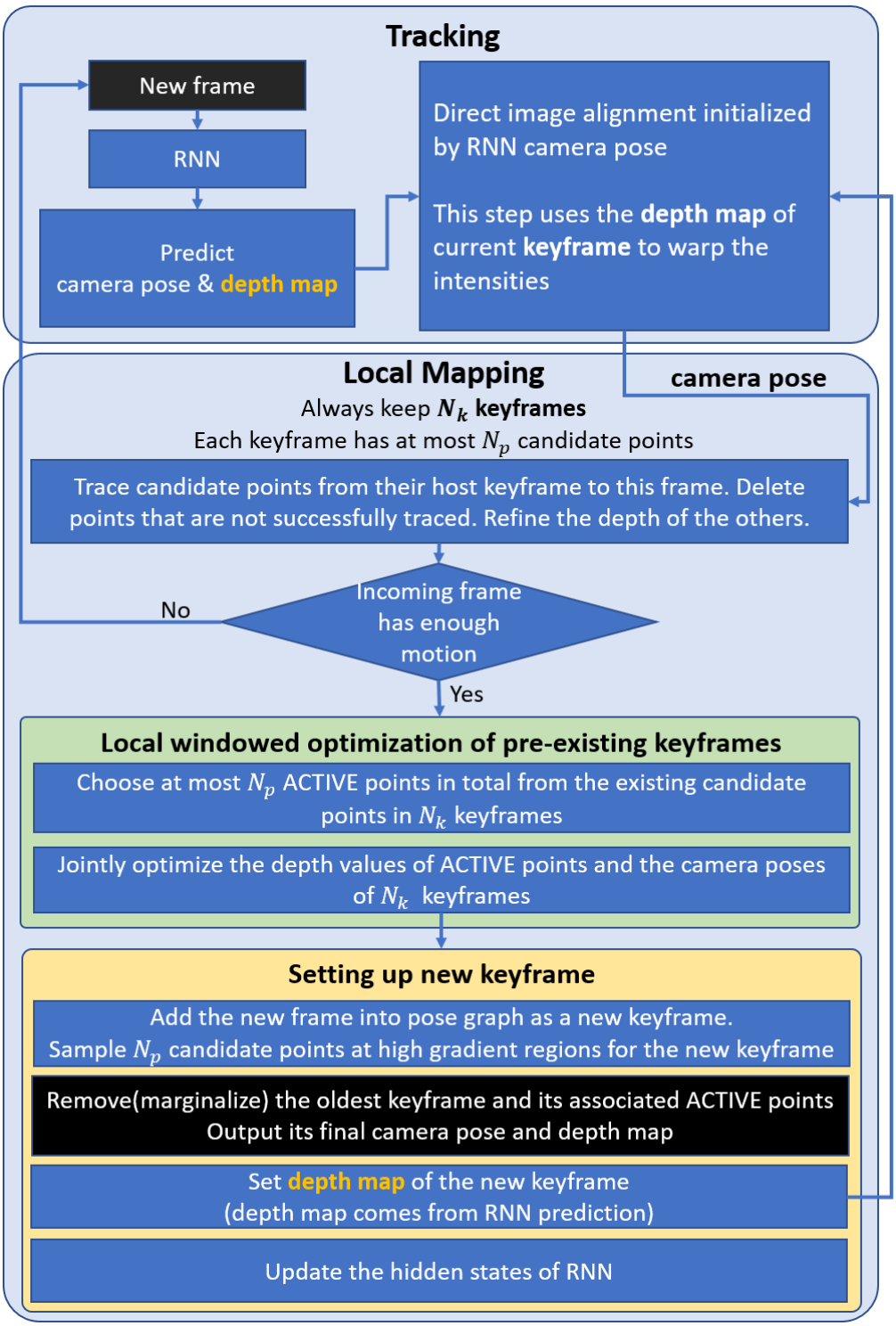


Figure 3.4: The pipeline interactively integrating RNN-DP and DSO.

The points that are not successfully projected will be removed. For the rest of the candidate points, their depth will be adjusted by searching on the epipolar lines in the new frame. For details, read

LSD-SLAM [5] and DSO [4].

If the new frame has enough motion, it will be established as a new keyframe, but before that, the local windowed optimization of pre-existing keyframes is conducted. Details of the local windowed optimization is in Section 2.5.3.

In the process of setting up the new keyframe, N_p (2000) candidate points are first sampled from it for future optimization. Its depth map predicted by the RNN-DP is then used for direct image alignment of the next incoming frame. Specifically, the points ($\mathcal{P}_K$) in Equation 2.29 will directly fetch depth values from this depth map rather than the semi-dense depth map used in original DSO. Next, the oldest keyframe (pose and associated points) is marginalized by the Schur Complement, finalizing its parameters. The background of marginalization was introduced in Section 2.5.2. Finally, the hidden states of the RNN-DP is updated. Therefore, the RNN-DP is equivalently processing a series of keyframes. For the other frames, it is only used for a forward prediction without changing the hidden states.

3.1.5 Fusion

A global model is needed to estimate and visualize missing surface. Each keyframe has a depth map that must be fused into a global mesh. In our pipeline we use a surfel-based method called SurfelMeshing [89] because surfel-based methods do not rely on a fixed 3D grid (detailed in Section 2.7.2).

The input to this step is the sequence of optimized camera poses, images and depth maps of the (marginalized) keyframes. The output is a 3D triangle mesh. As described in detail in [89], SurfelMeshing first creates a surfel cloud according to 3D coincidences computed from the depth values. Then it uses a novel meshing algorithm to generate a triangle mesh from the surfel cloud.

The quality of the output mesh highly depends on the consistency of successive depth maps. If the depth maps are not consistent, the mesh will be very noisy and will have a multi-layer artifact. Therefore, I included a windowed average algorithm to register the depth maps:

Windowed depth averaging In a local window containing the latest N (7) marginalized keyframes, I first in parallel project the depth maps of the $N-1$ old keyframes to the latest keyframe. Then, the new keyframe replaces its depth map with the weighted average of the projected

depth maps and its current depth:

$$D_N = \frac{\sum_{i=1...N} \frac{1}{N+1-i} \Phi(\Pi(T_{iN}\Pi^{-1}(D_i))}{\sum_{i=1...N} \frac{1}{N+1-i}} \tag{3.2}$$

D_i is the depth map of the i-th keyframe. Π is the projection from 3D to 2D. T_{iN} is the relative camera transformation between the i-th keyframe and the last keyframe. Φ is a resample-and-crop operation. The resulting average depth is used for the fusion. This step effectively eliminates the non-overlapping between depth maps at a cost of a slight smoothing.

3.2 Experiments

In this section I will first show some qualitative results of RNNSLAM and then show some quantitative evaluation results.

3.2.1 Qualitative Results

Depth Map Prediction Figure 2.10 shows depth maps estimated by RNN-DP. Given a colonoscopic video sequence, the RNN-DP produces high quality depth maps in real-time.

Fusion Process Figure 3.5 shows the incremental fusion process of a chunk of colon. The camera is moving along with the latest keyframe. The snapshots were captured in real time.

Figure 3.5: The fusion process for a chunk of a colon.

Reconstructions Figure 3.6 and 3.7 each show 6 reconstruction examples. To date, our method is the only one to generate high quality meshes of real colonoscopic videos. The reconstructions capture the curvature of the colonic surface such as the haustral ridges. They can be further used for missing region analysis.

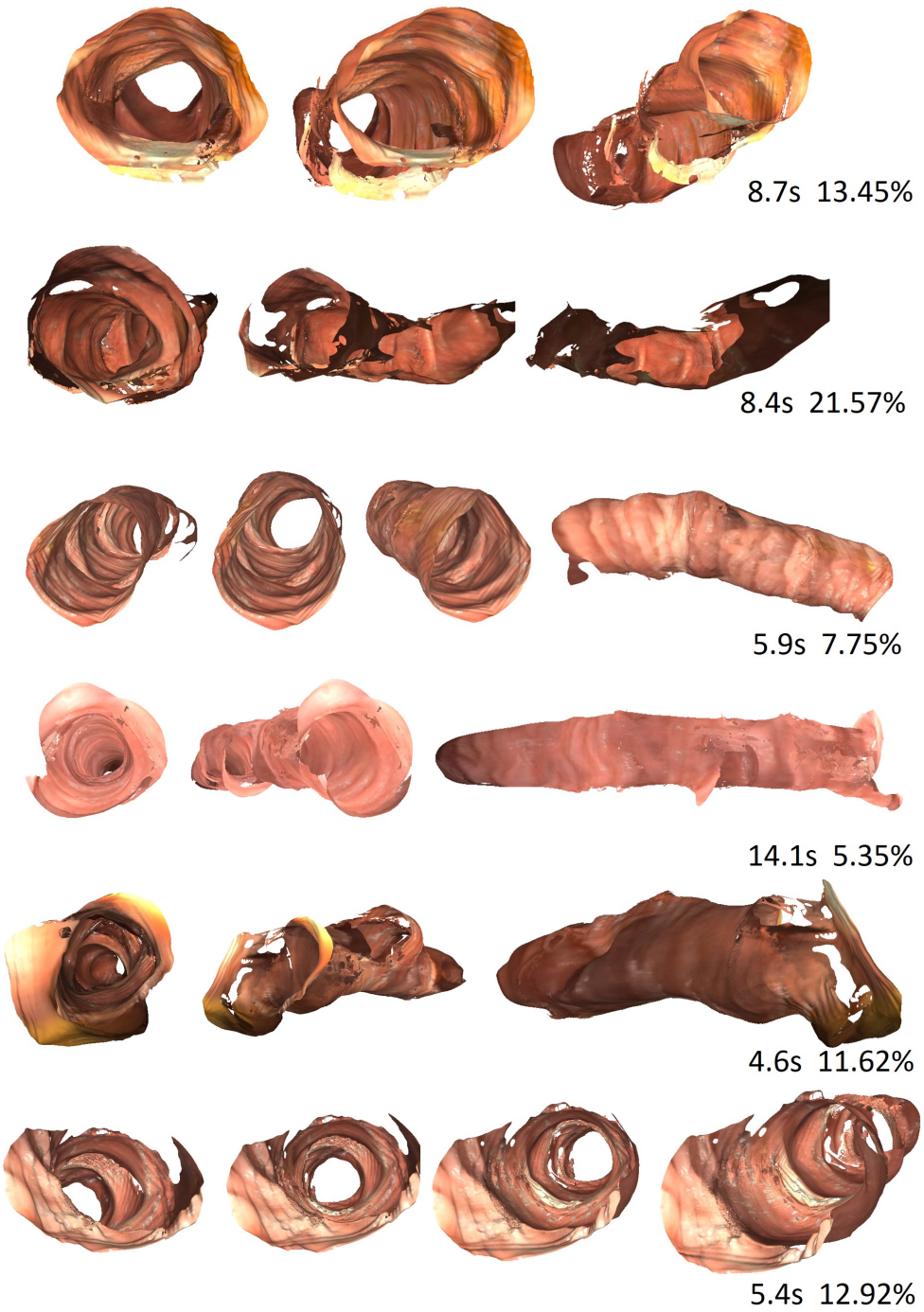


Figure 3.6: 6 reconstructed colon chunks from various points of view. The bottom-right number of each chunk is the length of its video and the estimated surface missing ratio. The estimation method is introduced in Section 3.2.3.

Missing Region Visualization The primary objective of this part of my dissertation is to visualize missing regions of a colonoscopy. Using my method, the missing regions are explicitly

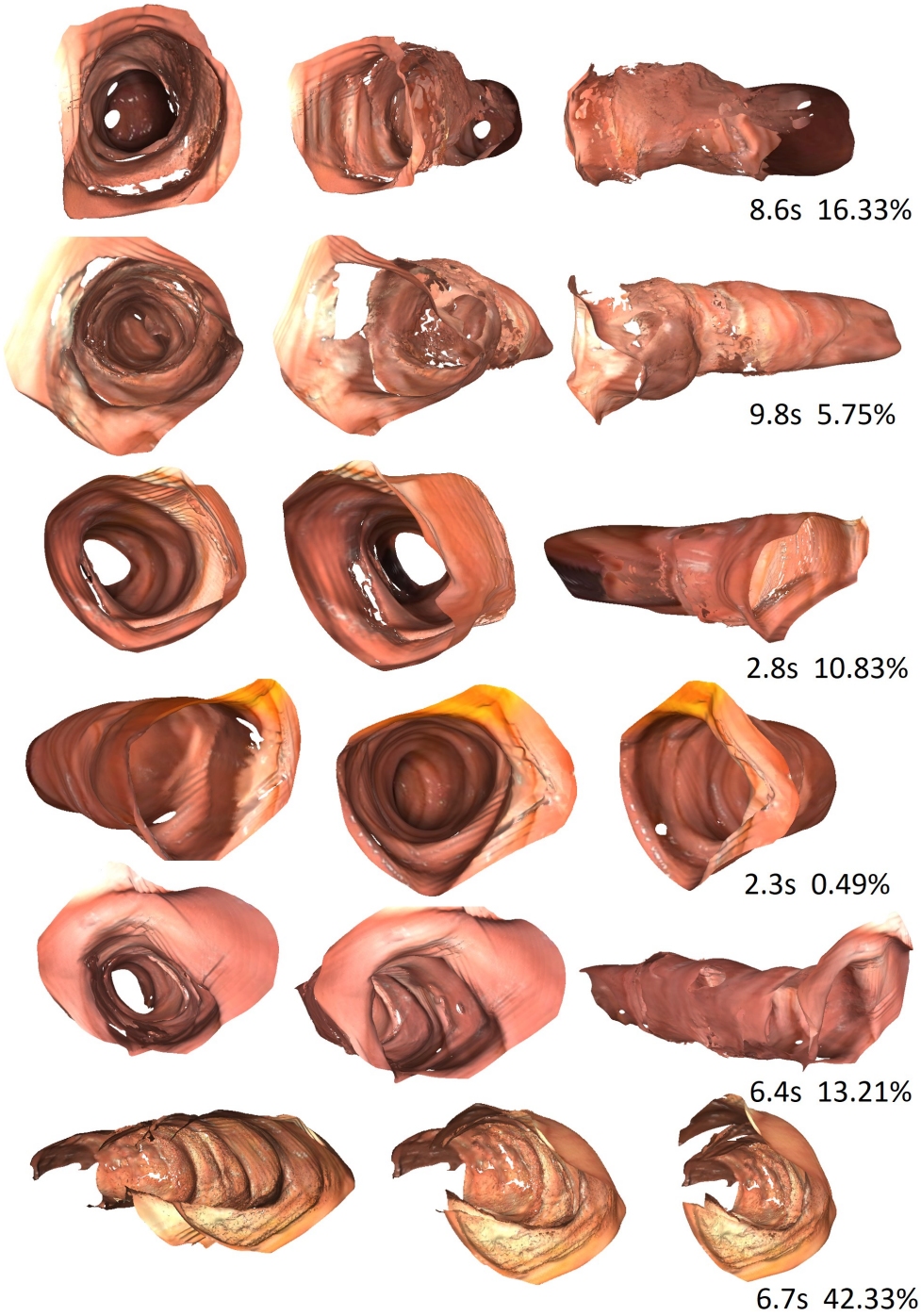


Figure 3.7: Another 6 reconstructed colon chunks from various points of view. The bottom-right number of each chunk is the length of its video and the estimated surface missing ratio. The estimation method is introduced in Section 3.2.3.

shown as the blank regions in the colon surface (wherever the generalized cylinder is not complete).

In Figures 3.8, 3.9 and 3.10 I show three examples of missing regions visualized by our system from

clinical colonoscopic videos. In Figure 3.8 the top part (red circle) of the colonic surface is not surveyed by the camera. Consequently, the respective part of the 3D model is missing. In Figure 3.9 there is a more severe missing of surface region because the camera never turned toward the left side of the images. Half of the colonic surface circumference is missing. In Figure 3.10 the region behind the haustral ridges were blocked by them. The missing regions are highlighted in the reconstruction (by black color). The missing regions in these examples have been verified by our colonoscopist coauthor [24], Dr. McGill.

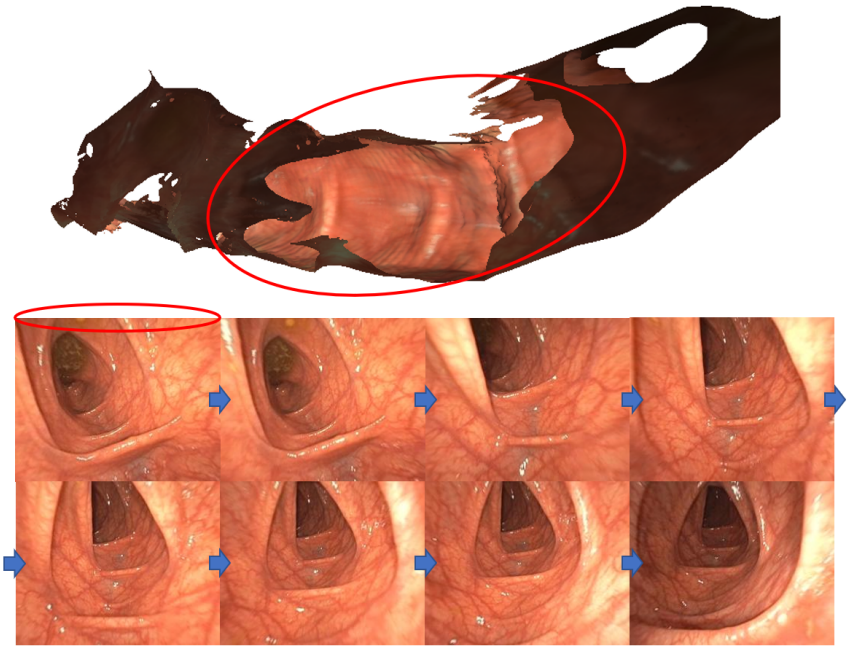


Figure 3.8: Missing region caused by lack of camera orientations. The blue arrows indicate the frame sequence.

3.2.2 Drift

This subsection addresses the drift problem. As mentioned in the Introduction, drift is typically caused by non-accurate tracking or lack of optimization. For an end-to-end deep learning method such as our trained RNN-DP, there is no inter-frame optimization applied such as local bundle adjustment. Thus, it will have accumulated camera pose drift. On the other hand, a traditional SLAM pipeline like the DSO is not robust enough against noise. In other words, its tracking is not accurate enough to prevent drift with only local bundle adjustment.

In Figure 3.13 I compare intermediate results (point cloud generated by the local mapping module)

of the original DSO and our RNNSLAM. In both cases, the cameras are moving from right to left. The scale of the reconstructed sparse scene is marked by yellow circles. For DSO, the scene scale is

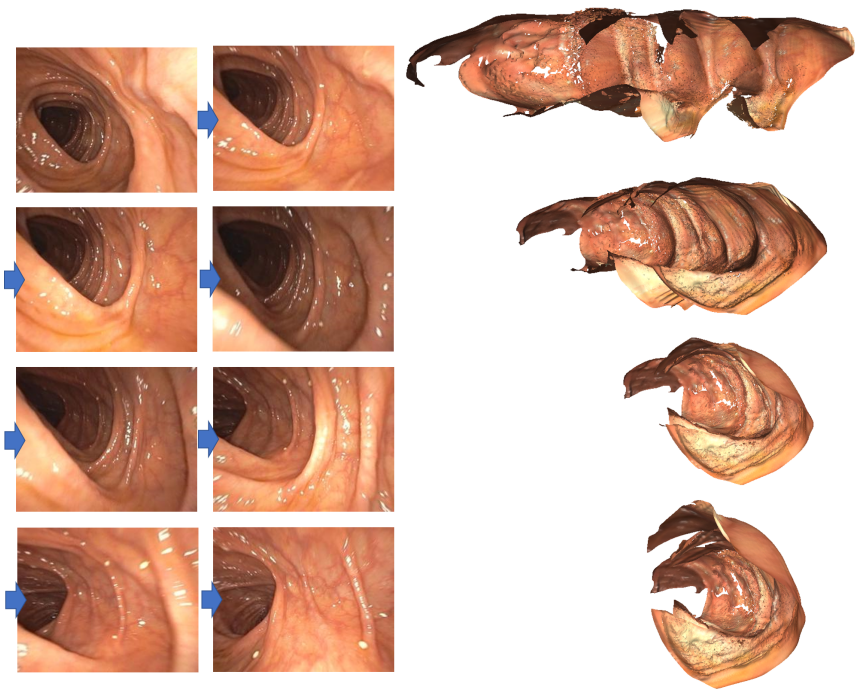


Figure 3.9: Missing region caused by lack of camera orientations. Half of the colon wall is missing because it was not surveyed by the camera (bottom right side of the snapshots). The blue arrows indicate the frame sequence.

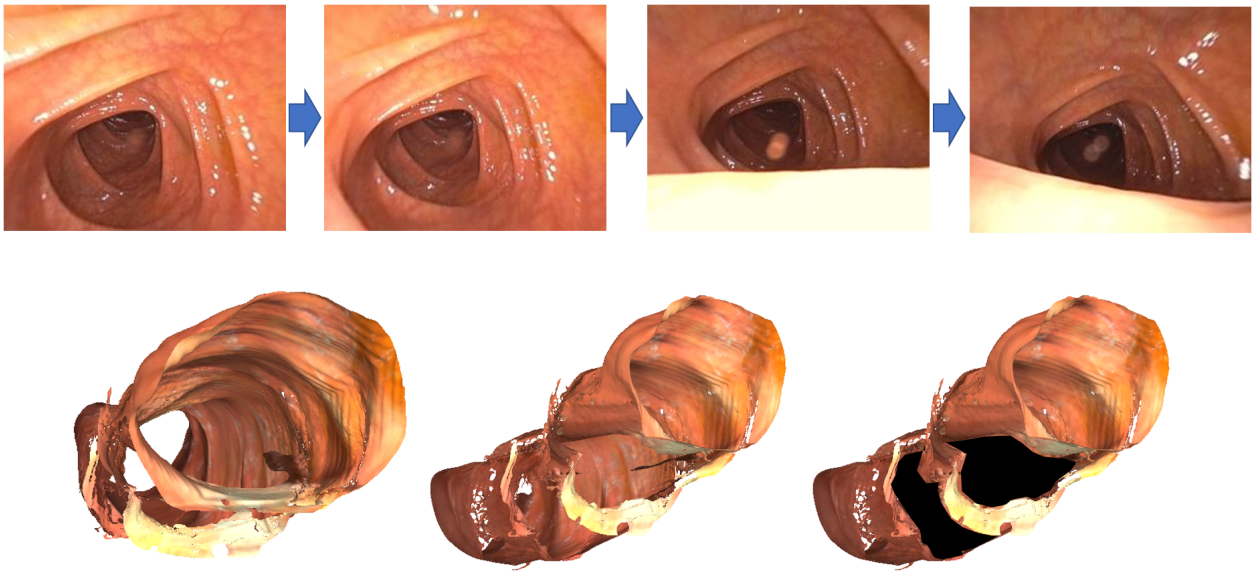


Figure 3.10: Missing region caused by haustral ridge occlusion.

changing rapidly, which is very non-realistic. For RNNSLAM, the scale is much more consistent throughout the chunk. This improvement comes from the better tracking driven by scale-consistent depth maps. The RNN-DP has learned robust prior knowledge to predict scale-consistent depth maps.

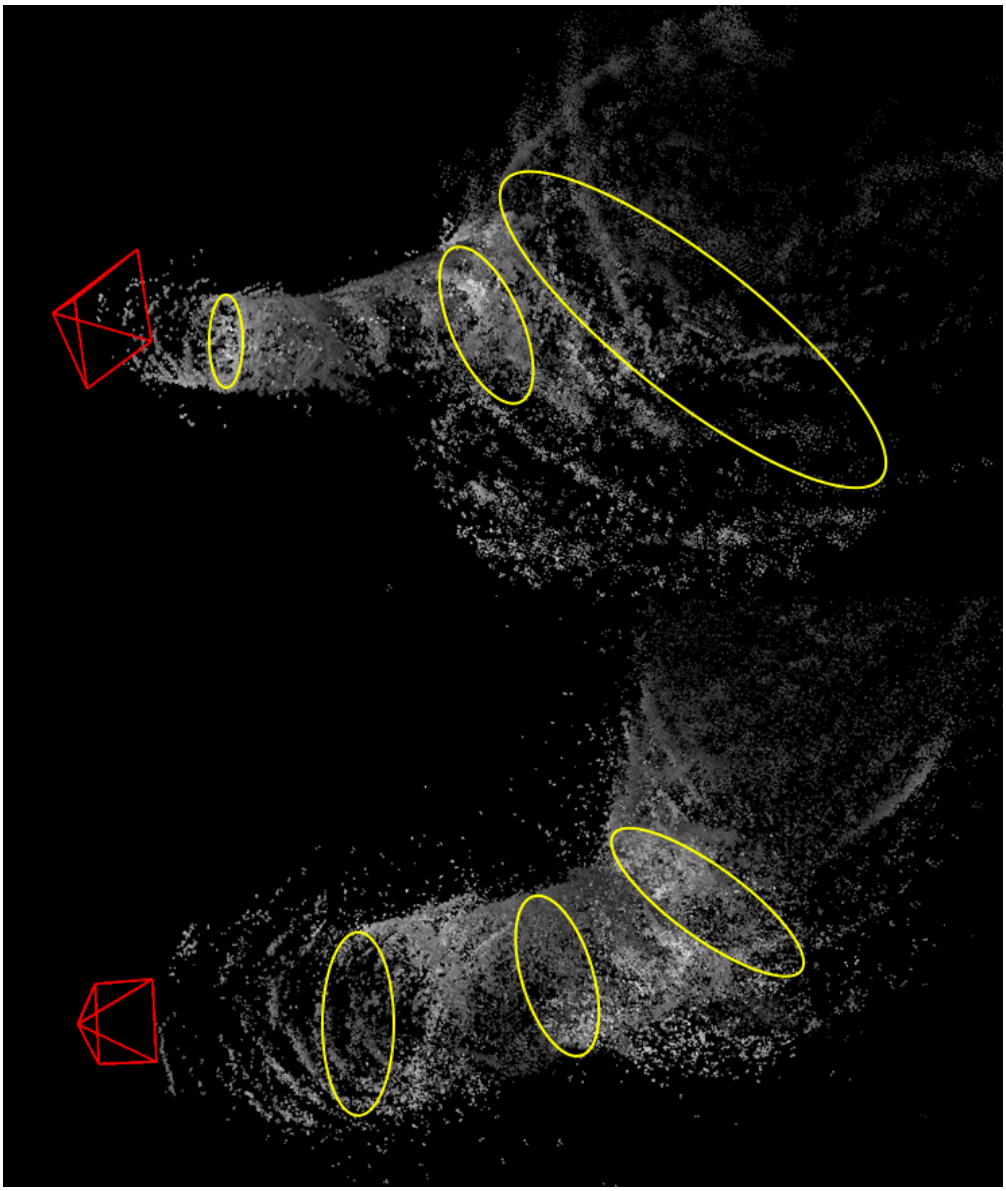


Figure 3.11: Top: scale drift of the DSO on colon data. The camera is moving from right to left. The scale of the colon is dramatically decreasing because of accumulated error. Bottom: no scale drift when using RNNSLAM. The scale-consistent depth maps used in RNNSLAM helps to build a scale consistent model.

To better compare the scale drift of DSO and RNNSLAM, I tested them on a standard SLAM dataset (KITTI [84]), which contains a loop. The dataset is captured by a camera installed on a car

driving through streets. The video is played twice, and the median of depth values is plotted for each frame. As shown in Figure 3.12, in DSO the depth median is dramatically increasing while RNNSLAM's depth median stays consistent.

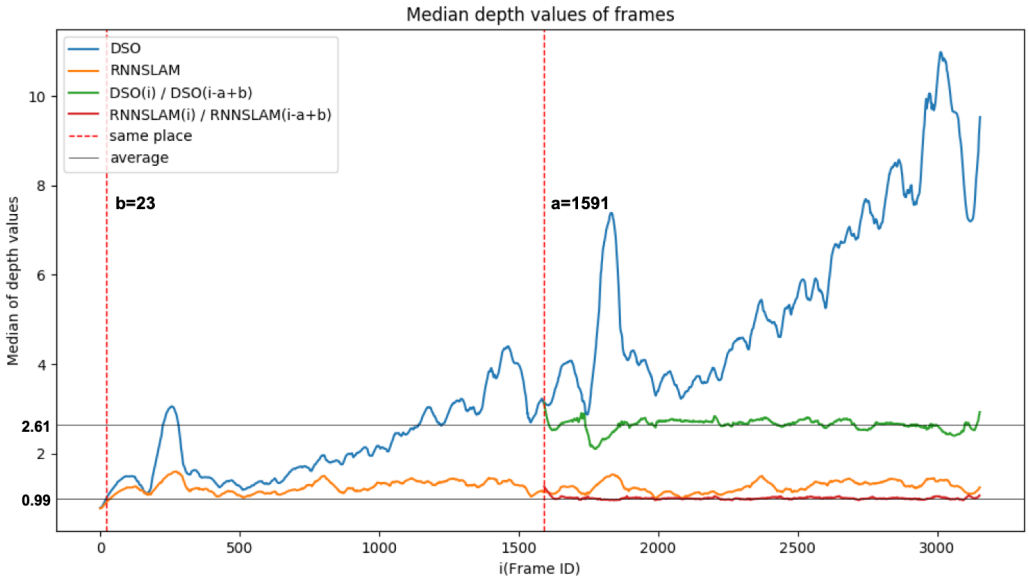


Figure 3.12: Comparison of scale (approximated by depth median of each frame) drift between DSO and RNNSLAM's local windowed optimization. A sequence containing a loop is played twice to compare scale drift. The scale drift of RNNSLAM is much smaller than DSO. The two vertical dashed lines mark two frames at the same place (frame id = a and frame id = b). In the second half, a ratio of current median depth value and its respective value in the first half is computed to characterize relative the scale change. The horizontal lines show the average values of the ratios. RNNSLAM has a average ratio almost equal to 1.0, but it is 2.61 for the original DSO.

Further on this test set, I compare the DSO, the retrained RNN-DP and the RNNSLAM in Figure 3.13. This is the same video I used in Figure 3.12. It drives around a loop, whose starting and ending points are marked in the images. In order to visualize the scale change and camera position drift, I concatenated a part of the beginning video to the end and let the methods continue to run through it. The scale drift of DSO is clearly shown by the blue boxes. For RNN-DP there is obvious camera pose drift because of accumulated pose error. Finally, for RNNSLAM, the scale of the scene is consistent and the camera pose drift is much more smaller. The improvement beyond DSO comes from a more accurate tracking (RNN-DP predicted depth maps); the improvement beyond RNN-DP comes from the local bundle adjustment in a SLAM system. This is a win-win strategy.

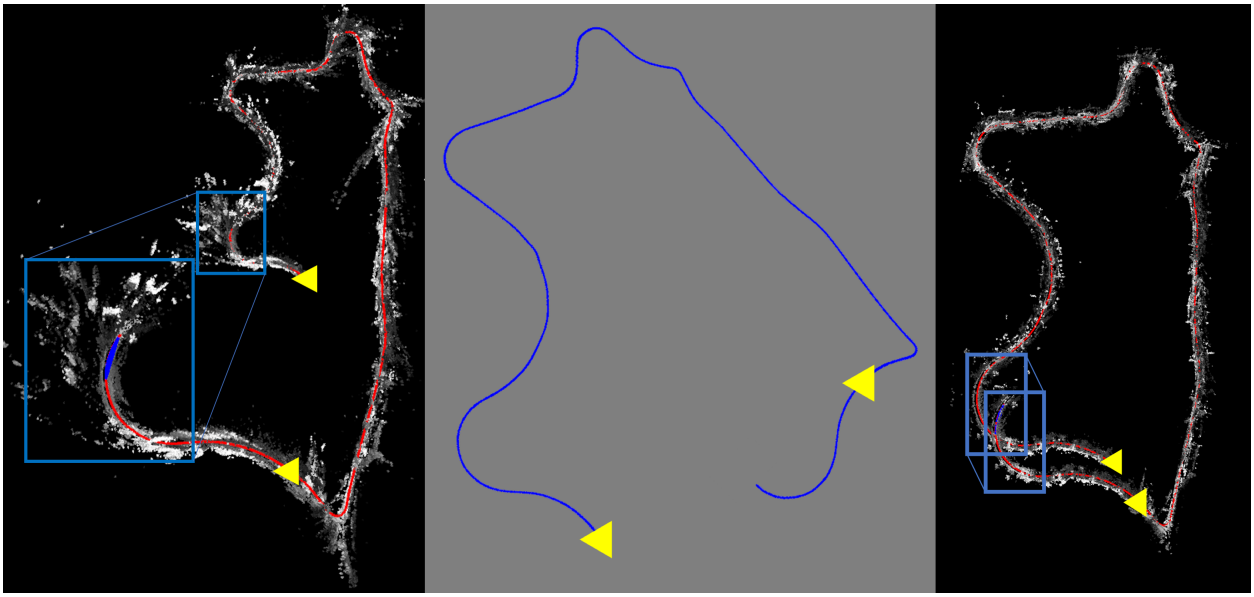


Figure 3.13: Comparison between tracks predicted by DSO (left), by raw RNN-DP prediction (middle), and by RNNSLAM (right). The dataset I used is the KITTI dataset for traditional SLAM, which contains a loop. This can provide a better visualization of the drift problem than colon images. The yellow triangles mark the beginning and ending point of a loop, which should be at the same point in the ground truth. I concatenated a chunk of the beginning frames to the end in order to visualize the scale difference. The region in the blue boxes are exactly the same region. In DSO, their scales and positions are very different. In raw RNN-DP prediction, although the length of the camera path is not stretched, the position still has severe drift. Comparatively, in RNNSLAM, the scale and positional drift dramatically decrease. There is no loop closure used.

3.2.3 Quantitative Results

In this section I provide some quantitative evaluation results of the RNNSLAM. In subsection "Quantify Missing Regions and Verify Their Reliability" I quantify the missing surface ratios and verify the missing regions' reliability. In subsection "Trajectory Accuracy" I show that by combining RNN and SLAM, much more accurate camera trajectories can be achieved. This chapter mainly focuses on integrating RNN and SLAM and reconstructing 3D colon surfaces. With regard to depth map accuracy, there exist concurrent works that can achieve more accurate depth maps [21, 114], but those methods do not produce camera poses or 3D reconstructions. Also, depth map accuracy is not the major focus of my dissertation.

Quantify Missing Regions and Verify Their Reliability In order to measure the surface missing area and quantify the missing ratio, I use a method developed by Yubo Zhang [115] that projects reconstructed colon surfaces onto the 2D plane and clearly reveals the missing regions.

She first establish the centerlines for the 3D colon surfaces, as shown as the black line in the left figure of Figure 3.14. Then for each vertex on the surfaces, she compute its location d along the centerline direction and its angle θ around the centerline. The vertices can now be projected onto the 2D plane according to these position information, forming a 2D flattened surface as shown in Figure 3.14 (right), where r is the average distance between vertices and the centerline. After being projected onto the 2D plane, the points' coordinates are discretized, forming 2D flattened images. Mathematical morphology methods are then used to clean up the noise of surface images. Then the areas of the image background and foreground are computed, which respectively correspond to the missing regions and the surface regions of the reconstructed colons, and the missing ratio of a reconstruction is computed as the area of the image background over the area summation of foreground and background. We estimated missing ratios for the 12 models in Figure 3.6 and 3.7. Their video lengths (in seconds) and their missing surface ratios are shown together with the models. The average missing surface ratio is 13.47%.

To verify the missing regions are reliable, we uniformly sample points from the background regions in the 2D flattened image and map them back to 3D according to their centerline position d and angle θ. The radii (points to centerline distances) used for mapping are interpolated from their nearest points on the surface regions. These "virtual" 3D sample points are shown in Figure 3.15 (left). Next, we render the 3D model, together with the sampled points, to each keyframe based on their camera poses. If a sample point is seen by any of the cameras, it is deemed an incorrect blank point (false positive), which means that we failed to reconstruct this region at this point. We estimated the error rate of the 12 models in Figure 3.6 and 3.7. The average error rate is 12.60%. This means our missing regions are 87% reliable. Besides this quantitative evaluation, we also had an endoscopist to visually compare the 3D model and the original frames and we got the same conclusion[1].

Trajectory Accuracy An open-source library named evo [117] was used for quantitative camera trajectory evaluation. To demonstrate the superiority of the RNNSLAM, I compare it to both

[1]In later work where the input to the SLAM step was preprocessed by a lighting correction [116], the missing surface was 100% reliable.

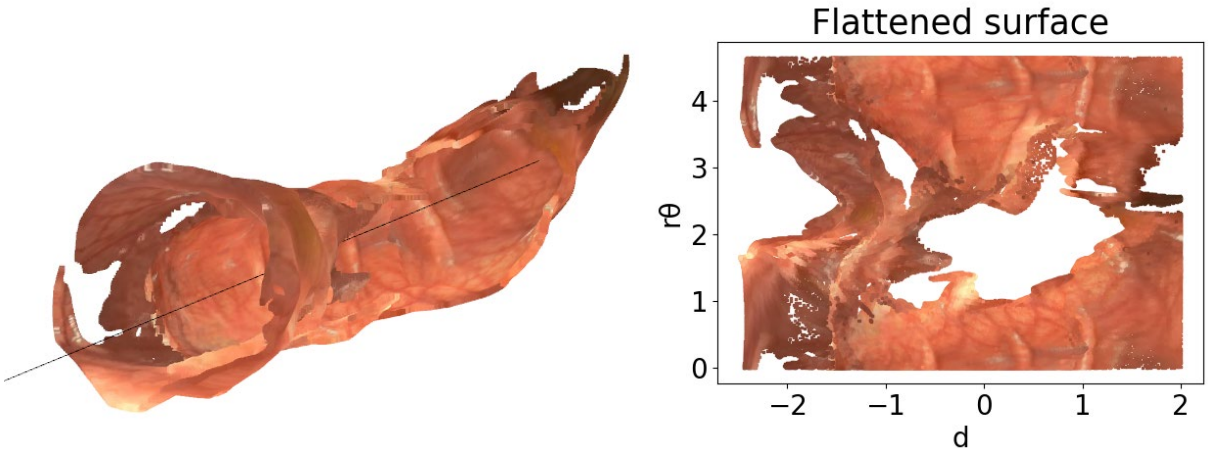


Figure 3.14: Evaluation of unsurveyed area from a 3D colon chunk. Left: A reconstructed chunk with its computed centerline. Right: The flattened surface after projecting each vertex of the 3D reconstruction onto the 2D plane.

DSO and RNN-DP. In order to conduct the quantitative comparison, a groundtruth trajectory is needed. To generate high quality camera trajectories in an offline manner, I used COLMAP [85], state-of-the-art SfM software that incorporates pairwise exhausted matching and global bundle adjustment. These trajectories were then used as "psuedo-ground-truth" for the evaluation.

Because this evaluation result can be affected by the accuracy of COLMAP itself, I also obtained an endoscopic sequence of a silicone phantom[2] whose ground truth camera trajectory was captured by an EM tracker. Fortunately, although our retrained RNN-DP was not trained on any phantom data, its texture is similar enough and our RNN-DP can still generate reasonable depth maps. This sequence can provide a relatively more objective comparison. A sample frame of the phantom sequence and its 3D reconstruction is shown in Figure 3.16.

Evaluation metrics The APE was used to evaluate global consistency between the real-time system estimated and the COLMAP-generated "ground truth" trajectory. The relative pose error E_i between two poses $P_{gt,i}, P_{est,i} \in \mathrm{SE}(3)$ at timestamp i is defined as

$$E_i = (P_{gt,i})^{-1} P_{est,i} \in \mathrm{SE}(3) \tag{3.3}$$

[2]This silicone phantom data was provided by Dr. Nicholas Durr's Lab at JHU

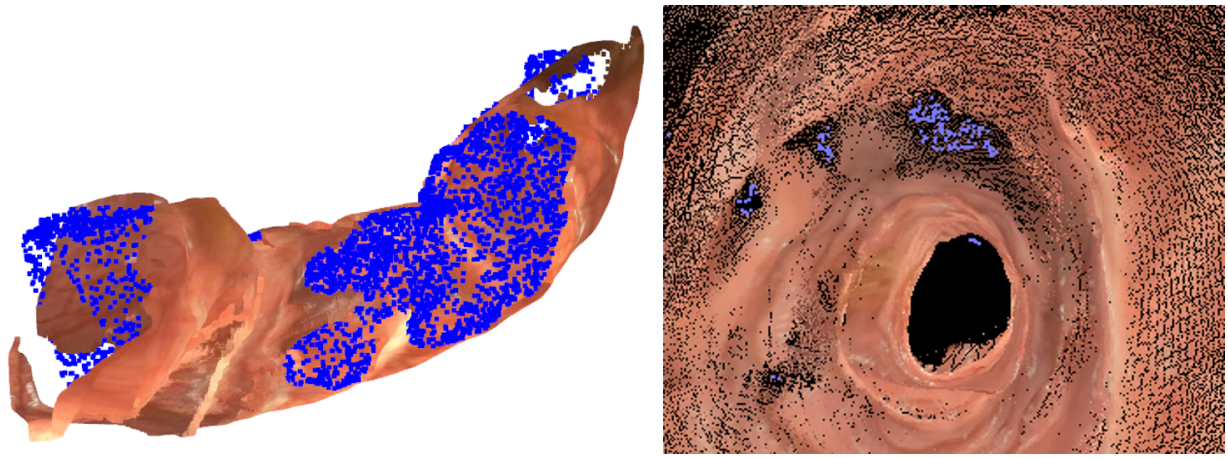


Figure 3.15: Verification of the detected unsurveyed regions. Left: Uniformly sampled points from the background regions in Figure 3.14 are mapped back to 3D and shown together with the original 3D model. Right: The 3D model, together with the sample points are rendered to each keyframe. Any seen blue points means the respective region failed to be reconstructed.

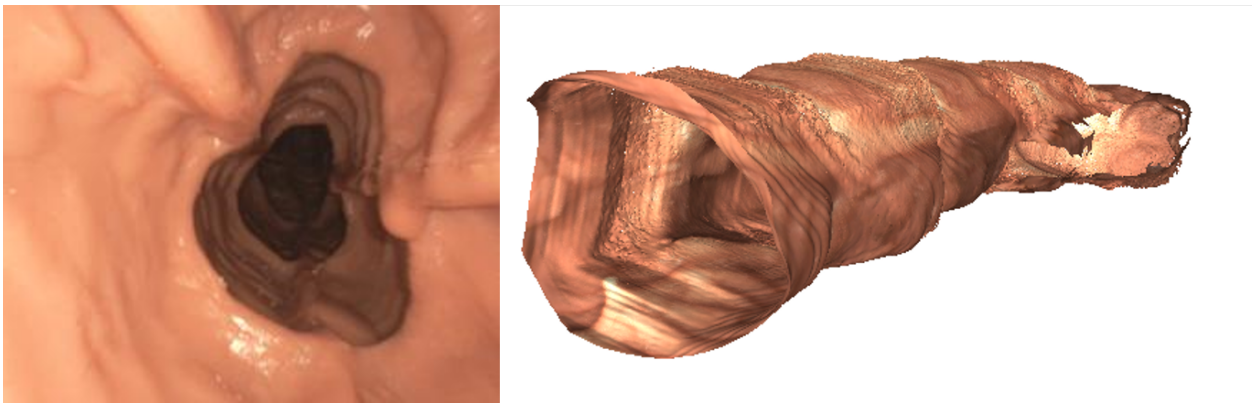


Figure 3.16: 3D reconstruction of a silicone phantom sequence. Left: A sample frame of an endoscopy taken in a silicone phantom. Right: The 3D reconstruction of the sequence.

The APE is defined as

$$\text{APE}_i = ||trans(E_i)||_2 \tag{3.4}$$

where $trans(E_i)$ refers to the translational components of the relative pose error. Then different statistics can be calculated on the APEs of all timestamps, e.g., the Root-Mean-Square Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\text{APE}_i^2} \tag{3.5}$$

The estimated trajectories from RNNSLAM, DSO and RNN-DP are at their own scale and

sampling rates. Therefore, before computing the APE, an additional scale alignment and data association step was applied (detailed in [117]).

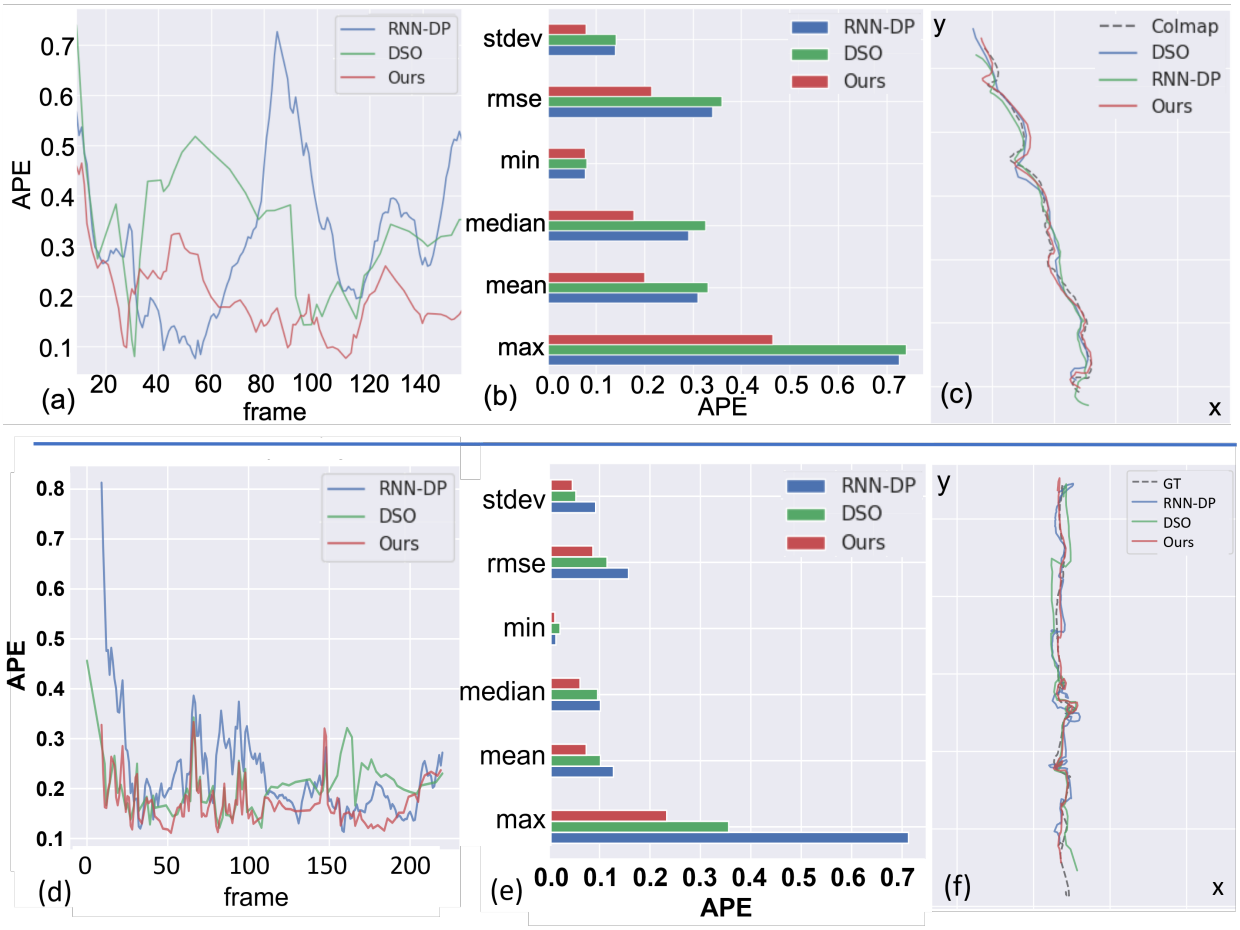


Figure 3.17: Evaluation of trajectory accuracy on two colonoscopy sequence. The top row (a-c) is a real colonoscopic sequence with 156 frames. Its ground truth is computed by COLMAP. The bottom row (d-f) is an endoscopic sequence of a silicone phantom with 220 frames. Its ground truth was captured by an EM tracker. (a/d) APE of the three approaches across the whole sequence. (b/e) Statistics based on APE. (c/f) A bird's-eye view of the full trajectories.

The quantitative evaluation results are shown in Figure 3.17 and Table 3.1. For all the results, the lower, the better since we are measuring the error. Figure 3.17 shows evaluation results on a real colonoscopic sequence (top row) and the silicone phantom sequence (bottom row). (a) and (d) show the absolute pose error (APE) of the three approaches across different time steps in the two sequences; it can be seen that RNNSLAM (red) has the lowest APE at most times. (b) and (e) each show corresponding statistics computed using the APE across the whole sequence; it is clear that RNNSLAM is significantly better than the other two approaches. (c) and (f) show top-down views of

the trajectories of the three approaches together with the ground truth (COLMAP-computed or EM tracker); and RNNSLAM trajectory (red) aligns more closely to the ground truth in both sequences.

Dr. Rui Wang and I repeated the evaluation in Figure 3.17 for 12 testing colonoscopic sequences, and in Table 3.1 I show the statistics of Figure 3.17.b but averaged across 12 colonoscopic sequences: RNNSLAM achieves the best result on all the metrics.

Method	rmse	std	min	median	mean	max
RNN-DP	0.617	0.253	0.197	0.518	0.560	1.229
DSO	0.544	0.278	0.096	0.413	0.465	1.413
RNNSLAM	**0.335**	**0.157**	**0.074**	**0.272**	**0.294**	**0.724**

Table 3.1: Average statistics based on the APE across 12 colonoscopic sequences.

CHAPTER 4

Place Recognition in Colonoscopy

Place recognition in colonoscopy between a present frame and a somewhat temporally separated frame is needed for various reasons. 1) It is needed when the endoscopist wants to recheck a certain region such as the detected unsurveyed regions. 2) When a temporal gap breaks the sequence and a similar view is observed after the gap, place recognition can detect this overlap and continue the reconstruction or connect the reconstructions before and after the gap together. 3) The colonoscopy procedure contains many "close-to-surface" operations such as rotating the endoscope locally to examine the colon surface closely. Similar views often happens before and after such operations. Detecting those overlaps can be useful to establish extra constraint to the reconstruction problem in order to make it more accurate (loop closure). All these recognition tasks are subject to image translation and rotation and a certain amount of surface deformation and lighting change that happen in colonoscopy. A complete unsurveyed region detection system should have these functions in order to be more accurate and practically useful. Implementing these functionalities are out of the scope of this chapter. This chapter tries to establish the prerequisite of them, namely, the place recognition technique in colonoscopy.

As detailed in Section 2.8, a place recognition or instance-level image retrieval task is as follows: For a query image (q), find images in a database (D) that share visual contents. Applying this to colonoscopic place recognition, the task is to find similar images in historical frames that were taken at the same colonic place.

In this chapter I present my work on colonoscopic place recognition by a deep learner. To the best of my knowledge, this is the first study in this area. I contribute a dataset (Colon10K) for colonoscopic image retrieval evaluation [31]. The dataset contains 20 short sequences (10216 images) with positive matchings manually labeled. I also summarize evaluation metrics because they are not always consistent across literatures and tasks. Moreover, there lacks a working method for colon place recognition because of the difficulties for colonoscopic images. I conducted an extensive

experiment to compare different existing methods, and I successfully trained working models for colonoscopic image retrieval. I also demonstrate the capability by applying a model on a recognition task for a long colonoscopic sequence and on a loop closure task for a reconstruction job. With these, I establish a benchmark for place recognition in colonoscopy.

This chapter is organized as follows: I will introduce the method I used for place recognition in Section 4.1. I will summarize the evaluation metrics in Section 4.2. In Section 4.3 I will introduce the Colon10K dataset. Finally, experiments will be presented in Section 4.4. The software of this project is in https://github.com/RuibinMa/ColonImageRetrieval.git.

4.1 Method

My method is heavily based on Radenović et al. [99], a strategy to fine-tune an off-the-shelf CNN for retrieval tasks using SfM results.

In detail, my method is composed of the following aspects:

Architecture An off-the-shelf CNN is taken from a trained classification network, e.g., VGG16 [83] for ImageNet [98] classification challenge. Instead of connecting the final convolutional layer to a dense layer, a pooling layer is connected to the final convolutional layer to form a global descriptor. The dimension of the descriptor equals the number of feature maps in the final convolutional layer. Different pooling strategies have been proposed, for example, max-pooling [27], sum-pooling [28], generalized mean pooling [30], and netvlad pooling [29]. So far, the generalized mean pooling [30] has given the best retrieval performance for common computer vision datasets such as Oxford5K and Paris6K [118]. For colonoscopy, I used two kinds of architectures, ResNet101 [112] and VGG16 [83]. I also tested all of the four aforementioned pooling strategies.

Training Strategy The whole network is fine-tuned with known positive matching pairs and hard-mined negative pairs by siamese training [119]. Siamese training is a strategy to train a neural network to find embeddings with expected behaviors. For example, the embeddings of images that share contents should should be closer in the Euclidean space. Figure 4.1 illustrates the siamese training strategy. In our case, the "inputs" corresponds to images and the "embeddings" corresponds to image descriptors which are vectors. In siamese training the network is run twice: once each on a pair of inputs to get two embeddings. A contrastive loss is computed based on the two embeddings

and whether they are a positive pair or a negative pair. If they are a positive pair, the contrastive loss will penalize the difference between them. If they are a negative pair, the contrastive loss will penalize the similarity between them.

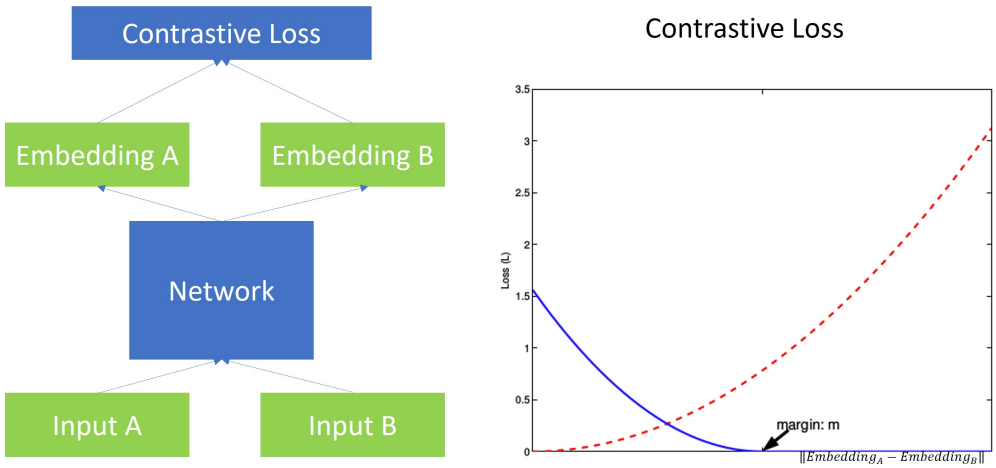


Figure 4.1: Left: siamese training strategy. Right: Contrastive loss, image taken from [120]. The red, dashed curve corresponds to the case when the two inputs are "positive" (have overlap). The blue, solid curve corresponds to the case when the two inputs are "negative" (do not have overlap).

Specifically, the contrastive loss is defined as

$$\text{Constrastive Loss} = \begin{cases} \|E_1 - E_2\|^2 & \text{positive} \\ \max(0, m - \|E_1 - E_2\|)^2 & \text{negative} \end{cases} \tag{4.1}$$

where E_1 and E_2 stand for the two embeddings. m is the so-called "margin". The right figure in Figure 4.1 shows the loss curves in the two scenarios, of negative or positive pairs. When the pair is negative and the magnitude of $\|E_1 - E_2\|$ is larger than the margin, the descriptor is deemed satisfactory and will not affect the loss. In other words, we just want the descriptors of negative pairs to be not too close to each other rather than as far as possible. On the other hand, we want the descriptors of positive pairs to be as close as possible.

The spirit of siamese training and contrastive loss is to leverage pair-wise ground truth instead of sample-wise ground truth. All desired behaviors of the embeddings are directly encoded in the contrastive loss. The challenge is to get pair-wise ground truth.

Positive Pairs The positive matchings between images are acquired automatically from SfM. Only matchings that survived the SfM pipeline are used as positive matchings; these matchings are reliable. For colon images, Dr. Rui Wang and I collected 60 full colon videos and split them into >12k 200-frame chunks. For each chunk, we ran SfM with exhaustive matching and global bundle adjustment. Each SfM model gives matchings between frames that share common keypoints. Duplicated pairs between chunks were deleted. Pairs of nearly identical images were also deleted for more efficient training. The remaining matchings were used as positive pairs for training. Because the positive pairs generated by SfM are those that can be successfully registered by the SfM algorithm, this strategy is biased toward easy-to-register frame pairs. Other hard-to-register images pairs end up being given less significance in the training dataset.

The training process is split into epochs. In each epoch, I randomly choose a number (2500 in practice) of positive pairs. One of the images in a pair is regarded as the query image for negative mining.

Negative Pairs Mining Negative pairs are automatically found online by hard negative mining. Specifically, after each epoch I randomly select a large number (50000 in practice) of images from all the images. I run the latest version of the model on these images to compute their "so-far-the-best" descriptors. For the query images in the next epoch, I also compute their descriptors. These two groups of images are matched using their descriptors. For each query image, I find the top 5 scored images that are not positively matched in ground truth. These 5 negative images will form 5 negative pairs for training.

4.2 Metrics

As groundtruth, D comprises positive images that share overlapping content with q and negative images that do not have overlap with q. When we have a set of tasks whose q's, D's and groundtruths are known, we can use the following three metrics to evaluate a model's performance.

1. **Recognition rate at rank** N: For each task, the images in D are ranked by their similarities to q. The task will be considered successful if at least one of the top N images are true positive. This metric is the percentage of successful tasks at rank N, also called "recall at rank N" in [121, 29]. This metric is usually applied for geo-localization tasks, which have relatively higher tolerance to false positives than in colonoscopy.

2. **Mean average precision**: For each task, a precision and a recall can be computed at a given threshold. When this threshold varies, an average precision can be computed by taking the area under the precision-recall curve. The mean average precision (mAP) is the mean of average precisions of all tasks, which was used in [30, 95]. This metric is a general measurement of the model's robustness against a varying threshold. It is widely used in instance-level image retrieval.

3. **Recall at 100% precision**: By adjusting a global threshold, we can compute the largest global recall (number of all true positives of all tasks / number of all groundtruth positives) that the system can achieve when the precision is 100%. This metric was used in [122] for loop closure, which requires no false positives.

4.3 Colon10K Dataset

4.3.1 Groundtruth labeling

I selected 20 colonoscopy subsequences whose numbers of frames range from 104 to 879. For each sequence, every 30th frame was chosen as a query frame. For each query, I did the following three steps to form its database:

1. All the frames in the other 19 sequences were used as negative images in its database.

2. I manually labeled the frames in the current sequence that were at the same colonic place (visually recognizable purely based on appearance). These annotated frames were used as positive images in the database for this query frame. This annotation was verified by an endoscopist.

3. The rest of the frames in this sequence can be ambiguous because it is hard to say whether they have overlap with the query or not with 100% certainty. They were thus not included in the database (not used for evaluation of the current query). Therefore, although some unannotated frames may still have some overlap, they will not negatively affect the evaluation quality. In other words, we will only care about the capability of recognizing the annotated frames.

4.3.2 "Indirect" matchings

In fact, a query can appear in multiple groups of consecutive frames (intervals) in a sequence because of loops or temporal gaps. The intervals that are not direct temporal neighbors (nearby frames) of a query are generally more interesting because they are real loop closure examples in colonoscopy instead of local similarity examples caused by camera continuity. I therefore treat each interval as an independent retrieval task. For example, if a query matches N intervals in its sequence,

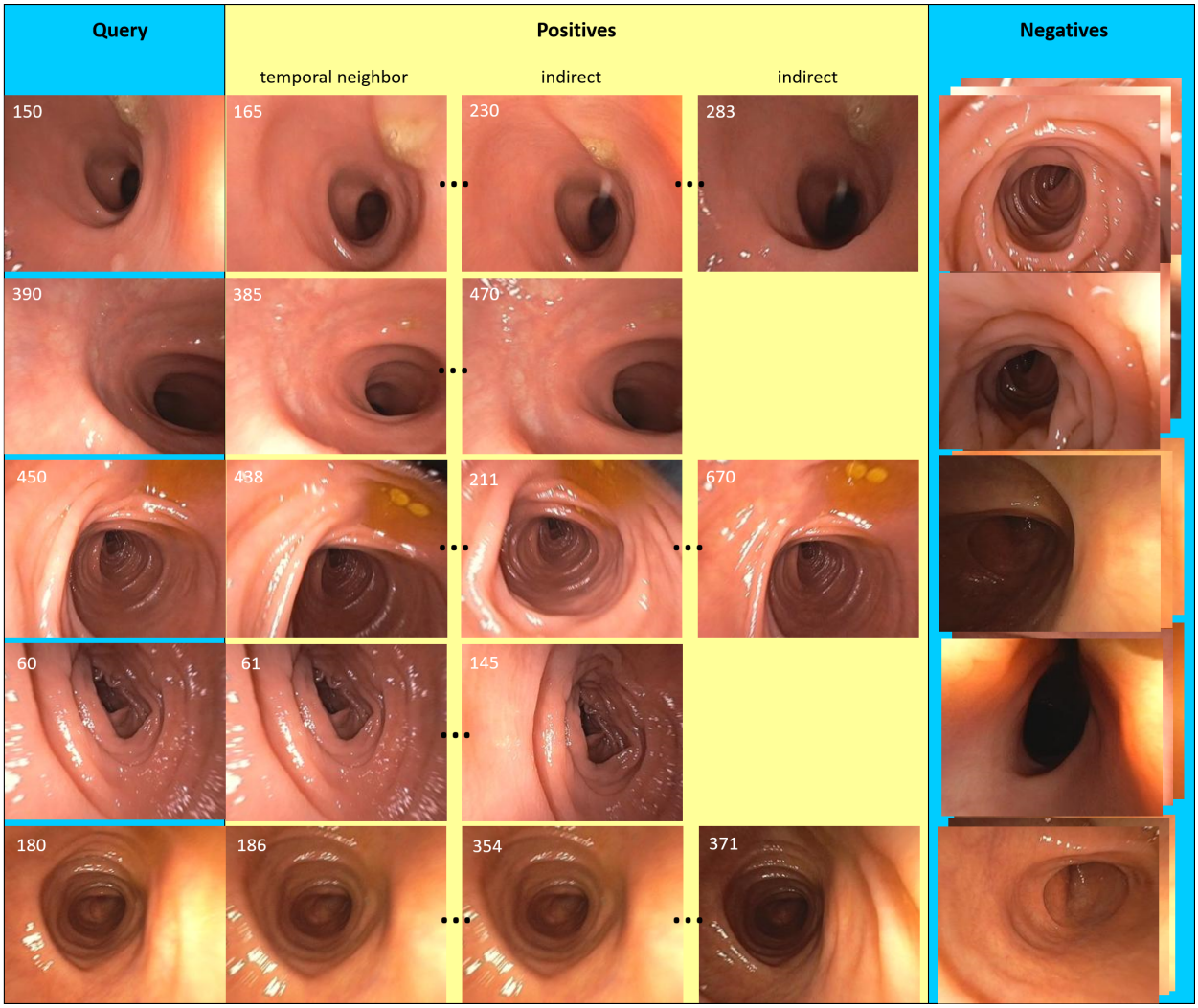


Figure 4.2: Five examples of query tasks in Colon10K dataset. A retrieval task comprises a query and a database. The database is composed of positives and negatives. For example, for the first query, there are three intervals matching to it in its sequence. I show one sample for each interval. It will contribute three retrieval tasks. The second query has two matching intervals and will contribute two retrieval tasks. The upper left number is the frame id in their respective sequences. The negatives are all the images in the sequences different from that of the query.

it will contribute to N recognition tasks that use the same query frame and negative images but different intervals of positive frames. In total, I generated 620 tasks (denoted by *all*). 309 tasks uses the intervals that are not direct neighbor frames of their queries as positives (denoted by *indirect*).

Figure 4.2 shows five examples of query tasks. The positive matchings show translational, rotational and lighting change and certain level of deformation. This Colon10K dataset is publicly available at https://endoscopography.web.unc.edu/place-recognition-in-colonoscopy/.

4.4 Experiments

I tested two architectures (ResNet101 [112] and VGG16 [83]) and four kinds of pooling (max-pooling [27], sum-pooling [28], generalized mean pooling [30] and netvlad pooling [29]). I also compared the performance of the off-the-shelf networks and their fine-tuned versions. I used cosine similarity and exhaustive matching for evaluation.

Baseline Schönberger et al. [95] proposed an instance-level image retrieval method based on SIFT feature [10] and BoW representation. A vocabulary tree is used to retrieve raw candidates and a fast spatial verification method (vote-and-verify) is performed to rank the candidates by the number of inlier keypoint matchings. I tested their method on two vocabularies: The first is learned from common images; this vocabulary is published in [100]. The second is learned from a large collection of colon images (no overlap with Colon5K). The local feature tuple I used is domain-size-pooling SIFT [11].

4.4.1 Comparison

I compare the performance of the SfM-result-trained CNNs and the baseline under the three metrics in Section 4.2 and on both "*all*" and "*indirect*" datasets.

Table 4.1 shows of the recognition rates at rank 1. It typically represents how likely it is to relocalize an image without error when we are taking the top-ranked database image. Table. 4.2 shows the mean average precision comparison. This criterion measures the average performance at all thresholds. This metric is used to pick the best model while training. Table. 4.3 shows the recall at 100% precision. From these three comparisons, I have the following observations:

1. Fine-tuning or domain adaptation is crucial for model performance. This is reflected in both deep learning methods (fine-tuning) and the traditional methods (custom vocabulary on colon images). Off-the-shelf networks almost cannot work on colon images because the features in

Recognition rate at rank 1		
Method	*all*	*indirect*
BoW-common vocabulary-sp	0.6129	0.2460
BoW-colon vocabulary-sp	0.6726	0.3560
VGG16-off the shelf-max	0.7097	0.4207
VGG16-max	0.8935	0.7864
VGG16-spoc	0.8823	0.7638
VGG16-gem	0.9000	0.7961
VGG16-netvlad	0.8661	0.7314
Resnet101-off the shelf-max	0.6871	0.3722
Resnet101-max	0.8952	0.7896
Resnet101-spoc	**0.9032**	**0.8058**
Resnet101-gem	**0.9032**	0.8026
Resnet101-netvlad	0.8516	0.7023

Table 4.1: Comparison of recognition rate at rank 1.

Mean Average Precision (mAP)		
Method	*all*	*indirect*
BoW-common vocabulary-sp	0.4116	0.0764
BoW-colon vocabulary-sp	0.5642	0.1925
VGG16-off the shelf-max	0.4463	0.0475
VGG16-max	0.8834	0.7702
VGG16-spoc	0.8762	0.7204
VGG16-gem	0.8869	0.7704
VGG16-netvlad	0.8612	0.6843
Resnet101-off the shelf-max	0.4453	0.0559
Resnet101-max	0.8972	0.7989
Resnet101-spoc	**0.9042**	**0.8245**
Resnet101-gem	0.9006	0.8159
Resnet101-netvlad	0.8582	0.6808

Table 4.2: Comparison of mean average precision on two manually labeled test set.

colon are very different from common images.

2. The performances of "*all*" and "*indirect*" datasets are significantly different. "*indirect*" is much more difficult. On the other hand, it is closer to realistic because the matchings are caused by camera re-visiting, not by local continuity. We should pay more attention to model behavior on this set.

3. Although there is a large gap between the performances of "*all*" and "*indirect*", the gap is mitigated after fine-tuning because the networks have learned robust features against the

Recall at 100% Precision		
Method	*all*	*indirect*
BoW-common vocabulary-sp	0.2213	0.0164
BoW-colon vocabulary-sp	0.3718	0.0754
VGG16-off the shelf-max	0.1676	0.0039
VGG16-max	0.5963	0.2773
VGG16-spoc	0.5583	0.1770
VGG16-gem	0.6269	0.2829
VGG16-netvlad	0.5815	0.2020
Resnet101-off the shelf-max	0.2199	0.0065
Resnet101-max	**0.7215**	0.4910
Resnet101-spoc	0.6911	0.3786
Resnet101-gem	0.7167	**0.4922**
Resnet101-netvlad	0.4268	0.0788

Table 4.3: Comparison of recall at 100% precision on two manually labeled test set.

change of content. This further proves that it is important to fine-tune networks by colonoscopic images for related tasks.

4. Resnet101 is the best performing architecture. The best pooling strategy is not always consistent across metrics. For Resnet101, generalized mean pooling generally performs better on recall at 100% precision, but sum pooling performs better on the other two metrics. Researchers should choose appropriate metrics according to specific applications.

5. Deep learning methods perform much better than the traditional methods. One reason is that colon images are very low-textured. There are not enough robust SIFT keypoints to detect.

Additionally, Figure 4.3 and 4.4 shows ten examples of randomly chosen true positives and top-ranked false positives for ten queries using network "Resnet101-spoc" fine-tuned on colon images. First, the true positives show that large translational and rotational change of the view, deformation and lighting change can be handled by the CNN image retriever. Second, the false positives are close to the queries in terms of high-level structure but their local details mismatch. This verifies that the network is learning reasonable details.

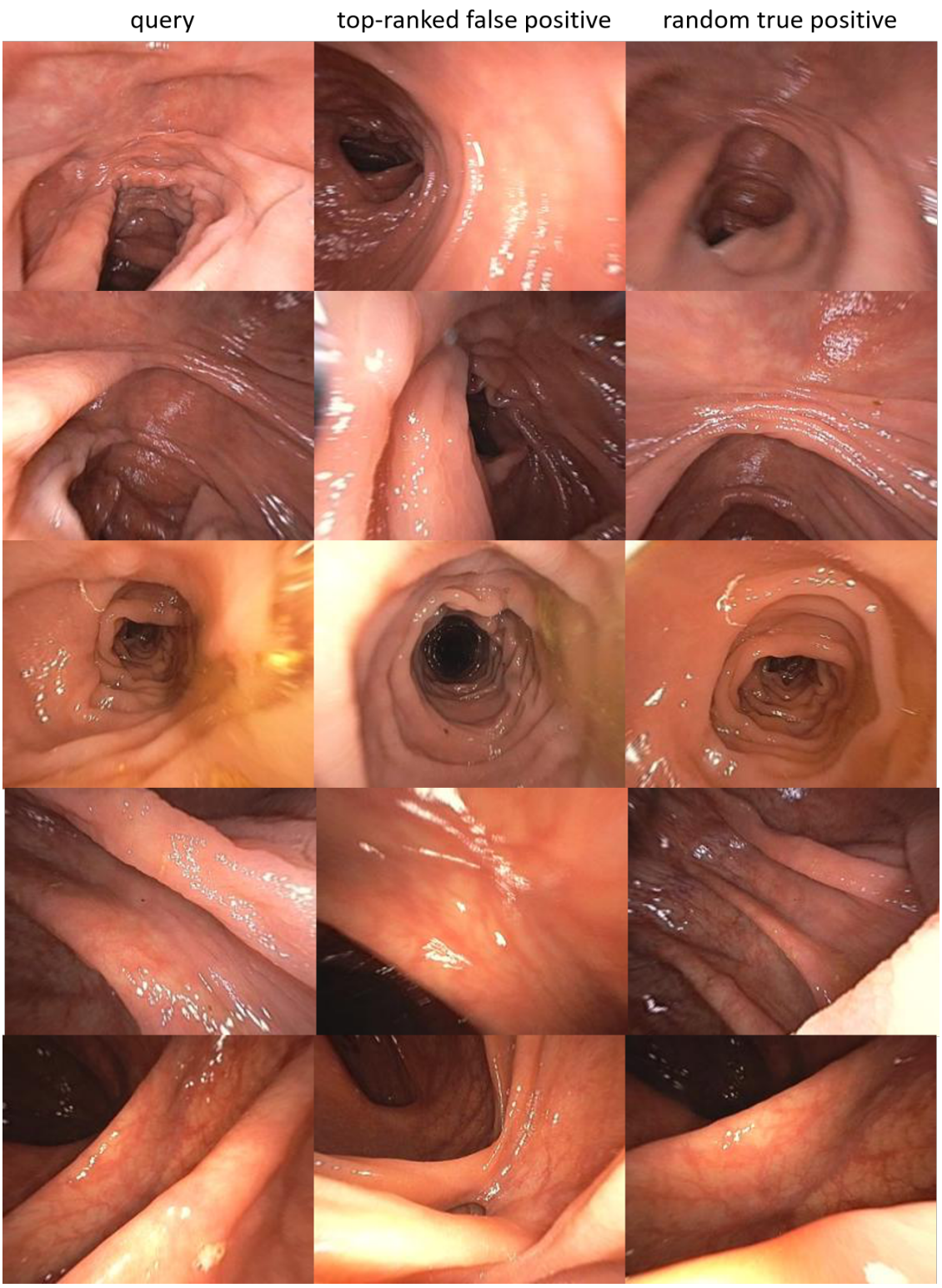


Figure 4.3: Five examples of top-ranked false positive examples and a random selected true positive (threshold=0.75) using Resnet101-spoc.

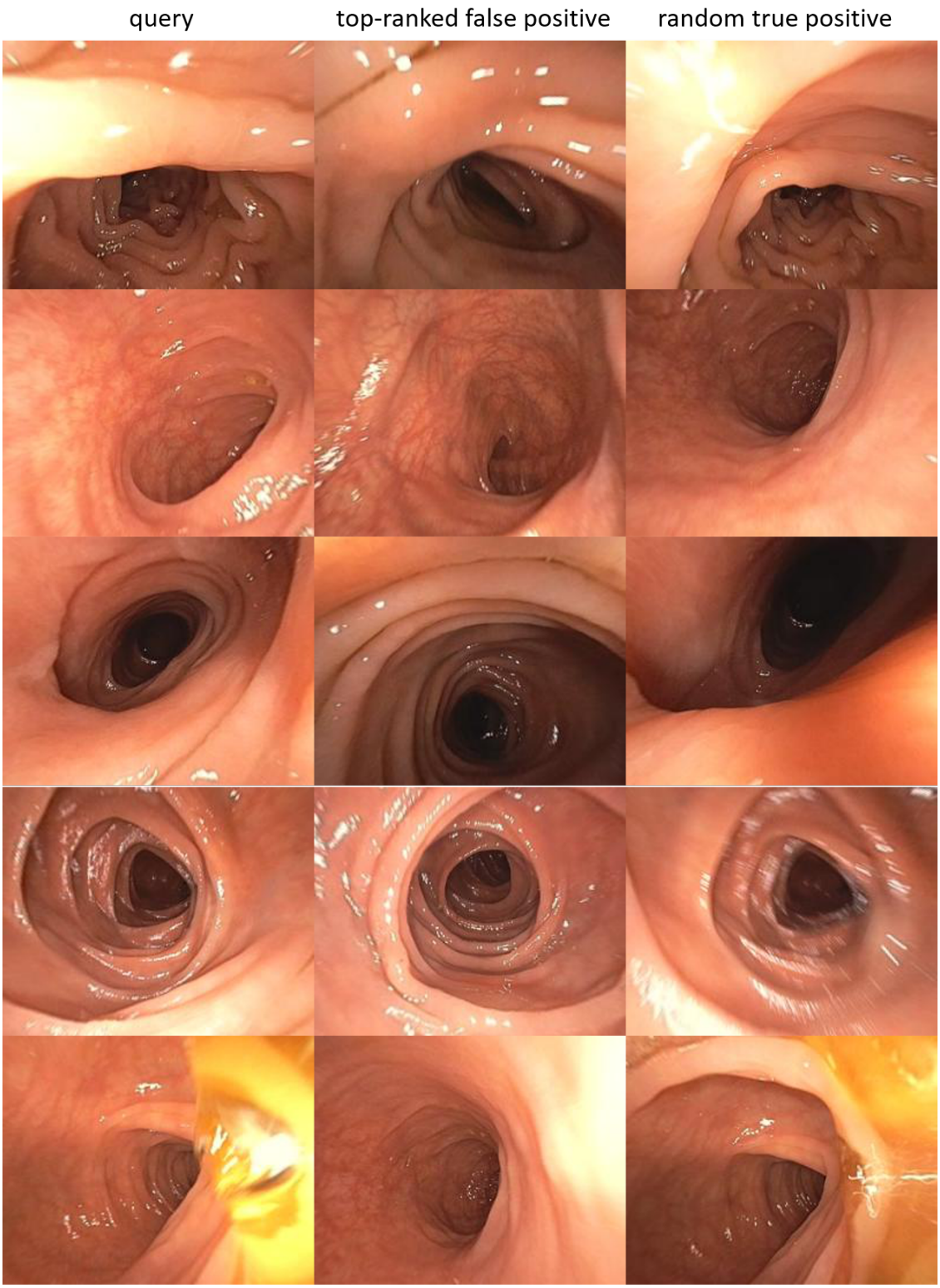


Figure 4.4: Another five examples of top-ranked false positive examples and a random selected true positive (threshold=0.75) using Resnet101-spoc.

4.4.2 Application Demos

In order to fully show my model's capability of recognizing colonic places, I present the following two demos.

Demo 1: Testing on a 2000-frame Colonoscopic Sequence. In this demo I run the trained Resnet101-spoc model on a real colonoscopic video that contains 2000 frames. As shown in Figure 4.5, the layout is composed of the current frame, the retrieved frames on the top, and a similarity score curve. The scores are computed by the dot products between the descriptor of the current frame and the descriptors of the historical frames. Historical frames whose score is higher than 0.75 and also higher than any other frames' scores in a 50-frame window are selected, ranked by their scores, and shown on the top. This CNN trained by SfM results is capable of retrieving frames subject to large deformation and lighting changes. In the bottom picture of Figure 4.5 the model is able to retrieve a frame that is almost 2000 frames away (more than 1 minute of video in between).

Demo 2: Loop Closure of Colonoscope Trajectory. Loop closure is an optional but very useful component in SLAM systems. To fully understand loop closure, we need to get familiarized with pose graph optimization.

Pose graph optimization is a technique to optimize the camera poses in a graph (typically formed by SLAM methods) when the constraints are redundant. A camera trajectory with N keyframes (vertices in the pose graph) is constructed with at least $N-1$ relative camera poses. Those are the edges in the pose graph. When the number of edges are larger than $N-1$, the system is over-defined, i.e., the constraints (edges) are redundant. From an optimization perspective, the edges are called "observations". The pose graph can be optimized based on those observations:

$$p_1, p_2, ..., p_N = \text{argmin}(\sum_{(i,j)\in edges} \mathbf{e}_{ij}^T \Omega \mathbf{e}_{ij}) \tag{4.2}$$

where p_1 to p_N are the camera poses in the world coordinate system, Ω is a weighting matrix (a diagnal matrix containing the significances of the translational and rotational components of poses) and $\mathbf{e}_{ij}$ is the residual of edge (i, j). Supposing p is composed of a translational vector t and a

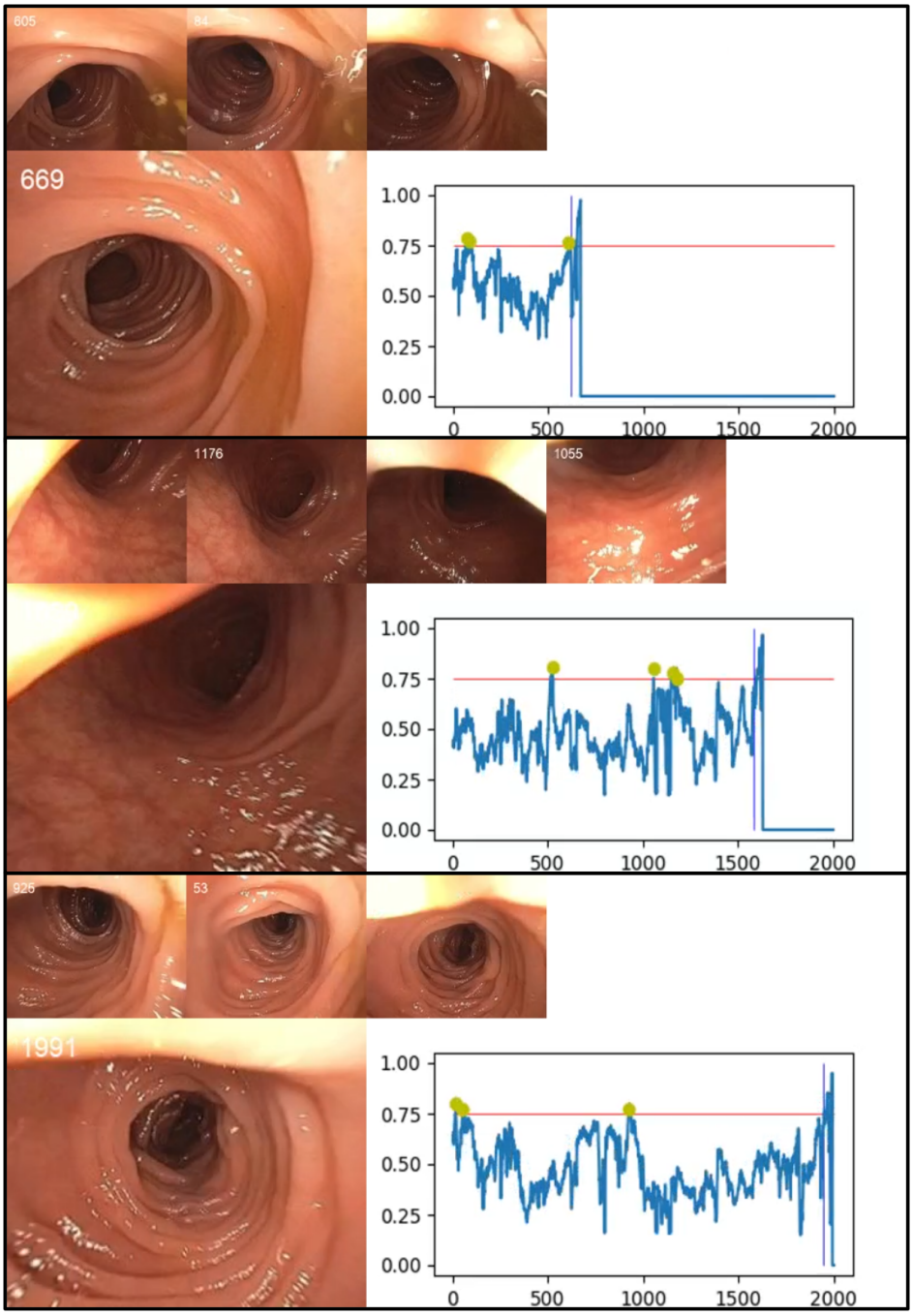


Figure 4.5: 3 pictures of a recognition demo. Resnet101-gem is run on a 2000-frame colonoscopic sequence. The current frame is shown in the bottom left. In the chart of each picture, the x-axis is frame number and the y-axis is the similarity score. The score curve is computed by the dot products between the current frame's descriptor and historical frames' descriptors. The latest 50 frames are ignored because they are very similar to the current frame. This is shown by the vertical thin blue line. The frames whose scores are higher than 0.75 (red line) and are also the highest in a temporal window are marked by the yellow dots. They are shown on the top of each picture.

quaternion q, $\mathbf{e}_{ij}$ is defined as

$$\mathbf{e}_{ij} = p_{ij}^{-1} \otimes (p_i^{-1} \otimes p_j) \tag{4.3}$$

where $p_{i,j}$ is the relative camera pose from i to j (the observation) and $\otimes$ is the motion composition operator:

$$p_i \otimes p_j = \begin{bmatrix} q_i(t_j) \\ q_i \cdot q_j \end{bmatrix} \tag{4.4}$$

This is a non-linear least squares problem. Efficient solvers for it have been implemented in libraries like G2O [79]. Through the use of such a solver, a more consistent pose graph can be achieved .

The redundant edges are usually added to the pose graph in a loop closure thread. Specifically, the basic process of loop closure is composed of three steps:

1. Detection: Find historical keyframes that share the same visual content with the current keyframe based on appearance.

2. Estimation: Estimate the relative camera pose between the current keyframe and the retrieved keyframe. This relative pose is added to the pose graph.

3. Optimization: Optimize the pose graph as detailed previously. Global bundle adjustment can also be used when 3D point correspondences are available.

In this demo I used the RNN-DP to predict relative camera poses between successive frames. The interface is shown in Figure 4.6. The current keyframe, its depth map and the latest detected matching image pair are shown on the left. The camera trajectory is shown on the right. The video of this demo was recorded from a silicone phantom and contains forward-backward motion. Some of the frames are expected to overlap during that motion.

Besides the thread of RNN-DP, there is a separate thread running in parallel for loop closure. The detection is based on my trained CNN image retriever. I set a very high threshold for the similarity in order to ensure the retrieved frame is physically close to the query frame. When a matching keyframe is detected for the current keyframe, a new RNN-DP is created. It takes these two frames sequentially and outputs a relative camera pose. This pose is added to the pose graph for optimization.

In Figure 4.7 I show the trajectories at seven time points. At each snapshot surrounded by a

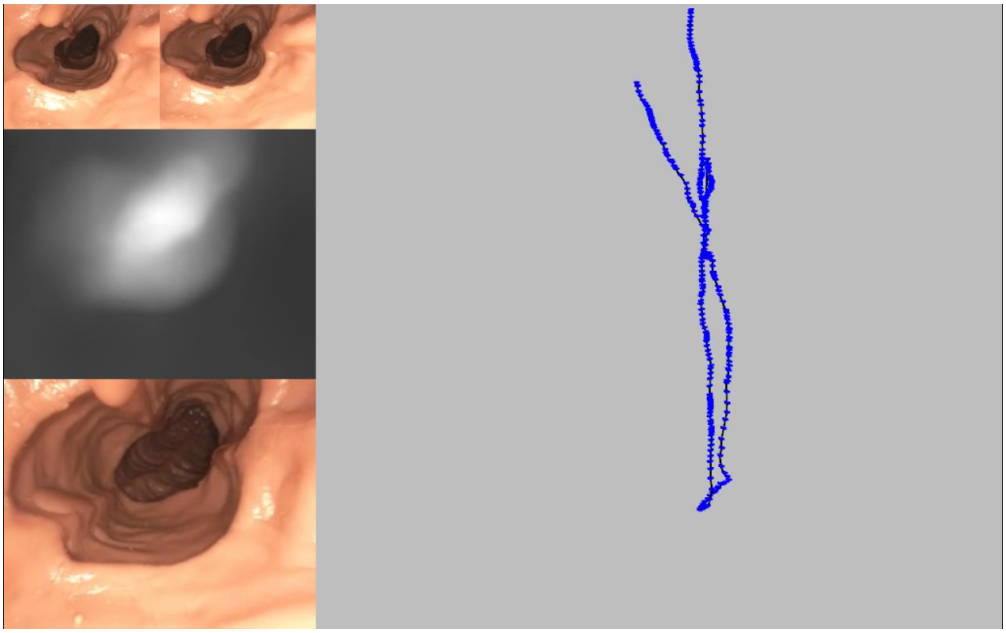


Figure 4.6: Loop closure demo interface. The bottom left figure shows the current frame. The middle left figure shows current frame's depth map estimated by RNN-DP. The top left two images shows two matching keyframes. This pair is detected when the camera revisits a place after a time threshold. The detection is done by my CNN image retriever. The right image shows the current camera trajectory.

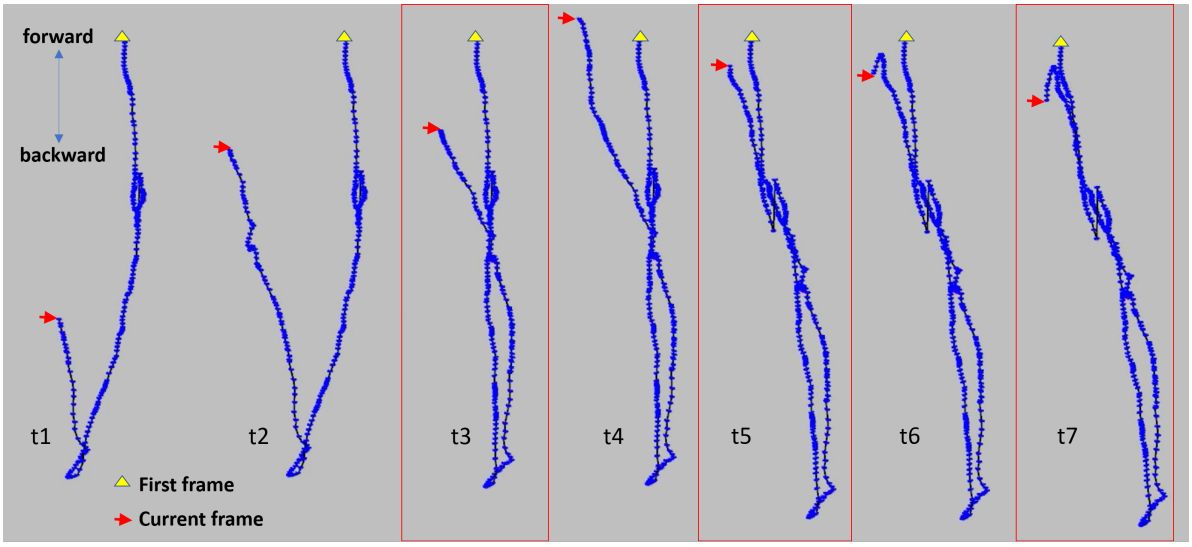


Figure 4.7: Loop closures in a colonoscope trajectory. The camera started from the yellow triangle, and moved downwards and then upwards. Pictures "t0" to "t7" shows seven snapshots of the camera trajectory at seven sequential time points. The pictures bounded by red boxes are snapshots after a loop closure's pose graph optimization, so there are three matching image pairs detected. It can be observed that a certain amount of drift is fixed after each loop closure.

red box, a matching keyframe was successfully detected and the trajectory got optimized. The camera trajectory is improved each time a loop is detected. Figure 4.8 compares the original (raw) trajectory produced by RNN-DP, the trajectory with loop closure, and the ground truth trajectory. The trajectory with loop closure is much more similar to the ground truth and its drift is significantly smaller. On the right of that figure I show a zoom-in picture of two parts of the trajectory where two loop closure edges were detected (the red lines). Again, these two poses were estimated by RNN-DP.

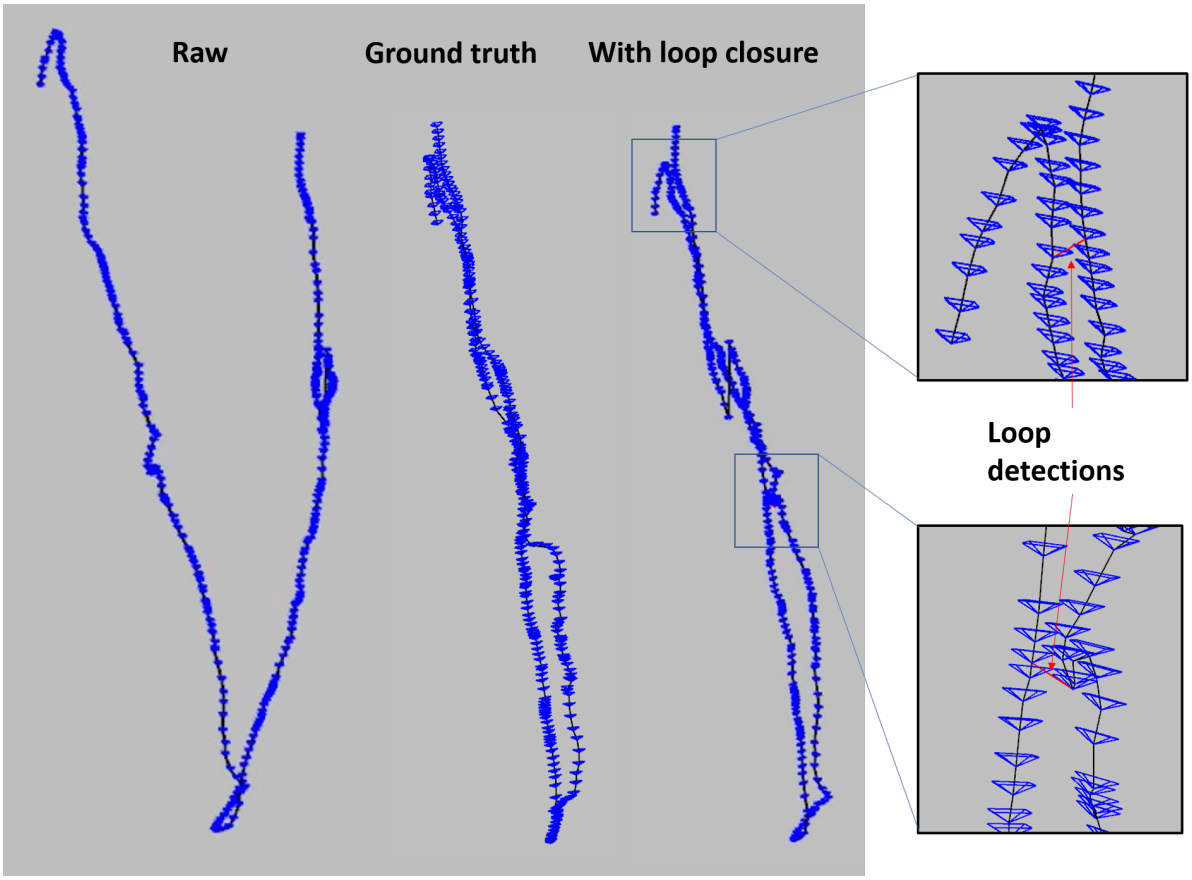


Figure 4.8: Comparison between the original (raw) trajectory predicted by RNN-DP (left), the trajectory fixed by loop closure (right), and the ground truth trajectory. The right pictures shows two parts of the trajectory in detail. The red line connects two matching keyframes detected by the CNN image retriever. Those relative poses are predicted by RNN-DP. The trajectory with loop closure is much closer to the ground truth because of the extra constraints (the red lines).

In Figure 4.9 I show the quantitative comparison between the trajectories with and without loop closure. The metric is the absolute pose error (APE, defined in Equation 3.4) to ground truth trajectory recorded by EM tracker. The result shows that the trajectory with loop closure is much more accurate than the one without loop closure. The mean APE reduced by more than 75%.

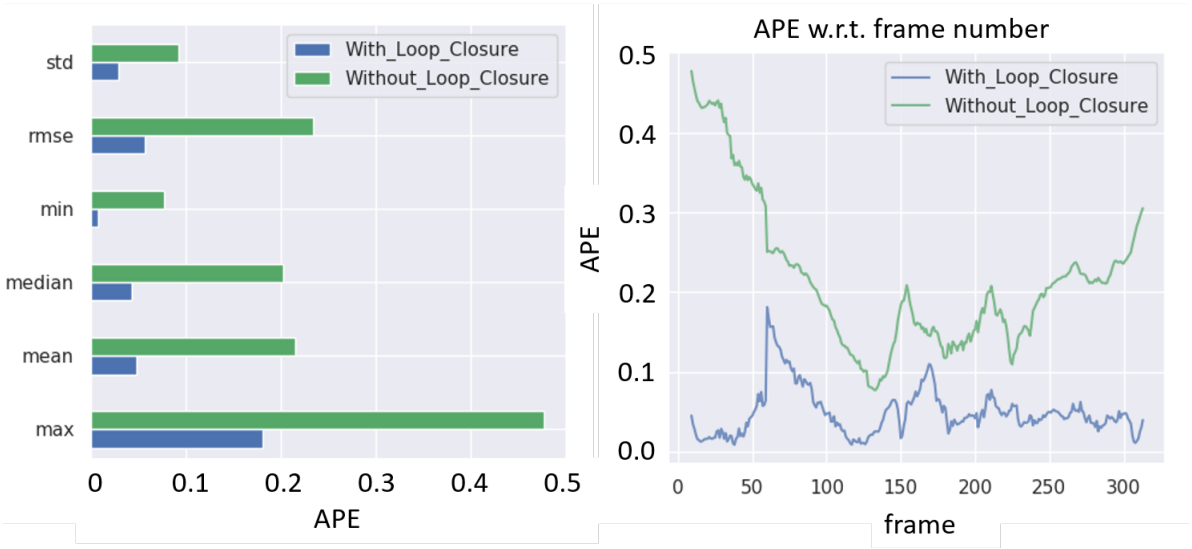


Figure 4.9: Loop closure result statistics. At each point in the pose graph an APE relative to the ground truth can be computed. The left figure shows the statistics of all the APEs throughout the trajectory. The error was significantly reduced by loop closure. The right figure shows the APE vs. the frame number. The trajectory with loop closure almost always have lower APE than the other one.

Our final goal is to improve reconstruction accuracy via loop closure. In Figure 4.10 I compare two colon reconstructions. One is fused using the raw camera trajectory in Figure 4.8, and the other is fused using the modified camera trajectory (with loop closure) in Figure 4.8. In the model of the raw trajectory, because of the heavy drift the first half of of the sequence does not overlap with the second half, leading to a "split" appearance in the final model. Comparatively, in the model of the trajectory with loop closure, because the trajectory does not have large drift the first and second half of the sequence overlap perfectly, leading to a consistent final model.

This demo verifies that this CNN image retriever is capable for detecting matching image pairs for a loop closure purpose. Given the fact that traditional loop detection systems cannot work on colonoscopic videos because of the challenges of colonoscopic images, this CNN image retriever is a hopeful alternative. The same work can be done on other difficult endoscopic images as well.

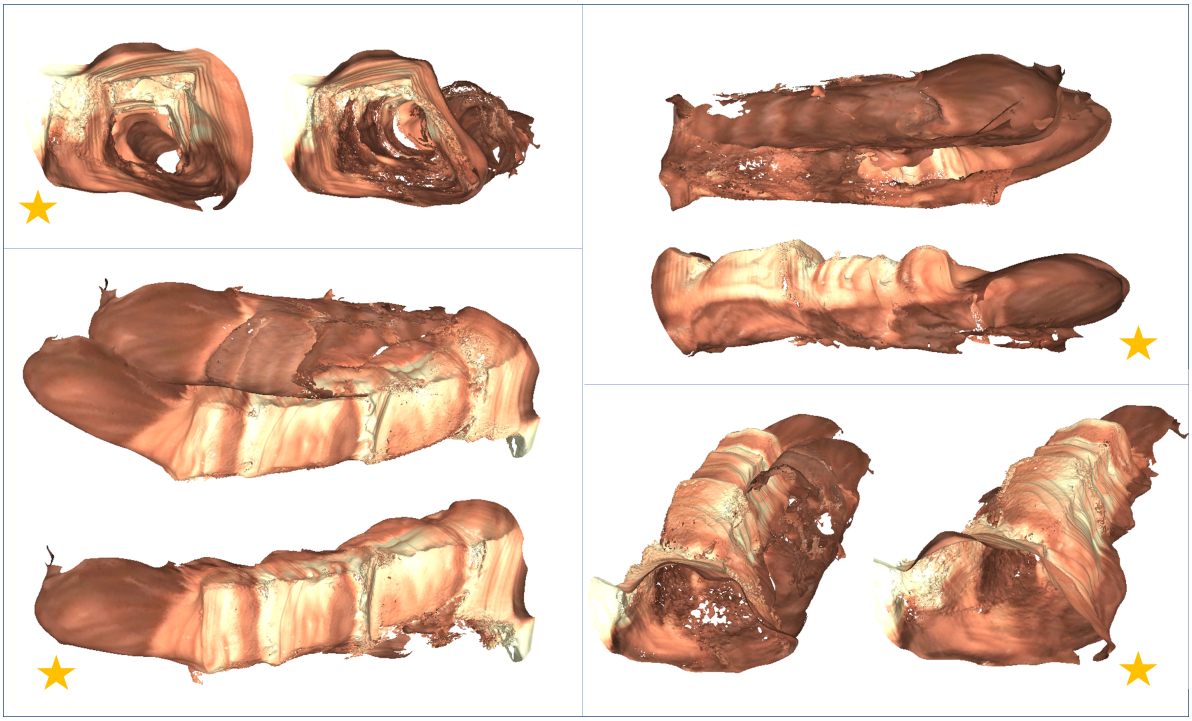


Figure 4.10: Comparison of reconstructions with and without loop closure from four different view-points. The yellow stars mark the one with loop closure.

4.4.3 Conclusion

In this chapter I have presented a colonoscopic place recognition method, some experimental results, and two application demos. While the method is somewhat robust against camera rotations and deformations of limited size, further work might allow even larger camera rotations and deformations.

This work should serve as a prerequisite for the desired functionalities of a complete unsurveyed region detection system. As future work, we should use this technique to improve the system's robustness against temporal gaps (continue reconstruction when a historical frame is retrieved after a gap). A loop closure module can be implemented in the reconstruction algorithm to take advantage of the local loops. Also, we need to implement a guidance-back module that can help endoscopists find the surrounding frames of an unsurveyed region.

CHAPTER 5

Generalized Cylinder Deformation under the Relative Cuvature Condition

This chapter introduces a generalized cylinder deformation algorithm [123, 124]. It was motivated by the need of straightening a colon surface and visualizing its interior. The software of this project is in https://github.com/RuibinMa/ColonVisualization.git.

5.1 Algorithm Pipeline

For an input triangle mesh of a generalized cylinder, my method outputs a mesh with a centerline following a target geometry. The target centerline geometry is pre-specified, e.g., a straight line or some other 3D parametric curve. Specifically, the method produces a vertex-wise mapping. The pipeline is illustrated in Figure 5.1. As a medium for this mapping, I use a discrete skeleton to perform this deformation. I sample discrete points on a centerline densely and uniformly and compute a cross section on each point. The whole process is divided into a skeleton extraction step and a deformation procedure.

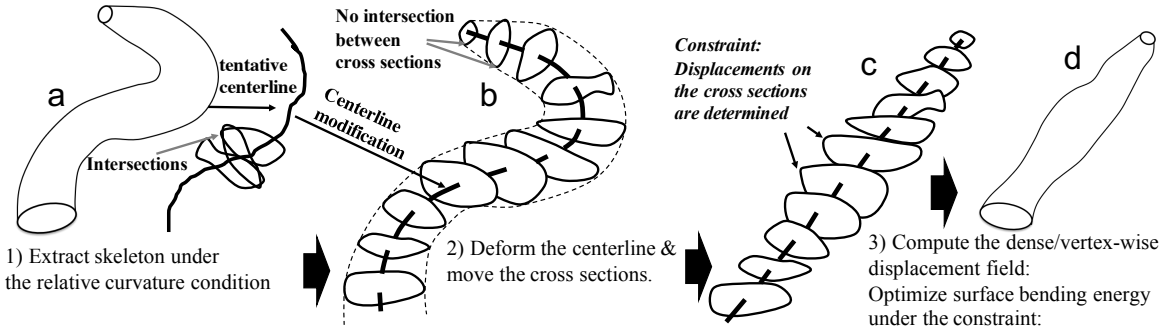


Figure 5.1: The pipeline of skeleton-based generalized cylinder deformation under the relative curvature condition. The input is a generalized cylinder and a target centerline shape. The output is a deformed version of that generalized cylinder.

In the skeleton extraction step I find a centerline for which the discrete cross sections do not intersect by incorporating the Relative Curvature Condition (RCC). In the first stage of the deformation procedure I reshape the centerline to the target parametric shape. Then the cross sections

are mapped accordingly using RMFs. Under the constraint that the displacements on the cross sections are determined and fixed, I solve for the displacements of all the other inter-cross-sectional regions by minimizing a Thin Shell bending energy.

5.2 Skeleton Extraction

5.2.1 Initial Centerline and Cross Sections

The method proposed by Antiga et al. [66] minimizes the following energy by finding a path between two user-given points $p0$ and $p1$.

$$E(\boldsymbol{c}(s)) = \int_{0=\boldsymbol{c}^{-1}(p0)}^{1=\boldsymbol{c}^{-1}(p1)} F(\boldsymbol{c}(s))ds \tag{5.1}$$

where F is a function that is lower for more internal positions (detailed in [66]). The functional is minimized on the Voronoi diagram of the generalized cylinder mesh [105, 66, 106].

I adopt this method to extract a tentative centerline $ctl_{initial}$ that is represented as a series of discrete points. Points on $ctl_{initial}$ are uniformly sampled. At each point, as defined in Section 2.9 I compute a cross section that is explicitly represented by a set of discrete points and edges. One important property we want the centerline to have is that it should imply no intersections between cross sections orthogonal to the centerline, which is violated because of sharp centerline curvatures, as shown in Figure 5.2 (left).

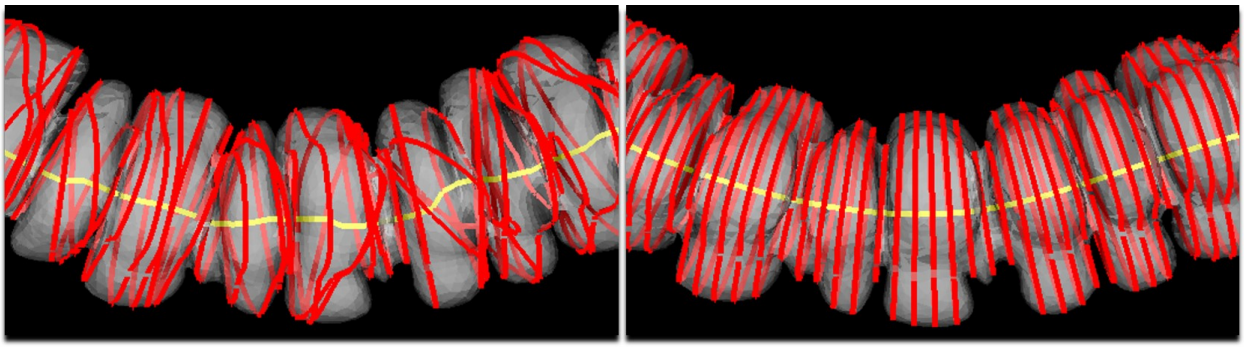


Figure 5.2: Yellow curve: centerline. Red curves: cross sections. The generalized cylinder surface is a human colon surface segmented from CT. Left: centerline computed by [105, 66, 106] which has large and even discontinuous curvatures; that leads to intersections. Right: centerline modified according to the RCC; there are no intersections between cross sections.

5.2.2 Centerline Modification Based on the Relative Curvature Condition

The cross sections of $ctl_{initial}$ cannot be used as the skeleton because cross-sectional intersections lead to surface folding, as in Figure 5.3. In [69] J. Damon presented the RCC for the non-singularity property of a generalized cylinder, *i.e.*, to detect cross-sectional intersections in the continuous situation. Accordingly, an algorithm based on the relative curvature condition and smoothing is performed on $ctl_{initial}$ to avoid intersection. The algorithm and its mathematical basis are as follows.

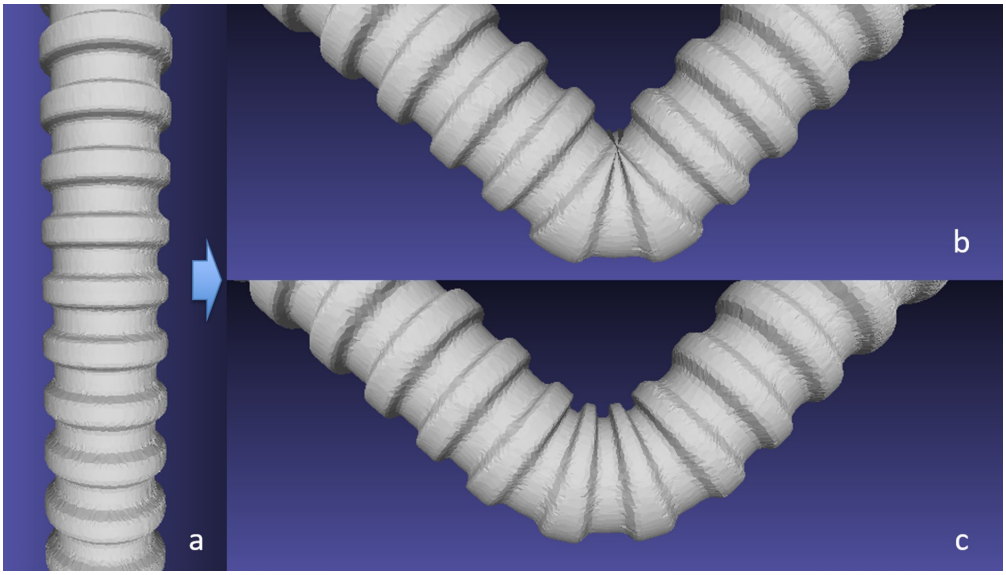


Figure 5.3: Self-intersections between cross sections will result in surface self-intersection. a) a straight tube to bend; b) tube bent with intersecting cross sections; c) tube bent without intersection between cross sections.

The RCC is illustrated in Figure 5.4. A cross section is locally orthogonal to the centerline. The (Frenet) normal direction is inside the orthogonal plane. Each point on the cross section can be represented by a vector from the centerline point to itself. I represent the angle between the vector and the normal direction by θ and the length of the vector by r. For a non-intersection cross section, all the boundary points must satisfy

$$r(\theta, s) < \frac{1}{\kappa(s) cos(\theta)} \quad \text{when } cos(\theta) > 0, \tag{5.2}$$

where $\kappa(s)$ is the local curvature of the centerline. The RCC only constrains the points on the side of the normal ($cos(\theta) > 0$). The term $\kappa(s)\ cos(\theta)$ is the so-called relative curvature. The condition suggests that regions with larger relative curvatures should have smaller radii, which is consistent

with intuition. The RCC detects intersections between cross sections in the continuous situation. If each point on all cross sections satisfy that condition, no cross section will intersect another cross section.

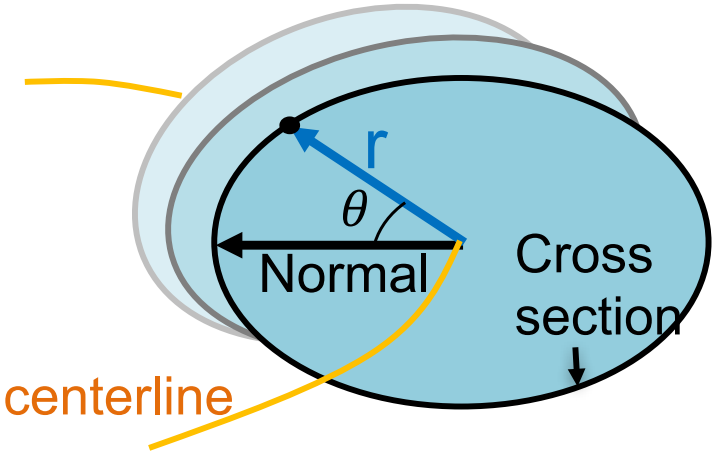


Figure 5.4: Relative Curvature Condition

In my discretized implementation, each cross section is represented by a number of points; I only deal with those points. This inequality condition must hold for all the points on the cross sections but is violated for regions of $ctl_{initial}$ with large centerline curvatures. I thus modify $ctl_{initial}$ by Algorithm 1 below. The cross sections are updated based on the modified centerline $ctl_{modified}$.

Algorithm 1 has two stages. I first perform iterative Gaussian smoothing and centerline adjustment according to local curvatures, trying to satisfy the RCC for each cross section. Secondly, for cross sections with large radii that are difficult to satisfy the RCC, I perform an interpolation between neighboring satisfactory cross-sectional planes to compute the cross section, as shown in Figure 5.5. The interpolation ensures uniform change of the orientations of the interpolated planes. It is done in a Lorentzian metric space [125]. Orthogonality can be slightly relaxed for the interpolated planes, and this non-orthogonality will be preserved during the mapping procedure in Section 5.3.1. Details can be found in the pseudo-code. The resulting skeleton can be regarded as a sweeping procedure along the centerline. Although the orthogonality is relaxed in some parts of the centerline, the normal and interpolated planes can be together regarded as a continuous Lorentzian parallel vector field.

Algorithm 1 Centerline Modification Algorithm

$it = 0, satisfactory = False, maxIter = 20$
$ctl = ctl_{initial}$, N = number of points on ctl
$crosssection$ is an N-tuple of the N cross sections.
T is an N-tuple of N tangent vectors of ctl
$crosssection_{final}$ is an N-tuple of the N final resulting cross sections.
(Smoothing)
while $it < maxIter$ and not $satisfactory$ **do**
  1-D Gaussian smoothing of the three coordinates of ctl ($\boldsymbol{x}_{n\times 1}, \boldsymbol{y}_{n\times 1}, \boldsymbol{z}_{n\times 1}$) separately.
  δctl = N-tuple of zero vectors, $satisfactory = True$
  for $i = 1 : N$ **do**
    Compute local ctl curvature κ_i and the (Frenet) Normal direction $\boldsymbol{n}_i$.
    Update $T[i]$ and $crosssection[i]$ using ctl
    if any point on $crosssection[i]$ violates RCC **then**
      $\delta ctl[i] = -\kappa_i \boldsymbol{n}_i$, $satisfactory = False$
    end if
  end for
  1-D Gaussian smoothing on the 3 coordinates of δctl seperately
  $ctl = ctl + \delta ctl$, $it = it + 1$
end while
(Interpolation)
for $i = 1 : N$ **do**
  if any point on $crosssection[i]$ violates RCC **then**
    if i==1 **then**
      $newT_i = T[2]$
    else if i==N **then**
      $newT_i = T[N-1]$
    else
      Find the two bounding cross sections that are satisfactory ($crosssection[l]$ and $crosssection[r]$).
      $t = (i-l)/(r-l), \alpha = arccos(T[l] \cdot T[r])$
      $\lambda(t,\alpha) = sin(t\alpha)/sin(\alpha)$ when $\alpha \neq 0$, otherwise $\lambda(t,\alpha) = t$
      $newT_i = \lambda(1-t,\alpha)T[l] + \lambda(t,\alpha)T[r]$
    end if
    $crosssection_{final}[i]$ = intersecting the mesh with the plane defined by $ctl[i]$ and $newT_i$, and taking the connected component closest to $ctl[i]$.
  else
    $crosssection_{final}[i] = crosssection[i]$
  end if
end for
output $crosssection_{final}$, $ctl_{modified} = ctl$

The resulting skeleton and its associated cross sections have no intersections. $ctl_{modified}$ is then to be deformed. The right frame of Figure 5.2 shows an example of $ctl_{modified}$ and the non-intersecting cross sections from real data (the human colon).

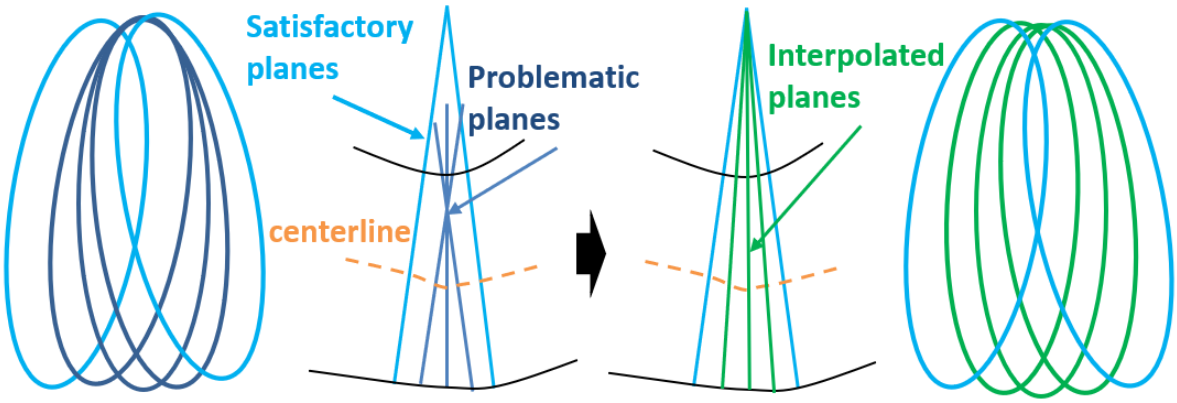


Figure 5.5: Replacing intersecting cross sections with interpolated ones

5.3 Deformation

The deformation procedure consists of a skeleton (geometrical) deformation stage (subsection 5.3.1) and fine-scale Thin Shell deformations of subdivided regions (subsection 5.3.2). The skeleton deformation stage deforms the centerline into a desired parametric curve. Relative rotation between cross sections is controllable; if no artificial twisting is desired, I minimize relative rotation between cross sections using the rotation minimizing frames (RMF) [70]. The mesh deformation stage takes the deformation of the skeleton as input and computes the vertex-wise displacement vectors by minimizing a bending energy.

5.3.1 Skeleton Deformation

I design how the surface deforms by specifying how the centerline and the position of each cross section are changing. Because we only focus on the shape change of the centerline, the shape of the cross-sectional curves remains unchanged during the process. They will move along with the centerline.

Method Detail

1. $ctl_{modified}$ is considered as a discrete curve with each point assigned a location parameter proportional to arc length. The target centerline is a parametric space curve sampled uniformly to the same number of points as $ctl_{modified}$.

2. At each point on $ctl_{modified}$, compute an RMF. The tangent direction of an RMF is constrained to be the tangent direction of the centerline. The other two directions are orthogonal. Rotations

between neighboring frames are minimized according to [70].

3. The target centerline is defined as a parametric curve and is sampled uniformly to the same number of points as $ctl_{modified}$. The total length of the curve (point spacing) can change for stretching purpose.

4. At each point on the target centerline, construct a new frame (denoted by NEWF) whose tangent direction is also the centerline tangent. The other two orthogonal directions are set according to user-defined consistency. For example, in a straightening task, no twisting (Figure 5.6.b) is introduced, resulting in all NEWFs in the same direction. Otherwise, the twisting amounts can be parameterized (Fig 5.6.c).

5. Map (rigid transform) each cross section according to its RMF and NEWF. (Slight non-orthogonality in Algorithm 1 is preserved.)

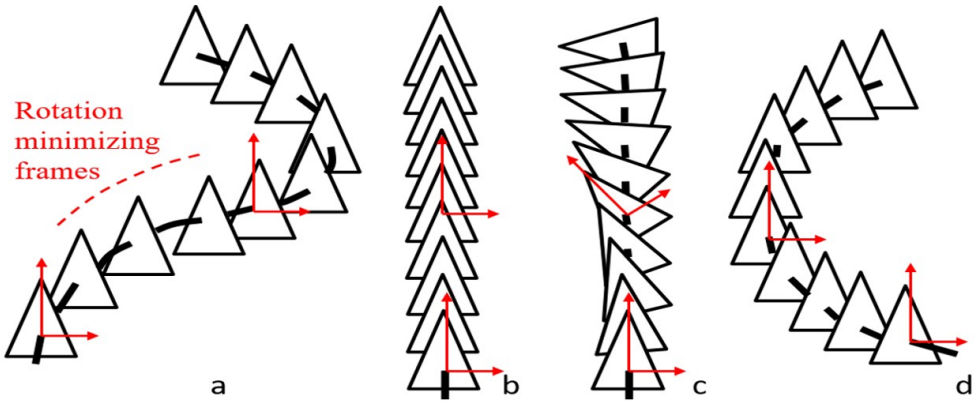


Figure 5.6: Three examples showing the process of deforming the centerline and moving the cross sections. a) A centerline & cross sections computed under the relative curvature condition. There is no intersection between cross sectional curves inside the generalized cylinder. At each cross section, two directions are calculated using the rotation minimizing frames. b) Target centerline shape: straight. No twisting. c) Target centerline shape: straight. Artificial twisting introduced. d) Target centerline shape: half circle.

A parametric curve is powerful in forming shapes that can not be easily achieved by direct manipulation, such as a helix with chosen radius and vertical spacing. This allows continuous shape variations of the generalized cylinder with shape parameters gradually changing or moving along a path in a shape space. We can perform various types of manipulations on the centerline as long as the cross sections preserve their relative *local* positions to the centerline and the relative curvature

condition is not violated. For example, the three schemes in Figure 5.6 respectively describe the manipulations of straightening, twisting, and bending.

Why not use the Frenet frame? A Frenet frame is composed of a local tangent direction of the curve, a normal direction and a binormal direction. The normal direction is the direction of the derivative of the tangent, and the binormal direction is the direction of the derivative of the normal. Although the normal directions are used in the RCC, I do not use the Frenet frames in skeleton deformation because the normal directions are not continuous. Because of the twisting caused by the torsion of the centerline [48], it is hard to maintain or adjust relative position (rotation) between neighboring cross sections during the deformation.

Comparatively, although RMFs have the same tangent direction as the Frenet frame, the two other orthogonal directions are computed consecutively from end to end so that the rotation between any two neighboring frames is minimal. Orientation consistency between NEWFs becomes intuitive for users if the source frames already have minimal rotation between each other. For example, in Figure 5.6(b), to straighten a generalized cylinder without twisting, we can simply set each pair of adjacent frames to be identical so that they also have minimum rotation between each other. It is hard to figure out what the orientations in Figure 5.6(b) should be if the orientations in Figure 5.6(a) are not rotation minimizing frames.

Our skeleton is compatible with the traditional free-form deformation. Using control points, the centerline can be easily modeled as a spline and fit into our current skeleton deformation scheme. If the centerline is bent too much and the relative curvature condition is violated, self-intersection can happen. My method cannot handle that case but can identify where the problematic bending occurs. One example is shown in Section 5.4.5.

5.3.2 Mesh Deformation

The skeletal deformation implies the underlying surface deformation. Our final objective is to compute the deformation of the whole surface; that is, the result should be in the form of displacement vectors of all the vertices on the surface. The deformation of the centerline precisely gives the displacements on the cross-sectional curves. Because these curves are calculated by intersecting the surface and planes and do not necessarily pass through any vertex on the surface, I instead first determine the displacements of the vertices that are close to any cross section (I

use a distance threshold $\epsilon = 0.1 \times$ the centerline point spacing) and that also satisfy the relative curvature condition. These vertices are then mapped by the method in the "The Constraint: Fixed Displacements" paragraph below.

After fixing the displacements of those vertices close to cross sections, the remaining problem becomes the following: given the constraint of the fixed displacements of a subgroup of vertices V^{fixed}, solve for the displacements on all the other vertices ($V^{unknown}$) on the surface such that local curvature patterns are preserved. I solve this problem by minimizing a bending energy defined on the triangle mesh under the constraint of holding the displacements of V^{fixed} unchanged.

The Constraint: Fixed Displacements The subgroup V^{fixed} of vertices are selected as those whose distance to any cross-sectional curve is smaller than a threshold and also satisfy the RCC. For each of these vertices, we can find its positional parameter (s) along the centerline. This is done by finding the vertex's closest point on the interpolated polynomial curve of the centerline between the two bounding cross sections. With parameter s, we can compute the local RMF(s) and the target new frame NEWF(s); the transform from the RMF(s) to NEWF(s) maps this vertex .

Reason: I do this only for the vertices V^{fixed} for two reasons: 1) Not all the vertices satisfy the relative curvature condition in practice, especially those outside the ϵ-neighborhood of the cross sections, but we still desire to solve for their displacements under the constraint of the displacements of the others to achieve optimum in sense of an bending energy (detailed in the next part). 2) The two bounding cross sections for the other vertices can be hard to determine. Without this information, the s parameter can be ambiguous because a vertex's closest point on the whole centerline may not be the correct corresponding centerline point. This is especially common for surfaces, such as the colon, that have a lot of local geometric details or so-called topological noise [126, 40, 127], like small handles.

Minimizing Surface Bending Energy The displacements of the other vertices $V^{unknown}$ are calculated by minimizing the Thin Shell bending energy [72, 128, 71, 129], which can be approximated as the squared Laplacian [72, 71] of the displacement function defined on the mesh ($(\Delta f)^2$). It essentially characterizes the change of mean curvatures in a local surface patch, which serves as a proxy for measuring local elastic bending of the structure (thin shell). As such, it requires the

definition of a discrete Laplace-Beltrami operator on a triangle mesh [130, 131]:

$$\Delta f(v_i) := \frac{1}{2A_i} \sum\nolimits_{v_j \in N(v_i)} (cot(\alpha_{i,j}) + cot(\beta_{i,j}))(f_j - f_i) \tag{5.3}$$

In this equation f is any function defined on the mesh, v stands for the vertices on the mesh, and α and β stand for the two opposing angles in the two triangles that share the (i, j) edge. A_i represents the area of the mixed Voronoi cell [130] for vertex v_i. $N(v_i)$ is the set of neighbors of vertex v_i.

In practice, because my centerline is densely sampled (up to 400 for long tubes like colons), the mesh is divided by V^{fixed} into many small regions. For each of these regions, the boundary is composed of some vertices in V^{fixed} whose displacements are fixed. Those boundary displacements are the constraint to solve for the unknown displacements inside this region. I initialize the displacements of $V^{unknown}$ with a rigid transformation, a global transformation that does not produce local elastic bending energy and thus not affect the final optimization. Such a rigid transformation is initialized to best fit the fixed displacements of V^{fixed} in terms of sum of squared error. The process is detailed in the following paragraph.

In the following, I denote vertices/regions of original positions using the subscript o, denote those rigidly transformed by the subscript r, and denote the target by t. Formally, let R_o be a region composed of V_o^{fixed} and $V_o^{unknown}$ on the original mesh. V_t^{fixed} are the target positions of V_o^{fixed}. T is a rigid transformation defined by V_o^{fixed} and V_t^{fixed} s.t. the sum of squared distances between $T(V_o^{fixed})$ and V_t^{fixed} is minimum. The vertices after the transformation are denoted by V_r^{fixed} and $V_r^{unknown}$, respectively:

$$\begin{aligned} V_r^{fixed} &= T(V_o^{fixed}) \\ V_r^{unknown} &= T(V_o^{unknown}) \end{aligned} \tag{5.4}$$

Denote the transformed region by $R_r = V_r^{fixed} + V_r^{unknown}$. Our target is to compute R_t, part of which is known as V_t^{fixed} and the other part of which is unknown ($V_t^{unknown}$). Let f be the displacement function from R_o to R_t, and let g be the displacement function from R_r to R_t. Again, part of g is known as $V_t^{fixed} - V_r^{fixed}$. Assuming $V_r^{unknown} = \{v_i^{unknown} | i = 1, ..., n\}$ and

$V_r^{fixed} = \{v_i^{fixed} | i = 1, ..., m\}$, the Thin Shell bending energy of g will be

$$E = \sum_{j \in \{x,y,z\}} (\sum_{i=1}^{n} (\Delta g^j(v_i^{unknown}))^2 + \sum_{i=1}^{m} (\Delta g^j(v_i^{fixed}))^2) \tag{5.5}$$

The domain of the second summand includes the fixed vertices because the Laplace-Beltrami operator may involve their neighboring vertices whose displacements are unknowns. This energy can be cast into a quadratic form and solved efficiently.

Finally, with g, we can solve for f of $V_o^{unknown}$:

$$f(v) = g(T(v)) \tag{5.6}$$

I do this for each region separately. As a result, we get a displacement vector for each vertex. Theoretically, this linear solution does not guarantee a non-intersection solution. However, because my method decomposes the total large deformation into many small inter-cross-sectional deformations using relatively dense cross sections and the geometry of each piece of sub-surface is simple enough, I have not encountered intersection in practice.

5.4 Results

The computation time varies from 15s to 60s on an Intel i7 single thread CPU. The most time-consuming step is to calculate cross sections, which is highly parallelizable.

In Section 5.4.1 I show examples of the deformation of five meshes as a proof-of-concept. In Section 5.4.2 I show an example to illustrate the capability of dealing with high curvature regions and local geometrical details. In Section 5.4.3 I show examples of shape variations controlled by skeletons whose parameters are continually changing. In Section 5.4.4 I show an application of my method to human colon visualization.

5.4.1 Deformation Examples

As a proof-of-concept (Figure 5.7), I tried my method on five generalized cylindrical objects: a flexible tube, a pencil, a human colon, a snake and a cane. For each of these objects, I show two deformation results with different target centerline shapes that are given as parametric space curves. My geometrical (skeleton) deformation efficiently deformed the objects to the target centerline shape at large scale, giving precise control of the centerline curvature. Cross-sectional intersection was

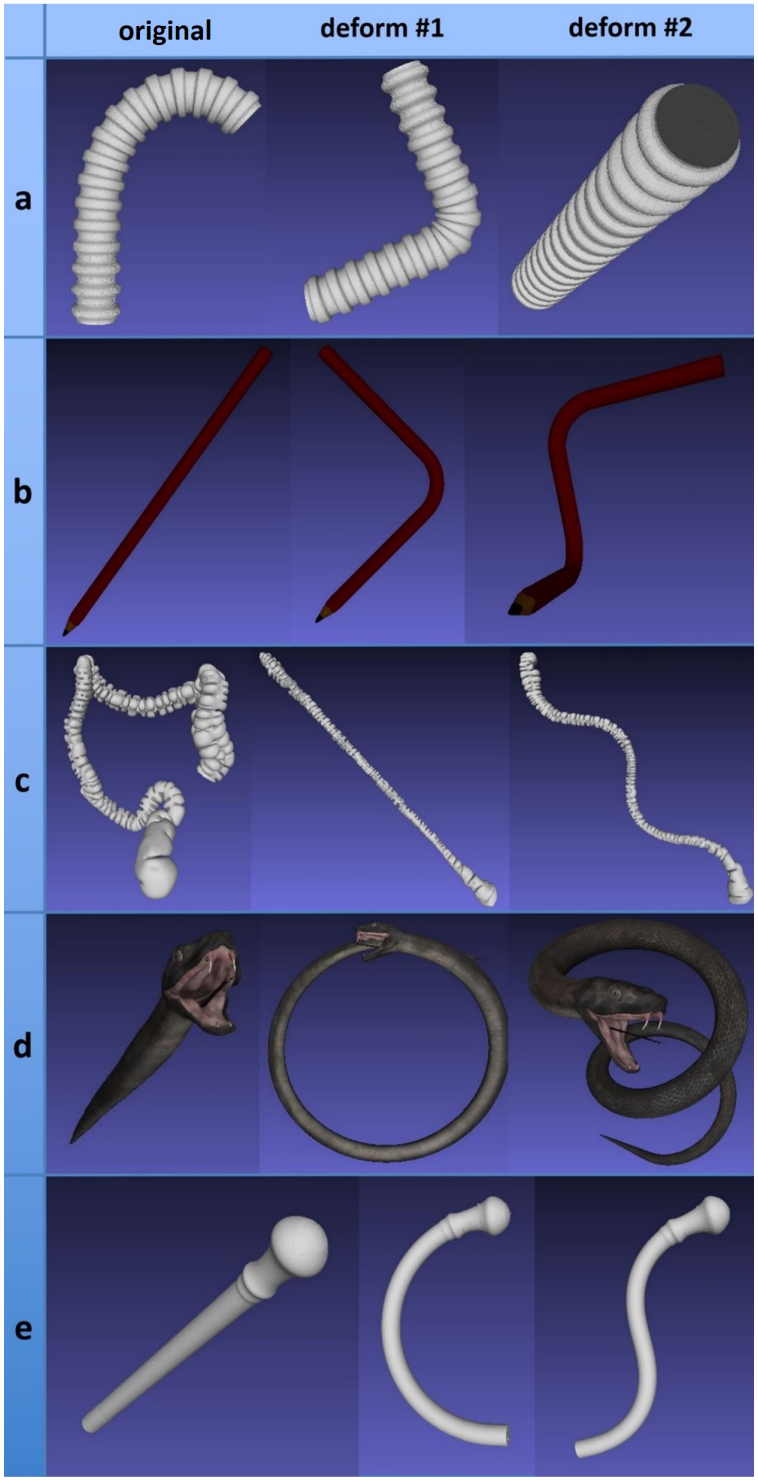


Figure 5.7: Five examples of generalized cylinder deformations. The target centerline shapes are parametric curves (in the middle and the right columns). a) flexible tube; b) pencil; c) human colon; d) snake; e) cane.

prevented for the example cases. Additionally, thanks to the use of Thin Shell model, my method does not introduce unnecessary distortion; that is, local curvature patterns are preserved.

5.4.2 High-curvature Regions

Figures 5.8 and 5.9 show two parts of a colon of high curvature. While straightening the colon, allowing for the visualization of its interior at-a-glance (discussed in detail in Section 5.4.4), my method prevents skeleton-level intersection and keeps the local curvature patterns. Moreover, for the so-called topological noise [126, 40] or small handles, my method does not need special processing.

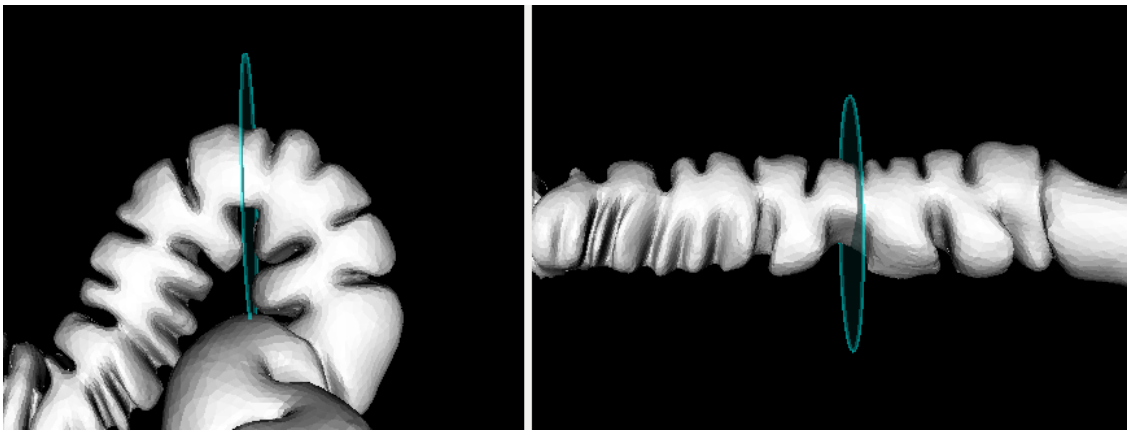


Figure 5.8: Deformation (left into right) of a human colon. The blue circles show corresponding positions of a high curvature region. Cross-sectional intersections are prevented, and local curvature patterns are preserved.

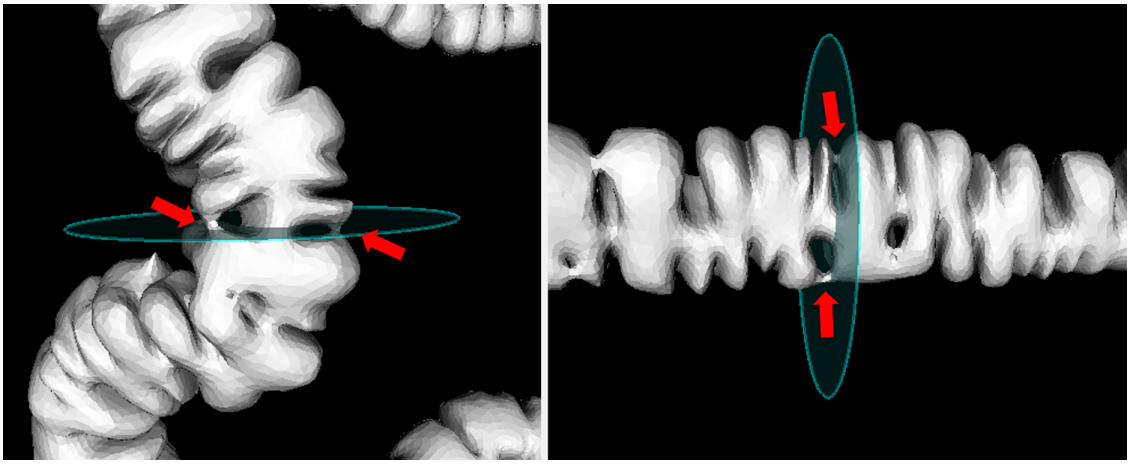


Figure 5.9: The red arrows point to two small handles. My method is robust to topological noise. In this situation there may be more than one connected component in the cross-sectional plane locally. In practice I keep the one closest to the centerline as the cross section.

5.4.3 Parametric Mesh Morphing

A parametric curve is an efficient tool to guide the deformation of a generalized cylinder because the target skeleton is inherently consistent with the skeleton of a generalized cylinder. Moreover, defining the target mathematically enables us to control the centerline curvature precisely, which can be hard for direct manipulations. To address this point, I show some examples whose shapes are morphing according to the underlying skeleton parameters. Figure 5.10 shows the shape variations of the snake the centerlines of which are bent into helices whose parameter is continually changing. Figure 5.11 shows the shape variations of the colon mesh whose centerlines are parameterized as sine curves of different frequencies. In Section 5.3.1 I mentioned the capability of controlling the twisting. This is shown in Figure 5.12. The orientations of the new frames on the target curve are parameterized. In these examples the target curve is of the same length as the $ctl_{modified}$. However, I can change the spacing of the points on the target curve. This is shown in Figure 5.13 where I stretch a flexible tube to different lengths.

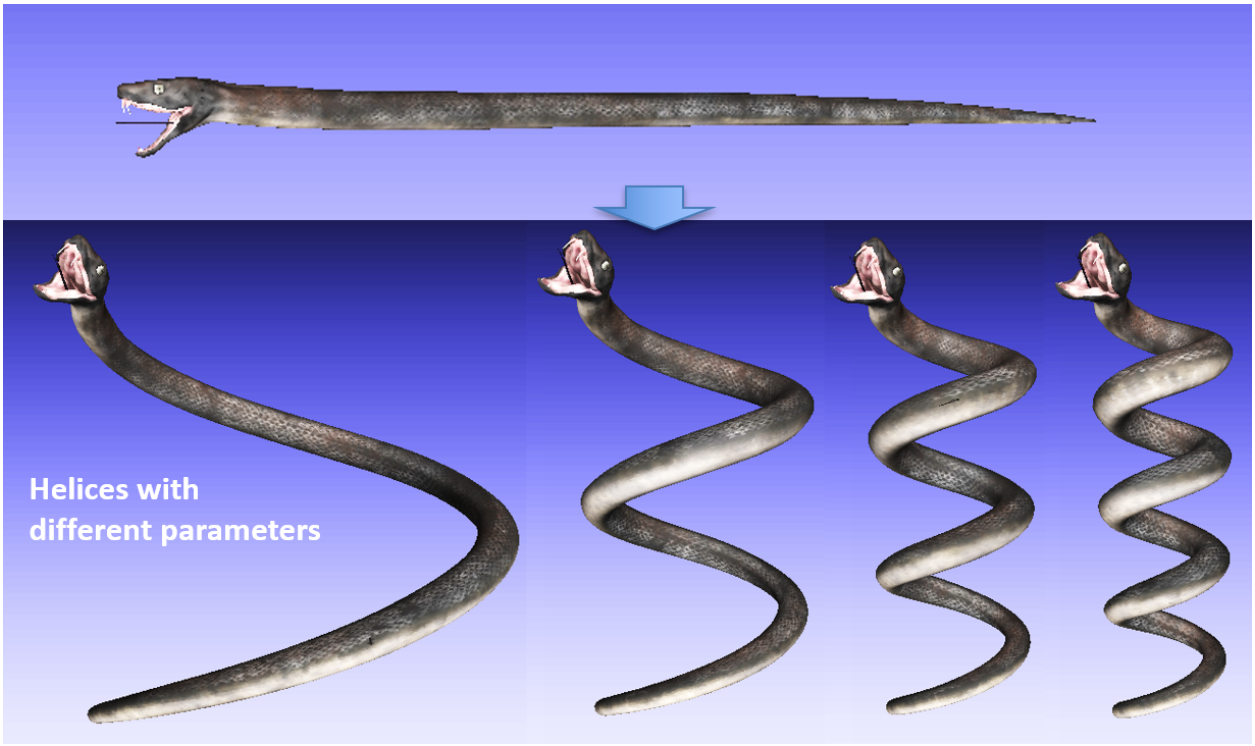


Figure 5.10: A snake is deformed into helices with different parameters. The densely-sampled centerline lies precisely on a mathematical helix. It is easy to impose precise curvature control using our method.

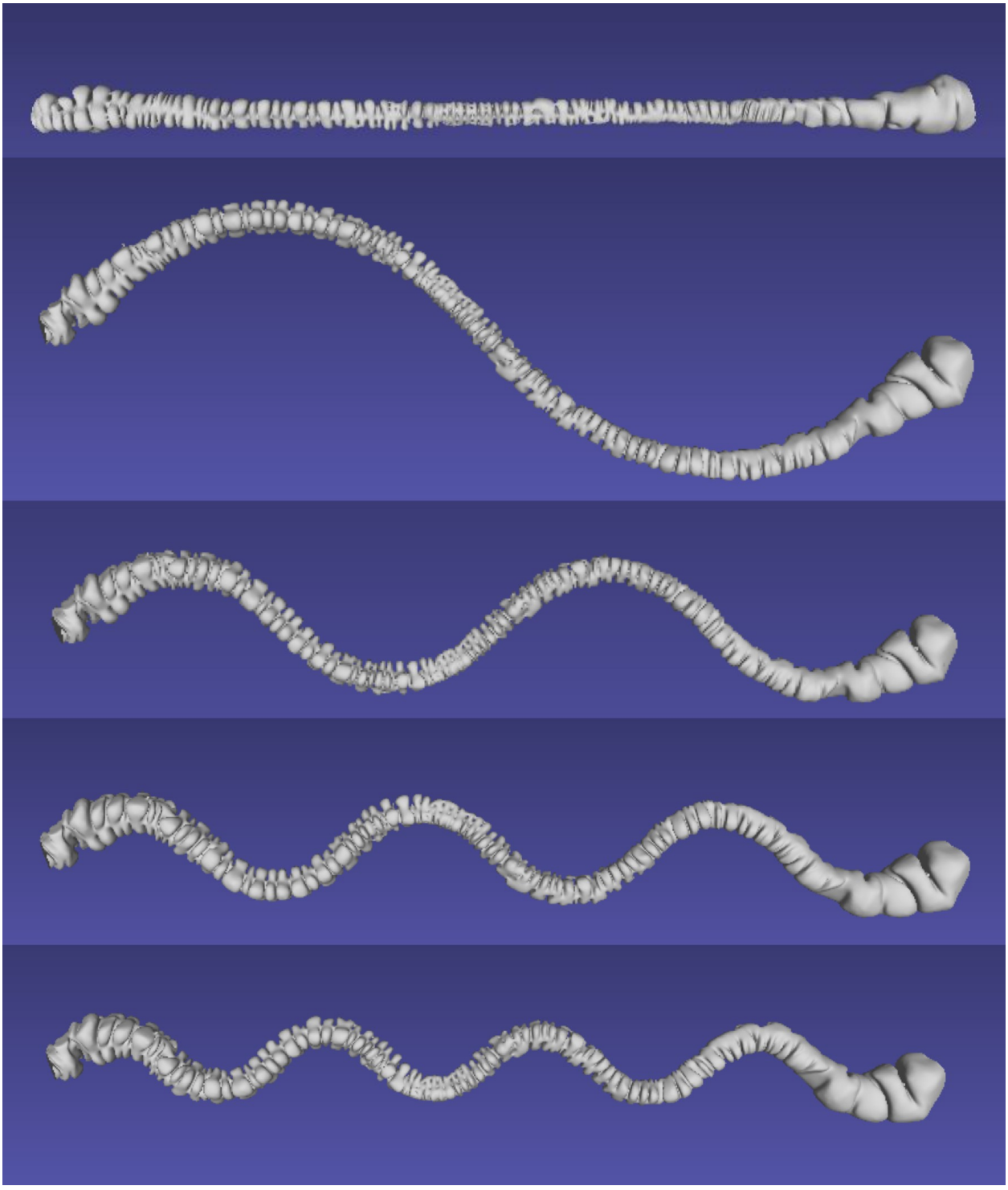


Figure 5.11: The centerline of a colon mesh is mimicking sine curves of different frequencies.

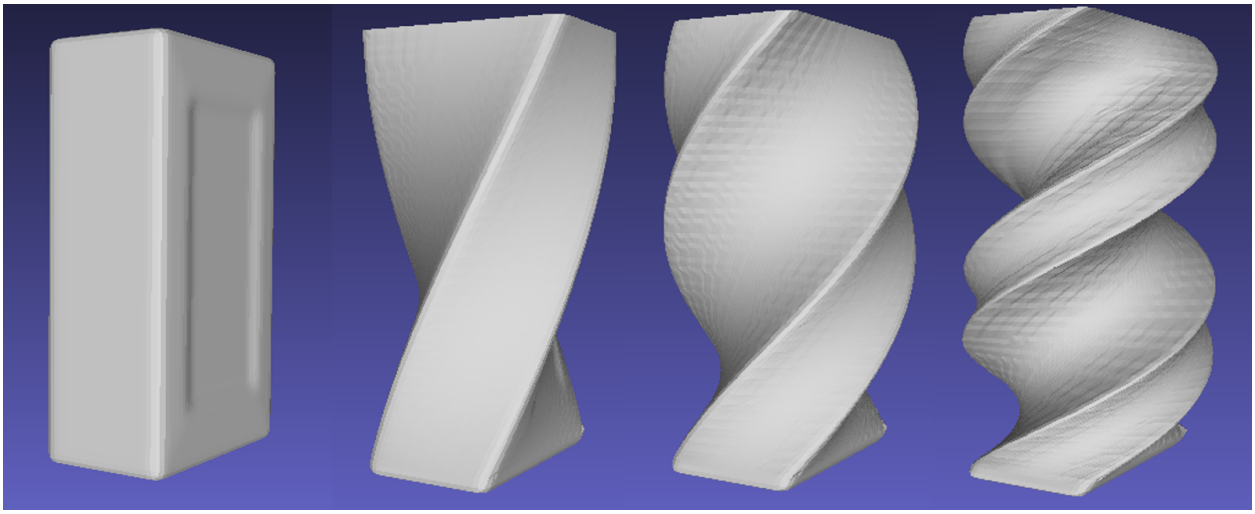


Figure 5.12: a) A cuboid mesh. b) c) d) Artificial twisting is applied to the cuboid. The resulting mesh is varying as the twisting degree is increasing. Although the points on the centerline stay static in this example, the relative rotations of the cross sections are changing.

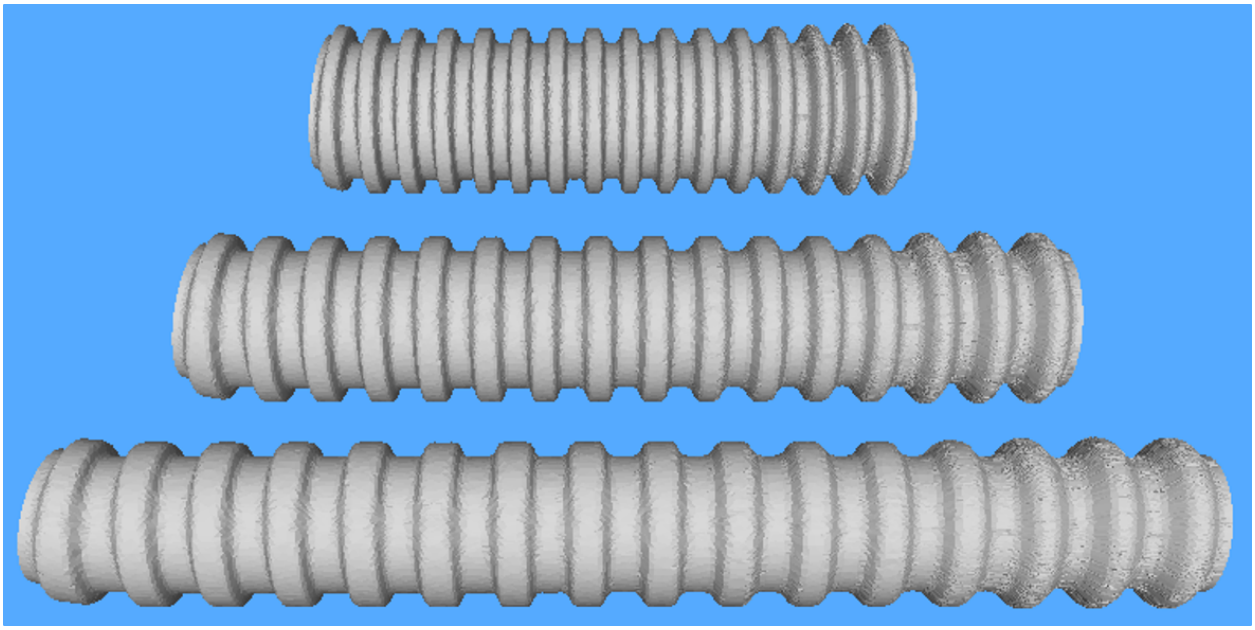


Figure 5.13: A flexible tube stretch to different lengths. This is done by changing the point spacing on the target parametric curve.

5.4.4 Colon Visualization

My method is applicable to colon visualization. Colon visualization has been an important research area in medical image analysis. Due to the highly-curved characteristic of the colon surface, it is very difficult for physicians to examine the complete surface of a colon during colonoscopy or the virtual colonoscopy based on modalities like CT. Therefore, unfolding approaches were proposed

to help physicians view a colon at-a-glance. One popular category of unfolding methods is conformal mapping [126, 67, 132, 40, 133, 134, 135, 136]. Conformal mapping maps a surface of tubular topology to a plane. In those works the resulting 2D surface is colored by curvature information or volume rendering intensities. However, traditional conformal mappings eliminate 3D geometry and only keep that geometric information as intensities. As addressed in [133, 134], this can hide important diagnostic information. There have been efforts to improve conformal mapping by preserving more geometric context. Based on conformal mapping, Nadeem *et al.* [133], proposed a shape-preserving mapping that enlarges the examined surface area of virtual colonoscopy; that is, they proposed conformal deformation of local regions of interest. Their resulting mesh is still tubular in 3D. It is applicable to any arbitrary genus surface and topology. Wang *et al.* [134] also proposed the so-called 2.5D mapping by assigning heights to planar pixels by computing distances to local fitted cylinders. Marino *et al.* [135, 136] proposed a context-preserving mapping that maps a tubular surface to a plane according to its projected 2D skeleton. This approach keeps very high-level geometry context of a tubular/tree-stuctured object like the skeleton shape in order to stretch the generalized cylinder.

As an alternative to the conformal mapping approaches, I apply my method on colon visualization. A colon surface is first straightened while keeping the tubular structure and local curvature patterns. Straightening is one of the simplest applications of my method. The straightened colon is then slit open longitudinally. I then simply map the quasi-tube from a cylinder to a semi-cylinder. As such, the interior of a colon surface is displayed succinctly. This process is shown in Figure 5.14(a,b,c). A rectangle produced by conformal mapping [132] is shown in Figure 5.14(d).

Compared with conformal mappings, my method has the following three advantages:

1. The area distortion is considerably less in my method. My method uses geometrical deformation as the constraint, keeping the shape of the cross sections and also the length of the centerline. Local curvature patterns are preserved by minimizing the Thin Shell bending energy. This guarantees that no unnecessary distortion is introduced in addition to the target deformation.

2. My method keeps geometry context in 3D rather than flattening the surface. The surface can be colored with additional texture such as the texture extracted from colonoscopic videos [72] (illustrated in Figure 5.15 with synthesized texture). This idea is similar to the approaches in [133, 134], but my method has a more explicit geometric constraint.

3. My method does not need special processing of topological noise [135, 136]. I illustrate these advantages in Figure 5.14, 5.15, and 5.16.

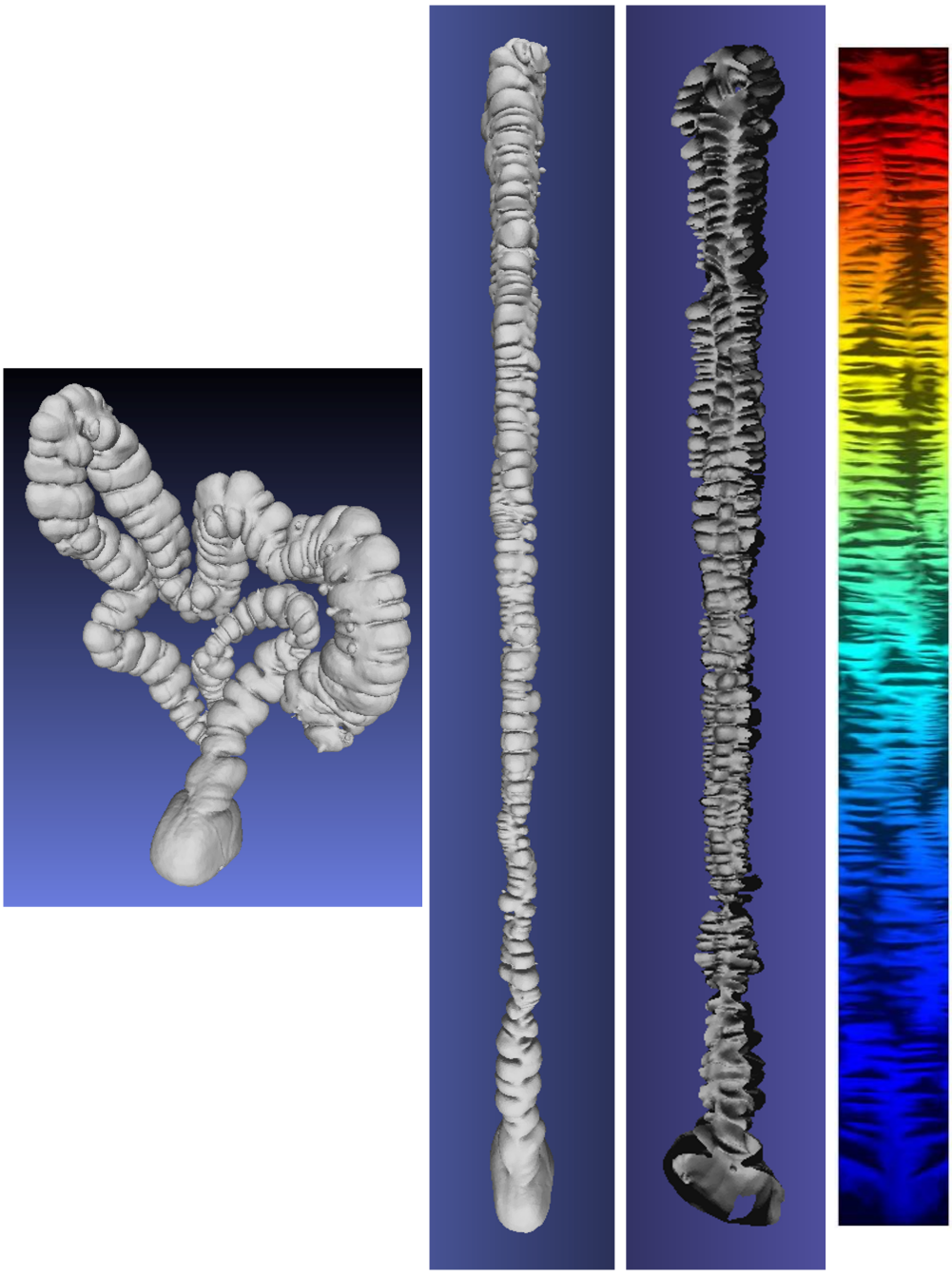


Figure 5.14: Colon visualization results. From left to right: 1) the original colon mesh; 2) straightened colon by my method; 3) slit-open colon; 4) conformal flattening colored with Fiedler information [132]

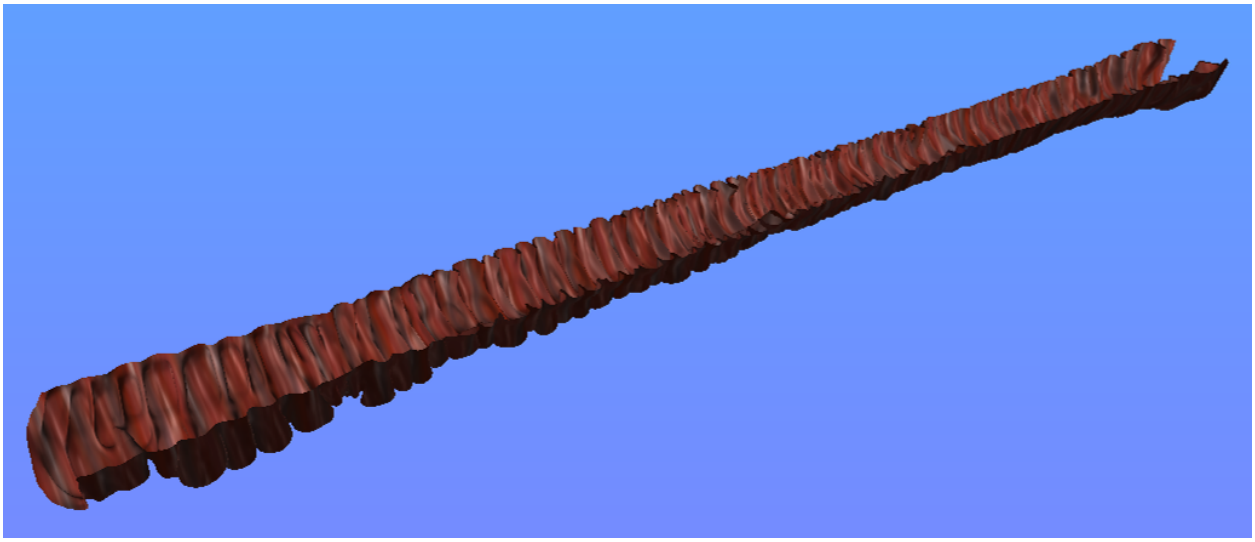


Figure 5.15: Visualization of the interior of a straightened colon. Geometry and (synthesized) texture are displayed together.

Figure 5.16 compares my semi-flattening and conformal flattening of a colon section. The four pictures on the right are at four slightly different views (rolled back and forth, left and right). Keeping local geometry makes the visualization more intuitive because it is in 3D.

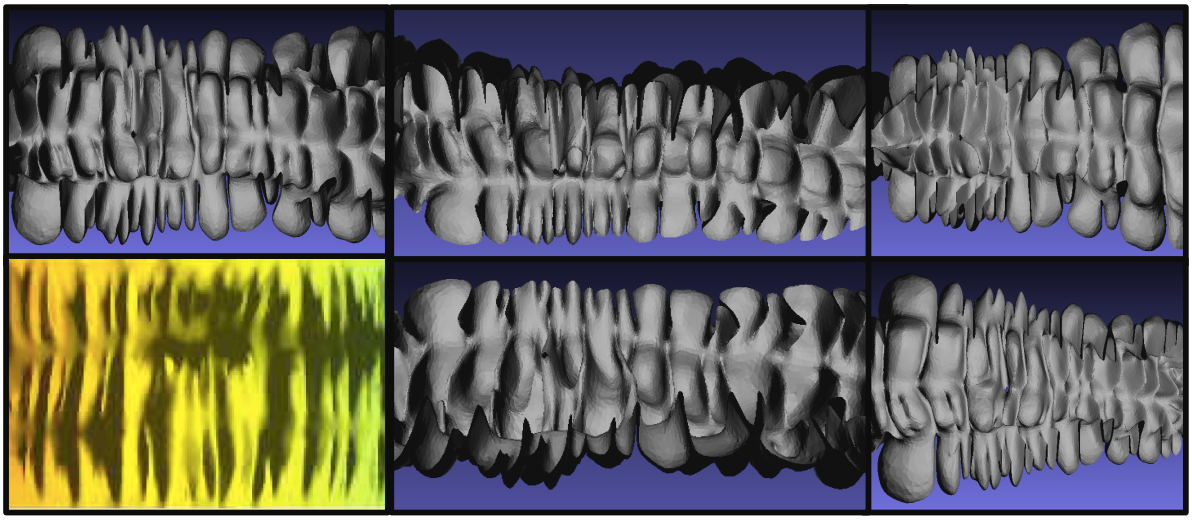


Figure 5.16: Top left: a semi-flattened colon section by proposed method, also with top view (top middle), bottom view (bottom middle), right view (top right) and left view (bottom right.). Bottom left: the same section by conformal flattening [132].

On the other hand, the advantage of conformal mapping approaches lies in their robustness. Because there is no skeletonization step, they do not rely on the centerline quality and are more

robust to various mesh resolutions.

The current slit-open operation in my current approach is a simple cylinder to semi-cylinder mapping. However, a straight longitudinal slitting line may not be the best choice because it can break the geometry of haustral folds (especially through the highly curved regions, aka flexures). A better strategy to find the slitting line was proposed in [132], which can be incorporated in the future. Another slitting line can be found according to the taenia coli (introduced in Section 2.1), which matches colon anatomy better.

5.4.5 Failure Case

My method fails when the target centerline is bent too much (relative to the tube radius) such that the relative curvature condition is violated in the deformed skeleton even if the condition is not violated in the initial skeleton. In such a case, surface folding will happen. In the example (Figure 5.17), a mesh of human stomach is bent sharply at a region with large radius. The three zoomed-in figures shows how the cross sections are intersecting. Techniques dealing with mesh contact [137, 64] could be incorporated to solve this large bending problem. My method provides a fast detection of possible intersection regions.

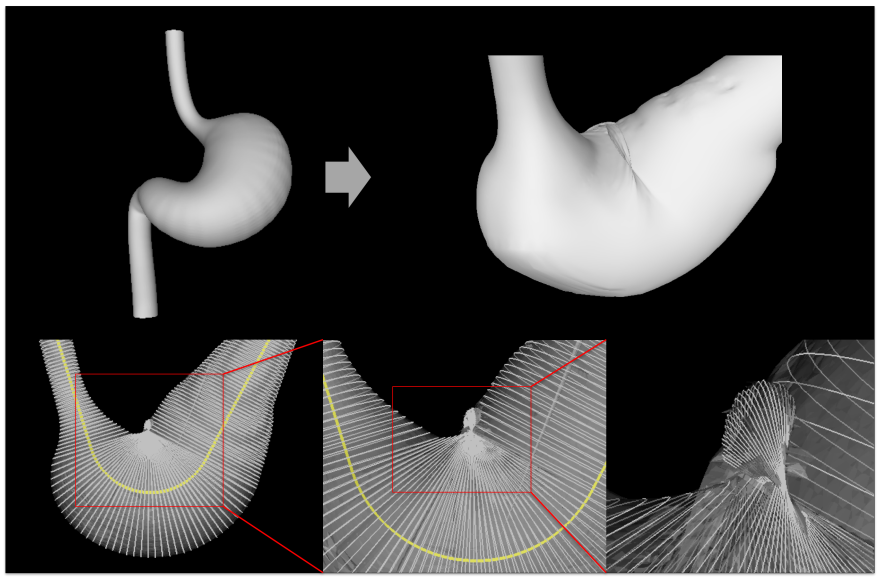


Figure 5.17: A mesh of human stomach is bent by 135 degrees, leading to surface folding. The bottom figures are zoomed-in pictures of the problematic region after deformation. The cross sections are in white, and the centerlines are in yellow.

CHAPTER 6

Conclusion and Discussion

This chapter reviews the contributions of this dissertation (in Section 6.1) and discusses some important problems of each chapter and associated future work (in Section 6.2).

6.1 Summary of Contributions

In this dissertation I have introduced the following contributions:

1. *A SLAM system (RNNSLAM) that works for colonoscopic images has been designed.* This SLAM system combines a traditional SLAM system (DSO) and an end-to-end depth and pose estimation neural network (RNN-DP). I integrated RNN-DP into DSO by replacing DSO's keyframe depth map by RNN-DP's depth prediction and initializing the camera pose in tracking by RNN-DP's pose prediction. This integration turned out to be a win-win strategy. It is a combination of a neural network's robust prior knowledge and a SLAM system's optimization mechanism (local bundle adjustment). Compared to DSO, the resulting camera trajectory has less drift and the scene scale is much more consistent thanks to the scale-consistent depth maps from RNN-DP. Compared to RNN-DP, RNNSLAM has much less camera pose drift because the optimization helps to reduce accumulated error. The idea of combining deep learning with optimization is a promising direction for more diverse applications.

2. *A pipeline for generating a dense 3D colon model for a short period of colon video to detect unsurveyed regions has been designed.* I used a RGB-D fusion algorithm to merge the RNNSLAM's output into a unified 3D mesh. A windowed depth map smoothing algorithm was designed to ensure the consistency needed for the fusion. I showed 3D reconstruction results of colonoscopies to demonstrate its capability of handling difficulties in colonoscopic images. The 3D models have high quality and capture the local details like haustra well. Most importantly, they explictly visualized the unsurveyed regions in the video, which are very hard to notice from first person perspective on the video.

3. *A benchmark for colonoscopic image retrieval / place recognition has been contributed.* This includes a testing dataset with manually labeled groundtruth and includes working deep neural networks (CNNs) trained novelly using the SfM results of a large collection of unlabeled colonoscopic videos. High performance has been achieved under multiple metrics. This work establishes a baseline for future research on image retrieval in colonoscopy and proves the feasibility of finetuning the CNNs with SfM results on colon images. It is the fundamental tool for many tasks related to colonoscopy visualization such as navigation back to the unsurveyed regions, handling temporal gaps formed by low-quality frames, and loop closure for reconstruction.

4. *A skeleton-driven generalized cylinder deformation algorithm has been designed.* Given a triangle mesh representing a generalized cylinder, I first skeletonize it by extracting a centerline and its orthogonal cross sections. The relative curvature condition is incorporated to eliminate (or minimize the number of) intersections between cross sections to prevents surface folding. The resulting centerline is deformed into the target shape, and the cross sections are mapped using rotation minimizing frames. The deformation vector field between the cross sections are solved by minimizing a Thin Shell bending energy in linear time. An accurate large-scale deformation is achieved while keeping local curvature patterns. The novelty lies in three aspects. 1) The relative curvature condition [69] was proposed detecting singularities in polar swept surfaces in continuous situations. We are the first to use it for resisting cross-sectional intersections. 2) I used rotation minimizing frames [70] to control twisting. 3) I cast a large-scale deformation into a geometrical deformation and a group of fine-scale Thin Shell optimization problems, which can be solved efficiently.

5. *The above centerline deformation algorithm was novelly applied to the colon semi-flattening problem.* Using the generalized cylinder deformation algorithm, I first straighten a colon surface by specifying the target centerline shape. Local curvature patterns are preserved during this process. Then it is slit open along a longitudinal line on the surface and mapped from a cylindrical surface to a semi-cylindrical surface. The result is a semi-flattened surface showing the interior surface succinctly without losing the local geometry. Compared to conformal mapping approaches, this algorithm has less area distortion and keeps 3D geometry. It can

handle colonic regions with very large curvature.

Thesis: *Our team's colonoscopic visualization system makes colonoscopy more accurate while maintaining efficiency by addressing the unsurveyed surface problem and relocalization difficulty. My contributions in this system of improved simultaneous localization and mapping in colon videos, colonic place recognition, and surface deformation of the colon model are critical to the system's success.*

6.2 Discussion and Future Work

6.2.1 Colon Reconstruction

Depth Ground Truth of Endoscopic Image Analysis Different from common images in traditional computer vision tasks, endoscopic images usually have less salient features, variable illumination and large scene deformation. This makes it hard to apply traditional methods like keypoint matching and tracking. Fortunately, developments in deep learning can provide a way to learn features that are suitable for endoscopic images. The challenge for deep learning lies in the difficulty to acquire groundtruth because endoscopes are not equipped with tools for measurement and human labeling is extremely expensive due to the need of specialty.

In [24] I presented an approach to collect depth groundtruth for colonoscopic images by SfM. Another way to generate ground truth is to use synthetic data. Using colon surfaces segmented from CT as determined by virtual colonoscopy [126, 132, 133], synthetic fly-through can generate endoscopic videos with depth ground truth. These have been used to train depth estimation neural networks [138, 21, 114]. Some transfer learning techniques such as adversarial training have been used to deal with the texture difference between real colonoscopic images and synthesized images. Mathew et al. [114] achieved state-of-the-art performance on colonoscopic depth estimation, as shown in Figure 6.1. Their depth map changes more sharply across the boundaries and is overall more accurate. This is because of the large data volume of synthetic data and transfer learning.

Dense depth ground truth is important not only for training, but also for evaluation. Without accurate depth ground truth, we currently have no way for evaluating the depth estimation network quantitatively.

One future direction is to build a colonoscopy simulator in order to generate depth ground truth. The virtual colon should have reasonable amount of texture. Some transfer learning methods should

also be used to fill the texture difference gap.

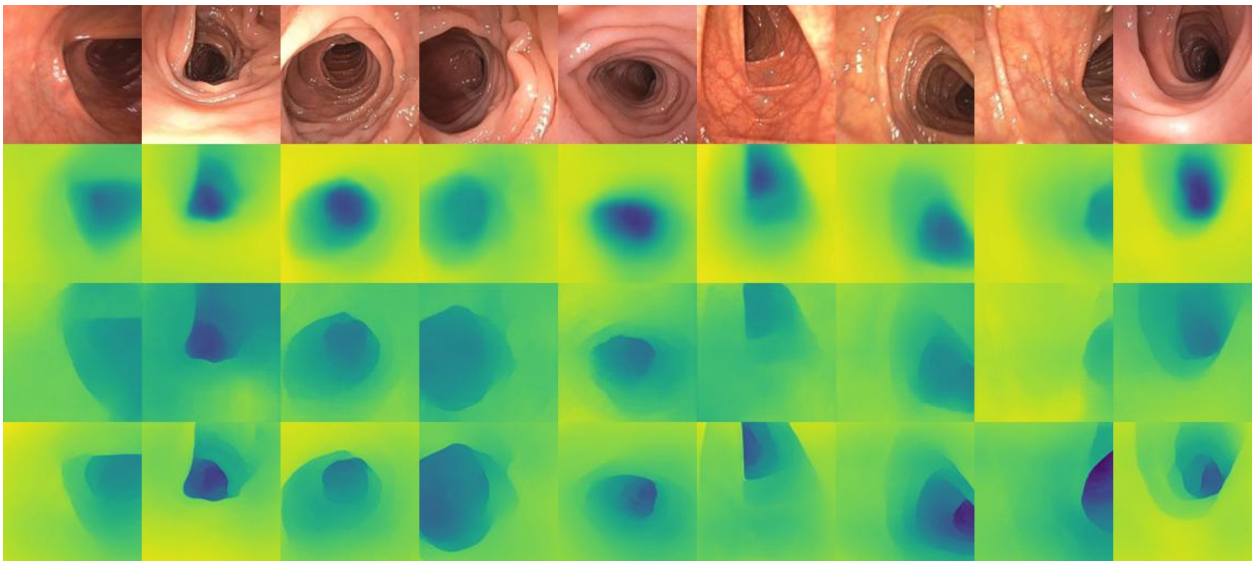


Figure 6.1: Qualitative comparison between our depth maps (2nd row) and those predicted by Chen et al. [21] (3rd row) and by Mathew et al. [114] (4th row).

Depth Ground Truth from SfM: Scale Issue Our current system uses an RNN-DP trained on SfM results of 12K 200-frame colonoscopic sequences. In order to deal with the scale difference between different reconstructions, we currently simply divided the depth values in each frame by the median depth value. This is based on the assumption that the depth maps of colon frames are roughly the same. Failure of that assumption may introduce large noise in the training data. One work addressing this issue is MegaDepth [139]. It introduced a scale-invariant loss based on the observation that "the ratios of pairs of depth are preserved under scaling". In log scale, the differences are preserved. Assuming we have an estimated log-depth L and the ground truth log-depth L', the scale-invariant depth loss is

$$loss = \frac{1}{n}\sum_{i=1}^{n}(R_i)^2 - \frac{1}{n^2}(\sum_{i=1}^{n} R_i)^2, R_i = L_i - L_i^* \tag{6.1}$$

where n is the number of locations where depth is available in an image. This scale-invariant depth loss is a hopeful direction for improving depth map accuracy of our current RNN-DP.

Depth Map Consistency Issue Recently, in depth prediction area more attention has been paid to consistent depth prediction [140, 141]. Being "consistent" means if we project all the depth maps to the 3D space, their reconstructed surfaces aligns well with each other. This is an important property for 3D-graphics related tasks such as 3D fusion, because in fusion algorithms [89, 87] the basic logic is to build 3D elements based on surface overlapping or coherence. Although these algorithms have some level of tolerance against noise or inconsistency between depth maps, heavy inconsistency will definitely harm the output completeness and quality. In fact, most existing depth prediction works have this issue because they predict depth maps on a frame-by-frame basis, and these include our RNNSLAM.

As mentioned in Chapter 3, my current solution for this inconsistency is a windowed depth averaging step, which registers the depth maps. A better solution is to have the depth maps reconstructed consistently in the beginning. This should leverage inter-frame information and should add explicit constraints to depth map consistency. Currently there are two influential works: Luo et al. [140] proposed running back-propagation at prediction time. Back-propagation is used as an optimization tool to explicitly reduce the depth map consistency loss. Fundamentally the algorithm is to "overfit" to the test data. They achieved very consistent depth maps that were used in downstream graphics application. The major limitation of this work is its high time consumption caused by back-propagation. Tang et al. [141] proposed the Bundle Adjustment Network (BANet). They designed a BA Layer to "optimize" the depth map and camera pose output from several frames. The major novelty is that they designed a differentiable Levenberg-Marquardt (LM) algorithm whose step size is predicted by neural networks. Although it is questionable whether the algorithm is actually doing optimization, their BA layer indeed explicitly adjusted the depth maps and camera poses, which constrained the degrees of freedom of the problem. Because this algorithm predicts depth maps by forward passes, its speed is not a problem. As a future direction, I suggest adopting the BA layer into our depth prediction network.

SLAM + Deep Learning Is a Win-Win Strategy An important lesson I have learned from my RNNSLAM work is the importance of the combination of deep learning and SLAM. More generally, this is a combination of deep learning and an optimization framework, and it improves the accuracy of the overall system. This general idea applies more generally to other problems

that have high accuracy requirements. Next I will explain why this combination helps based on my understanding.

From a maximum-a-posteriori perspective, the maximization of the posterior probability of the observation (a colonoscopic video) is equivalent to minimizing an energy function composed of a data matching term (maximum likelihood) and a prior term. Typically, the prior term in a standard SLAM pipeline is limited to regularization or smoothness terms. Our RNN-DP prediction (depth and pose) injects robust prior knowledge into the system although it is not explicitly used as a regularization term in the optimization formulation. This is why RNNSLAM performs much better than DSO. Particularly, it is scale-consistent. Also, an end-to-end deep neural network predicts depth and pose by forward passes. There exists no optimization, so it can be understood as purely prior-based prediction. That is why it cannot handle accumulated camera pose drift. Combining the two strategies is a big step toward maximum-a-posteriori.

Extreme Tracking Conditions The RNNSLAM cannot handle extreme tracking conditions such as very fast camera motion, very blurry images, large scenes deformation, large illumination changes and many successive bad frames (like frames polluted by liquids, fecal materials, etc.). This is why we currently reconstruct in chunks. Also, our pipeline cannot handle extremely low-textured frames such as a frame whose camera is very close to the colon surface and the whole image is homogeneous. These two challenges are illustrated in Figure 6.2.

Therefore, one future direction is to improve the robustness against extreme tracking conditions. One possible direction is to incorporate better depth prediction networks such as the one trained by virtual colonoscopy [114]. Another possibility is to improve the completeness of the reconstruction by merging two reconstructions based on the image retrieval technique; this can make up for the failed part in one reconstruction caused by bad tracking condition if it is better visualized in another reconstruction.

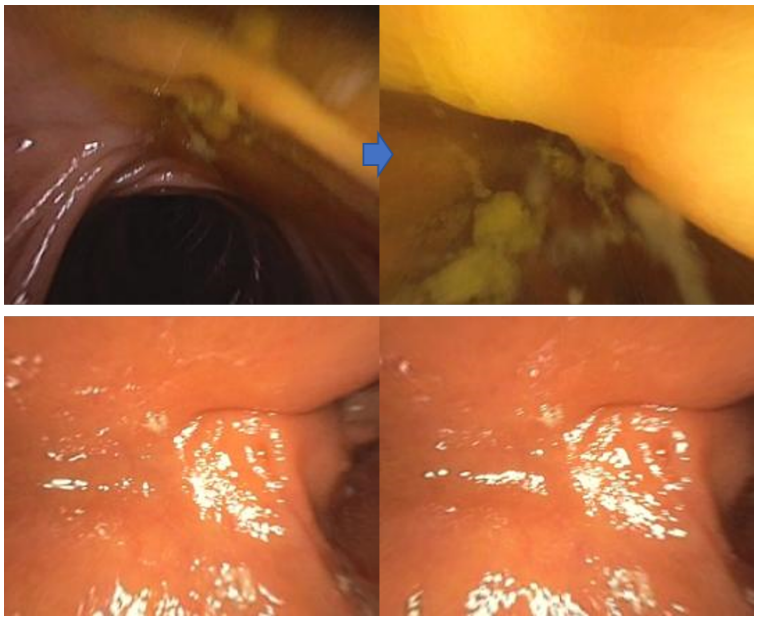


Figure 6.2: Two difficult scenarios for RNNSLAM. Top: tracking will fail when the change of content is very big, especially when the image is polluted by garbage. Bottom: tracking cannot work for extremely homogeneous images whose cameras are very close to the colon surface.

6.2.2 Colonoscopic Image Retrieval

In this section I discuss some future directions of the colonoscopic image retrieval work.

Extracting Local Features The CNN image retriever introduced in Chapter 4 computes a global descriptor for each input image. Because the global descriptor comes from the pooling of the final feature maps, many important local features may be lost during the procedure. However, in a lot of cases, especially the cases with large deformation, local features are very helpful for recognizing a place. For example, we may just pay attention to a patch of vessel texture to determine whether it is the same colonic location.

Traditional local features including SIFT and ORB do not work well in colonoscopic images because of the lack of distinguishable salient points. There have been works in the deep learning area about extracting local feature points and descriptors. For example, the TFeat [142] is a neural network that predicts a descriptor for a local patch. The patch is extracted by other feature detectors. The Learned Invariant Feature Transform (LIFT) [143] is an end-to-end alternative of SIFT that uses deep learning on both feature extraction and descriptor computing. The MatchNet [144] tries to also learn the metric function. It uses a network to compute a score from two descriptors. These

mentioned deep learning methods all used a siamese training (or similar variant) strategy. Although they have shown promising results on their test set, it is still questionable whether they are as robust as SIFT on all kinds of images. Given the significant texture difference in colonoscopy, some finetuning or even architecture modification of these networks may be necessary.

These potential deep features also can be applied into the 3D reconstruction. One way is to feed neural-network-predicted keypoints and descriptors into SfM pipelines. Since SfM has been used for generating ground truth for both RNN-DP and our CNN image retriever, better SfM results are very hopeful to improve reconstruction quality and increase the number successfully reconstructed sequences. For SLAM, deep features can also be used to replace the traditional features in tracking and make tracking more stable and robust in colonoscopies.

Regional Features and Attention Mechanism An important area in image retrieval is the "region-based" variant. Generally a region proposal network is concatenated with a retrieval network. For example, Gordo et al. [145] proposed using a region proposal network to select regions of interest from an image and then to use a CNN to compute a descriptor for each of the regions. Finally, they used maximum pooling to get a compact global descriptor. This is called regional maximum activations of convolutions (R-MAC). Region-based descriptors have the advantage of focusing on the object of interest or the best textured part of an image. However because in colonoscopic images, there is no well-defined "object", the region proposal network may need certain retraining or modification.

A strategy that may fit better with colonoscopic images is the attention mechanism. Basically this explicitly computes a weighting mask and multiplies it with feature maps in order to emphasize features at certain locations. Kim et. al. [146] proposed the Contextual Reweighting Network (CRN). The weighting mask computed showed that the network is paying much more reasonable attention to the discriminative regions, thereby leading to better retrieval performance. In colonoscopic images because we often pay attention to regions like the haustral ridges and the vessel patterns, it can be helpful to add an attention module to the current retrieval network.

Implementing Fast Retrieval Currently my way to match the descriptors is simply to compute dot products between the current descriptor and historical descriptors and find the highest scores.

This approach to find the nearest neighbors is in linear time. It is fast enough for thousands of images but may be too slow for a much longer video (like 10^6 images[1]). In this scenario some approximate nearest neighbor search algorithm should be employed. For example, Jégou et al. [147] proposed the product quantization algorithm, which decomposes the high-dimensional vector space into the Cartesian product of subspaces and then quantizes these subspaces separately.

Integrating Retrieval into Reconstruction In order to fully apply the image retrieval introduced in Chapter 4, it is better to integrate the image retriever with the reconstruction system introduced in Chapter 3. Specifically, this includes the following aspects:

1. **Jumping Temporal Gaps**: RNNSLAM's tracking sometimes fails for a short temporal gap formed by low-quality images. If some frames after the gap are similar enough to the frames before the gap, a relative camera pose can be estimated to resume the tracking or connect the models before and after the gap together. This recognition can be done by the image retriever presented in Chapter 4.

2. **Refining Reconstruction after Loop Closure**: I presented a loop closure demo in Section 4.4.2. It adjusts the camera trajectory. When integrating this CNN-based loop closure into a reconstruction system, the 3D model needs to be adjusted accordingly. This requires rerunning the fusion algorithm.

3. **Guidance Back Module**: When an unsurveyed region is detected, the endoscopist needs navigation back to the colonic location. This is an important component for a complete unsurveyed region detection system. The image retriever can be used for this navigation. It needs to be integrated into the reconstruction system so that the location can be shown w.r.t. the existing reconstruction.

6.2.3 Colon Surface Deformation

Some possible improvements of my generalized cylinder deformation method include the following:

[1]Not likely for a 0.5 hr colonoscopy, but may be true for other applications more generally.

1. Although my current method can handle quite complicated generalized cylinders like the human colon, we lack a proof of the existence of a centerline that prevents intersections in all cases. When the surface deviates from a smooth generalized cylinder, e.g., it has a protruding hump on the side of the Frenet normal, it is more difficult to find a satisfactory centerline by smoothing. I currently use interpolation as a maneuver.

2. My method is not applicable to tree-structured tubular objects. One of the further developments can be extending our skeleton representation for the branching tubular structures like blood vessels. Xu, X. *et al.* [148] proposed the method based on electrical field theory to model a branching generalized cylinder. Zhou *et al.* [68] proposed a generalized cylinder decomposition method that allows for dealing with objects composed of multiple generalized cylinders. Mortara *et al.* in [149] proposed a method to decompose a mesh into shape features and in particular in [150] proposed an algorithm to decompose the mesh into tubular primitives. The special challenge to use our method in this case, to be solved in the future, is to represent the branch point and how the cross sections come together there.

3. The centerline can be outside the input mesh as a result of the smoothing. This is why I set a maximum iteration count and use Lorentzian interpolation to deal with remaining regions that violates the relative curvature condition. Although the current method is sufficient for the examples being used even for high curvature regions, a better smoothing algorithm like the free-knot-spline method [151] can be incorporated.

4. Deformations with more abundant details and better realism have been achieved by example-driven approached such as [56, 152, 153]. In [152], Wampler et al. proposed the "as-multi-rigid-as-possible" method that used spatially localized weights to blend artist-designed models. In [153] Gao *et al.* proposed a data-driven method that achieved quite smooth and complex shape morphing using very sparse control points. My deformation strategy leverages geometric deformation in the early stage in which the shapes of cross sections are not changed; this could limit the capability for detailed deformation. Although for certain objects the shape examples can be difficult to get, e.g., human colon surfaces, for those with cheap shape examples, my extracted centerline could serve as an efficient geometric constraint.

5. In colon visualization, we can incorporate the method in [132] or find the taenia coli to find a better slitting line.

6.3 Conclusion

During the period of defending and polishing this dissertation, many other suggestions for improvement have been made. I leave these to future papers.

REFERENCES

[1] A. Jemal, M. M. Center, C. DeSantis, and E. M. Ward, "Global patterns of cancer incidence and mortality rates and trends," *Cancer Epidemiology and Prevention Biomarkers*, vol. 19, no. 8, pp. 1893–1907, 2010.

[2] W. Hong, J. Wang, F. Qiu, A. Kaufman, and J. Anderson, "Colonoscopy simulation," in *Proc.SPIE*, 2007.

[3] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, 12 2017.

[4] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Mar. 2018.

[5] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular slam," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 834–849, Springer International Publishing, 2014.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[7] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[8] E. Rosten, R. B. Porter, and T. Drummond, "Faster and better: a machine learning approach to corner detection," *CoRR*, vol. abs/0810.2434, 2008.

[9] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 778–792, Springer Berlin Heidelberg, 2010.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, Nov. 2004.

[11] J. Dong and S. Soatto, "Domain-size pooling in local descriptors: DSP-SIFT," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5097–5106, June 2015.

[12] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2650–2658, 2015.

[13] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, 2017.

[14] R. Wang, S. M. Pizer, and J.-M. Frahm, "Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[15] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[16] Z. Li and N. Snavely, "Megadepth: Learning single-view depth prediction from internet photos," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] R. Wang, J. Frahm, and S. M. Pizer, "Recurrent neural network for learning densedepth and ego-motion from video," *CoRR*, vol. abs/1805.06558, 2018.

[18] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular slam with learned depth prediction," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 00, pp. 6565–6574, July 2017.

[19] N. Yang, R. Wang, J. Stueckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *ECCV*, 2018.

[20] R. Wang, M. Schwörer, and D. Cremers, "Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras," in *International Conference on Computer Vision (ICCV), Venice, Italy*, 2017.

[21] R. J. Chen, T. L. Bobrow, T. Athey, F. Mahmood, and N. J. Durr, "SLAM endoscopy enhanced by adversarial depth prediction," *arxiv:1907.00283*, 2019.

[22] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3d reconstruction in dynamic scenes using point-based fusion," in *2013 International Conference on 3D Vision - 3DV 2013*, pp. 1–8, June 2013.

[23] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, pp. 1697 – 1716, 2016.

[24] Ma, Ruibin and Wang, Rui, S. Pizer, J. Rosenman, S. K. McGill, and J.-M. Frahm, "Real-time 3d reconstruction of colonoscopic surfaces for determining missing regions," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, (Cham), pp. 573–582, Springer International Publishing, 2019.

[25] H. Nosato, H. Sakanashi, E. Takahashi, M. Murakawa, H. Aoki, K. Takeuchi, and Y. Suzuki, "Image retrieval method for multiscale objects from optical colonoscopy images," *International Journal of Biomedical Imaging*, vol. 2017, pp. 1–13, 02 2017.

[26] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," 2014.

[27] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of cnn activations," 2015.

[28] A. Babenko and V. S. Lempitsky, "Aggregating deep convolutional features for image retrieval," *CoRR*, vol. abs/1510.07493, 2015.

[29] R. Arandjelovic, P. Gronát, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," *CoRR*, vol. abs/1511.07247, 2015.

[30] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning cnn image retrieval with no human annotation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 1655–1668, July 2019.

[31] R. Ma, S. McGill, R. Wang, J. Rosenman, J.-M. Frahm, and S. Pizer, "Colon10k: A benchmark for place recognition in colonoscopy," *in conference submission*.

[32] A. E. Kaufman, S. Lakare, K. Kreeger, and I. Bitter, "Virtual colonoscopy," *Communications of the ACM*, vol. 48, no. 2, 2005.

[33] M. Wan, Q. Tang, A. Kaufman, Z. Liang, and M. Wax, "Volume rendering based interactive navigation within the human colon," in *IEEE Visualization*, pp. 397–400, 1999.

[34] S. McGill, J. Rosenman, Q. Zhao, R. Wang, R. Ma, M. Niethammer, R. Alterovitz, J.-M. Frahm, J. Tepper, and S. Pizer, "3d reconstruction of colonoscopy videos: A new approach to improving the quality and effectiveness of colonoscopy," *In Preparation*.

[35] Q. Zhao, T. Price, S. Pizer, M. Niethammer, R. Alterovitz, and J. Rosenman, "The endoscopogram: A 3d model reconstructed from endoscopic video frames," in *MICCAI*, 2016.

[36] T. Price, Q. Zhao, J. Rosenman, S. Pizer, and J.-M. Frahm, "Shape from motion and shading in uncontrolled environments," tech. rep., Department of Computer Science, University of North Carolina at Chapel Hill, 2016.

[37] R. Wang, T. Price, Q. Zhaoa, J.-M. Frahm, J. Rosenman, and S. Pizer, "Improving 3d surface reconstruction from endoscopic video via fusion and refined reflectance modeling," in *Proc. of SPIE Vol*, vol. 10133, pp. 101330B–1, 2017.

[38] D. S. Paik, C. F. Beaulieu, R. B. J. Jr, C. A. Karadi, and S. Napel, "Visualization modes for CT colonography using cylindrical and planar map projections," *J. Comput. Assist. Tomogr.*, vol. 24, no. 2, pp. 179–188, 2000.

[39] A. V. Bartrolí, R. Wegenkittl, A. König, and M. E. Gröller, "Virtual colon unfolding," Tech. Rep. TR-186-2-01-08, Institute of Computer Graphics and Algorithms, Vienna University of Technology, April 2001.

[40] S. Haker, S. Angenent, A. Tannenbaum, and R. Kikinis, "Nondistorting flattening maps and the 3d visualization of colon ct images," *IEEE Transactions on Medical Imaging*, vol. 19, no. 7, pp. 665–670, 2000.

[41] L. Zhu, S. Haker, and A. Tannenbaum, "Conformal flattening maps for the visualization of vessels," in *SPIE Medical Imaging, 4681:742748*, 2002.

[42] L. Zhu, S. Haker, and A. Tannenbaum, "Flattening maps for the visualization of multibranched vessels," *IEEE Transactions on Medical Imaging*, vol. 24, no. 2, 2005.

[43] M.-S. Kim, E. Park, and H. Lee, "Modeling and animation of generalized cylinders with variable radius offset space curves," *The Journal of Visualization and Computer Animation*, vol. 5, pp. 189 – 207, 10 1994.

[44] D. Ballard and C. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, Inc, 1982.

[45] U. Shani and D. H. Ballard, "Splines as embeddings for generalized cylinders," *Computer Vision, Graphics, and Image Processing*, vol. 27, no. 2, pp. 129 – 156, 1984.

[46] T.-I. Chang, J.-H. Lee, M.-S. Kim, and S. Je Hong, "Direct manipulation of generalized cylinders based on b-spline motion," *The Visual Computer*, vol. 14, pp. 228–239, 10 1998.

[47] B. Jüttler and M. G. Wagner, "Computer-aided design with spatial rational b-spline motions," *ASME J. Mechan. Des.*, vol. 118, pp. 193–201, 1996.

[48] T. O'Donnell, T. E. Boult, X.-S. Fang, and A. Gupta, "The extruded generalized cylinder: a deformable model for object recovery," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 174–181, Jun 1994.

[49] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *SIGGRAPH Comput. Graph.*, vol. 20, pp. 151–160, Aug. 1986.

[50] W. M. Hsu, J. F. Hughes, and H. Kaufman, "Direct manipulation of free-form deformations," in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, (New York, NY, USA), pp. 177–184, ACM, 1992.

[51] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, (New York, NY, USA), pp. 175–184, ACM, 2004.

[52] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, "Mesh editing with poisson-based gradient field manipulation," in *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, (New York, NY, USA), pp. 644–651, ACM, 2004.

[53] O. Sorkine, "Laplacian Mesh Processing," in *Eurographics 2005 - State of the Art Reports* (Y. Chrysanthou and M. Magnor, eds.), The Eurographics Association, 2005.

[54] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum, "Large mesh deformation using the volumetric graph laplacian," in *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, (New York, NY, USA), pp. 496–503, ACM, 2005.

[55] R. W. Sumner and J. Popović, "Deformation transfer for triangle meshes," in *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, (New York, NY, USA), pp. 399–405, ACM, 2004.

[56] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pp. 109–116, Eurographics Association, 2007.

[57] W.-W. Xu and K. Zhou, "Gradient domain mesh deformation — a survey," *Journal of Computer Science and Technology*, vol. 24, pp. 6–18, Jan 2009.

[58] M. Botsch and O. Sorkine, "On linear variational surface deformation methods," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 213–230, Jan. 2008.

[59] S. Zhang, J. Huang, and D. N. Metaxas, "Robust mesh editing using Laplacian coordinates," *Graph. Models*, vol. 73, pp. 10–19, Jan. 2011.

[60] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan, "Skinning with dual quaternions," in *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, (New York, NY, USA), pp. 39–46, ACM, 2007.

[61] L. Kavan and O. Sorkine, "Elasticity-inspired deformers for character articulation," *ACM Trans. Graph.*, vol. 31, pp. 196:1–196:8, Nov. 2012.

[62] D. Rohmer, S. Hahmann, and M.-P. Cani, "Local volume preservation for skinned characters," *Computer Graphics Forum*, 2008.

[63] Y. Kho and M. Garland, "Sketching mesh deformations," in *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, ACM, 2007.

[64] S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel, "Skeleton-based variational mesh deformations," *Computer Graphics Forum*, vol. 26, no. 3, pp. 255–264, 2007. Proc. of EUROGRAPHICS'07.

[65] G. Aujay, F. Hétroy, F. Lazarus, and C. Depraz, "Harmonic skeleton for realistic character animation," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pp. 151–160, Eurographics Association, 2007.

[66] L. Antiga, B. Ene-Iordache, and A. Remuzzi, "Centerline computation and geometric analysis of branching tubular surfaces with application to blood vessel modeling," in *WSCG*, 2003.

[67] W. Zeng, J. Marino, K. C. Gurijala, X. Gu, and A. Kaufman, "Supine and prone colon registration using quasi-conformal mapping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, pp. 1348–1357, Nov 2010.

[68] Y. Zhou, K. Yin, H. Huang, H. Zhang, M. Gong, and D. Cohen-Or, "Generalized cylinder decomposition," *ACM Trans. Graph.*, vol. 34, pp. 171:1–171:14, Oct. 2015.

[69] J. Damon, "Swept regions and surfaces: Modeling and volumetric properties," *Theoretical Comp. Science*, vol. 392, pp. 66–91, 2008.

[70] W. Wang, B. Jüttler, D. Zheng, and Y. Liu, "Computation of rotation minimizing frames," *ACM Trans. on Graph.*, vol. 27, no. 1, Article 2, 2008.

[71] M. Bergou, M. Wardetzky, D. Harmon, D. Zorin, and E. Grinspun, "A quadratic bending model for inextensible surfaces," in *Eurographics Symposium on Geometry Processing*, pp. 227–230, June 2006.

[72] Q. Zhao, T. Price, S. Pizer, M. Niethammer, R. Alterovitz, and J. Rosenman, "Surface registration in the presence of missing patches and topology change," in *Proceedings of the Medical Image Understanding and Analysis Conference*, 2015.

[73] N. C. Institute, "Large intestine," *NCI Dictionary of Cancer Terms*, 2011-02-02.

[74] A. Handa, M. Blösch, V. Patraucean, S. Stent, J. McCormac, and A. J. Davison, "GVNN: Neural network library for geometric computer vision," *CoRR*, vol. abs/1607.07405, 2016.

[75] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[76] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15–22, May 2014.

[77] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *2013 IEEE International Conference on Computer Vision*, pp. 1449–1456, Dec 2013.

[78] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*, pp. 2320–2327, Nov 2011.

[79] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2O: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, 2011.

[80] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "A first-estimates jacobian ekf for improving slam consistency," in *Experimental Robotics* (O. Khatib, V. Kumar, and G. J. Pappas, eds.), (Berlin, Heidelberg), pp. 373–382, Springer Berlin Heidelberg, 2009.

[81] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[82] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *CoRR*, vol. abs/1506.04214, 2015.

[83] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[84] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[85] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[86] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), pp. 303–312, ACM, 1996.

[87] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, 2011.

[88] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," SIGGRAPH 2000, (USA), pp. 335–342, ACM Press/Addison-Wesley Publishing Co., 2000.

[89] T. Schöps, T. Sattler, and M. Pollefeys, "Surfelmeshing: Online surfel-based mesh reconstruction," *CoRR*, vol. abs/1810.00729, 2018.

[90] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "Surface splatting," in *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, pp. 371–378, 07/2001 2001.

[91] Sivic and Zisserman, "Video google: a text retrieval approach to object matching in videos," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 1470–1477 vol.2, 2003.

[92] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 1704–1716, Sep. 2012.

[93] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3384–3391, June 2010.

[94] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2007.

[95] J. L. Schönberger, T. Price, T. Sattler, J.-M. Frahm, and M. Pollefeys, "A vote-and-verify strategy for fast spatial verification in image retrieval," in *Asian Conference on Computer Vision (ACCV)*, 2016.

[96] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2911–2918, 2012.

[97] M. Perd'och, O. Chum, and J. Matas, "Efficient representation of local geometry for large scale object retrieval," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9–16, 2009.

[98] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[99] F. Radenović, G. Tolias, and O. Chum, "CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples," in *ECCV*, 2016.

[100] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[101] G. Zheng, S. Li, and G. Szekely, *Statistical Shape and Deformation Analysis: Methods, Implementation and Applications. Chapter Six by S. Pizer and J. S. Marron.* Elsevier Science, 2017.

[102] A. Verroust and F. Lazarus, "Extracting skeletal curves from 3d scattered data," *Vis. Comput.*, vol. 16, pp. 15–25, Feb. 2000.

[103] M. Mortara and G. Patanè, "Shape-covering for skeleton extraction," *International Journal of Shape Modeling*, vol. 08, no. 02, pp. 139–158, 2002.

[104] J.-H. Chuang, C.-H. Tsai, and M.-C. Ko, "Skeletonisation of three-dimensional object using generalized potential field," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1241–1251, Nov 2000.

[105] L. Antiga, B. Ene-Iordache, and A. Remuzzi, "Computational geometry for patient-specific reconstruction and meshing of blood vessels from mr and ct angiography," *IEEE Transactions on Medical Imaging*, vol. 22, May 2003.

[106] L. Antiga, M. Piccinelli, L. Botti, B. Ene-Irodache, A. Remuzzi, and D. Steinman, "An image-based modeling framework for patient-specific computational hemodynamics," *Med Biol Eng Comput*, vol. 46, pp. 1097–1112, 2008.

[107] S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel, "Free-form skeleton-driven mesh deformations," in *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, SM '03, (New York, NY, USA), pp. 247–253, ACM, 2003.

[108] X. Shi, K. Zhou, Y. Tong, M. Desbrun, H. Bao, and B. Guo, "Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics," in *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, (New York, NY, USA), ACM, 2007.

[109] J. P. Lewis, M. Cordner, and N. Fong, "Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, (New York, NY, USA), pp. 165–172, ACM Press/Addison-Wesley Publishing Co., 2000.

[110] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman, "Context-aware skeletal shape deformation," *Computer Graphics Forum (Proceedings of EUROGRAPHICS)*, vol. 26, no. 3, pp. 265–273, 2007.

[111] G. Wang, E. McFarland, B. Brown, Z. Zhang, and M. Vannier, "Curved cross-section based system and method for gastrointestinal tract unraveling," Apr. 3 2001. US Patent 6,212,420.

[112] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[113] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras," 10 2006.

[114] S. Mathew, S. Nadeem, S. Kumari, and A. Kaufman, "Augmenting colonoscopy using extended and directional cyclegan for lossy image translation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[115] Y. Zhang and S. Pizer, "Unsurveyed region," *UNC Computer Science Technical Report (2021)*.

[116] Y. Zhang and S. Pizer, "Lighting," *UNC Computer Science Technical Report (2021)*.

[117] M. Grupp, "EVO: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.

[118] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Revisiting oxford and paris: Large-scale image retrieval benchmarking," in *CVPR*, 2018.

[119] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546 vol. 1, June 2005.

[120] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, 2006.

[121] H. J. Kim, E. Dunn, and J.-M. Frahm, "Learned contextual feature reweighting for image geo-localization," in *CVPR*, July 2017.

[122] M. Cummins and P. Newman, "Invited Applications Paper FAB-MAP: Appearance-Based Place Recognition and Mapping using a Learned Visual Vocabulary Model," in *27th Intl Conf. on Machine Learning (ICML2010)*, 2010.

[123] R. Ma, Q. Zhao, R. Wang, J. Damon, J. Rosenman, and S. Pizer, "Skeleton-based Generalized Cylinder Deformation under the Relative Curvature Condition," in *Pacific Graphics Short Papers*, The Eurographics Association, 2018.

[124] R. Ma, Q. Zhao, R. Wang, J. Damon, J. Rosenman, and S. Pizer, "Deforming generalized cylinders without self-intersection by means of a parametric center curve," *Computational Visual Media*, vol. 4, pp. 305–321, 2018.

[125] J. Damon, "Lorentzian geodesic flows and interpolation between hypersurfaces in euclidean spaces," *Preliminary preprint in MIDAG paper collection*, 2018.

[126] W. Hong, X. Gu, F. Qiu, M. Jin, and A. Kaufman, "Conformal virtual colon flattening," in *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling*, SPM '06, (New York, NY, USA), pp. 85–93, ACM, 2006.

[127] J. Peng, D. Kristjansson, and D. Zorin, "Interactive modeling of topologically complex geometric detail," in *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, (New York, NY, USA), pp. 635–643, ACM, 2004.

[128] Y. Gingold, A. Secord, J. Y. Han, E. Grinspun, and D. Zorin, "A discrete model for inelastic deformation of thin shells," tech. rep., Courant Institute of Mathematical Sciences, 2004.

[129] E. Grinspun, "A discrete model of thin shells," in *SIGGRAPH*, 2006.

[130] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In Visualization and Mathematics III*, pp. 35–57. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[131] M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun, "Discrete laplace operators: No free lunch," in *Eurographics Symposium on Geometry Processing*, 2007.

[132] S. Nadeem, J. Marino, X. Gu, and A. Kaufman, "Corresponding supine and prone colon visualization using eigenfunction analysis and fold modeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, pp. 751–760, Jan 2017.

[133] S. Nadeem, X. D. Gu, and A. E. Kaufman, "LMap: Shape-preserving local mappings for biomedical visualization," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2017.

[134] H. Wang, L. Li, H. Han, R. Shi, B. Song, H. Peng, Y. Liu, X. Gu, Y. Wang, and Z. Liang, "A 2.5d colon wall flattening model for ct-based virtual colonoscopy," in *Machine Learning in Medical Imaging* (G. Wu, D. Zhang, D. Shen, P. Yan, K. Suzuki, and F. Wang, eds.), (Cham), pp. 203–210, Springer International Publishing, 2013.

[135] J. Marino, "Context preserving maps of tubular structures," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 1997–2004, 2011.

[136] J. Marino and A. Kaufman, "Planar visualization of treelike structures," vol. 22, pp. 1–1, 11 2015.

[137] R. Vaillant, L. Barthe, G. Guennebaud, M.-P. Cani, D. Rohmer, B. Wyvill, O. Gourmel, and M. Paulin, "Implicit Skinning: Real-Time Skin Deformation with Contact Modeling," *ACM Transactions on Graphics*, vol. 32, p. Article No. 125, July 2013. SIGGRAPH 2013 Conference Proceedings.

[138] S. Nadeem and A. E. Kaufman, "Depth reconstruction and computer-aided polyp detection in optical colonoscopy video frames," *CoRR*, vol. abs/1609.01329, 2016.

[139] Z. Li and N. Snavely, "Megadepth: Learning single-view depth prediction from internet photos," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[140] X. Luo, J.-B. Huang, R. Szeliski, K. Matzen, and J. Kopf, "Consistent video depth estimation," 2020.

[141] C. Tang and P. Tan, "BA-Net: Dense bundle adjustment network," 2018.

[142] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proceedings of the British Machine Vision Conference (BMVC)* (E. R. H. Richard C. Wilson and W. A. P. Smith, eds.), pp. 119.1–119.11, BMVA Press, September 2016.

[143] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned invariant feature transform," 2016.

[144] Xufeng Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "Matchnet: Unifying feature and metric learning for patch-based matching," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3279–3286, 2015.

[145] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," 2016.

[146] H. J. Kim, E. Dunn, and J. Frahm, "Learned contextual feature reweighting for image geo-localization," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3251–3260, 2017.

[147] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, pp. 117–128, Jan. 2011.

[148] X. Xu, J. M. Reinhardt, Q. Hu, B. Bakall, P. S. Tlucek, G. Bertelsen, and M. D. Abràmoff, "Retinal vessel width measurement at branchings using an improved electric field theory-based graph approach," *PLoS ONE*, vol. 7, no. 11, 2012.

[149] M. Mortara, G. Patané, M. Spagnuolo, B. Falcidieno, and J. Rossignac, "Blowing bubbles for multi-scale analysis and decomposition oftriangle meshes," *Algorithmica*, vol. 38, pp. 227–248, Jan 2004.

[150] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, and J. Rossignac, "Plumber: A method for a multi-scale decomposition of 3d shapes into tubular primitives and bodies," in *Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications*, SM '04, (Aire-la-Ville, Switzerland, Switzerland), pp. 339–344, Eurographics Association, 2004.

[151] L. M. Sangalli, P. Secchi, S. Vantini, and A. Veneziani, "Efficient estimation of three-dimensional curves and their derivatives by free-knot regression splines, applied to the analysis of inner carotid artery centrelines," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 58, no. 3, pp. 285–306, 2009.

[152] K. Wampler, "Fast and reliable example-based mesh ik for stylized deformations," *ACM Trans. Graph.*, vol. 35, pp. 235:1–235:12, Nov. 2016.

[153] L. Gao, Y.-K. Lai, D. Liang, S.-Y. Chen, and S. Xia, "Efficient and flexible deformation representation for data-driven surface modeling," *ACM Trans. Graph.*, vol. 35, pp. 158:1–158:17, July 2016.