Endoscopic Motion Estimation Using Video and CT

by

Linhao Jin

A thesis submitted to The Johns Hopkins University

in conformity with the requirements for the degree of

Master of Science in Engineering

Baltimore, Maryland

March, 2019

Abstract

Functional Endoscopic Sinus Surgery (FESS) is a surgical procedure that otolaryngologists have adopted to treat sinus diseases. Aiming for accurate treatments and less complications, surgeons are usually guided with an endoscopic navigation system when performing the surgery. The state-of-the-art navigation systems report a submillimeter positioning error. This significantly reduces intraopertive time and improves surgical outcomes.

Navigating endoscope is similar to Visual Odometry (VO) or Simultaneous Localization and Mapping (SLAM), all of which require an estimation of camera poses and motions in an unknown environment. Feature-based methods and direct methods are two common approaches for VO and Visual SLAM for motion estimation, but both methods have drawbacks. Feature computation and feature extraction consume are usually not computationally effective, while direct methods suffer from local optima. One recent alternative is called Semi-Direct Method, or hybrid method, which overcomes the drawbacks by applying optimization that is used in direct method to the selected features.[15] In this work, we introduce a novel endoscopic navigation system for FESS which uses both prescanned CT model and 2D endoscope video. The system is able to texture map the CT model in real time for visualization and to refine the pose estimation of the endoscope from different prior estimates.

Thesis Committee

Primary Readers

Gregory D. Hager Ph.D. (Primary Advisor)
Mandell Bellmore Professor
Department of Computer Science
Johns Hopkins University

Acknowledgments

First of all, I would like to thank to my academic advisor and thesis advisor, Dr. Gregory Hager for all of his ideas, advice, and supports. No fantastic idea would be realized without this decent man.

Next thanks must go to the members in the Malone Center. I am grateful for the staffs and students who have helped with all aspects of my work including exchanging ideas, arranging meetings and giving feedback.

Two years at Johns Hopkins have been a great laugh due to my friends: Ji, Yixuan, Chaoyi, Bowen, and Yulai. You make a lot of things easier.

Last thanks my parents for the financial assistance, daily phone call, emotional supports and everything else.

Table of Contents

Table of Contents v

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Functional Endoscopic Sinus Surgery 1

1.2 Simultaneous Localization and Mapping 2

1.3 Outline 3

1.4 Motion Estimation 4

1.4.1 2D-2D Motion Estimation 4

1.4.1.1 Feature-based Methods 4

1.4.1.2 Direct Methods 5

1.4.1.3 Hybrid Methods 6

1.4.2 3D-2D Motion Estimation 6

1.4.3 3D-3D Motion Estimation 7

1.5 Motivation and Challenge 7

2 Overview 9

2.1 System Architecture 9

2.2 Terminology 11

2.3 Notation 12
2.4 Fundamentals 12
2.4.1 Camera Model and Calibration 12
2.4.2 3D Geometry and Transformation 14
2.4.2.1 Homogeneous Representation and Lie Group 14
2.4.2.2 Euler Angles 15
2.4.2.3 Axis-angle Representation 16
2.4.2.4 Quaternions 17
2.4.3 Non-linear Optimization 18
2.4.3.1 Line Search Methods 19
2.4.3.2 Trust Region Methods 19

3 CT Texture Mapping **21**
3.1 Texture Mapping Pipeline 21
3.2 Back-projection Texture Mapping 22
3.3 Image Rendering 23

4 Motion Estimation **26**
4.1 Prior Motion Estimation 26
4.2 Image Registration 27
4.3 Outlier Removal 30
4.4 Pose Estimation 31
4.4.1 3D-2D Perspective-n-points 31
4.4.2 2D-2D 8-point Method 33

5 Motion-only Bundle Adjustment **36**
5.1 Bundle Adjustment 36
5.2 Pose-graph Optimization 37

6 Performance Evaluations **40**

6.1 Energy Function 40

6.2 Minimization of Energy Function 41

6.3 Estimation results 41

6.4 Early Stopping 49

6.5 Error Statistics 50

7 Conclusion and Discussion **51**

7.1 Discussion 51

7.1.1 Mathematical Interpretation 51

7.1.2 Comparison with Direct Methods 52

7.1.3 Comparison with Feature-based Methods 53

7.2 Future Works 53

References **55**

List of Tables

6.1 Motion estimation starting from a manually selected position. The prior pose is -0.5mm to the current pose in x direction. This table shows the pose estimations for different number of iterations. The unit for movements in x, y, z directions is in millimeter, and for yaw, pitch, roll is in rads. The last column shows the ground truth for the estimated pose. 42

6.2 Motion estimation starting from a random position. This table shows the pose estimates for different iterations. The unit for movements in x, y, z directions is in millimeter, and for yaw, pitch, roll is in rads. The last column shows the ground truth for the estimated pose. 45

List of Figures

2.1 Back-projection texture mapping pipeline 9

3.1 Standard texture mapping pipeline 21

3.2 Back-projection texture mapping pipeline 22

3.3 A textured CT model back-projection texture mapping. Gray vertices in the model means either the vertex is not seen by the camera, or it is occluded, others are shaded with color from endoscopic image. 23

3.4 A real time visualization of textured point cloud in RViz . . . 24

3.5 A rendered image from the textured CT model. 25

4.1 A brief illustration of camera motion between two consecutive poses. Blues points represents reprojected points from the CT model, red camera frames represent the estimated pose $\hat{C}_{k-1}$ and $\hat{C}_k$ at keyframe $k-1$ and k from prior motion, and the black frame represents optimized pose C_{k-1}. At keyframe k we employ image registration and PnP/BA to determine T_{k-1}, aligning the reprojected points with image feature points. Then the refined pose is transformed by another prior motion $T_{k-1,k}$ to obtain a new prior estimated pose $\hat{C}_k$. 27

4.2 Illustration of the smoothness constraint. As each back-projected point is isolated, we select a 15-by-15 ROI (green square) around the point (blue) of interest as a neighborhood and combine all the back-projected points in the neighborhood for least squares. The solution (u_i, v_i) after smoothness constraint calculation is assigned to be searching direction of the back-projected point. 30

4.3 Result of image registration. Red points means the back-projected points have the same intensity as the original points. Blue arrows show the registration between the back-projected points and its corresponding points along the searching direction. Each pair of these points connected by the arrow has the same intensity. Green points means that these points cannot find any point with same intensity given a certain searching direction and step size. Red and green points will be discarded, only blue pairs will be used for pose refinement. 31

4.4 Epipolar geometry 33

5.1 A pose-graph representation of the motion estimation process. Every node in the graph corresponds to a pose of the endoscope. Only poses are connected by edges while the 3D points in the CT model are discarded. These edges represent the transformation obtained from motion estimation. 38

6.1 Back-projected points selection. Blue arrow means there is a valid movement between the back-projected points and corresponding points. Red points means there is no movement and green points means there is no corresponding point under certain distance. The red and blue will be discarded in each iteration. 41

6.2 The Euclidean distance of the estimate pose to the ground truth. 42

6.3 The Euclidean distance of the estimate pose to the ground truth in each direction. 43

6.4 The Euclidean distance of the estimate pose to the ground truth in each angle axis. 44

6.5 The Euclidean distance of the estimate pose to the ground truth in each direction. 46

6.6 The Euclidean distance of the estimate pose to the ground truth in each angle axis. 47

6.7 Visualization of improvements of each iteration. The back-projected points are shifting left to align with the original image. The total magnitude decreases and more points are aligned. Two images coincide in the end as suggested by their contours. 48

6.8 Iteration performance when the energy is no longer decreased. 49

6.9 Error statistics of the estimated pose given a random perturbation in a certain range. The horizontal axis represents the magnitude of the perturbation, and the vertical axis represents the error in Euclidean distance to the ground truth. 50

Chapter 1

Introduction

Estimating the ego-motion of a mobile robot remains a hot topic in Robotics and Computer Vision over the decades, and one challenging problem is to compute six degree-of-freedom motion from video stream or sequences of images [2, 16, 17, 33, 46, 47, 49]. Determining the ego-motion of an agent with a camera is called Visual Odometry (VO) [38, 44]. If a map of nearby environment is built simultaneously, this process is referred as Visual Simultaneous Localization and Mapping (V-SLAM) [7, 11, 25]. The goal for vision-based motion estimation is to obtain the highest positioning accuracy and robustness with a considerable cost. Recent state-of-the-art systems are capable of running on mobile devices with plausible accuracies [13, 15].

1.1 Functional Endoscopic Sinus Surgery

VO and V-SLAM have wide applications in Robotics and automotive industry as well as in surgical treatments. Functional Endoscopic Sinus Surgery (FESS), as an effective surgical procedure for sinus disease which is performed over 250,000 times annually in the United State, is to treat sinonasal pathology such as chronic sinusitis [6, 27]. It has a long history which dates back to the 1980s after the pioneering work of Messerklinger and Kennedy. [12]. Navigating an endoscope in nasal structures is difficult due to the complexity of the anatomy. The complication associated with endoscopic sinus procedures is approximately 6-8%, with about 1% of them result in major complications [10, 24]. Commercial navigation systems only guarantee a positioning error of 2mm while the state-of-the-art systems may limit the error under 1mm [26, 33]. However, the error is still large compared to the nasal pathways.

Similar to VO or V-SLAM, navigating endoscope in nasal structures also requires accurate position and orientation estimation and tracking of the endoscope. Well-established endoscope navigation methods use sequences of endoscope images, and implement Structure from Motion (SfM) for the endoscopic position and orientation estimation. One recent method incorporates both video streams and Computed Tomography (CT) models, and register reconstructed model from SfM to the prescanned CT model using Iterative Closet Points (ICP)-based 3D registration. Another alternative renders virtual images from the CT model, and compare the rendered image with real images for pose estimation. However, several practical issues still exist and are required to be addressed before they reach clinical acceptance. First, navigation methods need to be robust to various tissue surface appearances and anatomy distortions. Second, ICP related methods require high computational power and good initial registrations, while rendering-based methods waste too much power in virtual rendering.

1.2 Simultaneous Localization and Mapping

Traditional VO or V-SLAM methods are normally feature-based, which involve feature detection, extraction and registration across frames. Feature detection finds local features such as corners, blobs or more general features, while feature extraction computes the feature descriptors that distinguish each feature [45, 50]. When detectors and descriptors are obtained, feature matching is usually solved in two ways. For rich feature descriptors, either they are matched by brute-force methods, which each descriptor is compared with all others for the best match, or tree-based and vocabulary-based sorting methods such as Fast Library for Approximate Nearest Neighbor (FLANN) by K-Means tree [22, 32, 39]. Other descriptors employ optical flow and search for correspondence by identifying a step size and a direction [5, 30]. Ideally, when feature correspondences are established, camera motion can be easily estimated with epipolar geometry.

A typical problem associated with feature-based methods is the computational complexity caused by feature description and matching, as well as the necessity for robust outlier romoval techniques such as RANSAC and M-estimators [1, 31]. One alternative for feature-based VO or V-SLAM is called direct method, which operates directly on pixel-level intensities and uses intensity differences as a measurement of similarity [19, 36]. Feature-based methods minimize reprojection errors, which is a sum of Euclidean distances of each feature pair. This process is called Bundle Adjustment [51]. Direct methods minimize photometric error between corresponding pixels at same location directly in sensor space. The great advantage of this method is that it requires no prior data association hence largely reduce the computational costs caused by feature extraction, matching and outlier romoval. However, direct methods are still expensive, as they require dense or semi-dense reconstruction of the environment for joint optimization. One recent alternative is called hybrid method or semi-direct method, which combines the advantage of both indirect and direct methods. One example of hybrid method extracts blobs as features, optimize photometric error of the selected features. This method avoids the expenses of both feature description, matching and prior data association, and achieve comparable accuracy and robustness.[15]

1.3 Outline

This work is organized as follows. Chapter 1 starts with the introduction to Functional Endoscopic Sinus Surgery, VO and V-SLAM, motion estimation methods, and motivation and challenges for the work. Chapter 2 presents the endoscopic navigation system in general, as well as the terminologies and notations used in this work. It also describes the fundamental knowledge of computer vision, kinematics and optimization that constantly used in different chapters. Chapter 3 demonstrates the CT texture mapping method and shows the result for visualization and virtual image rendering. Chapter 4 illustrates the motion estimation method which is the core chapter for this work. Chapter 5 describes the optimization method for the system and chapter 6 shows the experiment results under different situations. In the end, chapter 7 draws the conclusion and proposes future works.

1.4 Motion Estimation

1.4.1 2D-2D Motion Estimation

The objective for the front end module of VO and SLAM is to estimate the camera motion across frames. Recover camera positions and orientations with a sequence of images is generally divided into the following three classes.

1.4.1.1 Feature-based Methods

The standard procedure for feature-based motion estimation is shown by the following pipeline:

1. Input image sequences
2. Feature detection
3. Feature description
4. Feature matching/tracking
5. Motion estimation
6. Local optimization

Feature detection tracks high frequency textures such as corners and blobs [18], then describes these features with local information such as local gradients. Popular feature descriptors include SIFT [29], SURF [4], HOG [9] and ORB [43]. Feature matching is a procedure that matches features to create either 2D-2D, 3D-2D or 3D-3D correspondences. The correspondences are typically involved in the computation of the homography or transformation T_k, decomposition of R_k and t_k, and the computation of concatenate transformation $C_k = C_{k-1}T_k$. Where T_k, R_k and t_k are the homogeneous transformation, rotation and translation of the camera and C_k is the camera pose at keyframe k. For erroneous correspondences, outlier removal techniques such as RANSAC[1] and M-estimators[31] are implemented to find salient matches.

1.4.1.2 Direct Methods

Direct methods estimate motion directly by minimizing photometric error - an error measure that is based on the pixel-level intensities of two consecutive images. Compared to feature-based methods that only consider the distance to feature location, direct methods inherit the idea from optical flow, use local intensity gradient in the optimization framework. Pixel correspondences are given directly by the geometry of the problem, eliminating the need for robust data association.

Direct methods are well-established for monocular, RGB-D and stereo cameras [8, 20, 23]. For dense direct approaches, examples include DTAM [37, 48], REMODE [41] and LSD-SLAM [14]. In a complementary manner, for Sparse direct approaches, the formulation is proposed by Jin et al. [21] and DSO [13].

There are two major assumptions for direct methods which come from optical flow: brightness constraint, which assumes that the illumination of the image is constant and the motion is small, so the pixels are trackable across consecutive images; and smoothness constraint, which assumes that in a small neighborhood of an image which is usually called a patch, all the pixels inside the patch have similar movements. The optical flow problem can be formulated in different ways. Bruce D. Lucas and Takeo Kanade interpret the assumptions into two equations, where the detailed math can be found in Chapter 4.2. It is regarded as a local estimator of optical flow. The Horn-Schunck method is a global estimator as it formulates the optical flow problem in one objective function as:

$$E = \iint \left[(I_x u + I_y v + I_t)^2 + \alpha^2(\|u\|^2 + \|v\|^2)\right] dxdy \tag{1.1}$$

Where E is the global energy which the function seeks to optimize. The intuition behind the formulation is the overall pixel intensity difference can be minimized by finding an optimal to the first part of equation, at the same time imposing a penalty on the sum total movement u and v, as seen in the second part, to satisfy the assumptions.

1.4.1.3 Hybrid Methods

One recent hybrid method proposed by Scarramuza et al. is called SVO [15]. It uses a direct formulation called sparse image alignment to obtain correspondences, before switching to an indirect formulation for model optimization.

1.4.2 3D-2D Motion Estimation

Most existing 3D-2D motion estimation systems are rendering-based. By rendering a virtual image from the CT model using a graphical engine, it is possible to estimate motion by comparing the virtual image with the video stream obtained from the endoscope [40]. It is achieved by utilizing photometric rendering from the CT model following by analyzing the similarity metric between rendered image and video. Prior to these, Mori et al. [34]

first adopt optical flow to find the homography between two consecutive real images, then using the homography to render a virtual image from the CT model. Finally the cross-correlation is computed to refine the pose. Later on, Robu et al. [42] extend this work with another similarity measure - discriminative structural similarity (DSSIM), to compensate for illumination changes and tissue distortions.

1.4.3 3D-3D Motion Estimation

Most navigation systems that implements 3D-3D registration for motion estimation utilize the variations of ICP. Leonard et al. extract features with SURF plus hierarchical multi-affine (HMA) algorithm, and further apply SfM to generate 3D model [26]. The reconstructed 3D model is registered with the prescanned CT model for position and orientation estimation using trimmed-ICP. (TriICP) Billings et al.[6] implement another variation of ICP called Iterative Most-Likely Point Registration (IMLP) to find point correspondences, then apply similarity transform in point registration step. Instead of directly obtaining the endoscope pose after 3D-3D registration, Mirota et al.[33] track the 2D features using adaptive scale kernel consensus (ASKC) [52] and further calculating 2D-3D correspondences.

1.5 Motivation and Challenge

The best way to obtain a high navigation accuracy is to make fully use of the prescanned CT model of the patient, and incorporate the model with the real time video stream provided by the endoscope. If the CT model is textured, the back projection of the model vertices can be regarded as feature points. This could avoid the computation complexity caused by feature detection and feature extraction. Since each vertex of the CT model is assigned a unique index and a RGB value, feature matching can be greatly simplified by utilizing the indices of the potential points to trace the vertices, and comparing the intensities for determining correspondences.

One major challenge associated with the idea is the variation of illumination. When the endoscope moves inside the sinus cavity, the lighting condition might change with the shifting of the light source. This may result in the change of surface colors or intensities in different time stamps. To solve this problem, the illumination need to be normalized, and the registration algorithm should also be designed to be robust to lighting condition changes.

Another challenge is associated with the deformation of sinus structure during surgery. The sinus cavity is not static during the surgery by virtue of breathing and the surgical operation. The reflection of deformations on the video stream will lead to inaccurate registrations between the CT model and the endoscopic video, hence outputting an inaccurate transformation estimate. One possible solution is to segment the model and to select regions that are stationary. If the algorithm is able to run in a high frequency, the position of the 3D vertices shall also be adjusted adhere to the deformation in a mapping process.

Chapter 2

Overview

In this chapter, the system architecture is introduced. The terminologies and notations are then described following by the basics of camera calibration, kinematics and optimization that are used different parts of this thesis.

2.1 System Architecture

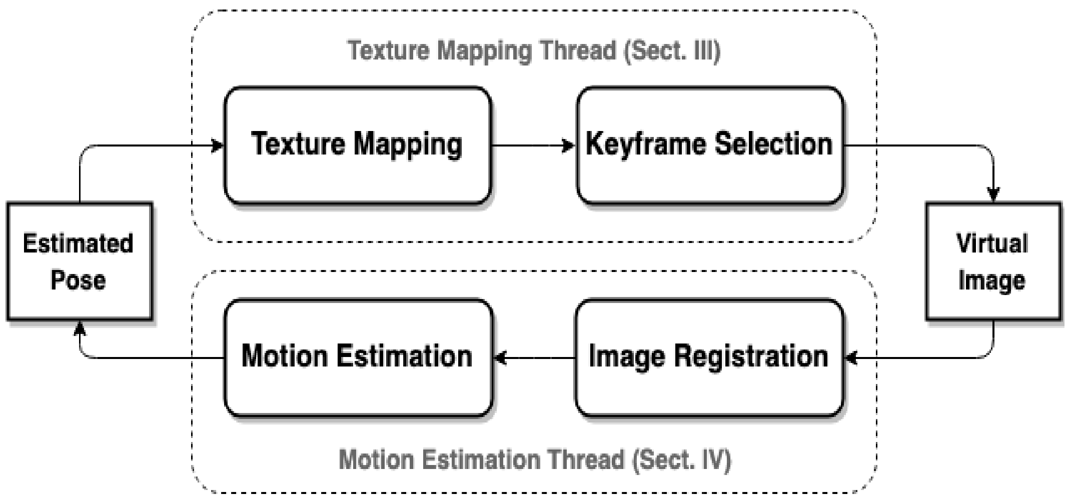


Figure 2.1: Back-projection texture mapping pipeline

Figure 2.1 provides an overview of the proposed endoscopic navigation system for sinus surgery. A complete VSLAM system generally contains five modules: Sensor Input, Front End VO, Back End Optimization, Loop Closing and Mapping. Sensor Input module reads and pre-processes raw sensor data from cameras and other sensors. The front end of a VSLAM system estimates the relative motion between two adjacent frames as well as determining the structure of the local map. This is typically referred as visual odometry. Back End Optimization takes different poses from VO and Loop Closure Detection as well as the 3D scene vertices, and optimizes all these parameters to get a globally consistent trajectory and a map. Loop closing detects if the robot or camera comes back to one of its previous location. If it has visited the same place, the loop closing module will send the information to the back end for trajectory refinement. Finally, Mapping establishes a corresponding 3D scene, or a map, with respect to the current trajectory.

As our system directly takes the prescanned CT model as the scene model, the Mapping module is no longer required. Our system has two threads - a Texture Mapping Thread and a Motion Estimation Thread. The texture mapping thread continuously takes the estimate of the current pose, and texture maps the CT scene model. When new images are received, the keyframe selection module will choose a frame of interest from a set of frames. This reduces the cost of computation and storage. When a new keyframe is selected, it renders a virtual image from the scene using the prior estimated pose and sends it to Motion Estimation Thread. In addition to updating the global scene model and publishing a virtual rendered image, it also publishes all corresponding indices of back-projected points in the rendered image. This allows the Motion Estimation Thread to trace the 3D position of the corresponding vertex of each back-projected point as well as its RGB value.

Once the rendered image and indices of back-projected points are received, the Motion Estimation Thread takes the rendered image and the current keyframe image, and registers the rendered image with the current endoscopic image using the method described in chapter 4 to establish 2D-2D correspondences. As the indices for each back-projected point can be exploited to obtain the original 3D vertex position, the 3D-2D correspondences are also established simultaneously. The correspondences are then sent to Motion Estimation module and new pose is hence estimated. Each newly estimated pose will be served as the prior motion estimate for the next iteration and be further used in the next estimation step.

As our system makes use of raw CT model as a reference map, no Mapping module is further needed. On the contrary, assuming a reasonably good initial endoscope position with registration error less than 1mm, the CT model is first texture mapped by the initial endoscopic image at the beginning of the process to start the iteration of pose estimation.

2.2 Terminology

Back-projection, or reprojection is the step of projecting the 3D vertices onto the image plane. Pose is an abbreviation for position and orientation which is often used in computer vision.

In this work, the prescanned CT model is referred as the model or the scene model for simplicity. The 3D points stored representing the model are called 3D vertices. The back-projected points refer to the 2D pixels on the image plane that are projected from 3D vertices. In general, each back-projected point has a 2D position in pixel, an index which is the same as the original 3D vertex, a 3D position that can be found using the index, and a RGB value also from the 3D vertex. For a back-projected point at pixel (u, v), the original pixel (u, v) in the image is called the original point. Most of the back-projected points are registered to the points called corresponding points or photometric equivalences. They are photometric equivalent, or equivalent in terms of intensity as they are registered by checking if their intensities are the same. If the intensities are the same, then they are aligned, otherwise they are misaligned and there will be an optical flow the point. The flow contains a direction and magnitude.

2.3 Notation

The position and orientation of each camera at keyframe k is defined as $\mathbf{C}_k$, while the estimate of the camera pose after prior motion is $\hat{\mathbf{C}}_k$. The rigid body transformation between frames $k-1$ and k in homogeneous coordinates is defined as $\mathbf{T}_{k-1,k} \in SE(3)$, where the rotation is $\mathbf{R}_{k-1,k}$ and the translation is $\mathbf{t}_{k-1,k}$. All the camera motion are calculated with respect to the world coordinates. For the transformation at the same keyframe k, the notations are simplified as $\mathbf{T}_k$, $\mathbf{R}_k$ and $\mathbf{t}_k$.

For each camera pose $\mathbf{C}_k$ at keyframe k, the associated image intensity for the endoscopic image is defined as $I_k^c : \Omega^c \subset \mathbb{R}^2 \mapsto \mathbb{R}$ and the back-projection image intensity that obtained directly from the back-projected CT model is defined as $I_k^m : \Omega^c \subset \mathbb{R}^2 \mapsto \mathbb{R}$, where Ω^2 is the image domain. The reprojection image contains only sparse points back-projected from CT model with its own intensity. The image coordinates for each back-projected point at keyframe k is defined as $\mathbf{u}_{i,k}^m(u_{i,k}^m, v_{i,k}^m) \in \mathbb{R}^2$ and the photometric equivalence, or the corresponding image feature point after registration is represented as $\mathbf{u}_{i,k}^c(u_{i,k}^c, v_{i,k}^c) \in \mathbb{R}^2$, where i is the point index. The intensity for a single pixel i is hence $I_{i,k}^m$, $I_{i,k}^c$ for the back-projected point and the original point intensity respectively. In addition, the position of a 3D vertex in the CT model with the same index i in the world coordinates is defined as $\mathbf{p}_i(x_i, y_i, z_i) \in \mathbb{R}^3$.

To project 3D vertices in world coordinates into image coordinates, the camera projection matrix at time k is defined as $\mathbf{K}_k$, hence $s\mathbf{u}_{i,k}^c = \mathbf{K}_k\mathbf{p}_i$, where s is the scale factor.

2.4 Fundamentals

2.4.1 Camera Model and Calibration

In order to transform the real world data onto the image plane, the position and orientation of the camera relates to the world coordinates need to be acquired as well as the projection between the objects in 3D space and 2D images. In computer vision, the transformation between world coordinates and camera coordinates is called Extrinsic Parameters, and the transformation from the camera coordinates to the image plane is called Intrinsic Parameters.

For a pinhole camera, the Intrinsic Parameters are defined as a 3×3 matrix as:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

where f_x, f_y are the focal lengths in x and y axis of the camera, and c_x, c_y are the principal points. The Extrinsic Parameters are defined as:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.2}$$

where R is the rotation matrix, and t is the translation vector

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}, \quad t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.3}$$

So to transform a 3D vertex $[x, y, z]^T$ in the world coordinates to a 2D pixel $[u, v, 1]^T$ in the image coordinates, we combine the formulas with augmentation as

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KT = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.4}$$

where s is a normalizing constant. Intuitively, the projection process first transforms a 3D point in world coordinates into camera coordinates by using the camera Extrinsic, and then projects the 3D point in camera coordinates onto the 2D image plane.

Due to the imperfection in lens manufacture and misalignment of lens and Charge-coupled Device (CCD) in camera assembly, the image obtained by the CCD is usually distorted. To remove lens distortions, we applied the following formula:

$$u_c = u(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.5}$$

$$v_c = v(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.6}$$

for the radial distortion, and

$$u_c = u + [2p_1 uv + p_2(r^2 + 2u^2)] \tag{2.7}$$

$$v_c = v + [2p_2 uv + p_1(r^2 + 2v^2)] \tag{2.8}$$

for the tangential distortion.

So for a pixel at $[u, v]^T$, the position of its undistorted corresponding pixel is noted as $[u_c, v_c]^T$, and the distortion model has 5 parameters $(k_1, k_2, p_1, p_2, k_3)$.

2.4.2 3D Geometry and Transformation

2.4.2.1 Homogeneous Representation and Lie Group

Every 3D rotation can be represented by a 3×3 matrix

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \tag{2.9}$$

if it satisfies the following property:

1. $R^T R = RR^T = 1$

2. det(R) = 1

Any matrix satisfies the above property is special orthogonal, denoted as SO(3)

$$SO(3) = \left\{ R \in \mathbb{R}^{3\times 3} : RR^T = I, det(R) = 1 \right\} \tag{2.10}$$

SO(3) is a group using matrix multiplication as the group operation and the identity matrix I as the identity element. To extend the idea of group into general rigid body transformations, we denote the special Euclidean group as

$$SE(3) = \left\{(t, R) : t \in \mathbb{R}^3, R \in SO(3)\right\} = \mathbb{R}^3 \times SO(3) \tag{2.11}$$

where t is the translation of the transformation and R is the rotation. An element of the special Euclidean group is called the homogeneous representation of transformation which is denoted as:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.12}$$

It is easy to check that $T \in SE(3)$ forms a group under matrix multiplication with the 4×4 identity matrix as the identity element. The inverse is denoted as:

$$T^{-1} = \begin{bmatrix} R^T & -R^T t \\ 0 & 1 \end{bmatrix} \tag{2.13}$$

2.4.2.2 Euler Angles

Euler Angles is one of the most intuitive ways to represent a 3D rotation of the rigid body with respect to a fixed coordinate frame. It is defined as the combination of $\alpha, \beta,$and γ. α is angle between x-axis and N-axis, β is the angle between z-axis and Z-axis, γ is the angle between N-axis and the X-axis.

$$R_x(\gamma) \doteq e^{\hat{x}\gamma} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix} \tag{2.14}$$

$$R_y(\beta) \doteq e^{\hat{y}\beta} = \begin{bmatrix} cos\beta & 0 & -sin\beta \\ 0 & 1 & 0 \\ sin\beta & 0 & cos\beta \end{bmatrix} \tag{2.15}$$

$$R_z(\alpha) \doteq e^{\hat{z}\alpha} = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

There are many conventions for the combination of $\alpha, \beta, and \gamma$ such as z-x-z, x-y-x, y-z-y, z-y-z, x-z-x and y-x-y. For instance for z-x-z convention, the rigid body first rotates around z-axis for an angle α, then y-axis for an angle β, and finally z-axis for an angle γ. To find the rotation of frame A with respect to frame B, using the above convention

$$R_{ba} = R_z(-\gamma)R_y(-\beta)R_z(-\alpha) \tag{2.17}$$

One application for Euler Angles is yaw, pitch and roll (x-y-z) which are usually used for describing the orientation of an aircraft. Yaw represents the bearing angle, pitch represents the elevation and roll represents the bank angle.

However, Euler Angle representation is always associated with a drawback called Gimbal Lock, which means a loss of one degree of freedom in a 3D system. Gimbal Lock happens when two axis are parallel or colinear, which introduces a singularity into the system causing the degree of freedom to drop.

2.4.2.3 Axis-angle Representation

Axis-angle representation is a more general form of representing a 3D rotation. It uses a unit vector as rotation axis and describes the change of orientation by an angle θ. Let $\omega \in \mathbb{R}^3$ be a unit vector which specifies the direction of rotation, and the angle θ , we can find the relationship of the 3×3 rotation matrix and the angle-axis representation using exponential map:

$$
\begin{aligned}
R(\omega,\theta) &= e^{\hat{\omega}\theta} \\
&= I + \theta\hat{\omega} + \frac{\theta^2}{2!}\hat{\omega}^2 + \frac{\theta^2}{3!}\hat{\omega}^3 + \cdots \\
&= I + \hat{\omega} sin\theta + \hat{\omega}^2(1 - cos\theta) \\
&= \begin{bmatrix} \omega_1^2 v\theta + c\theta & \omega_1\omega_2 v\theta - \omega_3 s\theta & \omega_1\omega_3 v\theta + \omega_2 s\theta \\ \omega_1\omega_2 v\theta + \omega_3 s\theta & \omega_2^2 v\theta + c\theta & \omega_v\theta + c\theta \\ \omega_1\omega_3 v\theta - \omega_2 s\theta & \omega_2\omega_3 v\theta + \omega_1 s\theta & \omega_3^2 v\theta + c\theta \end{bmatrix} \\
&= \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}
\end{aligned}
\tag{2.18}
$$

where $v\theta = 1 - cos\theta, s\theta = sin\theta, c\theta = cos\theta$. If an rotation matrix is given, the angle can be found by

$$\theta = cos^{-1}(\frac{trace(R) - 1}{2}) \tag{2.19}$$

if $\theta \neq 0$, we choose

$$\omega = \frac{1}{2s\theta}\begin{bmatrix} r_8 - r_6 \\ r_3 - r_7 \\ r_4 - r_2 \end{bmatrix} \tag{2.20}$$

2.4.2.4 Quaternions

To avoid the Gimbal Lock problem, another global parameterization of SO(3) is called the quaternions. Quaterions avoid the Gimbal Lock problem by generalizing complex numbers

$$Q = w + xi + yj + zk,\ [x,y,z]^T \in \mathbb{R}, i = 0, ..., 3 \tag{2.21}$$

where w is the scalar component of the quaternions, and $\vec{q}$ is the vector component. The relation of quaternions and a rotation matrix can be found as:

$$R = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2zw & 2xz+2yw \\ 2xy+2zw & 1-2x^2-2z^2 & 2yz2xw \\ 2xz-2yw & 2yz+2xw & 1-2x^2-2y^2 \end{bmatrix} \tag{2.22}$$

2.4.3 Non-linear Optimization

Nearly all computer vision problems can be formulated as optimization problems. An optimization problem is a problem of searching for the best solution, or the optimizer, that fulfills certain criteria. For instance for a nonlinear least squares problem.

$$min\frac{1}{2}\left\|f(x)\right\|_2^2 \tag{2.23}$$

To optimize an arbitrary nonlinear function f and to find the minimizer for the above equation, we can take the derivative with respect to x

$$\frac{df}{dx} = 0 \tag{2.24}$$

the solution for the equation will be the optimizer. In order to find the minimizer of a function, the optimization process usually starts with an initial value x. For the k-th iteration, we need to find an increment $\triangle x_k$ that makes $\|f(x_k + \triangle x_k)\|_2^2$ to be minimized. If $\triangle x_k$ is small enough, then the iteration stops, otherwise let $x_{k+1} = x_k + \triangle x_k$ and repeat the optimization step.

To find the best parameter, there are two major types of optimization methods - Line Search Method and Trust Region Method. Line search method first finds a search direction, then determines the step length in that direction. Trust Region method first selects a region of interest, then finds the optimal point inside the region.

2.4.3.1 Line Search Methods

One example of a Line Search Method is called Newton's Method. Newton's method first expands the objective function using Taylor Expansion.

$$\|f(x) + \triangle x\|_2^2 \approx \|f(x)\|_2^2 + \frac{1}{2}\triangle x^T H \triangle x \tag{2.25}$$

where J and H are the Jacobian and Hessian of the objective function. If only the first order derivative needs to be taken into consideration, the increment can be calculated as

$$\triangle x^* = -J^T(x) \tag{2.26}$$

Intuitively, this means each optimization step searches along the negative gradient of the objective function to maximize the increment. If we use both first order and second order derivative, then the increment becomes

$$H\triangle x^* = -J^T(x) \tag{2.27}$$

It has higher convergence rate over linear methods. However, calculating the inverse of the Hessian is computationally inefficient, so the Hessian is usually approximated by a combination of Jacobian

$$J(x)^T J(x)\triangle x = -J(x)^T f(x) \tag{2.28}$$

This method is called Gauss-Newton Method and it is often used in SLAM applications. But the disadvantage is that it is not guaranteed to converge even to a local minimum.

2.4.3.2 Trust Region Methods

Though Gauss-Newton Method has quadratic convergence rate, the algorithm is unstable if the initialization is closed enough to the ground truth. It is natural to think about adding a trust region for the optimization to guarantee its convergence. Inside the region the optimization is valid and otherwise the region needs to be adjusted.

The trust region can be selected based on the difference between the change of our approximated model and the change of the objective function for iteration. If the difference is small, the radius of the trust region could be increased to allow bigger step size, otherwise it is decreased to guarantee the decline of error. In practice, we can use

$$\rho = \frac{f(x + \triangle x - f(x))}{J(x)\triangle x} \tag{2.29}$$

to determine if the Taylor Expansion well approximates the objective function. If ρ is close to 1, then the approximation is valid. If ρ is too small, then it is invalid and the trust region needs to be compressed.

Then we can incorporate the region by transforming the constrained optimization problem into an unconstrained optimization using Lagrange Multiplier:

$$\min_{\triangle x_k} \frac{1}{2} \left\| f(x) + J(x_k)\triangle x_k \right\|^2 + \frac{\lambda}{2} \left\| D\triangle x \right\|^2 \tag{2.30}$$

where D can either be an identity matrix, or the square root of diagonal elements of $J^T J$. By taking the derivative with respect to x, the increment function becomes

$$(H + \lambda D^T D)\triangle x = -J(x)^T f(x) \tag{2.31}$$

or

$$(H + \lambda I)\triangle x = -J(x)^T f(x) \tag{2.32}$$

if D is chosen to be the identity matrix. This method is called Levenberg-Marquad Method. It leverages the Gauss-Newton Method and gradient descent method. When λ is small which means the Hessian is dominant, the Levenberg-Marquad is more similar to the Gauss-Newton. Otherwise if λ is small, it is more similar to the gradient descent.

Chapter 3

CT Texture Mapping

3.1 Texture Mapping Pipeline

Texture mapping is a process of shading an object not only with solid colors, but also a textured pattern or an image. A standard texture mapping pipeline contains: a vertex puller module, which retrieves all the vertices; a geometry processing module, which performs the camera transformation, lighting, projection, clipping and window viewport transformation; and a rasterization module, which linearly interpolates the data to access the color.

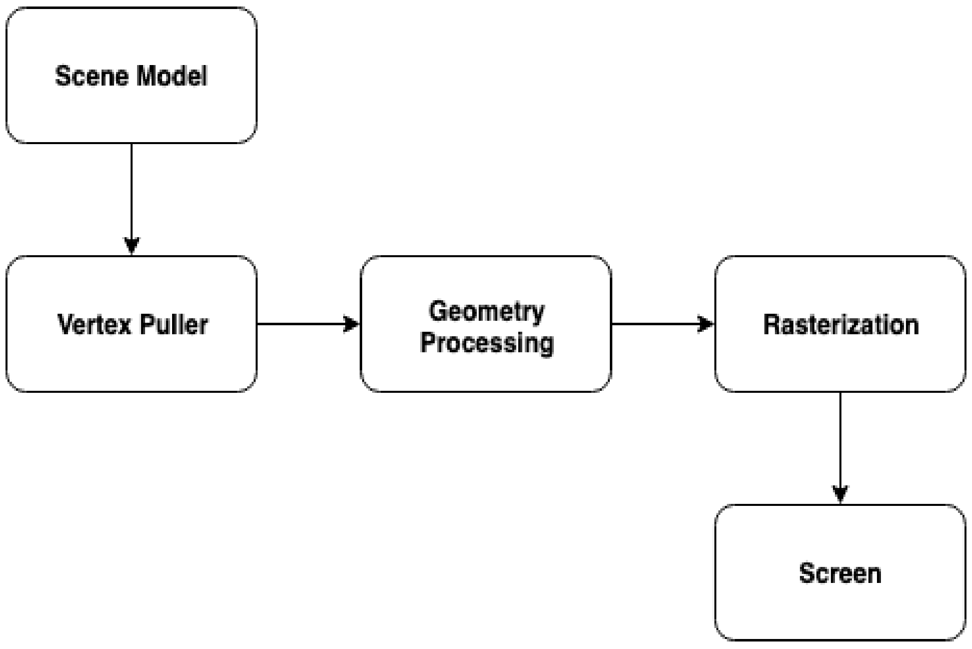


Figure 3.1: Standard texture mapping pipeline

3.2 Back-projection Texture Mapping

We implement the back-projection texture mapping pipeline as follows:

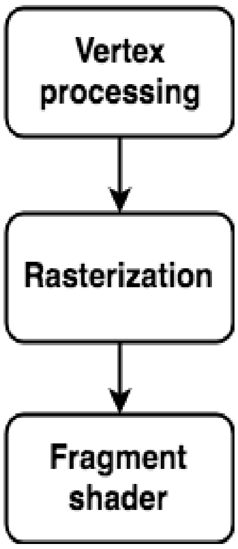


Figure 3.2: Back-projection texture mapping pipeline

The CT model vertices are stored in an ordered list with each vertex having a distinct index. Therefore, the vertex specification and tessellation are not included in this pipeline. The pipeline only contains vertex processing, rasterization and fragment shader.

The basic operating unit for texture mapping is called a primitive, the boundary of which is defined by each vertex. The vertex processing includes clipping and culling. Clipping means that primitives lying on the boundary between the inside of the viewing volume and the outside are split into several primitives, keeping visible primitives lying inside the volume. Triangle primitives can be culled to avoid rendering triangles facing away from the viewer.

Rasterization is to determine which pixels are drawn into the framebuffer, then interpolates parameters such as colors and texture coordinates for further processing. The result of rasterizing a primitive is a sequence of fragments. The state for a fragment includes its position in screen-space and depth information which is called Z-Buffer. When transforming each 3D

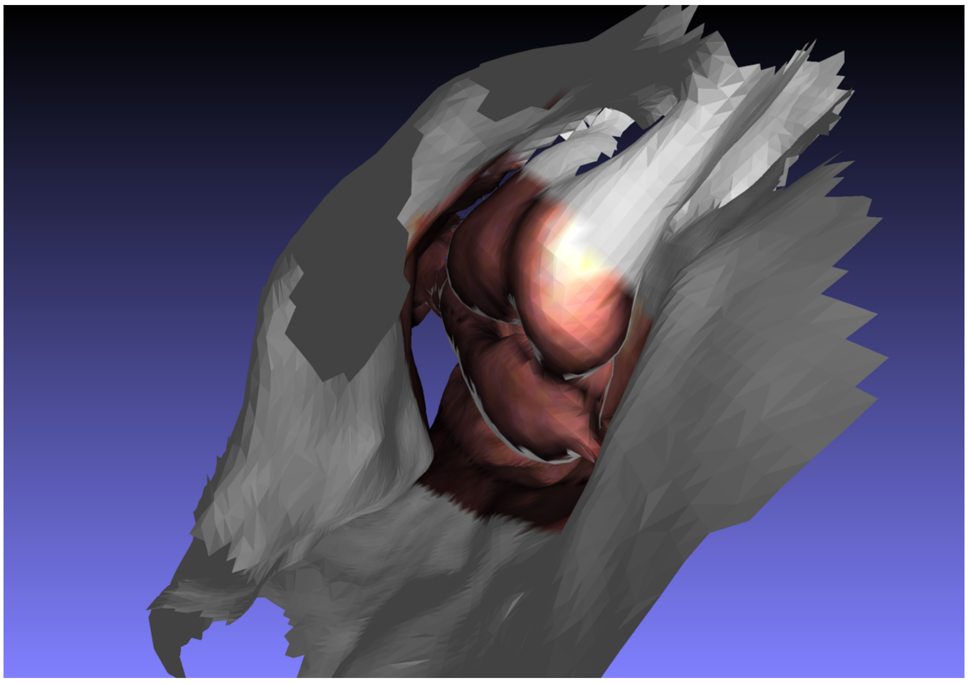


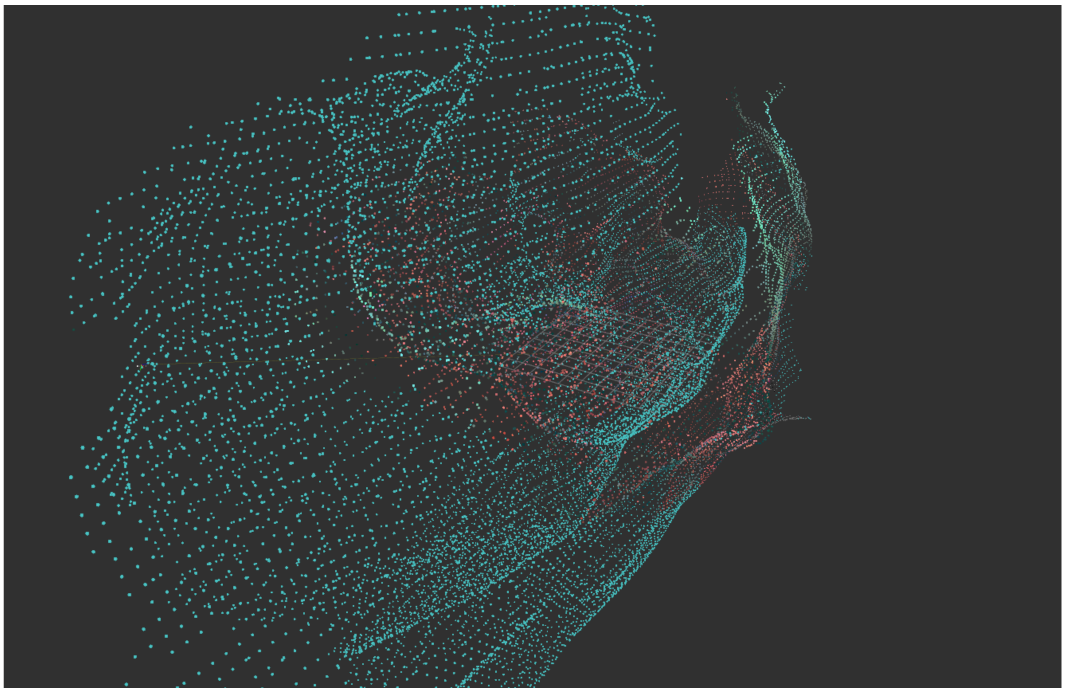
Figure 3.3: A textured CT model back-projection texture mapping. Gray vertices in the model means either the vertex is not seen by the camera, or it is occluded, others are shaded with color from endoscopic image.

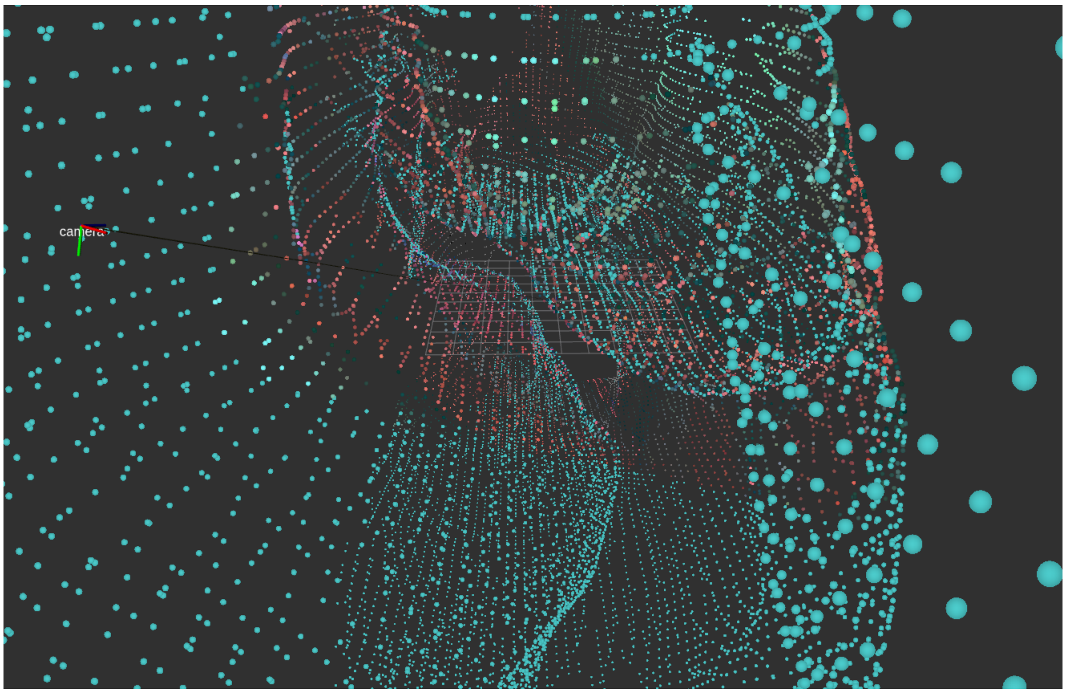
vertex from world coordinates to camera coordinates or image coordinates, Z-Buffer keeps track of the depth information of each individual vertex (*z*-axis value in camera coordinate), and retains only the fragment with the smallest depth value. The nature of Z-Buffer method is when more than one vertices fall onto the same pixel location on the image plane, the nearest one is kept as the farther ones are occluded.

Vertex shader performs basic color processing for each vertex. For vertices with smallest Z-Buffers, the corresponding 2D image pixels are shaded with the vertex intensities with proper interpolation.

3.3 Image Rendering

Image rendering is a process of generating photorealistic images from a 3D scene. The ultimate goal for computer graphics is to transform the 3D world objects into 2D images, which means to create images or films from computer-generated models. Rendering is a way to display such a model.


(a) Side view of textured sinus structure in RViz



(b) Front view of textured sinus structure in RViz

Figure 3.4: A real time visualization of textured point cloud in RViz

Figure 3.5: A rendered image from the textured CT model.

The image rendering pipeline also contains vertex processing, primitive assembly, rasterization and fragment shader which is nearly identical to the back-projection texture mapping pipeline. This allows us to employ the previous pipeline to generate a rendered image. See figure 3.5

Chapter 4

Motion Estimation

In this section, we describe the proposed textured CT-based motion estimation method for endoscopic navigation. The major assumptions are brightness constancy and smoothness constancy, which are also the major assumptions for optical flow and direct methods.

Our key idea is, given a high frequency but relatively noisy prior motion estimate, camera poses can be optimized by exploiting the CT model. One reasonable way is to use Perspective-n-Points or Bundle Adjustment, which requires a registration of 3D vertices and 2D pixels. This registration can be obtained from optical flow. See figure 4.1

4.1 Prior Motion Estimation

There is a necessity for motion prior in this method as the nasal pathway is considered to be poor featured, and the camera motion is considered fast given the scale of the sinus cavity. Moreover, the implementation of high frequency prior motion estimation can also fill the gap between each camera keyframe therefore enabling a smoothness tracking of the endoscope throughout the process. The prior motion can be obtained with high frequency measurement devices such as an EM tracker of the endoscope, an add-on Inertia Measurement Unit (IMU), Ultra-wide Band (UWB). It can be also obtained by other sensor-based or vision-based method real time motion

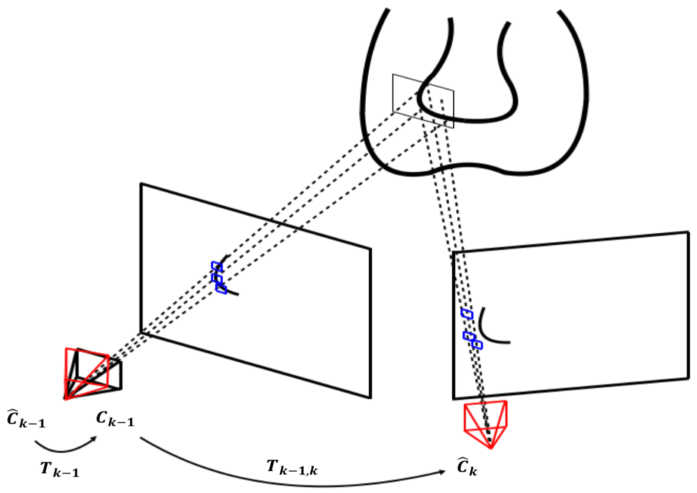


Figure 4.1: A brief illustration of camera motion between two consecutive poses. Blues points represents reprojected points from the CT model, red camera frames represent the estimated pose $\hat{C}_{k-1}$ and $\hat{C}_k$ at keyframe $k-1$ and k from prior motion, and the black frame represents optimized pose C_{k-1}. At keyframe k we employ image registration and PnP/BA to determine T_{k-1}, aligning the reprojected points with image feature points. Then the refined pose is transformed by another prior motion $T_{k-1,k}$ to obtain a new prior estimated pose $\hat{C}_k$.

estimation methods.

In this project, as the EM-tracker information is not available, we directly use the previous estimated pose as the prior motion estimate, and form an iterative estimation of movement. In other words, the estimated pose at time $t-1$ will be used to project the vertices onto the image plane at time t to estimate the pose at t.

4.2 Image Registration

The image registration algorithm is similar to the sparse direct approach, which uses Lucas-Kanade optical flow to track feature points. Given a 3D model, the first step is to project 3D vertices onto the image plane. These points after projection are referred as back-projected points. Due to the noise and uncertainty introduced by the motion prior, a minor deviation occurs that causes back-projected 3D points fall onto a pixel location away from their ground truth locations.

To map the back-projected points to the real corresponding pixels, at the same time avoiding the complexity of feature extraction and feature description, we applied a modified Lucas-Kanade optical flow to track the back-projected point. Starting from the back-projected point, by determining a searching direction and a magnitude, if any point along that direction is detected to have the same pixel intensity, this point is selected as the corresponding point and registered to the back-projected point. This means that the back-projected point "flows" or "shifts" from the original point to its photometric equivalence due to camera motion or noises. If there is no match of intensity within a certain searching distance, then the back-projected point is considered invalid and it is discarded. This normally happens in the region that has sharp intensity contrast such as corners and edges. There is also a case that some back-projected points have the same intensities as the original points on the image. This usually happens in flat regions and the points are also disregarded.

To determine the searching direction and magnitude, we formulate both brightness constraint and smoothness constraint. The brightness constraint can be computed using the image intensity of the original points $I_{i,k}$ at position $p_{i,k}$, and the intensity of back-projected points $I^m_{i,k}$ at the same pixel position at keyframe k as:

$$I^m_{i,k}(x_i, y_i, t-1) = I^c_{i,k}(x_i + u(x_i, y_i), y + v(x_i, y_i), t) \tag{4.1}$$

Using Taylor expansion, this can be written as:

$$I^m_{i,k}(x_i, y_i, t-1) \approx I^c_{i,k}(x_i, y_i, t) + I^c_x * u(x_i, y_i) + I^c_y * v(x_i, y_i) \tag{4.2}$$

Hence:

$$I^c_{i,k,x} u_{i,k} + I^c_{i,k,y} v_{i,k} + I_{t,i,k} \approx 0 \tag{4.3}$$

However, this equation is under-determined by using only one pair of corresponding points. The standard way to solve the equation is to apply another constraint, which is called the smoothness constraint, and then calculating the least squares solution. The smoothness constraint assumes that in the neighborhood of the pixel of interest, all pixels have an identical motion. Normally, the eight contiguous pixels in the 3×3 neighborhood of the feature point are selected together with the point.

$$\begin{bmatrix} I^c_{i,x} & I^c_{i,y} \end{bmatrix}_k \begin{bmatrix} u_i \\ v_i \end{bmatrix}_k = -I_{t,i,k} \tag{4.4}$$

Define:

$$A_k = \begin{bmatrix} I^c_{i,x,1} & I^c_{i,y,1} \\ I^c_{i,x,2} & I^c_{i,y,2} \\ \vdots & \\ I^c_{i,x,9} & I^c_{i,y,9} \end{bmatrix}_k, \quad b_k = \begin{bmatrix} I_{t,i,1} \\ I_{t,i,2} \\ \vdots \\ I_{t,i,9} \end{bmatrix}_k \tag{4.5}$$

Using Moore-Penrose pseudo inverse, the relative motion of a pixel can be computed as:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix}_k = -(A_k^T A_k)^{-1} A_k^T b_k \tag{4.6}$$

In this case, $I_{t,i,k}$ is the intensity difference of point i between original point intensity, $I^m_{i,k}$ and back-projected point intensity, $I^c_{i,k}$. $I_{i,x}$. $I_{i,y}$ are image derivatives in x and y direction calculated from endoscopic images. As back-projected points are sparse and isolated, which means no neighboring pixel intensity can be used for smoothness constraint. Hence, we adjust smoothness constraint by selecting a region of interest (ROI) and apply least square to all the reprojected points inside the ROI. See figure 4.2.

Once the relative motion $(u_{i,k}, v_{i,k})$ for a pixel is calculated, with the assumption of brightness constancy, the correspondence can be found by checking if any pixel along the direction $(u_{i,k}, v_{i,k})$ has the same intensity value. If, under a certain distance (usually 20 pixels), no pixel has the same intensity, then the point will be discarded. The result for registration is shown in figure 4.3.

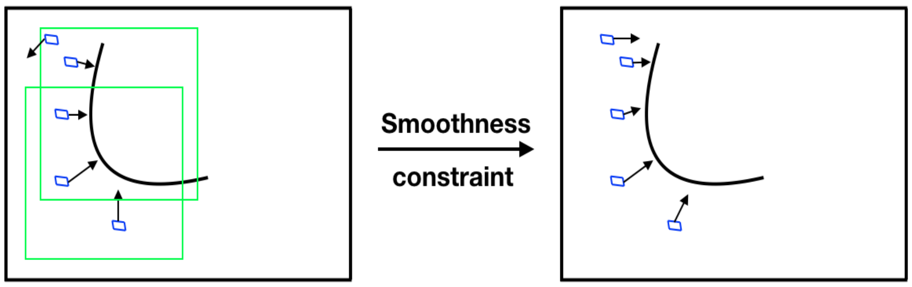


Figure 4.2: Illustration of the smoothness constraint. As each back-projected point is isolated, we select a 15-by-15 ROI (green square) around the point (blue) of interest as a neighborhood and combine all the back-projected points in the neighborhood for least squares. The solution (u_i, v_i) after smoothness constraint calculation is assigned to be searching direction of the back-projected point.

4.3 Outlier Removal

Random Sample Consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model. It is a counting-based method which counts the number inliers of a given model suggested by a subset of data to select the best model. The detailed steps are

1. Randomly select a subset of data to form a subset of hypothesis inliers.
2. Calculate the best model from the inliers
3. Test all other data using the model, count the number of total inliers that fit the model.
4. Resample the data and repeat step 1-4 until certain criteria are met.
5. Output the best model.

RANSAC-based rejection eliminates outliers based on thresholding of data. As it randomly selects a subset of data to approximate the real model, it is more robust to local minima and more effective when the data is noisy.

To determine the number of iterations needed for selecting the best model, we use the following formula

$$k = \frac{\log(1-p)}{\log(1-w^m)} \tag{4.7}$$

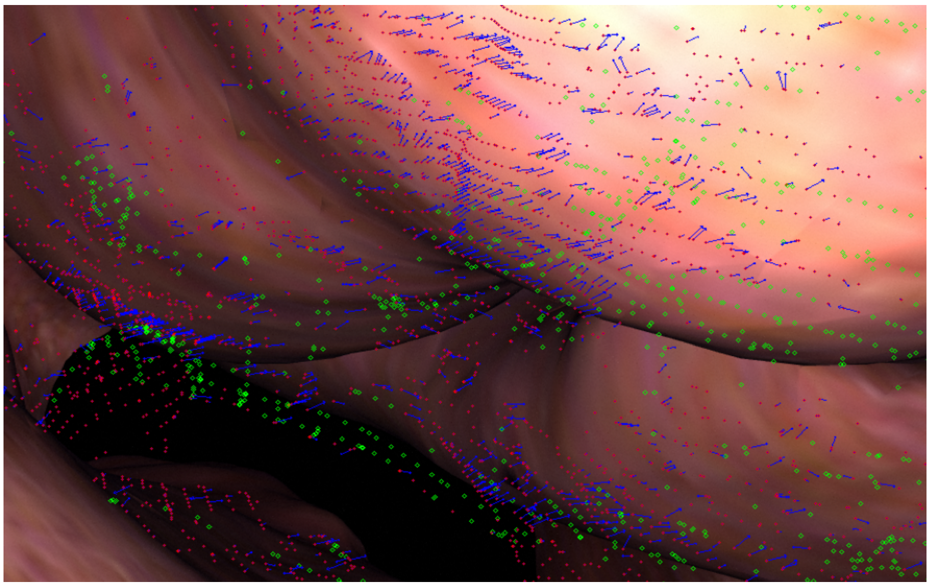


Figure 4.3: Result of image registration. Red points means the back-projected points have the same intensity as the original points. Blue arrows show the registration between the back-projected points and its corresponding points along the searching direction. Each pair of these points connected by the arrow has the same intensity. Green points means that these points cannot find any point with same intensity given a certain searching direction and step size. Red and green points will be discarded, only blue pairs will be used for pose refinement.

where p is the confidence value usually selected to be 0.95-0.99, and w is the ratio of inliers and m is the number of samples that is used to calculate the model. In practice, k can be manually fixed for simplicity.

4.4 Pose Estimation

4.4.1 3D-2D Perspective-n-points

Perspective-n-point is a general way of solving pose estimation problem of a calibrated camera, given a set of 3D vertices in the world coordinate and their corresponding 2D pixels in the image coordinate. From the above section, for each vertex i, we have obtained correspondences between its 3D vertex, 2D back-projection points and 2D registered points. Thus camera poses can be estimated by using either 2D-2D epipolar constraints or 3D-2D motion estimation techniques. However, algorithms using 2D-2D epipolar geometry such as 5-points Method or 8-points Method may encounter the problem of scale ambiguity because the scale of translation t cannot be determined through the calculation of homography.

Perspective-n-Point takes the depth information of a 3D vertex into account to overcome scale ambiguity, and applies Direct Linear Transform (DLT) to the projection model. For a 3D vertex i p_i, the relationship of its homogeneous coordinates $p_i(x, y, z, 1)$ with its image coordinates $\mathbf{u}_{i,k}(u, v)$ at keyframe k can be described as:

$$s \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}_k = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ t_5 & t_6 & t_7 & t_8 \\ t_9 & t_{10} & t_{11} & t_{12} \end{bmatrix}_k \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{4.8}$$

Canceling the scale factor s, we have the constraint:

$$u_i = \frac{t_1 x_i + t_2 y_i + t_3 z_i + t_4}{t_9 x_i + t_{10} y_i + t_{11} z_i + t_{12}} \tag{4.9}$$

$$v_i = \frac{t_5 x_i + t_6 y_i + t_7 z_i + t_8}{t_9 x_i + t_{10} y_i + t_{11} z_i + t_{12}} \tag{4.10}$$

Then we have:

$$\begin{bmatrix} p_1^T & 0 & -u_1 p_1^T \\ 0 & p_1^T & -v_1 p_1^T \\ \vdots & \vdots & \vdots \\ p_N^T & 0 & -u_N p_N^T \\ 0 & p_N^T & -v_N p_N^T \end{bmatrix} \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \mathbf{t}_3 \end{bmatrix} = 0 \tag{4.11}$$

where $\mathbf{t}_1 = (t_1, t_2, t_3, t_4)^T$, $\mathbf{t}_2 = (t_5, t_6, t_7, t_8)^T$ and $\mathbf{t}_3 = (t_9, t_{10}, t_{11}, t_{12})^T$. N for the total number of points.

For our case, the over-determined linear transform can be solved using SVD and least square, or other methods such as EPnP [28] and DLS. We choose EPnP proposed by V. Lepetit et al. [28].

4.4.2 2D-2D 8-point Method

To recover relative motion using only a set of 2D registered points for two different frames, a common way is to utilize epipolar geometry. In figure 4.4, assume the relative motion between two frames is R, t, the optical centers are O_1, O_2. Considering a feature point in the first frame is p_1, and its corresponding point in the second frame is p_2, if they are projected from the same 3D vertex, then their connections with the optical centers, $\underset{O_1P_1}{\rightarrow}, \underset{O_2P_2}{\rightarrow}$ should intersect at point P. In this case, the plane defined by O_1, O_2, P is called epipolar plane. The connection between O_1 and O_2 is called base line. The intersection between the base line $\underset{O_1O_2}{\rightarrow}$ and two frames are called epipoles, denoted as e_1, e_2. p_1e_1, p_2e_2 are called epipolar lines.

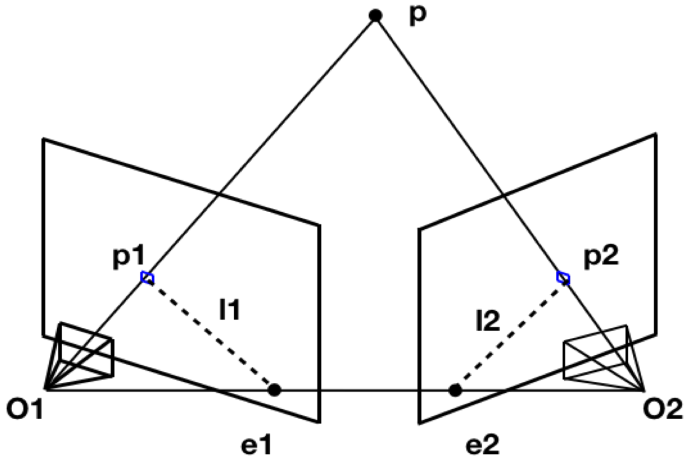


Figure 4.4: Epipolar geometry

To formulate the geometry mathematically, assume the 3D point P is $P = [X, Y, Z]$, the calibration matrix is K, so the projection of P on two frames are

$$p_1 = KP, \qquad p_2 = K(RP + t) \tag{4.12}$$

This equation can be rewritten as

$$x_1 = K^{-1}P_1, \qquad x_2 = K^{-1}P_2 \tag{4.13}$$

Where x_1, x_2 are the coordinates on the normalized plane of the pixels. From the equation we can obtain the relationship

$$x_2 = Rx_1 + t \tag{4.14}$$

Multiply $\hat{t}$ on both sides we have (outer product)

$$\hat{t}x_2 = \hat{t}Rx_1 \tag{4.15}$$

and x_2^T we have

$$x_2^T \hat{t}x_2 = x_2^T \hat{t}Rx_1 \tag{4.16}$$

Notice that ${}^T\hat{t}x_2$ is normal to both t and x_2, then the left side of the equation is 0. Hence we have

$$x_2^T \hat{t}Rx_1 = 0 \tag{4.17}$$

Substitute p_1, p_2 from equation 4.15 we have

$$p_2^T K^{-T} \hat{t}RK^{-1} p_1 = 0 \tag{4.18}$$

Equation 4.18 is known as epipolar constraint. It encodes the transformation of two frames in the relationship of two corresponding feature points. The matrix E is called Essential Matrix

$$E = \hat{t}R \tag{4.19}$$

and matrix F is called Fundamental Matrix

$$\begin{aligned} F &= K^{-T} \hat{t}RK^{-1} \\ &= K^{-T} EK^{-1} \end{aligned} \tag{4.20}$$

Generally speaking, given a set of corresponding point, we can recover camera motion by first calculating E or F, then decompose E or F to get R and t.

Essential Matix is a 3×3 matrix with 9 parameters. Thus the epipolar constraint can be written as

$$\begin{bmatrix} u & v & 1 \end{bmatrix} \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_7 & e_7 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \tag{4.21}$$

This can be further simplified linearly as

$$\begin{bmatrix} u_1^1 u_2^1 & u_1^1 v_2^1 & u_1^1 & v_1^1 u_2^1 v_1^1 v_2^1 & v_1^1 & u_2^1 & v_2^1 & 1 & \\ u_1^2 u_2^2 & u_1^2 v_2^2 & u_1^2 & v_1^2 u_2^2 v_2^1 v_2^2 & v_1^2 & u_2^2 & v_2^2 & 1 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_1^N u_2^N & u_1^N v_2^N & u_1^N & v_1^N u_2^N v_1^N v_2^N & v_1^N & u_2^N & v_2^N & 1 & \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \end{bmatrix} = 0 \tag{4.22}$$

This is called 8-point Method as eight points are sufficient to solve the equation so N is normally equal to eight. However in this project, the feature points are far more than eight, we can also find E by substituting all the points and solve the over-determined equation.

Once the Essential Matrix is calculated, the transformation between two adjacent frames can be recovered using Singular Value Decomposition (SVD). By SVD we will normally have four solutions, with only one that P has positive depths in both cameras. We can select the right transformation in this way and get rid of the invalid ones.

However, due to scale ambiguity, the 8-point Method is not able to output a transformation with the scale consistent to the CT scene model. Therefore, this method is only used to determine the relative direction of motion.

Chapter 5

Motion-only Bundle Adjustment

5.1 Bundle Adjustment

In practice, PnP is normally used as a prior pose prediction followed by a bundle adjustment formulation of PnP to refine the estimation. The PnP problem can be regarded as a nonlinear least square problem using Lie Algebra, hence the relative motion of two consecutive frames can be estimated using bundle adjustment. Bundle adjustment considers both camera poses and 3D vertices as optimization variables, and minimize reprojection error for an optimal solution.

Reformulate the above projection equation in Lie Algebra we have:

$$s \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K} \exp\left(\hat{\mathbf{C}}\right) \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{5.1}$$

In bundle adjustment, the error function is chosen to be the sum of Euclidean distances between back-projected points and their corresponding points.

$$\mathbf{C}^*_{k-1,k} = \underset{\mathbf{C}_{k-1,k}}{\arg\min} \frac{1}{2} \sum_{i=1}^{n} \left\| u_i - \frac{1}{s_i} \mathbf{K} \exp\left(\hat{\mathbf{C}}\right) p_i \right\|_2^2 \tag{5.2}$$

And globally

$$\mathbf{C}^* = \arg\min_{\mathbf{C}} \frac{1}{2} \sum_k \sum_{i=1}^{n} \left\| u_i - \frac{1}{s_i} \mathbf{K} \exp(\hat{\mathbf{C}}) p_i \right\|_2^2 \tag{5.3}$$

The error function can be linearized by finding the derivative of each variable with respect to the function, and solved with optimation methods such as Gauss-Newton Method and Levenberg-Marquardt Method to find the best solution.

As the prescanned CT scene model and intrinsic parameters are fixed, we apply motion-only bundle adjustment to optimize the only poses. In this case, the Hessian Matrix used for optimization is highly sparse with only camera poses block.

5.2 Pose-graph Optimization

The Front End VO is capable of generating a local trajectory and a scene map. Due to the noise associated with each step, the trajectory is not consistent in a large scale, hence requires a global optimization. Bundle adjustment is effective not only in solving optimization problem between two adjacent frames, but also the localization and mapping problem across frames in a large scale. One problem associated with graph-based optimization is, when the scale of poses and the map increases, the computational efficiency will drop. 3D points and poses will start to converge at the same time so the optimization will become less effective.

In our case, as 3D vertices are fixed and directly obtained from the 3D model, we are no longer interested in optimizing these parameters but only camera poses, hence we can keep track of only the camera poses to form a global Pose-Graph and apply graph optimization.

The nodes of the graph is represented by the camera poses, $\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_n$.

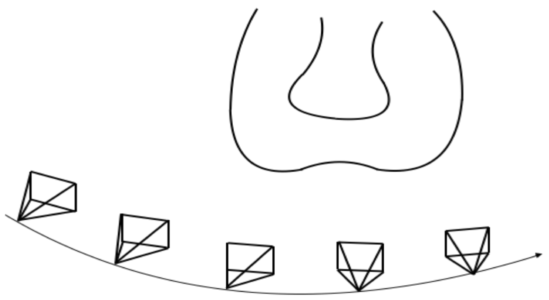


Figure 5.1: A pose-graph representation of the motion estimation process. Every node in the graph corresponds to a pose of the endoscope. Only poses are connected by edges while the 3D points in the CT model are discarded. These edges represent the transformation obtained from motion estimation.

Denote the relative motion between two nodes as $\Delta\mathbf{C}_{ij}$ we have:

$$\Delta\mathbf{C}_{ij} = \mathbf{C}_i^{-1} \circ \mathbf{C}_j = \ln(\exp((-\mathbf{C}_i)^{\wedge}\exp(\mathbf{C}^{\vee}))^{\wedge} \tag{5.4}$$

Or rewrite the equation using Lie Group we have:

$$\Delta\mathbf{T}_{ij} = \mathbf{T}_i^{-1}\mathbf{T}_j \tag{5.5}$$

In general, two sides of the equation may not be identical due to the accumulation of error and unpredictable noises. Hence, define the error term for the optimization as:

$$\begin{aligned}\mathbf{e}_{ij} &= \ln(\Delta\mathbf{T}_{ij}^{-1}\mathbf{T}_i^{-1}\mathbf{T}_j)^{\vee} \\ &= \ln(\exp((-\mathbf{C}_{ij})^{\wedge})\exp((-\mathbf{C}_i)^{\wedge}\exp(\mathbf{C}^{\vee}))^{\wedge}\end{aligned} \tag{5.6}$$

Then we can formulate the optimization problem by minimizing the cost function:

$$\min_{\mathbf{C}} \frac{1}{2}\sum_{i,j\in\varepsilon} \mathbf{e}_{ij}^T\Sigma_{ij}^{-1}\mathbf{e}_{ij} \tag{5.7}$$

where ε is the set of all vertices and Σ_{ij} is the covariance matrix between two poses.

Pose graph optimization is usually associated with a Loop Closure module. Loop closure is a module that detects if a robot or camera has revisited one of its previous location. It is crucial for enhancing the robustness of both topological and metrical SLAM algorithms. [3]

Loop closure contains two main module: a keyframe selection module which selects the keyframes based on the similarity between each frames; a loop detection module, which determines if two keyframes belong to the same location.

The selection of keyframe need to be designed to avoid redundant computation. Front end visual odometry is generally of high frequency, hence generating a sequences of pose estimation. Keyframe selection determines poses that contain distinctive information from sequence and forms nodes in the graph. In other words, keyframe selection is based on similarity between two adjacent frames.

To accelerate the key frame selection process, we utilize CT model indices. For each key frame, the indices of back-projected points form a index set. When new frames are received, indices in each new frame will be compared with all the existing index sets. If the new frame contains more than certain number of percentage of distinct indices compared to the previous keyframe, then the new frame will be selected as a new keyframe. If a certain percentage of feature points in the new keyframe can be viewed by at least three keyframes, the this frame is redundant and will be removed from the keyframes.[35]

Chapter 6

Performance Evaluations

6.1 Energy Function

In this project, we use Euclidean distance as a measurement of similarity for the estimated pose with respect to its ground truth. The energy function is defined as follows:

$$E = \sum_{i=1}^{n} \left\| \alpha^2 (u^2 + v^2) \right\|_2^2 \tag{6.1}$$

where n is the number of points back-projected onto the image plane. u and v are optical flows in x and y directions calculated in section 4.2. α is defined as:

$$\alpha = \begin{cases} 0 & \text{if } u^2 + v^2 = 0 \\ 1 & \text{if } 0 < u^2 + v^2 < d \\ 0 & \text{if } u^2 + v^2 > d \end{cases} \tag{6.2}$$

where d is a thresholding distance in pixel. The intuition behind this is if a registration is valid, which means the magnitude of the flow is small and positive, we choose to minimize all these flows disregarding the rest of the points. In this system we choose d to be 20 pixels which is corresponding to about 0.3mm pose error in world coordinates. It can be adjusted based on the endoscopic movements in practice.

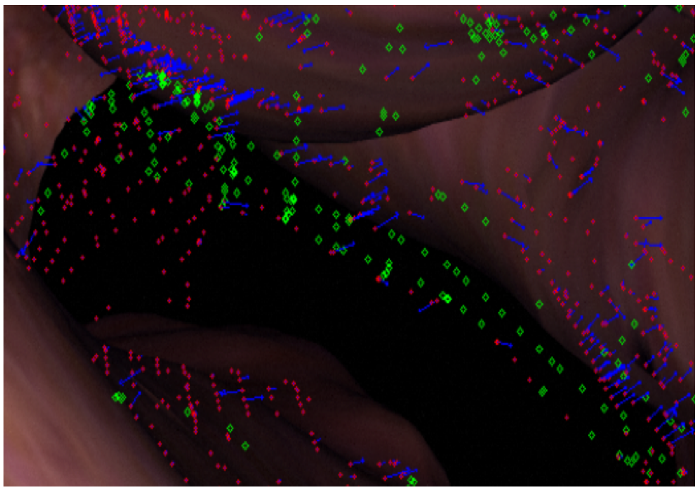


Figure 6.1: Back-projected points selection. Blue arrow means there is a valid movement between the back-projected points and corresponding points. Red points means there is no movement and green points means there is no corresponding point under certain distance. The red and blue will be discarded in each iteration.

6.2 Minimization of Energy Function

There are two scenarios that the energy of the system will drop. In the first scenario, the estimated pose shifts closer to its ground truth in each iteration causing the total length of flows to reduce.

In another scenario, when the prior pose estimation is completely off or the gradient leads to the wrong search direction, the number of valid registration significantly decreases. It results in the decreasing of energy function but generates a wrong estimation. This scenario also includes extreme cases such as the camera is facing out of the sinus cavity. As this violates the assumption of slow motion, vision-based methods are usually invalid. But it can be easily solved with other sensor-based prior motion estimation methods (see Sect. 4.1)

6.3 Estimation results

The following table shows a pose estimation of a single frame. The estimation process starts with the pose of the last frame, which is 0.5mm off in x direction to the current pose. The algorithm is designed to have 10 iterations. In each iteration, the algorithm minimizes the energy defined in the previous section, and returns a new value of estimated pose.

Iterative Pose Estimation for a Frame					
pose	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5
x	4.091057	4.045245	4.042591	4.045245	4.028958
y	26.974160	26.924137	26.902820	26.924137	26.915048
z	11.684938	11.655635	11.658340	11.655635	11.671567
x-axis	0.550026	0.544711	0.544711	0.544542	0.542748
y-axis	2.367989	2.368812	2.368812	2.368051	2.368256
z-axis	-1.827689	-1.826783	-1.826783	-1.827071	-1.826622
Iteration6	Iteration7	Iteration8	Iteration9	Iteration10	Ground Truth
3.989887	4.059595	4.047657	4.083908	4.078509	4.3593
26.911758	26.957401	26.892593	26.918454	26.939552	27.1535
11.694322	11.682742	11.659985	11.651248	11.685586	11.8704
0.545610	0.546054	0.546067	0.544414	0.547130	0.3598572
2.369949	2.368471	2.368985	2.367433	2.368512	0.0497875
-1.827068	-1.826612	-1.827072	-1.826988	-1.826609	-1.8895694

Table 6.1: Motion estimation starting from a manually selected position. The prior pose is -0.5mm to the current pose in x direction. This table shows the pose estimations for different number of iterations. The unit for movements in x, y, z directions is in millimeter, and for yaw, pitch, roll is in rads. The last column shows the ground truth for the estimated pose.

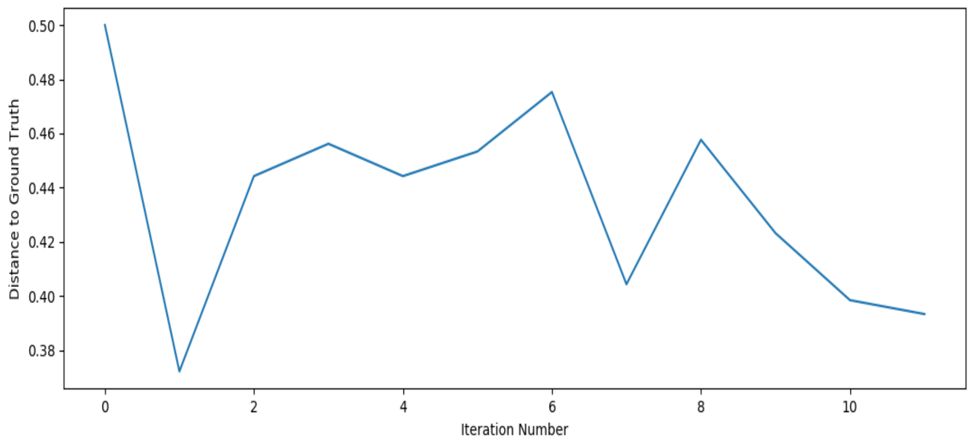
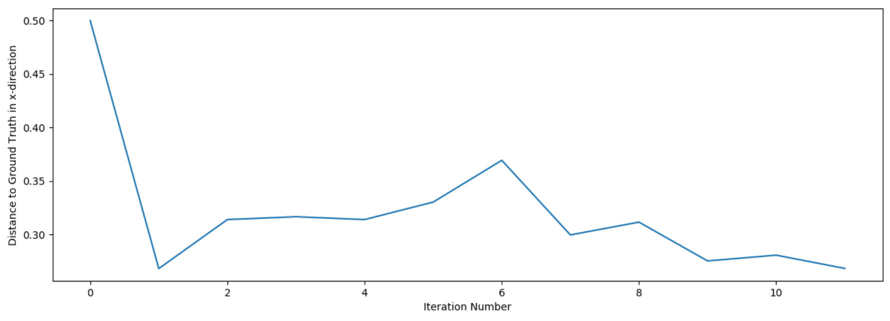
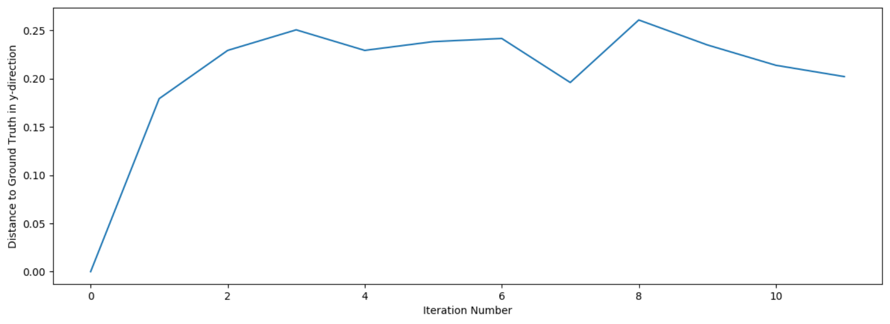


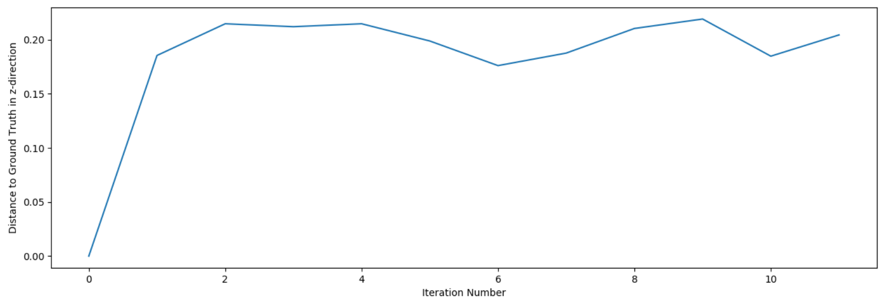
Figure 6.2: The Euclidean distance of the estimate pose to the ground truth.



(a) Error in x direction

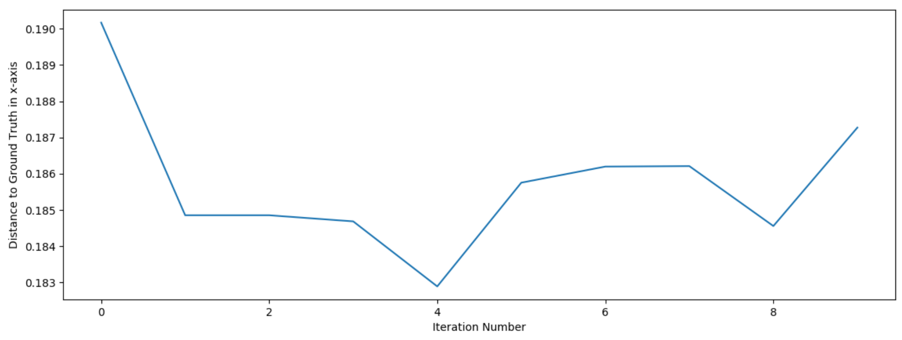


(b) Error in y direction

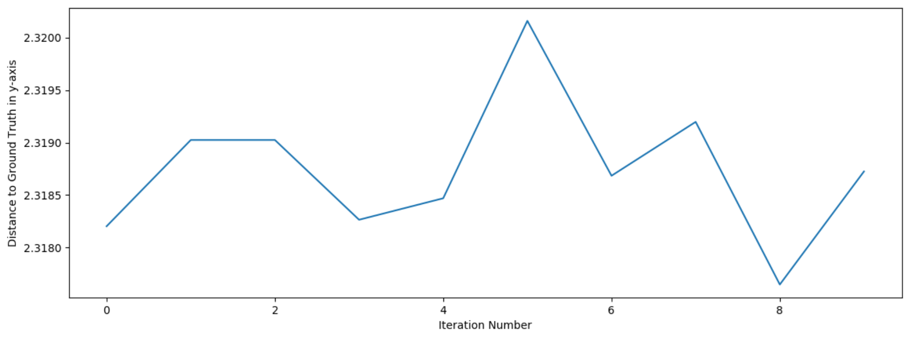


(c) Error in z direction

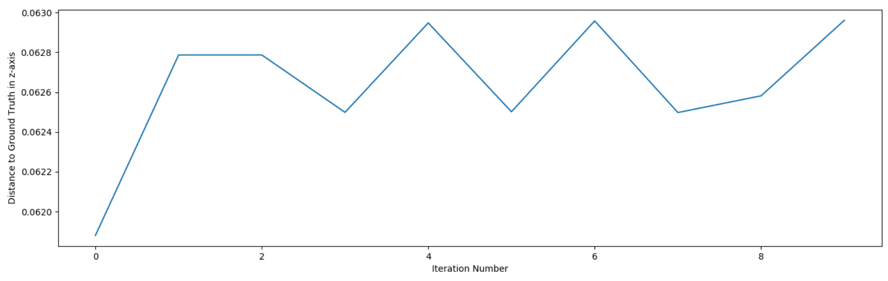
Figure 6.3: The Euclidean distance of the estimate pose to the ground truth in each direction.



(a) Error in x axis



(b) Error in y axis



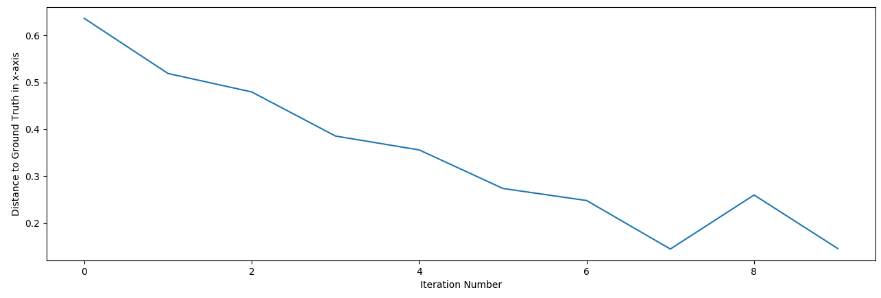
(c) Error in z axis

Figure 6.4: The Euclidean distance of the estimate pose to the ground truth in each angle axis.

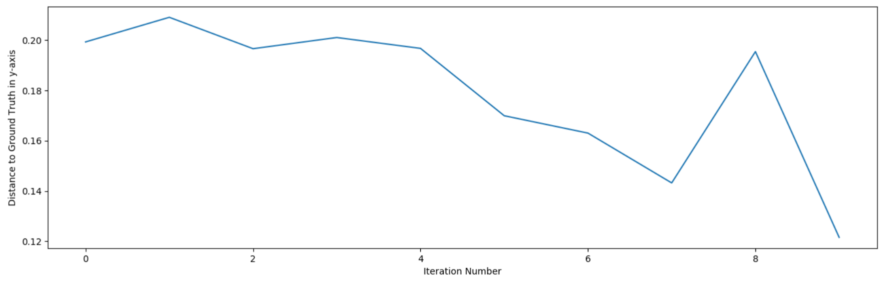
To further test our method, we start with a random position which is about 1mm away from its ground truth. The results is shown below.

Iterative Pose Estimation for a Frame					
pose	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5
x	5.022175	4.904615	4.865226	4.771162	4.741542
y	27.355992	27.365783	27.353296	27.357743	27.353421
z	11.794553	11.832653	11.861530	11.879385	11.818083
x-axis	0.556773	0.555786	0.557391	0.556387	0.555899
y-axis	2.354605	2.356595	2.357667	2.359798	2.358854
z-axis	-1.831809	-1.831824	-1.831696	-1.831950	-1.833448
Iteration6	Iteration7	Iteration8	Iteration9	Iteration10	Ground Truth
4.659306	4.633547	4.530066	4.645394	4.531065	4.38557
27.326630	27.319720	27.299903	27.352135	27.278241	27.1567
11.847258	11.882004	11.859021	11.882315	11.843280	11.9095
0.554183	0.555833	0.552045	0.553557	0.552027	0.360388,
2.360727	2.361816	2.362929	2.360879	2.363119	0.0522335
-1.833103	-1.832394	-1.833455	-1.832502	-1.833360	-1.88488

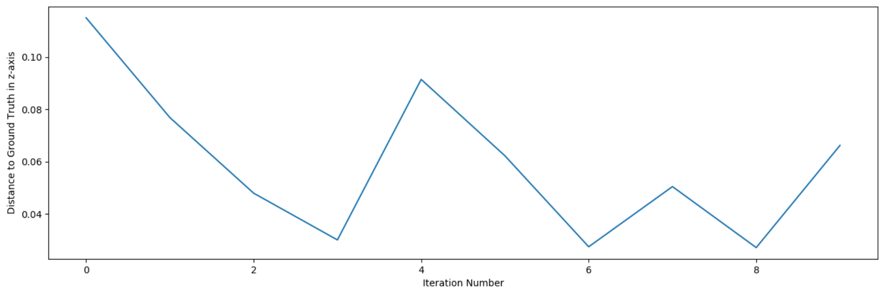
Table 6.2: Motion estimation starting from a random position. This table shows the pose estimates for different iterations. The unit for movements in x, y, z directions is in millimeter, and for yaw, pitch, roll is in rads. The last column shows the ground truth for the estimated pose.



(a) Error in x direction

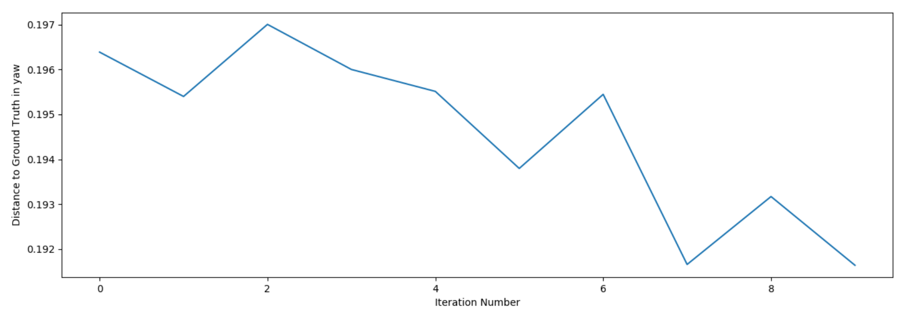


(b) Error in y direction

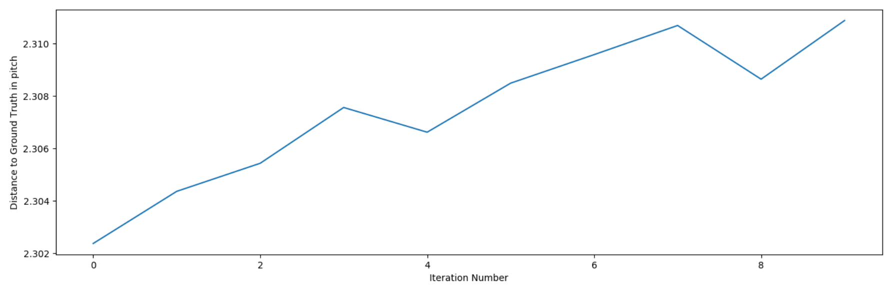


(c) Error in z direction

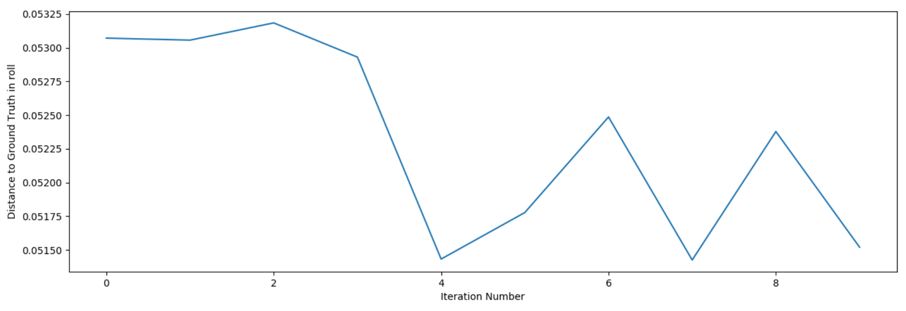
Figure 6.5: The Euclidean distance of the estimate pose to the ground truth in each direction.



(a) Error in x axis



(b) Error in y axis



(c) Error in z axis

Figure 6.6: The Euclidean distance of the estimate pose to the ground truth in each angle axis.

As seen from results of both manually selected starting position and randomly selected starting position, the iterative pose estimation will reduce the error to about 0.4mm, which corresponds to the resolution of the prescanned CT model. For the directions with large disparities (see x directions for both cases), the error drops rapidly and oscillates around its optimal solution. For the directions with small disparity, the estimated location oscillates with only small improvements.

Ideally, the iteration stops when a certain number of iteration is reached.

In our system, we use 10 iterations. It also stops when no vertex is observed by the camera. Figure 6.7 shows the visualize the performance of our method for each iteration. As we can see from the figure, after a certain number of iterations, the contours of the images are aligned and more points are mapped to its photometric equivalence.

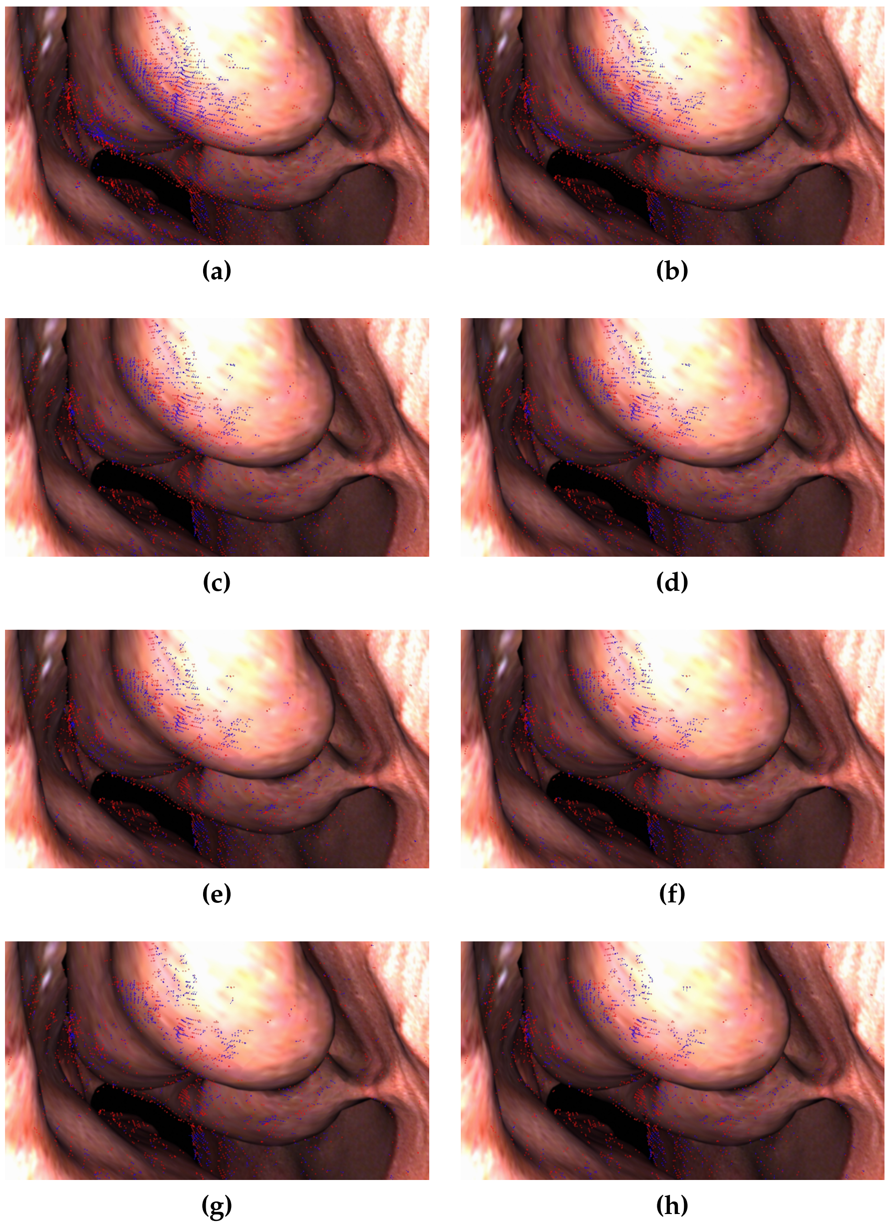
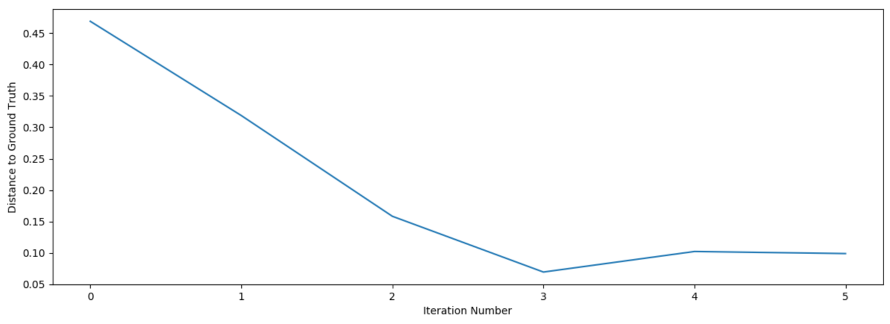
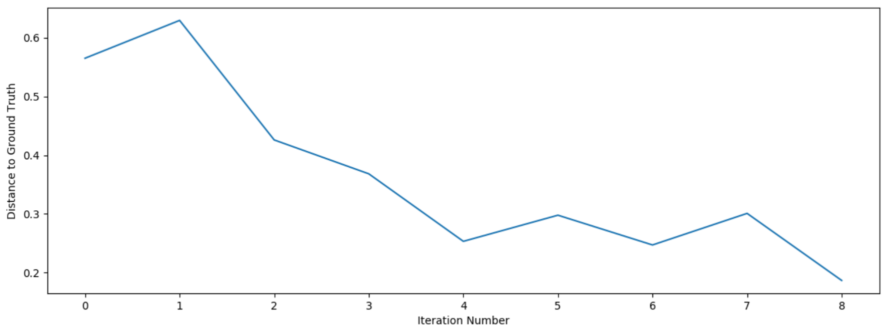


Figure 6.7: Visualization of improvements of each iteration. The back-projected points are shifting left to align with the original image. The total magnitude decreases and more points are aligned. Two images coincide in the end as suggested by their contours.

Due to the natural of optimization, it normally takes more iterations to get to an optima when the disparity is large.



(a)



(b)

Figure 6.8: Iteration performance when the energy is no longer decreased.

6.4 Early Stopping

To save computational power, we add one more constraint to decide if the iteration should continue. If the energy is reduced in the current iteration, the method continues; otherwise the iteration stops. The maximum number if iterations is set to 30 in this case.

From figure 6.8, we can see that the method is still able to converge to the optimal solution if it stops when the energy no longer decreases. It may take more iterations to converge if the disparity is comparatively large.

6.5 Error Statistics

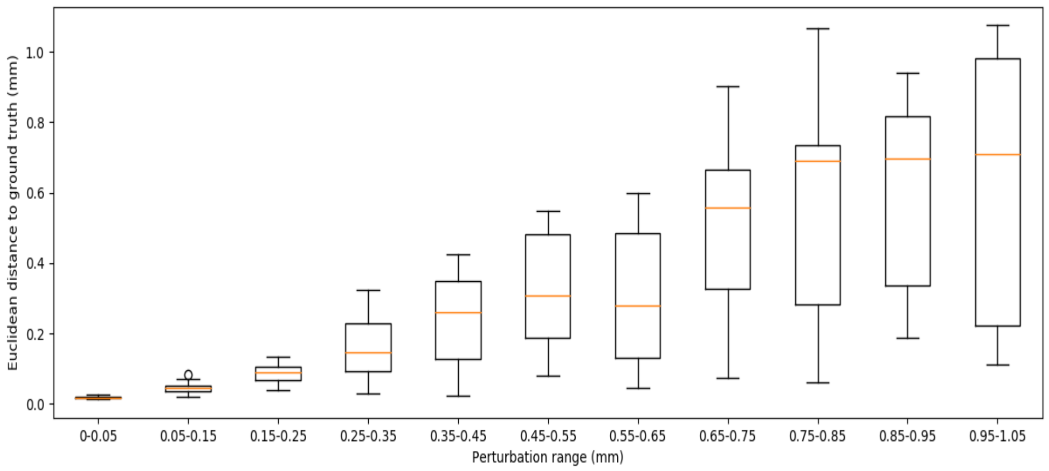


Figure 6.9: Error statistics of the estimated pose given a random perturbation in a certain range. The horizontal axis represents the magnitude of the perturbation, and the vertical axis represents the error in Euclidean distance to the ground truth.

To test how the system performs for different starting positions away from its ground truth, we randomly generate small perturbations from -1.05mm to 1.05mm away from the ground truth as starting positions. The error statistics of the estimate is shown in figure 6.9. In general, the mean and variance of the error increase as the perturbation range increases. This means if the prior pose estimate has a large disparity to its ground truth, the pose estimation iteration is more likely to run into local minima. But there is still a change to generate a comparatively good estimate suggested by the lower quartiles of the boxes.

Chapter 7

Conclusion and Discussion

In this work, we introduce a visual navigation system for sinus surgery with the ability of texture mapping the CT model in real time and localizing the endoscope for navigation. We also propose a motion estimation method and show that the method could refine the pose estimates from different prior estimates.

7.1 Discussion

7.1.1 Mathematical Interpretation

Our method is inspired by the idea of optical flow and bundle adjustment, and seeks to minimize global energy to find the optimal solution. This procedure can be interpreted as a nonlinear optimization using first order gradient descent. The direction of the flow can be interpreted as the gradient or search direction in steepest descent, and the flow magnitude can be interpreted by the step size.

The first order Newton's method is greedy, as it maximize the performance for each iteration by choosing the optimization direction to be the local gradient. But it also suffers from the gradient as it usually takes a zig-zag path for optimization which uses more iterations to converge. As seen from the figures, the estimated pose also oscillates around its optimum.

To solve this problem, we can adjust the search direction to make it less greedy by using the idea of Gauss-Newton Method as described in chapter 2.

$$J(x)^T J(x) \triangle x = -J(x)^T f(x) \tag{7.1}$$

In this case, we can approximate the gradient using the function itself and its Jacobians, and determine the step size in the similar manner. Similarly, given the Jacobian and Hessian of an image, Trust Regions Methods and Dog Leg Methods could also be investigated for future modifications.

7.1.2 Comparison with Direct Methods

Our method is naturally similar to Direct Methods as both methods are inspired by optical flow. Our method can be regarded as a discrete version of Direct Methods. Direct Methods formulate the problem using the equation

$$T^*_{k-1,k} = \underset{T_{k-1,k}}{\arg\min} \frac{1}{2} \sum_i \left\| I^c_k(\mathbf{K} T_{k-1,k} p_i) - I^c_{k-1}(u_i) \right\|^2_2 \tag{7.2}$$

So for each iteration, it returns a transformation T of rotation R and translation t, and uses this transformation for new iterations. It combines the registration and motion estimation in one optimization step. One advantage of combining all substeps into a single objective function is it can achieve high speed. By putting the optimization variables into an optimization framework such as G2O and Ceres, Direct Methods could achieve the frequency of more than 50Hz. It significantly outperforms our method in terms of speed as it doesn't require any image rendering or image registration.image rendering or image registration are the most time consuming steps in our method.

However, our method may outperform Direct Methods in terms of accuracy. As Direct Methods are usually dense or semi-dense, they are purely based on least squares and there is no outlier removal module. They suffer a lot from the outliers when the when the trackable points is sparse or the image has a lot of textures. In the similar manner, Direct Methods are also more sensitive to local mimima. Discretization of each optimization step of Direct

Methods allow us to process and modify the data throughout registration and motion estimation step. We can regulate the flow to compensate for the sparseness of the back-projected points, and remove the outliers to improve the estimation accuracy of each iteration.

7.1.3 Comparison with Feature-based Methods

Our method is also inspired from bundle adjustment as the method is trying to minimize the reprojection error between the back-projected points and its corresponding points. There are two major differences between our method and feature-based methods. Our method regards the back-projected points as feature points rather than extracting feature points. Bypassing feature extraction and feature description significantly saves computational power and processing time.

The second difference is, for feature-based method the registration is solid, which means the correspondence is fixed throughout the process. Therefore there is no need for an iterative optimization to get the best result. Principally, 3D-2D motion estimation methods use PnP to obtain a reasonable initial estimation, and refine the estimate using local bundle adjustment. However, for our method, the registration is soft. The source point may be registered to different photometric equivalence in different iterations. In this case, local Bundle Adjustment is not necessary as a result of the stochastic natural of the method.

7.2 Future Works

Our future work involves five parts. The first part is to extend our visual navigation system to a complete visual SLAM system. To achieve this, a pose graph optimization module and a loop closure module described chapter 5 need to be implemented. Moreover, in surgical data collection and real surgeries, surgeons need to manually create loops to enable global pose optimization.

The second part is to add an initial pose estimation module using other registration methods. This could be achieved using feature-based 3D-2D registration or ICP-based 3D-3D registration if an 3D reconstruction is given. The methods available are described in the introduction.

The third part is to integrate another sensor for prior motion estimation, such as an IMU or an EM tracker. If a reasonably good estimate (say the error is less than 0.5mm) is given, the method is more likely to converge with less time and less iterations.

The fourth is to conduct experiments with real endoscopes and CT data. The real endoscopic image is slightly different from the simulation image, which requires modifications and adjustments to the method.

The fifth is to implementing new ways of calculating gradient direction and step length, as described in Chapter 7.1.1. The could lead to better results only with simple modifications to the system. One more thing could be done is to transform all the iterations of our method into one optimization formulation like Direct Methods for faster processing speed. But it still requires modifications to regulate the flows as the back-projected points are highly sparse and the method is sensitive to outliers.

References

[1] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications To Image Analysis and Automated Cartography”. In: 24 (1981), pp. 381–395.

[2] Sameer Agarwal et al. “Building Rome in a Day”. In: *Commun. ACM* 54.10 (2011), pp. 105–112. ISSN: 0001-0782. DOI: 10.1145/2001269.2001293.

[3] A. Angeli et al. “Visual topological SLAM and global localization”. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 4300–4305. DOI: 10.1109/ROBOT.2009.5152501.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “SURF: Speeded Up Robust Features”. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[5] S. S. Beauchemin and J. L. Barron. “The Computation of Optical Flow”. In: *ACM Comput. Surv.* 27.3 (1995), pp. 433–466. ISSN: 0360-0300. DOI: 10.1145/212094.212141.

[6] Seth D. Billings et al. “Anatomically Constrained Video-CT Registration via the V-IMLOP Algorithm”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*. Ed. by Sebastien Ourselin et al. Cham: Springer International Publishing, 2016, pp. 133–141. ISBN: 978-3-319-46726-9.

[7] C. Cadena et al. “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age”. In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2624754.

[8] J. Civera, A. J. Davison, and J. M. M. Montiel. “Inverse Depth Parametrization for Monocular SLAM”. In: *IEEE Transactions on Robotics* 24.5 (2008), pp. 932–945. ISSN: 1552-3098. DOI: 10.1109/TRO.2008.2003276.

[9] Navneet Dalal and Bill Triggs. “Histograms of Oriented Gradients for Human Detection”. In: 1 (2005), pp. 886–893.

[10] Kim Dalziel et al. “Endoscopic sinus surgery for the excision of nasal polyps: A systematic review of safety and effectiveness”. In: 20 (2006), pp. 506–19.

[11] H. Durrant-Whyte and T. Bailey. "Simultaneous localization and mapping: part I". In: *IEEE Robotics Automation Magazine* 13.2 (2006), pp. 99–110. ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1638022.

[12] Kennedy DW. "Functional endoscopic sinus surgery: Technique". In: *Archives of Otolaryngology* 111.10 (1985), pp. 643–649. DOI: 10.1001/archotol.1985.00800120037003. eprint: /data/journals/otol/17089/archotol_111_10_003.pdf. URL: +http://dx.doi.org/10.1001/archotol.1985.00800120037003.

[13] J. Engel, V. Koltun, and D. Cremers. "Direct Sparse Odometry". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (2018), pp. 611–625. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2658577.

[14] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 834–849. ISBN: 978-3-319-10605-2.

[15] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast semi-direct monocular visual odometry". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 15–22. DOI: 10.1109/ICRA.2014.6906584.

[16] Gregory D. Hager and Peter N. Belhumeur. "Efficient region tracking with parametric models of geometry and illumination". In: *PAMI* (1998).

[17] K. J. Hanna. "Direct multi-resolution estimation of ego-motion and structure from motion". In: *Proceedings of the IEEE Workshop on Visual Motion*. 1991, pp. 156–162. DOI: 10.1109/WVM.1991.212812.

[18] Berthold Horn and Brian G. Schunck. "Determining Optical Flow". In: 17 (1981), pp. 185–203.

[19] Michal Irani. "All About Direct Methods". In: (2000).

[20] H. Jin, P. Favaro, and S. Soatto. "Real-time 3D motion and structure of point features: a front-end system for vision-based control and interaction". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*. Vol. 2. 2000, 778–779 vol.2. DOI: 10.1109/CVPR.2000.854954.

[21] Hailin Jin, Paolo Favaro, and Stefano Soatto. "A semi-direct approach to structure from motion". In: *The Visual Computer* 19.6 (2003), pp. 377–394. ISSN: 1432-2315. DOI: 10.1007/s00371-003-0202-6.

[22] M. Kaess, A. Ranganathan, and F. Dellaert. "iSAM: Incremental Smoothing and Mapping". In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1365–1378. ISSN: 1552-3098. DOI: 10.1109/TRO.2008.2006706.

[23] G. Klein and D. Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. 2007, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852.

[24] James G. Krings et al. "Complications of primary and revision functional endoscopic sinus surgery for chronic rhinosinusitis". In: *The Laryngoscope* 124.4 (), pp. 838–845. DOI: 10.1002/lary.24401. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/lary.24401.

[25] J. J. Leonard and H. F. Durrant-Whyte. "Simultaneous map building and localization for an autonomous mobile robot". In: *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*. 1991, 1442–1447 vol.3. DOI: 10.1109/IROS.1991.174711.

[26] S. Leonard et al. "Evaluation and Stability Analysis of Video-Based Navigation System for Functional Endoscopic Sinus Surgery on In-Vivo Clinical Data". In: *IEEE Transactions on Medical Imaging* (2018), pp. 1–1. ISSN: 0278-0062. DOI: 10.1109/TMI.2018.2833868.

[27] Simon Leonard et al. "Image-based navigation for functional endoscopic sinus surgery using structure from motion". English (US). In: *Medical Imaging 2016: Image Processing*. Vol. 9784. SPIE, 2016. DOI: 10.1117/12.2217279.

[28] V. Lepetit, F. Moreno-Noguer, and P. Fua. *EPnP: An Accurate O(n) Solution to the PnP Problem*. 2008.

[29] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*. 2003.

[30] Bruce D. Lucas and Takeo Kanade. "An iterative image registration technique with an application to stereo vision". In: *In IJCAI81*. 1981, pp. 674–679.

[31] K. MacTavish and T. D. Barfoot. "At all Costs: A Comparison of Robust Cost Functions for Camera Correspondence Outliers". In: *2015 12th Conference on Computer and Robot Vision*. 2015, pp. 62–69. DOI: 10.1109/CRV.2015.52.

[32] K. Mikolajczyk and C. Schmid. "A performance evaluation of local descriptors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.10 (2005), pp. 1615–1630. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2005.188.

[33] Daniel Mirota et al. "Toward Video-Based Navigation for Endoscopic Endonasal Skull Base Surgery". In: 12 (2009), pp. 91–9.

[34] K. Mori et al. "Tracking of a bronchoscope using epipolar geometry analysis and intensity-based image registration of real and virtual endoscopic imagesâĂăâĂăA preliminary version of this paper was presented at the Medical Image Computing and Computer-Assisted Intervention (MICCAI) Conference, Utrecht, The Netherlands (Mori et al., 2001)." In: *Medical Image Analysis* 6.3 (2002), pp. 321 –336. ISSN: 1361-8415. DOI: https://doi.org/10.1016/S1361-8415(02)00089-0. URL: http://www.sciencedirect.com/science/article/pii/S1361841502000890.

[35] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. "ORB-SLAM: a Versatile and Accurate Monocular SLAM System". In: *CoRR* abs/1502.00956 (2015). arXiv: 1502.00956. URL: http://arxiv.org/abs/1502.00956.

[36] Nassir Navab et al. "Direct Method for Motion Estimation: An Alternative to Decomposition of Planar Transformation Matrices". In: *Proceedings of the 24th DAGM Symposium on Pattern Recognition*. London, UK, UK: Springer-Verlag, 2002, pp. 575–582. ISBN: 3-540-44209-X.

[37] 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*. 2011, pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513.

[38] David NistÃľr, Oleg Naroditsky, and James Bergen. "Visual Odometry". In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPRâĂŹ04*. 2004.

[39] David NistÃľr and Henrik StewÃľnius. "Scalable Recognition with a Vocabulary Tree". In: 2 (2006), pp. 2161 –2168.

[40] Yoshito Otake et al. "Rendering-based video-CT registration with physical constraints for image-guided endoscopic sinus surgery". In: *Proceedings of SPIE–the International Society for Optical Engineering* 9415 (2015).

[41] M. Pizzoli, C. Forster, and D. Scaramuzza. "REMODE: Probabilistic, monocular dense reconstruction in real time". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 2609–2616. DOI: 10.1109/ICRA.2014.6907233.

[42] Maria Robu et al. "Intelligent viewpoint selection for efficient CT to video registration in laparoscopic liver surgery". In: *International Journal of Computer Assisted Radiology and Surgery*. 2017.

[43] Ethan Rublee et al. "ORB: an efficient alternative to SIFT or SURF". In: (2011), pp. 2564–2571.

[44] Davide Scaramuzza and Friedrich Fraundorfer. "Visual Odometry [Tutorial]". In: 18 (2011), pp. 80–92.

[45] Jianbo Shi and Tomasi. "Good features to track". In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[46] Noah Snavely, Steven M. Seitz, and Richard Szeliski. "Photo tourism: Exploring photo collections in 3D". In: *In Proc. ACM SIGGRAPH*. 2006.

[47] George Stockman and Linda G. Shapiro. *Computer Vision*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN: 0130307963.

[48] Jan Stühmer, Stefan Gumhold, and Daniel Cremers. "Real-Time Dense Geometry from a Handheld Camera". In: *Pattern Recognition*. Ed. by Michael Goesele et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 11–20. ISBN: 978-3-642-15986-2.

[49] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 2010.

[50] Carlo Tomasi and Takeo Kanade. *Detection and Tracking of Point Features*. Tech. rep. International Journal of Computer Vision, 1991.

[51] Bill Triggs et al. “Bundle Adjustment – A Modern Synthesis”. In: *VISION ALGORITHMS: THEORY AND PRACTICE, LNCS*. Springer Verlag, 2000, pp. 298–375.

[52] Hanzi Wang, Daniel Mirota, and Gregory D. Hager. “A generalized kernel consensus-based robust estimator”. English (US). In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (2010), pp. 178–184. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2009.148.

Linhao Jin

(410) 982-9679 | ljin18@jhu.edu | www.linhaojin.com

Education

May 2019	**JOHNS HOPKINS UNIVERSITY (JHU)**, Baltimore, MD M.S.E. in Robotics - Perception and Cognitive Systems Track. Courses: Over 20 graduate-level courses taken/audited related to computer vision and SLAM. Advisor: Dr. Gregory Hager.
May 2017	**NANYANG TECHNOLOGICAL UNIVERSITY (NTU)**, Singapore B.Eng in Mechanical Engineering (First-Class Honor) - Robotics and Mechatronics Track. Honors: Full Merit-Based Scholarship from Ministry of Education in Singapore, Dean's List 2014 - 2015, Presidential Research Scholarship, Singapore-MIT Undergraduate Research Fellowship.

Work Experience

Spring 2018	**Johns Hopkins University Department of Computer Science, Baltimore, MD** **Teaching Assistant** · Organized recitations and facilitated discussions for CS663 Algorithms for Sensor-based Robotics..
Jun - Aug 2016	**Singapore-MIT Alliance for Research and Technology Center (SMART), Singapore** **Undergraduate Research Fellow** · Developed a real-time gantry detection, tracking and distance estimation algorithm using line detection algorithms for RGB and depth inputs. · Implemented and tested the algorithm with a stereo vision camera (ZED) on SMART Self-Driving Car.
Aug-Dec 2015	**Rolls-Royce Advanced Technology Center, Singapore** **Computational Engineering Intern** · Facilitated Data Analytic as a Service (DaaS) Team front end development and data visualization. · Created a weekly report automatic compilation application in Visual Basic. · Assisted in Propeller Defections Detection Project and DaaS Team project/program management.

Research Experience

Present - 2018	**Real-time Endoscopic Navigation Using Video and CT: A SLAM Approach.** | **Master's thesis** · Developed a novel endoscopic navigation system using SLAM. · Utilized a computer graphics pipeline to texture map the CT model in real-time. · Developed an optical flow-based method for the CT model and the endoscopic video 3D-2D registration. · Estimated endoscope motion using PnP and optimized global poses by motion-only bundle adjustment. · Developing a fast vision-based loop closing algorithm for back end optimization.
Sep - Nov 2017	**Spacecraft Multi-layer Insulation Segmentation using Fully Convolutional Network** · Implemented Fully Convolutional Networks (FCN-8, FCN-16, FCN-32) to segment multi-layer insulation. · Achieved cross validation accuracies over 90% on LCSR MLI dataset. · Compared and analyzed the results with outputs from FCNs with a conditional random field.
Sep - Nov 2017	**Labeled-Face-in-the-Wilds Face Classification Using Siamese Network** · Constructed a 19-layer Siamese Network in PyTorch on Google Cloud to classify faces in the LFW dataset. · Achieved the test accuracy over 95% for BCE loss and 80% for contrastive loss.
May 2017 -	**Indoor 3D Reconstruction and Mapping for High Ceiling Spray Painting Robot** | **NTU Robotics Research Center** · Devised both software and hardware of a 3D reconstruction system for a mobile spray painting robot. · Incorporate a stepper motor and a 2D laser scanner with serial communication to generate 3D scans. · Determined the rigid transformations using 3D-3D point cloud matching methods.
Jan 2017 - Aug 2015	**Event-based Camera and Ultra-wide Band (UWB)-based UAV Localization** | **NTU Internet of Things Lab** · Developed both software and hardware of a visual-inertial system for pose estimation of a mobile robot. · Constructed the system with an event-based camera, UWBs, LED anchors and an IMU. · Implemented UKF for sensor fusion and 6-DOF pose estimation. Tested the system under a Vicon system.

Skills

Programming:	C++, Python, C, MATLAB, HTML, MS VBA
Software:	ROS, PyTorch, OpenCV, PCL, Linux, Gazebo, SOLIDWORKS, Github, QNX, Arduino