FROM IMAGES TO AUGMENTED 3D MODELS: IMPROVED VISUAL SLAM AND AUGMENTED POINT CLOUD MODELING

A Thesis
Presented to
The Academic Faculty

by

Guangcong Zhang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2015

FROM IMAGES TO AUGMENTED 3D MODELS: IMPROVED VISUAL SLAM AND AUGMENTED POINT CLOUD MODELING

Approved by:

Professor Patricio A. Vela, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Panagiotis Tsiotras
School of Aerospace Engineering
Georgia Institute of Technology

Professor Erik I. Verriest
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Fumin Zhang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Anthony J. Yezzi
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: Octorber 22, 2015

To my family,

and the people who have faith in me.

ACKNOWLEDGEMENTS

First and foremost, I would like to extend my sincerest gratitude to my adviser, Patricio A. Vela, for his support over the course of my Ph.D. studies. He introduced me to various interesting research topics and afforded me the opportunity to work with many excellent researchers. Most importantly, he encouraged me to explore and pursue the research areas where my passions and motivation lay. This thesis would not be possible without his insightful guidance and faith in my abilities.

I would also like to thank Dr. Panagiotis Tsiotras for serving on my dissertation committee. I collaborated with Dr. Tsiotras for an extended period of time on the vision-only robot navigation project. He not only gave me theoretical guidance related to many key problems, but also served as a role model through his rigorous research mindset. I am indebted to my committee members, Prof. Erik Verriest, who served as the chair of my proposal committee and provided very helpful feedback; Prof. Anthony Yezzi, who was also in my proposal reading committee and introduced me to computational computer vision through his remarkable course "PDEs for Computer Vision"; and Prof. Fumin Zhang, who provided significant support for my thesis work. I am also grateful to Prof. Ioannis Brilakis in University of Cambridge, UK, for his advice and collaboration on the point-cloud modeling project during my early years of doctoral research.

I am grateful to many of the collaborators and colleagues I have befriended and worked with. Particularly, I am thankful to Dr. Hassan Kingravi, whom I treat as my mentor, for his advice on both work and life; Dr. Michail Kontitsis for many valuable discussions and collaborations on challenging robotics experiments; as well as my long-term friends and lab-mates, Miguel Serrano and Gbolabo Ogunmakin, for

their generous day-to-day support over the years.

My appreciation also goes out to my other lab-mates, Peter Karasev, Alex Chang, Fu-jen Chu, and Luisa Fairfax; my colleagues in the robot navigation project, Dr. Nuno Filipe, Dr. Dae-min Cho, and Alfredo Valverde; and my colleagues in the point-cloud modeling project, Dr. Habib Fathi, Dr. Abbas Rashidi, Dr. Fei Dai, etc. I am additionally grateful to individuals I have met during my internships, especially Dr. Qingfeng (Elden) Yu, who extended selfless support in training me and advising my career path.

My research has been supported by the National Science Foundation and U.S. Air Force Research Laboratory, without whom this work would never have been completed.

Last, but most importantly, I am deeply grateful to my family. Over the years, regardless of the distance that separated us, they have always been the closest people in my heart. Their unconditional support has always given me the utmost motivation and courage to face new challenges and continue moving forward in my life. In particular, I would like to thank my dearest wife, Yiwei Yan. She continues to always love me, care about me, and believe in me. She is the sunshine of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Vision only navigation and modeling is an important problem in computer vision and robotics. This problem can be divided into two parts. The first part involves the *monocular* visual simultaneous localization and mapping (VSLAM), which takes image sequence as input, and outputs the estimated camera poses as well as the point cloud data (PCD) mapping the environment. Although research efforts have been devoted to data-driven algorithms for solving the VSLAM problem, it remains an open problem which features can result in more accurate estimate of camera poses and thus more accurate mapping results. The second part of the problem is the PCD modeling, which converts a raw PCD to an augmented model as the final output. Challenges exist in building compact, precises, and geometric-driven models from raw point clouds, which is particularly important when applied to as-built building information models.

The work in this thesis is dedicated to solve the above two problems. In the first part of the thesis, I propose algorithms to select the features of the most utilities to VSLAM estimation, using the system observability theories. I first model the VSLAM process by building a piece-wise linear system (PWLS) based on the camera $SE\langle 3\rangle$ dynamics and measurements. Then I derive the necessary conditions to make the PWLS model *completely observable*. Based on the conditions, I designed two algorithms to improve the VSLAM by selecting the subset of features which form the strongest observable sub-system and result in a more accurate estimate. The first algorithm, dubbed "Optimally Observable and Minimal Cardinality (OOMC) SLAM", exploits the instantaneously completely observable condition. The OOMC algorithm is formulated into an Extended Kalman Filter (EKF) based SLAM, and

is demonstrated to improve both the data association and localization. The second algorithm, called the "Good Features (GF) to track for VSLAM" algorithm, utilizes the temporally completely observable condition to rank each feature by the proposed τ-temporal observability scores. I evaluate this algorithm by integrating it with different designs of VSLAM systems, including the traditional filtering based VSLAM and the recent keyframe Bundle-Adjustment (BA) based VSLAM. Extensive evaluations are performed on large-scale benchmark data sets, demonstrating that the GF algorithm improves the accuracy in different VSLAM systems, with very little loss in runtime.

The second part of this thesis focuses on solving the PCD modeling problem in a geometry-driven manner. I address the PCD modeling by first developing an algorithm to model PCD with planar patches, the basic geometric elements, by formulating a sparsity-inducing optimization to retrieve the linear subspaces embedded in the PCD. Then I extend the family of geometric elements to all quadric surface primitives (e.g. ellipsoid, quadric cones, etc.). An algorithm to detect, fit, and classify these shape primitives from PCDs is presented. With these two algorithms, an approach is further designed for semantic modeling of PCDs, with applications in as-built building information models (BIM).

List of Acronyms

BA	Bundle Adjustment
BIM	Building Information Model
CAD	Computer-Aided Design
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
KF	Key Frames
GF	Good Features
MLESAC	Maximum Likelihood Sample Consensus
OOMC	Optimally Observable and Minimal Cardinality
PCD	Point Cloud Data
PWLS	Piece-Wise Linear System
RANSAC	Random Sample Consensus
SfM	Structure from Motion
SLAM	Simultaneous Localization and Mapping
SOM	Stripped Observability Matrix
SSC	Sparse Subspace Clustering
TOM	Total Observability Matrix
VSLAM	(monocular) Visual Simultaneous Localization and Mapping

CHAPTER I

INTRODUCTION

3D modeling from image sequences is a challenging problem which can be divided into two key problems in robotics and computer vision. The first problem is referred as "(monocular) visual simultaneous localization and mapping" [32, 5, 27]. For simplicity, I directly refer "monocular VSLAM" as "VSLAM" in this thesis. The second problem is point cloud data modeling [84]. As depicted in Figure 1, given an image sequence from a monocular camera, the VSLAM solves the localization by estimating the camera $SE\langle 3\rangle$ poses, and simultaneously building the map of the environment represented by a collection of 3D points, i.e. a PCD. It is important to note that the localization and mapping are coupled in one estimation problem. The output of the VSLAM, especially the 3D point cloud, is then fed into the point cloud modeling algorithm, which converts the raw spatial information into high level information by building a geometry-driven model with semantic labels. Notice that although the localization result is not directly used in the PCD modeling, it is of great significance in some applications such as robot navigation and control, etc. In this chapter, I will introduce first the VSLAM problem and then the PCD modeling problem, together with brief discussions of my solutions to these two problem. This chapter will be concluded with the outline of the remaining parts in this thesis.

1.1 Monocular Visual Simultaneous Localization and Mapping (VSLAM)

1.1.1 Problem formulation and typical VSLAM systems

Monocular Visual Simultaneous Localization and Mapping (VSLAM) [27] refers to the problem of estimating camera poses and building a 3D map representing the

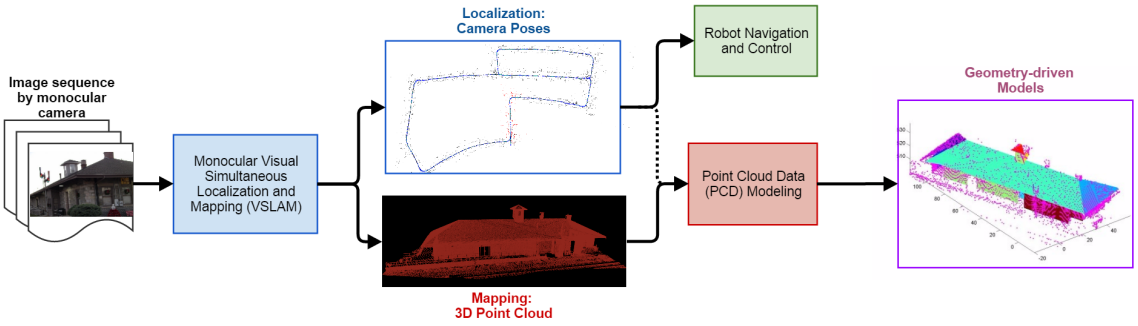


Figure 1: The pipeline of the solution to 3D modeling from image sequence. The whole pipeline is composed of two key components: monocular visual simultaneous localization and mapping, and point cloud data modeling.

environment simultaneously, given a sequence of images captured by a calibrated monocular camera. A similar problem is the "Structure from Motion (SfM)" [51] problem, which is often referred in computer vision community.

Figure 2 visualizes the dynamics and measurement process underlying this problem. As depicted in the figure, the camera moves in the 3D space with *unknown* 6 degrees-of-freedom (DOF) poses (3 DOFs in translation and 3 DOFs in orientation). The camera is assumed to be *calibrated*, meaning its intrinsic parameters are known. At each time instance, the camera captures objects in the 3D space by projecting them onto the 2D image plane through a pinhole projection model. The pinhole project is characterized by the camera $SE\langle 3\rangle$ transformation followed by a 3D-to-2D projective transformation. However, the connections between points across images, and the connections between 2D image points and the actual 3D objects, are both *unknown*.

The VSLAM problem is usually tackled in the following way. Given a frame during the VSLAM process, visual keypoints are first detected. Then the *data-association* problem is solved by matching the visual keypoints between the current and previous frames (as shown in Figure 3) as well as matching the 2D visual features with the 3D points based on the current estimates. With the data-association, an $SE\langle 3\rangle$

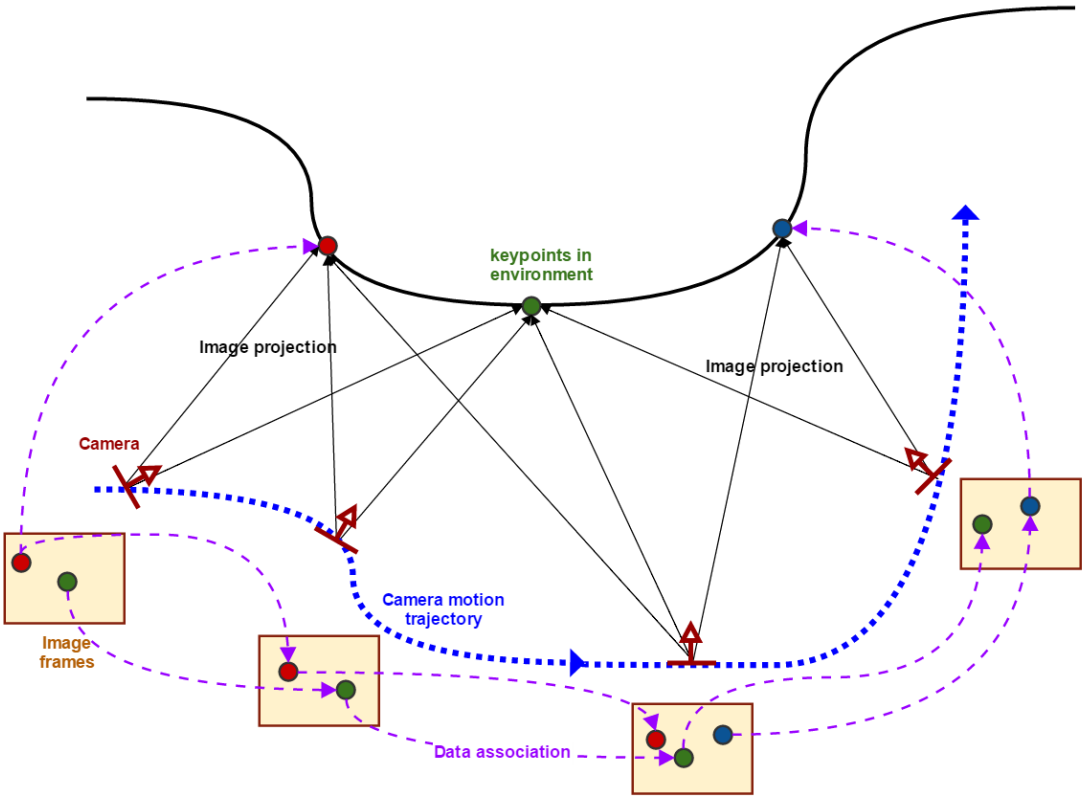


Figure 2: An illustration of the dynamics and measurements process involved in the VSLAM problem.

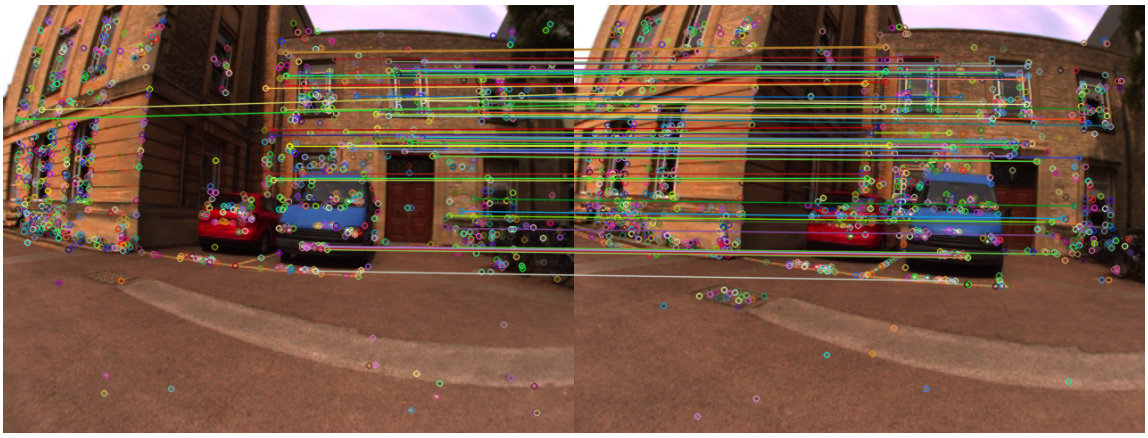


Figure 3: Matching the visual features across two consecutive frames, as part of the data-association process.

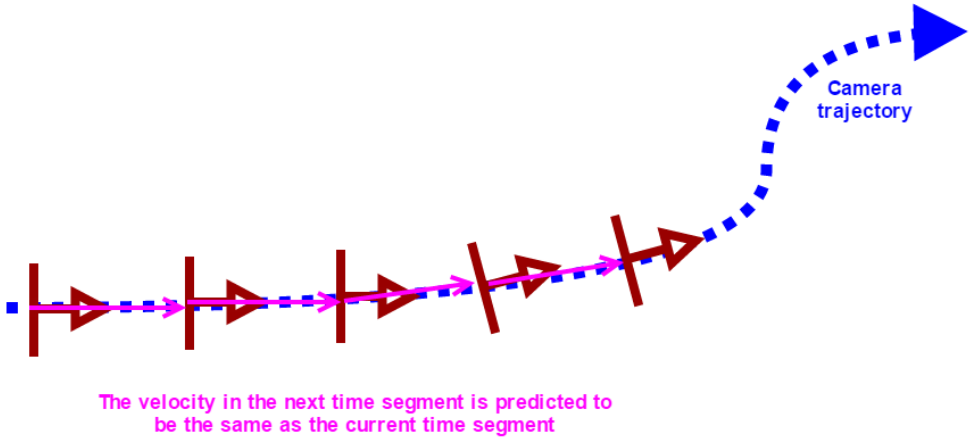


Figure 4: Constant velocity assumption

transformation of the camera poses from the previous frame to the current frame (i.e. localization), as well as the 3D positions of the features (i.e. mapping), can then be solved. This process continues to the next frame by predicting the next transformation. A key assumption here is the "constant velocity assumption" [27], as depicted in Figure 4, which assumes that for a camera with high enough frame rates, two consecutive transformations are approximately the same with very small deviations.

Researchers have proposed various ways to solve the core estimation problem of localization and mapping. In the early VSLAM systems, this problem is solved using

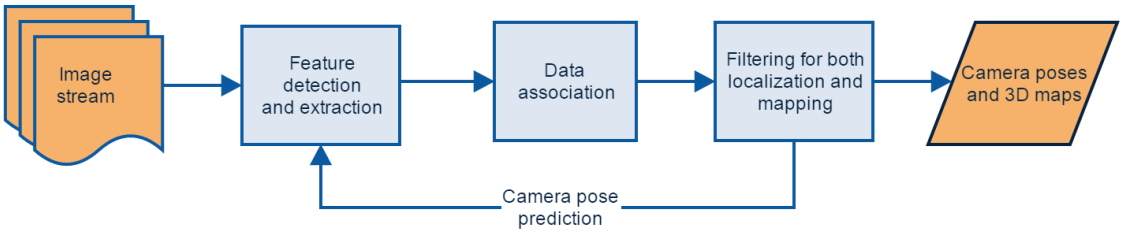


Figure 5: A typical design of a filtering based VSLAM system.

filtering methods. In the state-space model, the state vector is composed of both the camera state (including the pose and velocity) and 3D positions of the features. Figure 5 depicts a typical design of the filtering based VSLAM system. A milestone work is the "MonoSLAM" [27], in which the estimation is solved using an Extended Kalman Filter (EKF). Figure 6 shows an example of EKF-VSLAM system [81] in action, in which each feature is tracked and indexed across the whole trajectory. The major disadvantage of EKF-VSLAM is that it does not scale well as the map grows. Accordingly, [34] proposed to improve the filtering based VSLAM using a particle filter. Other methods also improve the filtering based VSLAM in different aspects: using a better landmark parametrization [21, 88], using other features like edges [33], reformulating the estimation problem as a smoothing [28]/ incremental smoothing [59, 58] problem, etc.

In recent years, as the computation capability develops, more sophisticated designs of VSLAM systems have been proposed. The most prevalent framework is the keyframe based bundle adjustment (BA) [101]. BA minimizes the image reprojection errors using nonlinear least-squares algorithms [62]. The keyframe approach selects a subset of frames for which the BA is solved, and thus greatly reduces the computational intensity compared to the frame-to-frame approach. Figures 7 illustrates a typical system design of keyframe based bundle adjustment VSLAM. The design features in multiple processing threads including a faster thread for frame-to-frame motion tracking, and a slower thread for mapping on keyframes with BA [60]. More

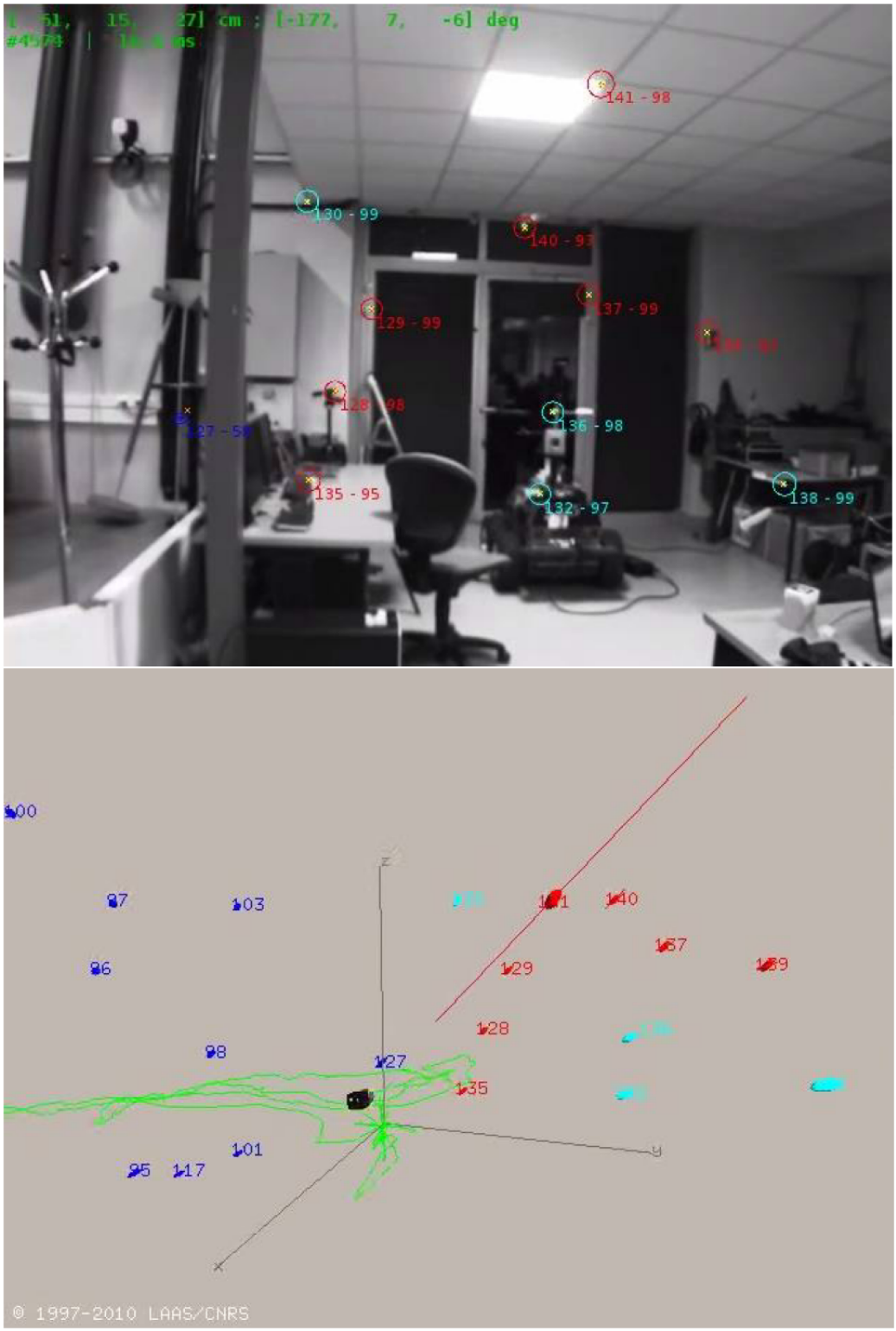


Figure 6: An example of EKF-VSLAM: RT-SLAM system [81]. Top: current frame plotted with tracked visual features. Bottom: estimated camera trajectory and 3D positions of the features.

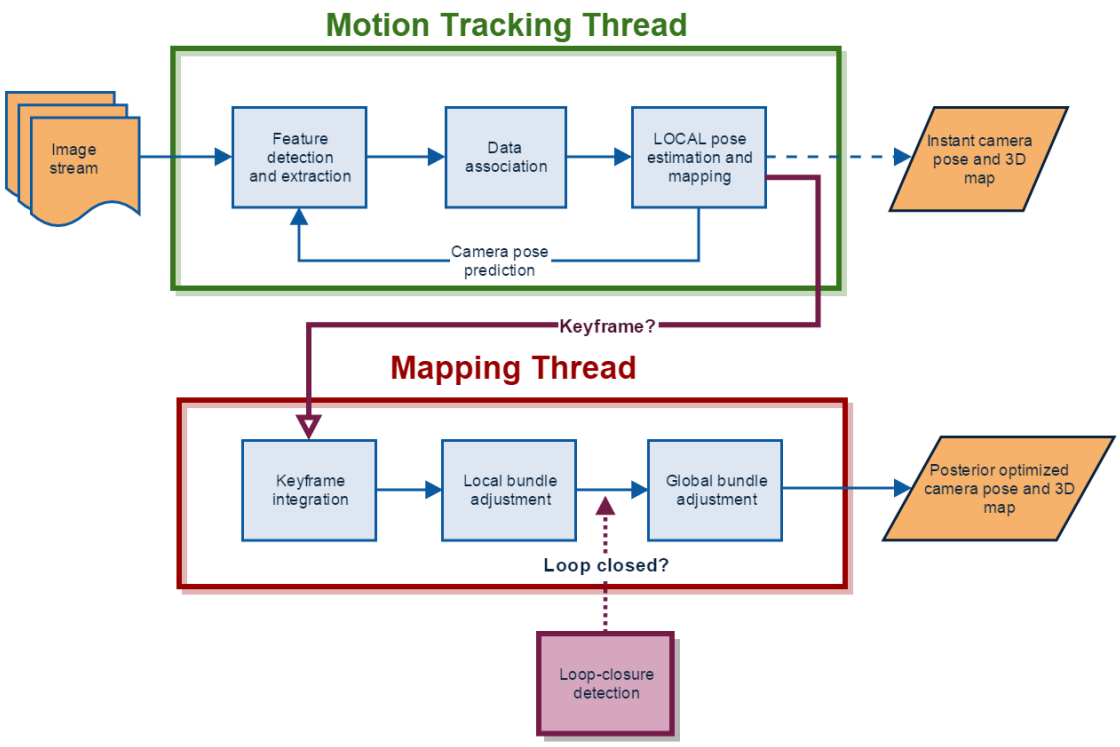


Figure 7: A typical design of a keyframe based bundle adjustment VSLAM system.

recent systems also have a third thread for loop-closure detection [70]. This type of system will be discussed in details in Chapter 4. Figure 8 provides an example from a representative keyframe based BA VSLAM system, from which it can be seen that keyframe based BA systems are able to track and map more points than filtering based systems.

1.1.2 Challenges in VSLAM and my solution

From the previous section, one can see that no matter what the system design is, the localization and mapping directly depend on the data-association results. However, in reality data-association may not be completely correct. For example, outliers can happen due to the feature mis-matching. Traditionally people use randomized data-driven methods, such as Random sample consensus (RANSAC) [39, 22] to remove the outliers. Such randomized methods are computational intensive. Moreover, even if the feature matching is correct without any outliers, not all the features contribute

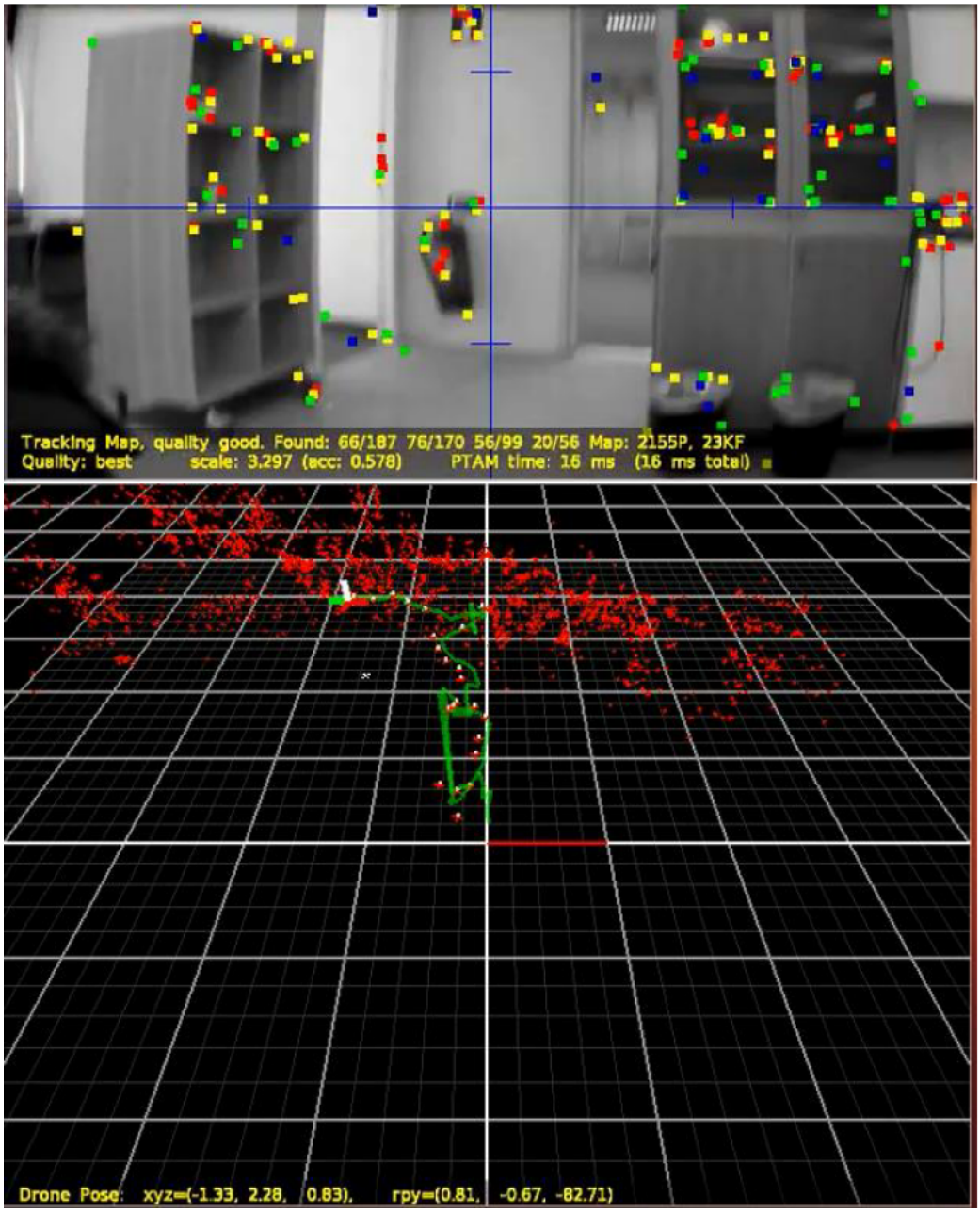


Figure 8: An example of keyframe based BA VSLAM: PTAM (parallel tracking and mapping) system [60]. Top: current frame plotted with tracked visual features. Bottom: estimated camera trajectory and 3D positions of the features.

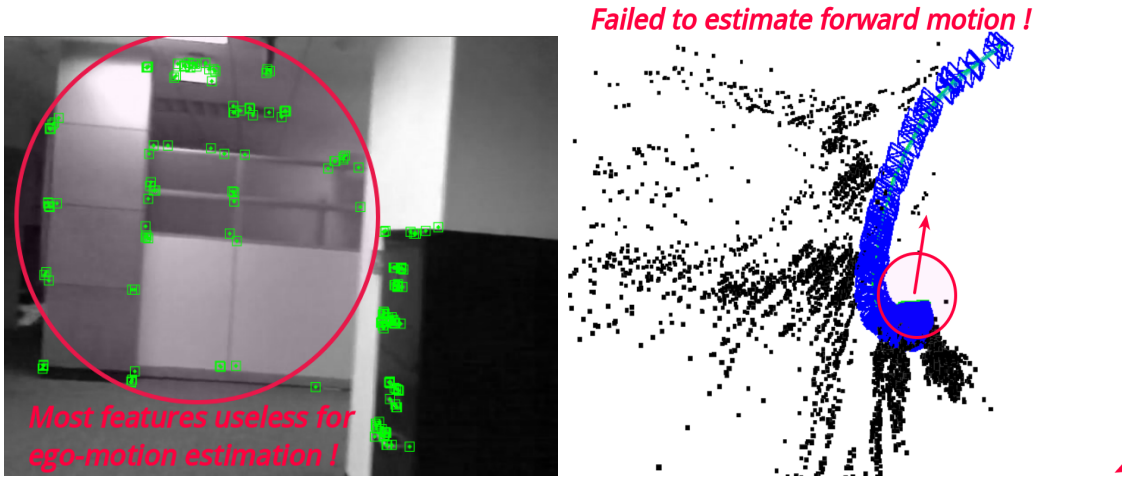


Figure 9: An example showing that not all the features contribute in the same level to the localization accuracy. Left: features detected and tracked in the current frame. Right: the incorrect camera trajectory estimated.

in the same level to the localization accuracy and thus the mapping accuracy. An example is described by Figure 9. In the example, the camera is moving forward approximately along the optical axis. Most of the visual features are detected from the area in front of the camera. These features are barely useful in the ego-motion estimation (heuristically they have very small baselines across frames). This results in failure of estimating the camera forward motion. Therefore, it is important to select the features with the most utility to the estimation process. Previously feature selection approaches such as information gain based methods [25] were proposed. However, as Chapter 4 will show, these approaches may not effectively solve the problem. Thus, selecting the useful features remains an open problem.

My solution differs from previous approaches in that it selects the most useful features via system observability. Observability is a condition which, when achieved, means that an estimation process will arrive at the correct estimate [121]. Often associated with observability are means to estimate the degree of observability, which provide a means to rate the degrees of *informativeness* of measurements. By analyzing the SLAM process we establish observability conditions that make the SLAM system

completely observable while also providing an observability score. Using the score, we are able to select a subset of tracked features to form a strongly observable subsystem. This subsystem provides the estimate of the whole SLAM state but with the best conditioning for improved noise tolerance and accuracy.

1.2 Point cloud modeling

In this section I will briefly discuss the PCD modeling problem and my solution.

1.2.1 PCD modeling Problem

The point cloud data (PCD), encoding raw spatial information and possibly with color information, has attracted more and more research effort in the recent years, as the depth sensors became more popular and cheaper. In this thesis, the scope is restricted to the PCDs with only spatial information but no color information. Moreover, the work focuses on the applications of as-built modeling for civil infrastructures.

A PCD can be captured using direct depth sensors such as professional total stations, time-of-flight cameras, or integrated RGB-D sensors (e.g. Microsoft Kinect). Alternatively, a PCD can be generated indirectly from image sequences using VSLAM or SfM systems. The most obvious advantages of using a VSLAM system versus a direct sensor are that, VSLAM can generate PCD in a much larger scale than the time-of-flight cameras or RGB-D sensors, and VSLAM uses a much cheaper sensor, i.e. a monocular camera, than a professional total station.

The problem of PCD modeling in the scope of this thesis, is to convert the raw PCDs from VSLAM with only spatial information, to augmented models with computer-aided design (CAD) representations and the semantic labels.

1.2.2 Challenges in modeling PCD from VSLAM

Figures 10 and 11 show two PCDs collected using a profession total station and a VSLAM system respectively. A PCD generated from VSLAM algorithms is typically

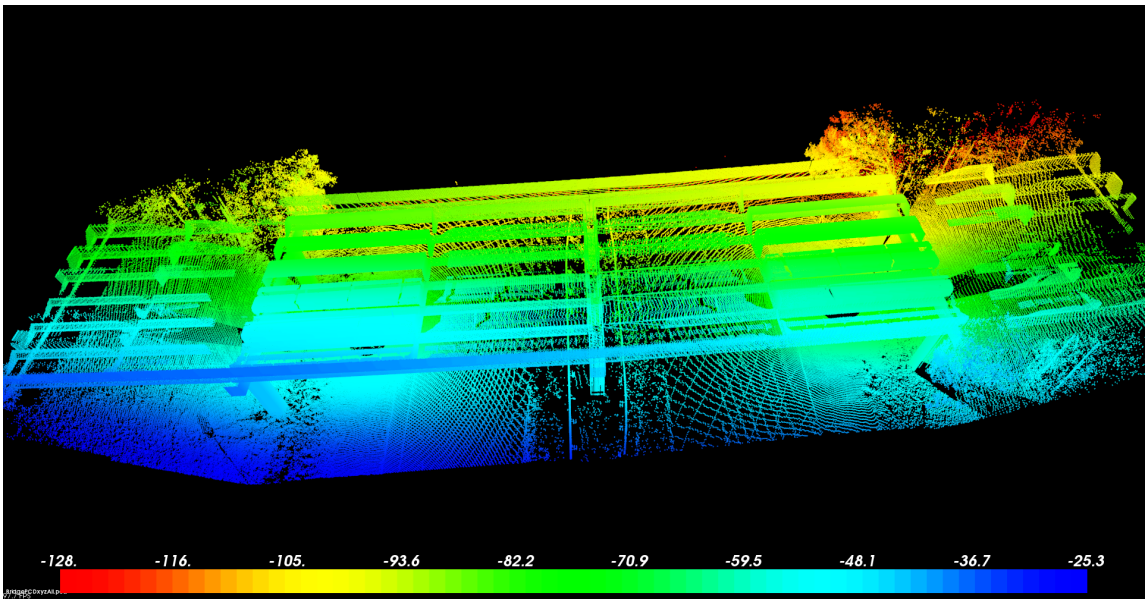


Figure 10: A PCD of a bridge, captured by a profession total station.

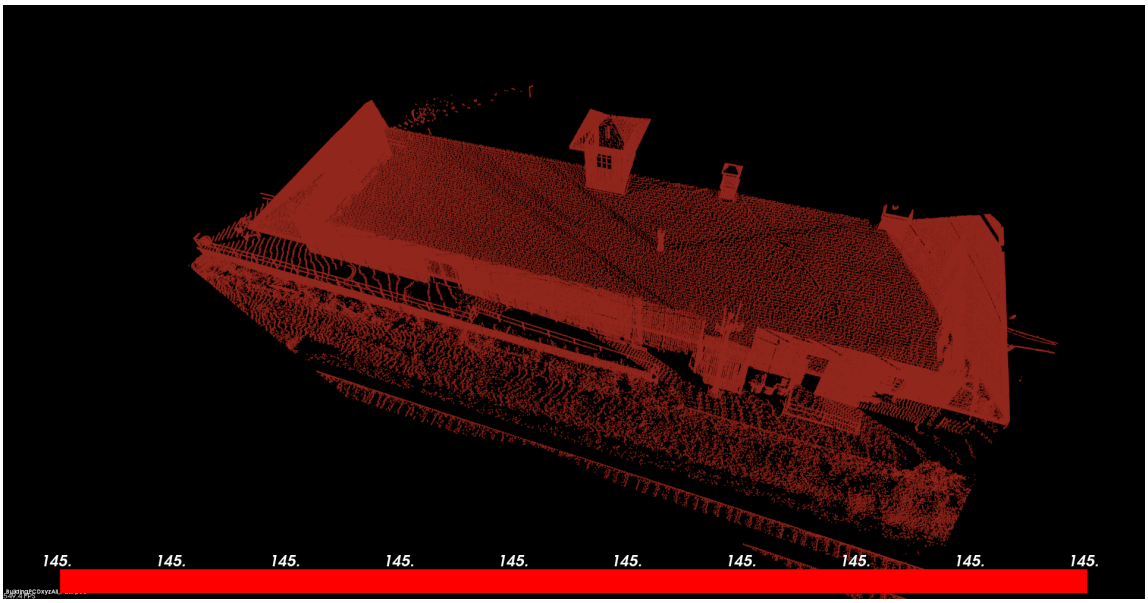


Figure 11: A PCD of a building, generated using VSLAM techniques.

of lower quality than a PCD captured with profession instruments. This lower quality results in challenges of modeling the PCDs from VSLAM, which can be summarized as:

1. A PCD from VSLAM has outlier points. This is due to the wrong data association or mapping estimation.

2. The distribution of 3D points in a PCD from VSLAM is not uniform. This is because VSLAM mapping relies on visual keypoints, which are not uniformly distributed on a real object especially man-made objects.

3. A PCD from VSLAM is usually incomplete, due to occlusions, feature-less facets, etc.

4. A PCD from VSLAM has a much larger reconstruction error (in centimeter levels) than a PCD from professional instruments (in millimeter levels). In nature, the pixel quantization effect of images limits the PCD accuracy.

5. The PCD has no scale information, as shown in Figure 11, because VSLAM is ambiguous to the absolute scale.

In addition to the above, a PCD of an infrastructure is of large scale, typically with millions of points. This makes the problem more challenging.

1.2.3 My strategists

My solution is based on a key observation that, most of the shapes in civil infrastructures are composed by surface primitives which can be represented by either planar patches or the more general quadratic surfaces. By imposing the constraints of surface primitives, together with the robust estimation framework, we can effectively handle the issues of outliers, non-uniform distributions of points, and large reconstruction errors. Moreover, models with only surface primitives can reach a much more compact

representation of shapes compared to a large collection of points. Notice that the scale ambiguity is not handled in my method, but can be easily addressed by recognizing an object in the scene with known absolute scale. On top of this geometry-driven approach, I further enhance the model with semantic labels.

1.3 Outline of the thesis

The rest of this thesis is organized as follows.

Chapter 2, 3 and 4 are devoted to the problem of feature selection in VSLAM. In particular,

- In Chapter 2, I start from the $SE\langle 3\rangle$ dynamic model and camera pinhole projection, to reach a piece-wise linear system modeling the VSLAM. Based on this model, I further derive two necessary conditions to make the VSLAM system completely observable with a detailed proof.

- In Chapter 3, I develop an algorithm called "Optimally Observable and Minimal Cardinality (OOMC) SLAM" based on the first condition, which tells us the minimum required number of features to make the instantaneous system completely observable. The OOMC algorithm is casted in the EKF-VSLAM framework to improve VSLAM localization accuracy and data-association.

- In Chapter 4, I further develop a more sophisticated algorithm dubbed "good features (GF) SLAM" to exploit the second condition of complete observability, which tells us as few as one tracked anchor across two time segments (three frames) is required to form a completely observable system. Based on this, I propose a score to rank each individual feature, along with an efficient algorithm to compute this score. I further propose an approach to group features when not enough good features are present, so as to reach a better conditioned system. Theoretical support is provided to prove that the approach is near-optimal.

Finally, I integrate the GF method into the two prevalent VSLAM systems: filtering based VSLAM and keyframe based BA VSLAM. Performance of the GF method is evaluated on large scale benchmark data sets, demonstrating that it outperform the former state-of-the-art.

Chapter 5 and Chapter 6 are for the problem of PCD modeling. In particular,

- In Chapter 5 I design an algorithm for modeling the PCD with planar patches. The algorithm is based on a sparsity-inducing optimization to retrieve the linear subspaces embedded in a raw point cloud. Then I apply the method on a PCD which is generated by VSLAM for a real building. Evaluation metrics are further proposed to quantitatively evaluate the performance. The experiments show promising results with better performance compared to other methods.

- Chapter 6 further extends the PCD modeling from planar patches to quadratic surface primitives. I use the general quadric model to accommodate widely-seen shapes in civil infrastructure; then develop an algorithm to detect and robustly fit these surface primitives to the PCD. In addition, I present the method for learning simple semantic labels for the PCD with surface primitives.

Finally, the whole thesis is concluded in Chapter 7 with further discussions and directions for future research.

CHAPTER II

VISUAL SLAM SYSTEM MODELING AND COMPLETELY OBSERVABLE CONDITIONS

During the VSLAM process, the measured image feature points used for estimation have varying degrees of informativeness and are subject to different levels of noise. Some of the detected and tracked feature points are detrimental to estimation and should be rejected as viable measurements. Therefore, finding the features that provide the best values for estimation is important when SLAM is to be used for practical purposes.

As the foundation, this chapter focuses on the state-space modeling of $SE\langle 3\rangle$ VSLAM and deriving the conditions to make the system completely observable. Particularly, given the nonlinear and time-varying nature of $SE\langle 3\rangle$ VSLAM, I model it as a piece-wise linear system. Such an approximation preserves the system characteristics but with little loss of accuracy [46]. Based on this, I examine its observability using the stripped observability matrix (SOM). Since the system is completely observable when its SOM is full-rank, by checking the full-rank conditions I obtain two necessary conditions for the system to be completely observable.

2.1 Background

System theory, especially observability theory, have been seen in robotics literature, but mostly restricted to 1D SLAM [45] and 2D (planar motion) SLAM [3, 64, 76, 106, 2] or for multi-sensor based SLAM [14, 53, 13], rather than monocular camera SLAM on $SE\langle 3\rangle$. Moreover, observability theory has mainly been for full rank observability condition analysis, much as in [106] which analyzes bearings-only SLAM.

For visual SLAM on $SE\langle 3\rangle$, [72] provides an analysis of observability, but it is for stereo-vision SLAM with planar displacement instead of full 6-DOF. [90] discusses observability tests for camera ego-motion from perspective views at time instances. Few works use observability in algorithm design rather than merely observable condition analysis /observability tests. [54] presented a framework for improving the consistency of EKF-based *planar SLAM*.The method focused more on the linearization step. EKF linearization points are selected in a way that ensures the observable subspace is of appropriate dimension for the linearized system. The authors solved this problem via two methods: observability constraints and First-Estimates Jacobian.

The work in this chapter was more inspired by [106] and the fundamental work in [46]. [46] investigates into the properties of piece-wise linear systems (PWLS). It proves that the Total Observability Matrix (TOM) can be used to characterize the observability properties of a PWLS; more importantly it proves that under certain conditions, Stripped Observability Matrix (SOM) can be used as a proxy of TOM, which avoids the expensive computation of TOM but with little loss in system characteristics. The work in [106], though mainly focuses on planar SLAM, formulate the continuous-time 2D SLAM system as a PWLS and successfully applies the SOM theories to examine the observable conditions of planar SLAM.

2.2 VSLAM System Modeling

2.2.1 Motion and Observations for $SE\langle 3\rangle$ SLAM

Here, the SLAM scenario with features and anchors is considered. The SLAM system dynamics are modeled under the hybrid $SE\langle 3\rangle$ state common to robotics (position in world frame $\mathcal{W}$, with orientation in body frame $\mathcal{R}$) [71], with a perspective camera measurement model.

2.2.1.1 Dynamic and Measurement Models

For a system with discrete observations, a constant velocity motion model suffices [27]. Accordingly, given the $SE\langle 3\rangle$ position and orientation $\mathbf{r}_{\mathcal{R}_k}^{\mathcal{W}}$, $\mathbf{q}_{\mathcal{R}_k}^{\mathcal{W}}$ (vector, quaternion), and associated velocities $\mathbf{v}_{\mathcal{R}_k}^{\mathcal{W}}$, $\omega^{\mathcal{R}}$, at time t, the camera state

$$\mathbf{x}_{\mathcal{R}_k}^{\mathcal{W}} = \begin{pmatrix} \mathbf{r}_{\mathcal{R}_k}^{\mathcal{W}} & \mathbf{q}_{\mathcal{R}_k}^{\mathcal{W}} & \mathbf{v}_{\mathcal{R}_k}^{\mathcal{W}} & \omega^{\mathcal{R}} \end{pmatrix}^{\intercal} \tag{1}$$

is updated per:

$$\mathbf{x}_{\mathcal{R}_{k+1}}^{\mathcal{W}} = \begin{pmatrix} \mathbf{r}_{\mathcal{R}_{k+1}}^{\mathcal{W}} \\ \mathbf{q}_{\mathcal{R}_{k+1}}^{\mathcal{W}} \\ \mathbf{v}_{\mathcal{R}_{k+1}}^{\mathcal{W}} \\ \omega^{\mathcal{R}} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_{\mathcal{R}_k}^{\mathcal{W}} + (\mathbf{v}_{\mathcal{R}_k}^{\mathcal{W}} + \mathbf{V}^{\mathcal{W}})\Delta t \\ \mathbf{q}_{\mathcal{R}_k}^{\mathcal{W}} \times \exp\left(\left[\omega^{\mathcal{R}} + \mathbf{\Omega}^{\mathcal{R}}\right]\Delta t\right) \\ \mathbf{v}_{\mathcal{R}_k}^{\mathcal{W}} + \mathbf{V}^{\mathcal{W}} \\ \omega^{\mathcal{R}} + \mathbf{\Omega}^{\mathcal{R}} \end{pmatrix}, \tag{2}$$

where $\mathbf{V}^{\mathcal{W}}, \mathbf{\Omega}^{\mathcal{R}}$ are zero-mean Gaussian noise. The measurement model for the i-th feature ${}^{(i)}\mathbf{p}_k^{\mathcal{W}} \in \mathbb{R}^3$ is pinhole projection:

$${}^{(i)}\mathbf{p}^{\mathcal{R}_k} = [p_x^{\mathcal{R}_k}, p_y^{\mathcal{R}_k}, p_z^{\mathcal{R}_k}]^{\intercal} = \mathrm{R}_{\mathcal{W}}^{\mathcal{R}_k}\left(\left(\mathbf{q}_{\mathcal{R}_k}^{\mathcal{W}}\right)^{-1}\right)\left({}^{(i)}\mathbf{p}_k^{\mathcal{W}} - \mathbf{r}_{\mathcal{R}_k}^{\mathcal{W}}\right), \tag{3}$$

$$\mathbf{h}_i^{\mathcal{R}_k} = \text{Distort}\left[\begin{pmatrix} u \\ v \end{pmatrix}\right] = \text{Distort}\left[\begin{pmatrix} u_0 - fk_u\frac{p_x^{\mathcal{R}_k}}{p_z^{\mathcal{R}_k}} \\ v_0 - fk_v\frac{fp_y^{\mathcal{R}_k}}{p_z^{\mathcal{R}_k}} \end{pmatrix}\right], \tag{4}$$

where $\mathrm{R}(\mathbf{q})$ is the rotation matrix of $\mathbf{q}$; fk_u, fk_v, u_0, v_0 are the camera intrinsic parameters; and Distort$[\cdot]$ is nonlinear image distortion [26]. Given the camera position $\mathbf{r}_{\mathcal{R}_k}^{\mathcal{W}}$ and orientation described by the quaternion $\mathbf{q}_{\mathcal{R}_k}^{\mathcal{W}}$, Equation (3) formulates the relationship between the 3D coordinates ${}^{(i)}\mathbf{p}_k^{\mathcal{W}}$ of a feature point in the world frame and the transformed 3D coordinates ${}^{(i)}\mathbf{p}^{\mathcal{R}_k}$ in the camera/robot frame. Then, as described by Equation (4), the 3D coordinates ${}^{(i)}\mathbf{p}^{\mathcal{R}_k}$ are projected to the 2D image place through a 3D-to-2D projective transformation followed by a 2D-to-2D nonlinear distortion. The camera intrinsic parameters, including the (skewed) focal lengths (fk_u and fk_v) and the principal point offsets (u_0 and v_0) fully determine the relationship between the 3D coordinates ${}^{(i)}\mathbf{p}^{\mathcal{R}_k}$ and the undistorted 2D image coordinates $(u, v)^{\intercal}$. The final measurement $\mathbf{h}_i^{\mathcal{R}_k} = (u_d, v_d)^{\intercal} \in \mathbb{R}^2$ is obtained by distorting $(u, v)^{\intercal}$.

2.2.1.2 Piece-wise Linear System (PWLS) modeled for SLAM

Assume the system has N_f features and N_a anchors. An anchor is a 3D point in $\mathcal{W}$ whose position is known, while a feature is 3D point whose position is not certain (at least initially). Both are observed by the camera as per Equation (4).

For the k-th time segment $\mathcal{T}_k \equiv [t_k, t_{k+1})$ (from time k to time $k+1$), the dynamics of the whole system with input $\mathbf{u}_k$ are

$$\mathbf{X}_{k+1}^{\mathcal{W}} \triangleq \begin{pmatrix} \mathbf{x}_{\mathcal{R}_{k+1}}^{\mathcal{W}} \\ \mathbf{P}_{k+1}^{\mathcal{W}} \end{pmatrix} = \mathrm{f}\left(\begin{pmatrix} \mathbf{x}_{\mathcal{R}_k}^{\mathcal{W}} \\ \mathbf{P}_{k}^{\mathcal{W}} \end{pmatrix} \middle| \mathbf{A}_k^{\mathcal{W}} \right) + \mathbf{u}_k, \tag{5}$$

$$\mathbf{h}^{\mathcal{R}_{k+1}} = \mathrm{h}^{\mathcal{R}_{\mathbf{k+1}}}\left(\begin{pmatrix} \mathbf{x}_{\mathcal{R}_k}^{\mathcal{W}} \\ \mathbf{P}_{k}^{\mathcal{W}} \end{pmatrix} \middle| \mathbf{A}_k^{\mathcal{W}} \right), \tag{6}$$

where $\mathbf{P}_k^{\mathcal{W}} \triangleq ({}^{(1)}\mathbf{p}_k^{\mathcal{W}}, {}^{(2)}\mathbf{p}_k^{\mathcal{W}}, ..., {}^{(N_f)}\mathbf{p}_k^{\mathcal{W}})^\intercal \in \mathbb{R}^{3N_f}$ is the map state vector by stacking the feature vectors, $\mathbf{A}_k^{\mathcal{W}} \triangleq ({}^{(1)}\mathbf{a}_k^{\mathcal{W}}, {}^{(2)}\mathbf{a}_k^{\mathcal{W}}, ..., {}^{(N_a)}\mathbf{a}_k^{\mathcal{W}})^\intercal \in \mathbb{R}^{3N_a}$ is the anchor state vector, and $\mathbf{h}^{\mathcal{R}_{k+1}} \triangleq (\mathbf{h}_1^{\mathcal{R}_{k+1}}, \mathbf{h}_2^{\mathcal{R}_{k+1}}, ..., \mathbf{h}_N^{\mathcal{R}_{k+1}})^\intercal \in \mathbb{I}^{2(N_f+N_a)}$ is the measurement vector at time $k+1$ with measurements from both features and anchors.

With the smooth motion assumption, the system at each time segment $\mathcal{T}_k$ is linearized. The linearized systems across time segments form the **piece-wise linear system (PWLS)** in (7) below, which approximates the time-varying system in (5).

$$\begin{cases} \mathbf{X}_{k+1}^{\mathcal{W}} = \mathrm{F}^{\mathcal{R}_\mathbf{k}} \mathbf{X}_k^{\mathcal{W}} + \mathbf{u}_k \\ \delta\mathbf{h}^{\mathcal{R}_k} = \mathrm{H}^{\mathcal{R}_\mathbf{k}} \mathbf{X}_k^{\mathcal{W}} \end{cases} \quad \text{for } t \in \mathcal{T}_k \tag{7}$$

The PWLS preserves the characteristic behavior of the original time-varying system with little loss of accuracy [46].

2.2.2 Computation of Process and Measurement Matrices in VSLAM PWLS

2.2.2.1 Process Matrix in PWLS

For $\mathfrak{S}_{\text{SLAM}}(\mathcal{F}; \mathcal{G}_{\text{cam}}^{SE\langle 3\rangle})$ with N_f features and N_a anchors, the PWLS matrices are

$$\mathrm{F}^{\mathcal{R}_{\mathbf{k}}} = \begin{pmatrix} \mathrm{F}_{\mathbf{x}_{\mathcal{R}_k}^{\mathcal{W}}} & \mathbf{0}_{13\times 3N_f} \\ \mathbf{0}_{3N_f\times 13} & \mathbf{I}_{3N_f\times 3N_f} \end{pmatrix}, \tag{8}$$

and

$$\mathrm{F}_{\mathbf{x}_{\mathcal{R}_k}^{\mathcal{W}}} = \left(\begin{array}{cc|cc} \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 4} & \Delta t\cdot\mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{4\times 3} & \mathbf{Q}_{4\times 4} & \mathbf{0}_{4\times 3} & \mathbf{\Omega}_{4\times 3} \\ \hline \mathbf{0}_{6\times 7} & & \mathbf{I}_{6\times 6} & \end{array}\right)_{13\times 13}, \tag{9}$$

For simplicity, denote $\omega = \omega^{\mathcal{R}}$.

To compute $\mathbf{Q}$, let $\Delta\omega = \omega^{\mathcal{R}} \times \Delta t$, and

$$\Delta\mathbf{q}_\omega = \begin{pmatrix} \cos(\frac{||\Delta\omega||}{2}) \\ \sin(\frac{||\Delta\omega||}{2})\cdot\frac{\Delta\omega}{||\Delta\omega||} \end{pmatrix} \triangleq (q_\omega^R, q_\omega^x, q_\omega^y, q_\omega^z)^\intercal \in \mathbb{R}^{4\times 1} \tag{10}$$

then,

$$\mathbf{Q} = \begin{pmatrix} q_\omega^R & -q_\omega^x & -q_\omega^y & -q_\omega^z \\ q_\omega^x & q_\omega^R & q_\omega^z & -q_\omega^y \\ q_\omega^y & -q_\omega^z & q_\omega^R & q_\omega^x \\ q_\omega^z & q_\omega^y & -q_\omega^x & q_\omega^R \end{pmatrix}, \tag{11}$$

To compute $\mathbf{\Omega}$, let $\mathbf{q}_{\mathcal{R}_k}^{\mathcal{W}} = (q_k^R, q_k^x, q_k^y, q_k^z)^\intercal$; then

$$\mathbf{\Omega} = \begin{pmatrix} q_k^R & -q_k^x & -q_k^y & -q_k^z \\ q_k^x & q_k^R & -q_k^z & q_k^y \\ q_k^y & q_k^z & q_k^R & -q_k^x \\ q_k^z & -q_k^y & q_k^x & q_k^R \end{pmatrix} \cdot f_q(\omega, \Delta t). \tag{12}$$

Given Δt and ω (and thus $||\omega||$ is known), $f_q(\omega, \Delta t)$ can be expressed with the following generator functions:

$$\zeta_1(x) = -\frac{\Delta t}{2} \cdot \frac{x}{||\omega||} \cdot \sin\left(||\omega|| \frac{\Delta t}{2}\right) \tag{13}$$

$$\zeta_2(x) = \frac{\Delta t}{2} \cdot \frac{x^2}{||\omega||^2} \cdot \cos\left(||\omega|| \frac{\Delta t}{2}\right) + \frac{1}{||\omega||}\left[1 - \frac{x^2}{||\omega||^2} \cdot \sin\left(||\omega|| \frac{\Delta t}{2}\right)\right] \tag{14}$$

$$\zeta_3(x, y) = \frac{xy}{||\omega||^2}\left[\frac{\Delta t}{2}\cos\left(\frac{||\omega||\Delta t}{2}\right) - \frac{1}{||\omega||}\sin\left(\frac{||\omega||\Delta t}{2}\right)\right] \tag{15}$$

where x, y are elements in ω, and thus $\zeta_1(x), \zeta_2(x), \zeta_3(x, y) \in \mathbb{R}$. With these generator functions, we have

$$f_q(\omega, \Delta t) = \begin{pmatrix} \zeta_1(\omega_1) & \zeta_1(\omega_2) & \zeta_1(\omega_3) \\ \zeta_2(\omega_1) & \zeta_3(\omega_1, \omega_2) & \zeta_3(\omega_1, \omega_3) \\ \zeta_3(\omega_2, \omega_1) & \zeta_2(\omega_2) & \zeta_3(\omega_2, \omega_3) \\ \zeta_3(\omega_3, \omega_1) & \zeta_3(\omega_3, \omega_2) & \zeta_2(\omega_3) \end{pmatrix} \tag{16}$$

2.2.2.2 Measurement Matrix in PWLS

The measurement Jacobian is

$$\mathrm{H}^{\mathcal{R}_k} = \tag{17}$$

$$\left(\begin{array}{ccc|ccc} \frac{\partial \mathbf{h}_1^{\mathcal{R}_k}}{\partial \mathbf{r}_{\mathcal{R}}^{\mathcal{W}}} & \frac{\partial \mathbf{h}_1^{\mathcal{R}_k}}{\partial \mathbf{q}_{\mathcal{R}}^{\mathcal{W}}} & \mathbf{0}_{2\times 6} & \frac{\partial \mathbf{h}_1^{\mathcal{R}_k}}{\partial \mathbf{p}_k^{\mathcal{W}}} & \cdots & \mathbf{0}_{2\times 3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{h}_{N_f}^{\mathcal{R}_k}}{\partial \mathbf{r}_{\mathcal{R}}^{\mathcal{W}}} & \frac{\partial \mathbf{h}_{N_f}^{\mathcal{R}_k}}{\partial \mathbf{q}_{\mathcal{R}}^{\mathcal{W}}} & \mathbf{0}_{2\times 6} & \mathbf{0}_{2\times 3} & \cdots & \frac{\partial \mathbf{h}_{N_f}^{\mathcal{R}_k}}{\partial \mathbf{p}_k^{\mathcal{W}}} \\ \hline \frac{\partial \mathbf{h}_{(N_f+1)}^{\mathcal{R}_k}}{\partial \mathbf{r}_{\mathcal{R}}^{\mathcal{W}}} & \frac{\partial \mathbf{h}_{(N_f+1)}^{\mathcal{R}_k}}{\partial \mathbf{q}_{\mathcal{R}}^{\mathcal{W}}} & \mathbf{0}_{2\times 6} & & & \\ \vdots & \vdots & \vdots & & \mathbf{0}_{2N_a\times 3N_f} & \\ \frac{\partial \mathbf{h}_{(N_f+Na)}^{\mathcal{R}_k}}{\partial \mathbf{r}_{\mathcal{R}}^{\mathcal{W}}} & \frac{\partial \mathbf{h}_{(N_f+Na)}^{\mathcal{R}_k}}{\partial \mathbf{q}_{\mathcal{R}}^{\mathcal{W}}} & \mathbf{0}_{2\times 6} & & & \end{array}\right). \tag{18}$$

The first N_f rows are w.r.t. the features while the last N_a rows are w.r.t. the anchors. When $N_f = 0$, the measurement Jacobian for one anchor is:

$$\mathbf{H} = \begin{pmatrix} \frac{\partial \mathbf{h}^{\mathcal{R}_k}}{\partial \mathbf{r}_{\mathcal{R}}^{\mathcal{W}}} & \frac{\partial \mathbf{h}^{\mathcal{R}_k}}{\partial \mathbf{q}_{\mathcal{R}}^{\mathcal{W}}} & \mathbf{0}_{2\times 6} \end{pmatrix} \triangleq \begin{pmatrix} \frac{\partial h}{\partial r} & \frac{\partial h}{\partial \mathbf{q}} & \mathbf{0}_{2\times 6} \end{pmatrix} \tag{19}$$

. In details, the first block of $\mathbf{H}$ is

$$\frac{\partial h}{\partial r} = -\frac{\partial h}{\partial \mathbf{p}^{\mathcal{R}_k}} \cdot R_w^r = -\left(\frac{\partial h}{\partial h_u} \cdot \frac{\partial h_u}{\partial \mathbf{p}^{\mathcal{R}_k}}\right) \cdot \mathrm{R}\left[\mathbf{q}_{\mathcal{R}_k}^{\mathcal{W}}\right] \tag{20}$$

and

$$\frac{\partial h_u}{\partial \mathbf{p}^{\mathcal{R}_k}} = \begin{pmatrix} -\frac{f \cdot k_u}{p_z^{\mathcal{R}_k}} & 0 & -\frac{f \cdot k_u \cdot p_x^{\mathcal{R}_k}}{\left(p_z^{\mathcal{R}_k}\right)^2} \\ 0 & \frac{f \cdot k_v}{p_z^{\mathcal{R}_k}} & -\frac{f \cdot k_v \cdot p_y^{\mathcal{R}_k}}{\left(p_z^{\mathcal{R}_k}\right)^2} \end{pmatrix} \tag{21}$$

where $[p_x^{\mathcal{R}_k}, p_y^{\mathcal{R}_k}, p_z^{\mathcal{R}_k}]^\intercal = \mathbf{p}^{\mathcal{R}_k} = \left(\mathrm{R}\left[\mathbf{q}_{\mathcal{R}_k}^{\mathcal{W}}\right]\right)^{-1}\left(\mathbf{p}_k^{\mathcal{W}} - \mathbf{r}_{\mathcal{R}_k}^{\mathcal{W}}\right)$, and $\mathrm{R}[\mathbf{q}]$ is the conversion function converting the quaternion to rotation matrix as define in 22.

$$\mathrm{R}[\mathbf{q}] = \begin{pmatrix} \left(q^R\right)^2 + \left(q^x\right)^2 - \left(q^y\right)^2 - \left(q^z\right)^2 & 2(q^x q^y - q^R q^z) & 2(q^z q^x + q^R q^y) \\ 2(q^x q^y + q^R q^z) & \left(q^R\right)^2 - \left(q^x\right)^2 + \left(q^y\right)^2 - \left(q^z\right)^2 & 2(q^y q^z - q^R q^x) \\ 2(q^z q^x - q^R q^y) & 2(q^y q^z + q^R q^x) & \left(q^R\right)^2 - \left(q^x\right)^2 - \left(q^y\right)^2 + \left(q^z\right)^2 \end{pmatrix} \tag{22}$$

$\frac{\partial h}{\partial h_u}$ is computed according to the wide-angle camera model [26]:

$$\frac{\partial h}{\partial h_u} = \left(\mathbf{J}_{\text{undist}}\right)^{-1} \tag{23}$$

where $\mathbf{J}_{\text{undist}}$ is the Jacobian matrix of the distortion function.

Given the pixel measurement of the anchor on the current frame as $[z_x, z_y]$, and the camera intrinsic parameter set $[f, C_x, C_y, d_x, d_y, k_1, k_2]$, we have

$$x_d = (z_x - C_x)d_x, \tag{24}$$

$$y_d = (z_y - C_y)d_y, \tag{25}$$

$$r_d = d_x^2(z_x - C_x)^2 + dy^2(z_y - C_y)^2; \tag{26}$$

and

$$\frac{\partial h_{ux}}{\partial x} = (1 + k_1 r_d + k_2 r_d^2) + (z_x - C_x) \cdot (k_1 + 2k_2 r_d) \cdot (2d_x^2 z_x - C_x), \tag{27}$$

$$\frac{\partial h_{ux}}{\partial y} = (z_x - C_x) \cdot (k_1 + 2k_2 r_d) \cdot 2d_y^2(z_y - C_y), \tag{28}$$

$$\frac{\partial h_{uy}}{\partial x} = (z_y - C_y) \cdot (k_1 + 2k_2 r_d) \cdot 2d_x^2(z_x - C_x), \tag{29}$$

$$\frac{\partial h_{uy}}{\partial y} = (1 + k_1 r_d + k_2 r_d^2) + (z_y - C_y) \cdot (k_1 + 2k_2 r_d) \cdot 2d_y^2(z_y - C_y); \tag{30}$$

then,

$$\frac{\partial h}{\partial h_u} = (\mathbf{J}_{\text{undist}})^{-1} \tag{31}$$

$$= \begin{pmatrix} \frac{\partial h_{ux}}{\partial x} & \frac{\partial h_{ux}}{\partial y} \\ \frac{\partial h_{uy}}{\partial x} & \frac{\partial h_{uy}}{\partial y} \end{pmatrix}^{-1} \tag{32}$$

The second block of $\mathbf{H}$ is

$$\frac{\partial h}{\partial \mathbf{q}} = \frac{\partial h}{\partial \mathbf{p}^{\mathcal{R}_k}} \cdot \frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \mathbf{q}} = \frac{\partial h}{\partial h_u} \cdot \frac{\partial h_u}{\partial \mathbf{p}^{\mathcal{R}_k}} \cdot \frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \mathbf{q}} \tag{33}$$

where $\frac{\partial h}{\partial h_u}$ and $\frac{\partial h_u}{\partial \mathbf{p}^{\mathcal{R}_k}}$ have been discussed.

To compute $\frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \mathbf{q}}$, first denote the conjugate of $\mathbf{q}$ as:

$$\bar{\mathbf{q}} \triangleq \text{conj}(\mathbf{q}) = \begin{pmatrix} q^R, -q^x, -q^y, -q^z \end{pmatrix}^\intercal \tag{34}$$

and

$$\frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \bar{\mathbf{q}}^R} = \begin{pmatrix} 2\bar{q}^R & -2\bar{q}^z & 2\bar{q}^y \\ 2\bar{q}^z & 2\bar{q}^R & -2\bar{q}^x \\ -2\bar{q}^y & 2\bar{q}^x & 2\bar{q}^R \end{pmatrix} \tag{35}$$

$$\frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \bar{\mathbf{q}}^x} = \begin{pmatrix} 2\bar{q}^x & 2\bar{q}^y & 2\bar{q}^z \\ 2\bar{q}^y & -2\bar{q}^x & -2\bar{q}^R \\ 2\bar{q}^z & 2\bar{q}^R & -2\bar{q}^x \end{pmatrix} \tag{36}$$

$$\frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \bar{\mathbf{q}}^y} = \begin{pmatrix} -2\bar{q}^y & 2\bar{q}^x & 2\bar{q}^R \\ 2\bar{q}^x & 2\bar{q}^y & 2\bar{q}^z \\ -2\bar{q}^R & 2\bar{q}^z & -2\bar{q}^y \end{pmatrix} \tag{37}$$

$$\frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \bar{\mathbf{q}}^z} = \begin{pmatrix} -2\bar{q}^z & -2\bar{q}^R & 2\bar{q}^x \\ 2\bar{q}^R & -2\bar{q}^z & 2\bar{q}^y \\ 2\bar{q}^x & 2\bar{q}^y & 2\bar{q}^z \end{pmatrix}. \tag{38}$$

Let $\tilde{\mathbf{p}} = \mathbf{p}_k^{\mathcal{W}} - \mathbf{r}_{\mathcal{R}_k}^{\mathcal{W}}$, then $\frac{\partial h}{\partial \mathbf{q}}$ is computed as in Eq 39.

$$\frac{\partial h}{\partial \mathbf{q}} = \left(\frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \bar{q}^R} \cdot \tilde{\mathbf{p}} \mid \frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \bar{q}^x} \cdot \tilde{\mathbf{p}} \mid \frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \bar{q}^y} \cdot \tilde{\mathbf{p}} \mid \frac{\partial \mathbf{p}^{\mathcal{R}_k}}{\partial \bar{q}^z} \cdot \tilde{\mathbf{p}} \right)_{3\times 4} \cdot \begin{pmatrix} 1 & & & \\ & -1 & & \\ & & -1 & \\ & & & -1 \end{pmatrix} \tag{39}$$

2.3 VSLAM System Observability

The *observability* and *complete observability* of a system are defined as below.

Definition 2.3.1. *A system is **observable** at time t_0 if the state vector at time t_0, $\mathbf{x}(t_0)$, can be determined from the output function $\mathbf{y}_{[t_0,t_1]}$ (or the output sequence in discrete time case), where $t_1, t_0 < t_1$ is some finite time. If this is true for all t_0 and $\mathbf{x}(t_0)$, the system is said to be **completely observable**.*

For a discrete PWLS, the observability condition is characterized by the Total Observability Matrix (TOM).

Definition 2.3.2. *[46] The Total Observability Matrix (TOM) of a discrete PWLS is defined as*

$$\mathcal{Q}_{\mathrm{TOM}}(j) = \begin{pmatrix} \mathcal{Q}_1 \\ \mathcal{Q}_2 F_1^{n-1} \\ \vdots \\ \mathcal{Q}_j F_{j-1}^{n-1} F_{j-2}^{n-1} \cdots F_1^{n-1} \end{pmatrix} \tag{40}$$

*where F_j is the process matrix and H_j is the measurement matrix for time segment j. $\mathcal{Q}_j$ is the **linear observability matrix**, $\mathcal{Q}_j^\intercal = \left[H_j^\intercal | (H_j F_j)^\intercal | \cdots | (H_j F_j^{n-1})^\intercal \right]$.*

The sufficient and necessary condition for the system to be completely observable is given by the following Theorem:

Theorem 2.3.1. *A discrete PWLS is completely observable if and only if TOM is full-rank.*

Computation of the TOM is expensive. However, for the SLAM system described in Equation 7, $\mathcal{N}(\mathcal{Q}_j) \subset \mathcal{N}(F_j)$. In this case, the Stripped Observability Matrix (SOM) defined below can be used to analyze the observability of a VSLAM PWLS.

Definition 2.3.3. *The Stripped Observability Matrix (SOM) of a discrete PWLS is defined as*

$$\mathcal{Q}_{\text{SOM}}(r) = \begin{pmatrix} \mathcal{Q}_1 \\ \mathcal{Q}_2 \\ \vdots \\ \mathcal{Q}_r \end{pmatrix} \tag{41}$$

where $\mathcal{Q}_j^{\intercal} = \left[H_j^{\intercal} | (H_j F_j)^{\intercal} | \cdots | (H_j F_j^{n-1})^{\intercal}\right]$*;* F_j *is the process matrix and* H_j *is the measurement matrix for time segment* j*.*

The following theorem justifies that SOM provides a proxy to examine the full rank condition of the system.

Theorem 2.3.2. *[46] For PWLS, when* $\mathcal{N}(\mathcal{Q}_j) \subset \mathcal{N}(F_j)$*, the SOM has the same nullspace as TOM, i.e.*

$$\mathcal{N}(\mathcal{Q}_{\text{SOM}}(j)) = \mathcal{N}(\mathcal{Q}_{\text{TOM}}(j)) \tag{42}$$

For the $SE\langle 3\rangle$ VSLAM, Theorem 2.3.2 applies.

Based on the previous two theorems, the following theorem for the necessary conditions of $SE\langle 3\rangle$ VSLAM can be proved.

Theorem 2.3.3. *When* $N_f = 0$*, within* r *time segments, the necessary conditions for system (7) to be completely observable are:*

1. $r = 1$ *and* $N_a \geq 3$*, or*

2. $r \geq 2$ *and* $N_a \geq 1$*.*

Proof. Consider the linear observability matrix in one time segment:

$$\mathcal{Q} = \begin{pmatrix} \mathbf{H} \\ \mathbf{HF} \\ \vdots \\ \mathbf{HF}^{12} \end{pmatrix} \in \mathbb{R}^{26\times 13}. \tag{43}$$

Note that in the actual computation, each block of $\mathcal{Q}$ can be computed efficiently with

$$\mathbf{HF}^n = \begin{pmatrix} \mathbf{H}_{<1:3>} & \mathbf{H}_{<4:7>}\mathbf{Q}^n & \mathbf{H}_{<1:3>} n\Delta t & \mathbf{H}_{<4:7>} \sum_{i=0}^{n-1} \mathbf{Q}^i \mathbf{\Omega} \end{pmatrix} \tag{44}$$

where $\mathbf{H}_{<i:j>}$ is a sub-block of $\mathbf{H}$ composed by **columns** from i to j.

By examining the rank condition of the a single linear observability matrix, we know that

$$rank(\mathcal{Q}) = 8 \tag{45}$$

In particular, the first eight rows of $\mathcal{Q}$ are of rank eight.

Based on this, we examine the rank conditions of SOM in two cases: (1) single time segment, multiple anchors, and (2) single anchor, multiple time segments.

Case # 1: single time segment, multiple anchors.

In this case, the $\mathbf{F}$ matrices of all anchors are the same. Assume the measurement jacobians subblocks that corresponding to anchors $A1, A2, A3, \cdots, AN_a$ are $\mathbf{H}_{A1}, \mathbf{H}_{A2}, \mathbf{H}_{A3}, \cdots, \mathbf{H}_{AN_a} \in \mathbb{R}^{2\times 13}$, and assume that the measurements and estimated positions of all anchors are different, the overall measurement jacobian is then

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{A1} \\ \mathbf{H}_{A2} \\ \vdots \\ \mathbf{H}_{AN_a} \end{pmatrix} \in \mathbb{R}^{2N_a\times 13} \tag{46}$$

The rank of SOM in this single time segment with all N_a anchors is

$$rank\left(\mathcal{Q}_{\text{SOM}}(1|N_a)\right) = rank\begin{pmatrix}\mathbf{H}\\ \mathbf{HF}\\ \vdots\\ \mathbf{HF}^{12}\end{pmatrix} = rank\begin{pmatrix}\mathcal{Q}(\mathbf{F},\mathbf{H}_{A1})\\ \mathcal{Q}(\mathbf{F},\mathbf{H}_{A2})\\ \vdots\\ \mathcal{Q}(\mathbf{F},\mathbf{H}_{AN_a})\end{pmatrix} \tag{47}$$

where $\mathcal{Q}(\mathbf{F},\mathbf{H})$ denotes the linear observability matrix generated by Jacobians $\mathbf{F}$ and $\mathbf{H}$.

By examining the rank conditions of Eq. 47, we have:

When $N_a = 1$

$$rank\Big(\mathcal{Q}(\mathbf{F},\mathbf{H}_{A1})\Big) = 8 \tag{48}$$

When $N_a = 2$

$$rank\begin{pmatrix}\mathcal{Q}(\mathbf{F},\mathbf{H}_{A1})\\ \mathcal{Q}(\mathbf{F},\mathbf{H}_{A2})\end{pmatrix} = 11 \tag{49}$$

When $N_a = 3$

$$rank\begin{pmatrix}\mathcal{Q}(\mathbf{F},\mathbf{H}_{A1})\\ \mathcal{Q}(\mathbf{F},\mathbf{H}_{A2})\\ \mathcal{Q}(\mathbf{F},\mathbf{H}_{A3})\end{pmatrix} = 13 \textbf{ (Full-rank)} \tag{50}$$

When $N_a > 3$, the SOM remains full-rank.

Thus, within one single time segment ($r = 1$), the necessary condition for the SLAM system to be completely observable is $N_a \geq 3$.

Case # 2: single anchor, multiple time segments.

In this case, assume the anchor is tracked across r time segments, and the Jacobian matrices across time are $\mathbf{F}_1, \mathbf{F}_2, \cdots, \mathbf{F}_r$ and $\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_r$. To derive the necessary condition, we also assume that all $\mathbf{F}$ and $\mathbf{H}$ matrices are different, i.e. they are

generated by different camera states and features measurements and estimates. The SOM is then

$$\mathcal{Q}_{\text{SOM}}(r|1) = \begin{pmatrix} \mathcal{Q}(\mathbf{F}_1, \mathbf{H}_1) \\ \mathcal{Q}(\mathbf{F}_2, \mathbf{H}_2) \\ \vdots \\ \mathcal{Q}(\mathbf{F}_r, \mathbf{H}_r) \end{pmatrix} \in \mathbb{R}^{26r \times 13} \tag{51}$$

When $r = 1$,

$$rank\Big(\mathcal{Q}(\mathbf{F}_1, \mathbf{H}_1)\Big) = 8 \tag{52}$$

When $r = 2$,

$$rank\begin{pmatrix} \mathcal{Q}(\mathbf{F}_1, \mathbf{H}_1) \\ \mathcal{Q}(\mathbf{F}_2, \mathbf{H}_2) \end{pmatrix} = 13 \textbf{ (Full-rank)} \tag{53}$$

When $r > 2$, the SOM remains full-rank.

Thus, for a single anchor, the necessary condition for the SLAM system to be completely observable is $r \geq 2$, i.e., the anchor is tracked across no less than two time segments. □

Remark 2.3.1. *In Theorem 2.3.3, the first condition is consistent with the well-known result for recovery of rigid body motion: for each pair of images, if there are more than or equal to three known scene-to-image (3D-to-2D) correspondences, localization and mapping can be solved [6, 118, 18, 114]. Condition #1 is illustrated in Figure 12.*

Remark 2.3.2. *According to the second condition, if a feature is tracked across three frames, the system composed of the camera motion and the feature may become observable, and the corresponding SOM full-rank [115]. Degenerate conditions such as the point lying on the translation vector of a camera undergoing pure translation would fail to be observable (as would pure rotation). Condition #2 is illustrated in Figure 13.*

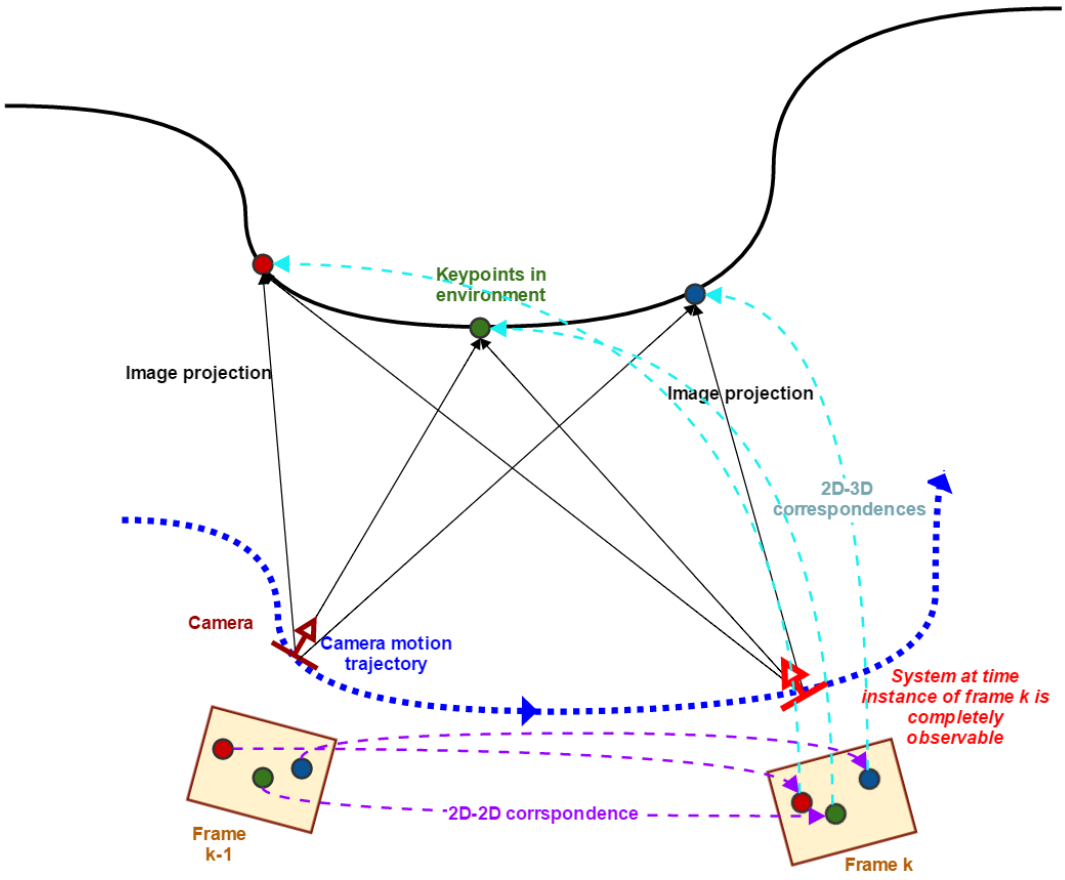


Figure 12: VSLAM system completely observable condition #1.

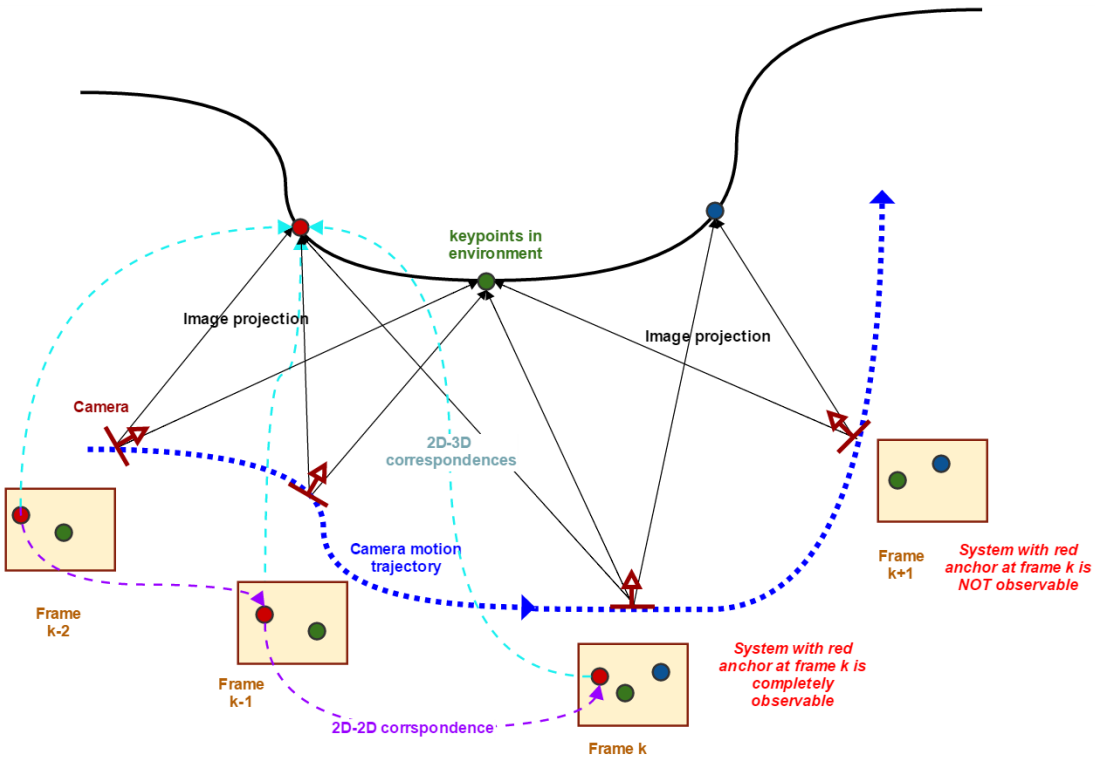


Figure 13: VSLAM system completely observable condition #2.

2.4 Conclusion

In this chapter, the observability conditions of the $SE\langle 3\rangle$ VSLAM systems are investigated. I start from the process model based on constant velocity assumption and the measurement model based on pinhole projection, and reach a piece-wise linear system model for VSLAM. In particular, the details for computing the PWLS are presented. Then, the observability of the VSLAM system is analyzed with the stripped observability matrix. To investigate the conditions for making the VSLAM system completely observable, I examine the full-rank conditions of the stripped observability matrix. Finally, two necessary conditions are obtained. The conditions state that the VSLAM system can be completely observable if (1) no less than three anchors are tracked in one time segment (i.e. between two frames), or (2) as few as one anchor is tracked across two time segments (i.e. across three frames).

Based on the results in this chapter, an algorithm dubbed "Optimally Observable

and Minimal Cardinality (OOMC) SLAM" [116] is developed in Chapter 3 by exploiting the first condition, and a feature selection algorithm dubbed "good features (GF) SLAM" [115] is developed in Chapter 4 by exploiting the second condition.

CHAPTER III

OPTIMALLY OBSERVABLE AND MINIMAL CARDINALITY (OOMC) VSLAM

The work in this chapter utilizes system observability condition #1 derived in Chapter 2 to guide monocular SLAM [116]. Instead of providing all measured features then performing data-driven outlier rejection (such as with RANSAC), I propose to identify only the minimal subset of features which form an optimally observable SLAM subsystem for localization in each time instance. A means to test the observability conditioning of candidate feature point groupings is proposed. Based on the conditioning, an efficient algorithm for picking the optimally observable feature subset is derived by incorporating the image geometric measures. The proposed monocular SLAM algorithm, called Optimally Observable and Minimal Cardinality (OOMC) SLAM is formulated as an EKF process. OOMC SLAM is first validated using a 6-DOF localization experiment; the results demonstrate accuracy comparable to the state-of-art SLAM algorithm with significantly improved computational efficiency. A longer sequence on a 620-meter trajectory is also tested. The algorithm achieves 0.9178% relative error against the GPS ground truth.

3.1 Introduction

Conventional monocular SLAM like MonoSLAM [27] deals with feature rejection by using data-driven and randomized approaches like RANSAC [39, 19] to increase the accuracy and robustness of SLAM estimation. Randomized, data-driven, model fitness approaches make little use of prior information and result in high computation

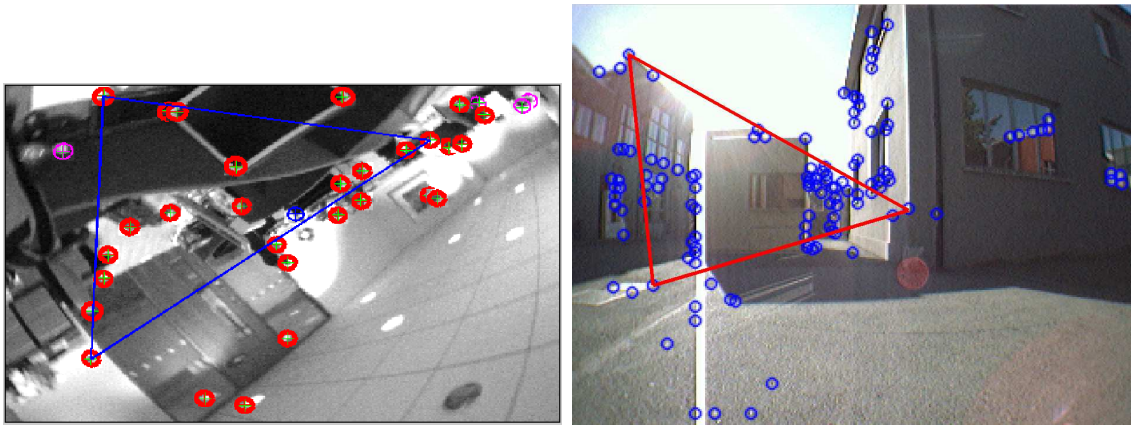


Figure 14: At each time segment, the OOMC algorithm identifies the feature triplet (defines the triangles) which forms the subsystem of **optimal observability** and **minimal cardinality** with the camera state. Localization and mapping is then performed with this subsystem.

cost. Thus in recent years, research efforts have sought to incorporate prior information into SLAM to facilitate the feature selection process. These methods include prior probability of camera state [22] or information gain [25]. However, they only seek to mitigate the cost of randomized estimation and still rely on it to increase robustness.

The work described herein is motivated by the same goal: the exploitation of known computable measures to select the most informative and reliable features for localization and mapping. The solution differs from previous approaches in that it selects the most informative features in a deterministic fashion via two properties: system observability, and reliability of feature detection and matching.

Using the score, it is able to select a minimal subset of tracked features to form an optimally observable subsystem, c.f. Figure 14. This subsystem provides the estimate of the whole SLAM state but with the best conditioning for improved noise tolerance and accuracy.

In particular, this chapter presents

1. a framework for using observability to guide SLAM, which can increase both the accuracy (better inlier ratio) and efficiency. This is done by identifying

the subsystem with **optimal observability** and **minimal cardinality** for localization and mapping;

2. a scoring strategy based on the first condition of VSLAM system complete observability;

3. the presentation of OOMC SLAM with efficient selection of observable triplets cast using an EKF, with evaluation results demonstrating the OOMC EKF VSLAM improves over previous methods

3.2 Background

Since the related work of using system observability in SLAM has been reviewed in Chapter 2, here I discuss other work of searching feature consensus in $SE\langle 3\rangle$ VSLAM.

Other work focuses on fusing prior information to improve the feature consensus search. Active Matching [25] improves the search by guiding the search action at each step according to Shannon information gain. The Shannon information gain is derived from a Bayesian framework modeling the dynamics. However, the Active Matching approach is expensive. Later Scalable Active Matching [50] was proposed to increase the efficiency and scalability by relaxing the Active Matching method. Nevertheless, these two methods focus on the feature matching step of the SLAM system, instead of the overall estimation process.

The most similar work is the 1-Point RANSAC based EKF SLAM [22]. Estimation (i.e., the EKF update step) incorporates the prior motion model information. By doing so, the claim is that the minimal sample size for RANSAC is reduced to one, gaining a large computational savings with little loss of discriminative power. Though theoretical justifications of the algorithm were not provided, the authors demonstrated the high localization accuracy of 1-Point RANSAC by comparing to bundle adjustment results in a 6-DOF test. Due to its solid performance, 1-Point RANSAC is the baseline method used for comparison.

3.3 Optimally Observable and Minimal Cardinality (OOMC) SLAM

3.3.1 OOMC Formulation

Consider a set of point features $\mathcal{F}$ matched across two frames. The key idea of the algorithm stems from the fact that different subsets of $\mathcal{F}$ provide different degrees of *informativeness*. Instead of using all features in a randomized method (e.g. RANSAC), using only the most reliable and informative subset of $\mathcal{F}$ should provide a better conditioned subsystem. The degree of *informativeness* is reflected in the observability conditioning of the SLAM system. Our algorithm aims to find the subset of features $\mathcal{F}'$ maximizing SLAM system ($\mathfrak{S}_{\text{SLAM}}$) observability in order to increase the robustness to noise and thus increase the accuracy of estimation. Concurrently, for improved efficiency, the cardinality of this subset is minimized to the minimal cardinality allowed for the system to be **completely observable** ($\aleph_{\textbf{CompletelyObservable}}$). Our objective function is therefore:

$$\begin{aligned} \mathcal{F}' = \arg\max_{\mathcal{F}' \subset \mathcal{F}} & \textbf{ObsScore}\left(\mathfrak{S}_{\text{SLAM}}(\mathcal{F}'; \mathcal{G}_{\text{cam}}^{SE\langle 3\rangle})\right) \\ \text{s.t.} \quad & |\mathcal{F}'| = \min\left(\aleph_{\textbf{CompletelyObservable}}\right) \end{aligned} \tag{54}$$

Recall that the VSLAM system $\mathfrak{S}_{\text{SLAM}}(\mathcal{F}; \mathcal{G}_{\text{cam}}^{SE\langle 3\rangle})$ is modeled as a piece-wise linear system as follows

$$\begin{cases} \mathbf{X}_{k+1}^{\mathcal{W}} = \mathrm{F}^{\mathcal{R}_k}\mathbf{X}_k^{\mathcal{W}} + \mathbf{u}_k \\ \delta\mathbf{h}^{\mathcal{R}_k} = \mathrm{H}^{\mathcal{R}_k}\mathbf{X}_k^{\mathcal{W}} \end{cases} \quad \text{for } t \in \mathcal{T}_k. \tag{55}$$

According to condition #1 in Theorem 2.3.3, which states that a necessary condition to make $SLAMsys$ completely observable is that no less than three anchors are tracked in one time segment (i.e. between two frames). Therefore, in Equation 54,

$$|\mathcal{F}'| = \min\left(\aleph_{\textbf{CompletelyObservable}}\right) = 3. \tag{56}$$

3.3.2 Observability Score

In a typical SLAM scenario, the number of scene-to-image correspondences is usually more than three. Thus, the first condition is easy to guarantee. When the completely observable conditions hold, the degree of observability can be further evaluated: the more observable the system is, the more accurate the motion estimation is. Within one time segment (when $N_a > 3$), a total of N_a choose three subsystems can be generated with three anchors in each subsystem. Ideally, each of these subsystems gives the exact camera localization result. However, in presence of noise, the system with the most favorable observability matrix conditioning gives the most accurate estimate of camera motion among all the subsystems. In order to find the triplet that forms the most observable subsystem, the degree of observability is measured by the smallest singular value of SOM:

$$\textbf{ObsScore}\left(\mathfrak{S}_{\text{SLAM}}\right) = \psi = \sigma_{\min}(\mathcal{Q}_{\text{SOM}}(r)) \tag{57}$$

Here $\mathcal{Q}_{\text{SOM}}(r)$ is from one time segment ($r = 1$) and three anchors.

The score ψ reflects how sensitive the state estimate is w.r.t the process and measurement noise. The larger ψ is, the less sensitive the estimate is to the noise, and thus the more accurate the estimated state is. Since both $\mathrm{F}^{\mathcal{R}_\mathbf{k}}$ and $\mathrm{H}^{\mathcal{R}_\mathbf{k}}$ are sparse, and $\mathcal{Q}_{\text{SOM}}(r)$ is of low dimension for three anchors, the computation of ψ is efficient. Thus there is little overhead in evaluating the observability score for candidate triplets.

3.3.3 Triplet Selection Strategy

The complete observability conditions and score presented in Theorem 2.3.3 may guide the visual SLAM estimation process. In practice, anchors may not be available. However, at each time instance the features with scene-to-image correspondences can play the role of anchors (a convenience used in conventional visual SLAM/SfM

Algorithm 1: Top $N_{\mathfrak{T}}$ strongest observable triplets $\{\mathfrak{T}_s\}$ selection algorithm

Data: $\{\mathbf{Z}\}$ – Initial matched feature set (mismatches may exist);
$\{S\}$ – Matching scores for initial matches $\{\mathbf{Z}\}$;
$N_{\mathfrak{T}}$ – Number of required triplets.

Result: $\{\mathfrak{T}_s\}_{s=1}^{N_{\mathfrak{T}}}$ – a set of triplets that form the strongest observable subsystems.

[Stage 1. Generate triplet pool $\{\mathfrak{T}'_{s'}\}$]

$\{S_{\mathfrak{P}}\} \leftarrow$ Compute the pairwise sums of matching scores $\{S\}$ for each pair in $\{\mathbf{Z}\}$; // $\mathcal{O}(N_{\mathbf{Z}}^2)$

$\{\mathfrak{P}\}_{N_{\mathfrak{P}}} \leftarrow$ Find the $N_{\mathfrak{P}}$ pairs of features from $\{\mathbf{Z}\}$ with the largest $S_{\mathfrak{P}}$; // $\mathcal{O}(N_{\mathbf{Z}}^2 \log N_{\mathfrak{P}})$

$\{A\} \leftarrow$ For each pair in $\{\mathfrak{P}\}$, compute triplet area A with a third feature in $\{\mathbf{Z}\}$; // $\mathcal{O}(N_{\mathbf{Z}} N_{\mathfrak{P}})$

$\{\mathfrak{T}'_{s'}\} \leftarrow$ Get triplets with top $N_{\mathfrak{T}'}$ area from $\{A\}$; // $\mathcal{O}(N_{\mathbf{Z}} N_{\mathfrak{P}} \log N_{\mathfrak{T}'})$

[Stage 2. Select strongest observable triplets $\{\mathfrak{T}_s\}_{N_{\mathfrak{T}}}$]

$\{\psi_{s'}\} \leftarrow$ Compute the observability scores for each triplet candidate in $\{\mathfrak{T}'_{s'}\}$; // $\mathcal{O}(N_{\mathfrak{T}'})$

$\{\mathfrak{T}_s\} \leftarrow$ Select $N_{\mathfrak{T}}$ with largest ψ ; // $\mathcal{O}(N_{\mathfrak{T}'} \log N_{\mathfrak{T}})$

systems). At each time, a subset of scene-to-image correspondences are used to solve camera localization. This subset is usually selected in a randomized and data-driven process, such as using RANSAC paradigms [27, 22]. Instead the algorithm presented here promotes only three features (denoted as a triplet $\mathfrak{T}_s$) to be anchors, such that the subsystem with $\mathfrak{T}_s$ forms the optimally observable subsystem. This deterministic strategy of constant computational cost eliminates the random selection step, thereby gaining computational savings while having a systematic guarantee of the estimation accuracy.

In addition to the observability score of $\mathfrak{T}_s$, the SLAM estimate is also affected by the feature matching step (it must be correct). Thus, the proposed triplet selection strategy also considers the feature matching score S of the matching algorithm used. In our system, the initial matched feature set $\{\mathbf{Z}\}$ is obtained via an individually compatible matching similar to [27], which matches wrapped image patches using normalized cross-correlation as the matching score S. Moreover, to avoid the selected

features being too close on the image, triplets with larger area is favored, where the triplet area for the triplet composed of features i_1, i_2, i_3 is

$$A_{<j_1,j_2,j_3>} = \frac{1}{2}\left((\mathbf{h}_{i_1}^{\mathcal{R}_k} - \mathbf{h}_{i_2}^{\mathcal{R}_k}) \times (\mathbf{h}_{i_1}^{\mathcal{R}_k} - \mathbf{h}_{i_3}^{\mathcal{R}_k})\right) \tag{58}$$

To incorporate the above two scoring system into OOMC triplet selection, a two-stage triplet selection algorithm is proposed. In the first stage of the algorithm, matching score S is used to find pairs of features which are best matched; then the triplet candidates $\{\mathfrak{T}'_{s'}\}$ are generated by finding the third feature that forms the largest A with each previously selected pair. Because the S comes directly from the feature matching step and computation of A is trivial, this stage is computationally efficient. In the second stage, the observability metric ψ of each triplet candidate in $\{\mathfrak{T}'_{s'}\}$ is computed. The $N_{\mathfrak{T}}$ triplets that forms the $N_{\mathfrak{T}}$ strongest observable subsystems are selected and output. The two-pass strategy avoids exhaustive search for triplets with high Observability Score by first finding the most likely candidates.

The triplet selection algorithm is summarized in Algorithm 1. The final computation complexity order is dominated by line 3, which has the order $\mathcal{O}(N_{\mathbf{Z}}^2 \log N_{\mathfrak{P}})$. This complexity is due to the Top-K selection algorithm with max-heaps and each atomic operation is trivial. Thus the overall algorithm is efficient.

3.3.4 OOMC SLAM in EKF Framework

With the completely observable conditions presented in Theorem 2.3.3, visual SLAM process can be guided by the observable status of the system. In practice, anchors may not be available. However, in each time instance the features with scene-to-image correspondences can be seen as the anchors. This strategy is used in conventional visual SLAM/SfM systems in a way that: at each time, a subset of scene-to-image correspondences are used to solve camera localization. This subset is usually selected in a randomized and data-driven process, such as using RANSAC paradigms [27] [22]. Different from the conventional approaches, the algorithm presented here augments

only three features (denoted as a triplet $\mathfrak{T}_s$) as anchors, such that the subsystem with $\mathfrak{T}_s$ forms the strongest observable subsystem. This deterministic strategy gets rid of the randomized manner used in conventional approaches. Thus, the algorithm is able to gain a great computational saving over the conventional approaches while have a systematic guarantee of the estimation accuracy.

With Algorithm 1, once the strongest observable triplets are selected, the system state vector can be estimated with the triplets. For state estimation, an Extended Kalman Filter (EKF) estimator is used to update the system state. Note that although the EKF is used here, other pose estimators can also be used as long as they can perform pose estimation with three scene-to-image correspondences. Examples include the nonlinear optimizer for poses used in PTAM [60] and the coupled estimator used in CLAM [6].

The EKF estimator is formulated as **prediction:**

$$\hat{\mathbf{X}}^{\mathcal{W}}_{k|k-1} = \mathrm{f}(\hat{\mathbf{X}}^{\mathcal{W}}_{k-1|k-1}) + \mathbf{u}_k, \tag{59}$$

$$\mathbf{P}_{k|k-1} = \mathrm{F}^{\mathcal{R}_k}\mathbf{P}_{k-1|k-1}\mathrm{F}^{\mathcal{R}_k\mathsf{T}} + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^{\mathsf{T}}, \tag{60}$$

$$\hat{\mathbf{h}}^{\mathcal{R}_k}_k = \mathrm{h}^{\mathcal{R}_k}(\hat{\mathbf{X}}^{\mathcal{W}}_{k-1|k-1}), \tag{61}$$

$$\mathbf{S}_k = \mathrm{H}^{\mathcal{R}_k}\mathbf{P}_{k|k-1}\mathrm{H}^{\mathcal{R}_k\mathsf{T}} + \mathbf{R}_k; \tag{62}$$

followed by **update:**

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathrm{H}^{\mathcal{R}_k\mathsf{T}}\left(\mathrm{H}^{\mathcal{R}_k}\mathbf{P}_{k|k-1}\mathrm{H}^{\mathcal{R}_k\mathsf{T}} + \mathbf{R}_k\right)^{-1}, \tag{63}$$

$$\hat{\mathbf{X}}^{\mathcal{W}}_{k|k} = \hat{\mathbf{X}}^{\mathcal{W}}_{k|k-1} + \mathbf{K}_k\left({}^{(\mathrm{s})}\mathbf{h}^{\mathcal{R}_k} - {}^{(\mathrm{s})}\mathrm{h}^{\mathcal{R}_k}(\hat{\mathbf{X}}^{\mathcal{W}}_{k|k-1})\right), \tag{64}$$

$$\mathbf{P}_{k|k} = \left(\mathbf{I} - \mathbf{K}_k\mathrm{H}^{\mathcal{R}_k}\right)\mathbf{P}_{k|k-1}; \tag{65}$$

where $\mathbf{P}$ is the covariance matrix of state vector in the multidimensional Gaussian distribution, $\mathbf{Q}$ is the covariance matrix of process noise in a zero-mean Gaussian distribution, $\mathbf{G}$ is the Jacobian of process noise w.r.t. the state vector; $\mathbf{R}$ is the covariance matrix of measurement noise, $\mathbf{S}$ is the covariances of the corresponding

image projections; $\mathbf{K}$ is the estimated Kalman gain.

In each time segment, Equation (64) is used to update both the camera pose and the map. ${}^{(\mathbf{s})}\mathbf{h}_k^{\mathcal{R}_k}$ represents the subset $\mathbf{s}$ of measurements used to update the state vector. The key difference with respect to the standard EKF-SLAM is that the state vector is first partially updated with the strongest observable triplets $\mathfrak{T}_s$ from Algorithm 1. To increase the robustness, the estimation is performed with a set of strongly observable triplets $\{\mathfrak{T}_s\}_{s=1}^{N_{\mathfrak{T}}}$ in a two-phase process, similar to two-phase model estimation in a RANSAC iteration. However, this process is performed in a deterministic manner instead of randomized manner as in RANSAC paradigm: In the first phase, for each $\mathfrak{T}_s \in \{\mathfrak{T}_s\}$, only the state vector is updated using Equation (64) with $\mathbf{s} = \mathfrak{T}_s$; then a consensus set is retrieved from the updated model. In the second phase, after all the triplets have been tested, both the state vector and covariances are fully updated with the most supported consensus set. This fully deterministic algorithm overcomes the problem of expensive computation and unstable frame-rates due to the randomized process, resulting in a more efficient and stable algorithm. The proposed OOMC-SLAM using the EKF framework is summarized in Algorithm 2.

3.4 Experiments

OOMC-SLAM using EKF is validated through two sets of experiments: one for 6-DOF localization in a smaller-scaled indoor environment, and the other one for a larger-scaled outdoor environment. Both experiments use publicly available datasets for benchmarking.

3.4.1 6-DOF Experiment with Comparison to 1-Point RANSAC

This experiment has two goals. Firstly, it validates the performance of OOMC in 6-DOF SLAM. Secondly, a thorough comparison under a controlled experiment is made between OOMC and the state-of-art algorithm 1-Point RANSAC (1pRANSAC) [22]. 1pRANSAC is compared to because of its balance of efficiency and high accuracy

Algorithm 2: OOMC-SLAM under EKF framework.

Data: $\{\mathcal{I}_k\}$ – Image source;
$\{\mathbf{u}_k\}$ – Excitation source;
$\mathbf{\Pi}_{cam}$ – Calibrated camera intrinsic matrix.

Result: $\{\hat{\mathbf{X}}^{\mathcal{W}}_{k|k}\}$ – Sequence of estimated state vectors with 6-DOF camera poses and features in the scene.

while $\{\mathcal{I}\} \neq \emptyset$ **do**
  $(\hat{\mathbf{X}}^{\mathcal{W}}_{k|k-1}, \mathbf{P}_{k|k-1}, \hat{\mathbf{h}}^{\mathcal{R}_k}_k, \mathbf{S}_k) \leftarrow$ EKF prediction ; `// Equations (59)∼(62)`
  $\{\mathbf{Z}\} \leftarrow$ Match features on $\mathcal{I}_k$;
  $\{\mathfrak{T}_s\}_{s=1}^{N_{\mathfrak{T}}} \leftarrow$ Select strongest observable triplets ; `// Algorithm 1`
  $\mathbf{K}_k \leftarrow$ Compute Kalman gain;
  ${}^{(\mathfrak{T}_*)}\mathbf{h}^{\mathcal{R}_k}_{\text{cs}} \leftarrow \emptyset$;
  for each $\mathfrak{T}_s$ **in** $\{\mathfrak{T}_s\}$ **do**
    ${}^{(\mathfrak{T}_s)}\hat{\mathbf{X}}^{\mathcal{W}}_{k|k} \leftarrow$ Partially update EKF state with $\mathfrak{T}_s$; `// Equation (64)`
    ${}^{(\mathfrak{T}_s)}\mathbf{h}^{\mathcal{R}_k}_{\text{cs}} \leftarrow$ Retrieve consensus set of ${}^{(\mathfrak{T}_s)}\hat{\mathbf{X}}^{\mathcal{W}}_{k|k}$;
    if $\left|{}^{(\mathfrak{T}_s)}\mathbf{h}^{\mathcal{R}_k}_{\text{cs}}\right| > \left|{}^{(\mathfrak{T}_*)}\mathbf{h}^{\mathcal{R}_k}_{\text{cs}}\right|$ **then**
      ${}^{(\mathfrak{T}_*)}\mathbf{h}^{\mathcal{R}_k}_{\text{cs}} \leftarrow {}^{(\mathfrak{T}_s)}\mathbf{h}^{\mathcal{R}_k}_{\text{cs}}$; `// Most supported concensus set`
  $\left(\hat{\mathbf{X}}^{\mathcal{W}}_{k|k}, \mathbf{P}_{k|k}\right) \leftarrow$ Fully update EKF with ${}^{(\mathfrak{T}_*)}\mathbf{h}^{\mathcal{R}_k}_{\text{cs}}$; `// Equation (64),(65)`

compared to other methods. Moreover, both 1pRANSAC and OOMC SLAM are formulated with EKF framework.

To control the comparison experiment, the 1pRANSAC dataset (`https://openslam.org/ekfmonoslam.html`) is used. Furthermore, for comparison of execution time, I implemented the codes for this experiment with the same tool (MATLAB) as 1pRANSAC source codes. Both methods share the same camera calibration, feature detector, and initial feature matcher. Therefore, the differences in performance between these two methods are mainly due to the difference of algorithms themselves.

Frames numbers 101 to 2100 of the dataset were used. The triplet selection parameters were $N_{\mathfrak{P}} = 50$, $N_{\mathfrak{T}'} = 400$, $N_{\mathfrak{T}} = 30$. Example frames from OOMC are shown in Figure 15 with the triplet plotted. One observation is that the triplets alway span a large area in 3D space (but may not in image space). Empirically, the triplet selection was temporally consistent over small time, i.e. the features augmented as a triplet in one frame were more likely to be augmented in consecutive frames.

The localization results of OOMC and 1pRANSAC are shown in Figure 16 and the absolute differences are in Figure 17. The mean and standard deviation of position differences are $[0.0013 \pm 0.0017, 0.0003 \pm 0.0017, 0.0003 \pm 0.0012]$, while the orientation differences are $[0.0011 \pm 0.0018, 0.0002 \pm 0.0016, 0.0002 \pm 0.0014]$ (rad).

The processing time of both methods are compared in Figure 18, with the map size plotted. The result shows that OOMC has a better stability in speed than 1pRANSAC. Further more, the speed of OOMC is significantly faster than 1pRANSAC. For 1pRANSAC, the average processing time is 0.4345 sec. Compared to this, the average processing time of OOMC is 0.0596 sec. The speedup of OOMC over 1pRANSAC is 7.4776 ± 1.6190 times.

Finally, the inlier ratios (denoted as Γ) of max-support consensus set are compared in Figure 19. This metric reflects the degrees of informativeness of the features used for EKF partial updates. The higher Γ is, the more informative the selected

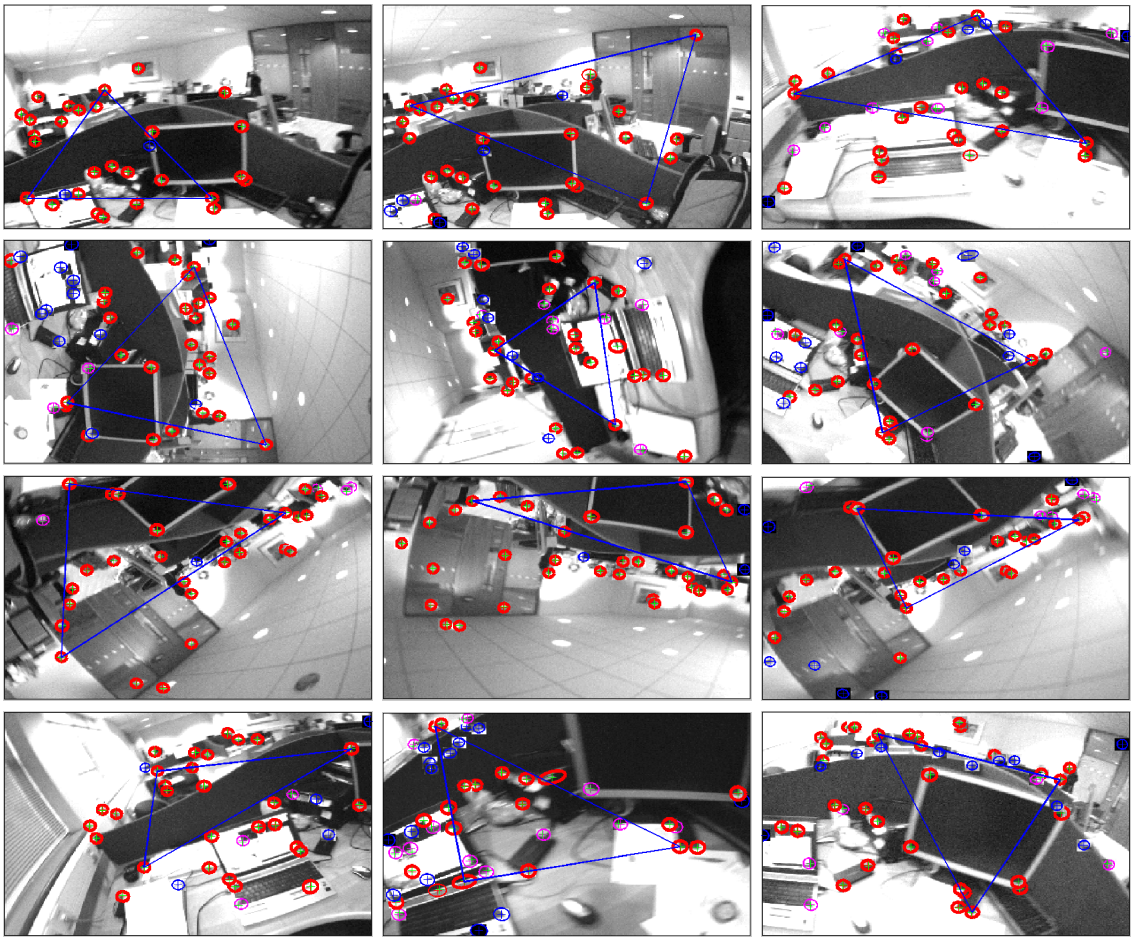


Figure 15: The triplets (depicted as blue triangles) selected for EKF update in example frames from the 1pRANSAC dataset. The measurements are plotted in eclipses of 1-σ regions with various markers: thick red – low-innovation inliers; thin red – high-innovation inliers; magenta: rejected spurious matches; blue – no match found by cross-correlation.

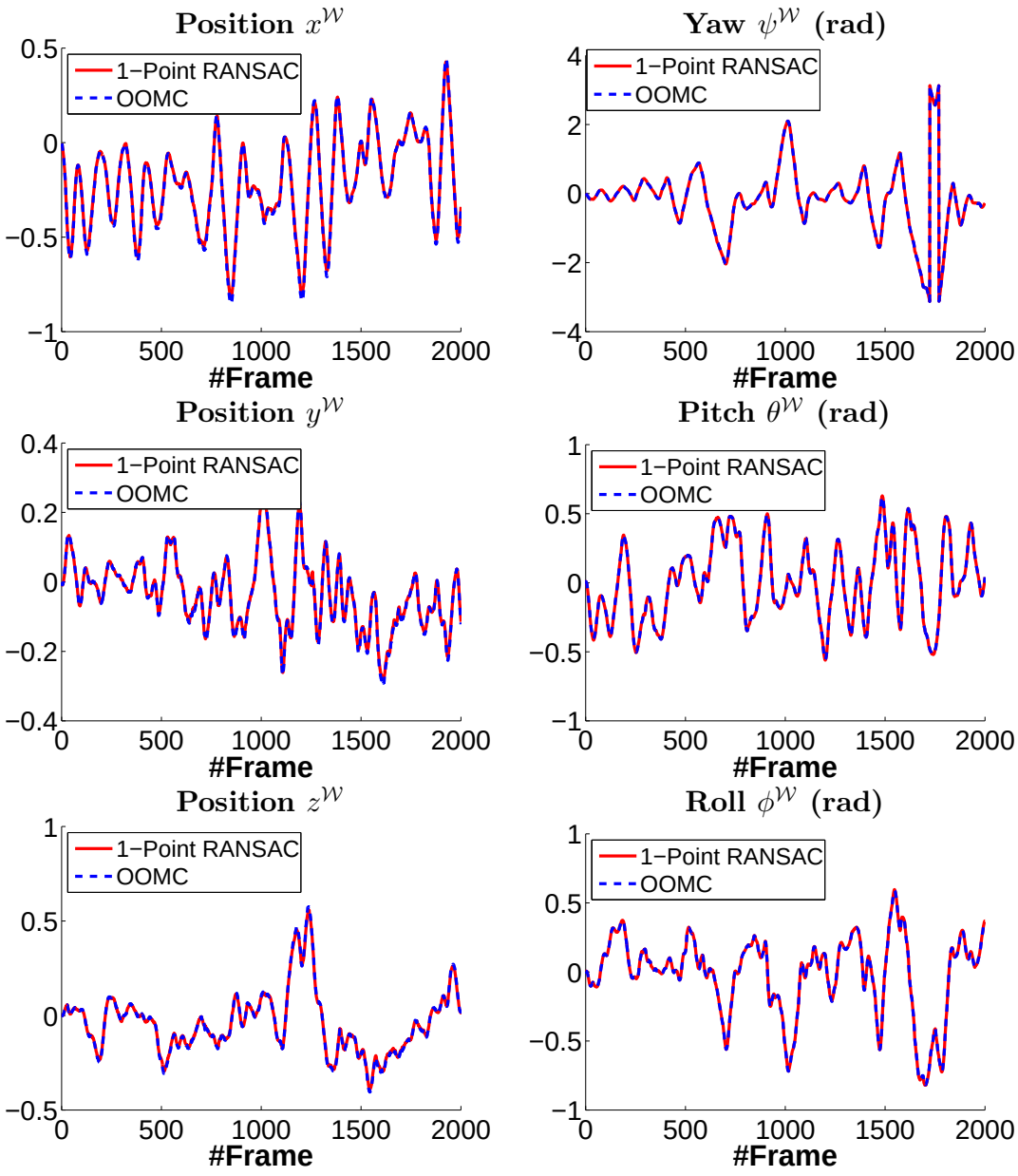


Figure 16: Camera localization results of 1pRANSAC SLAM and OOMC SLAM

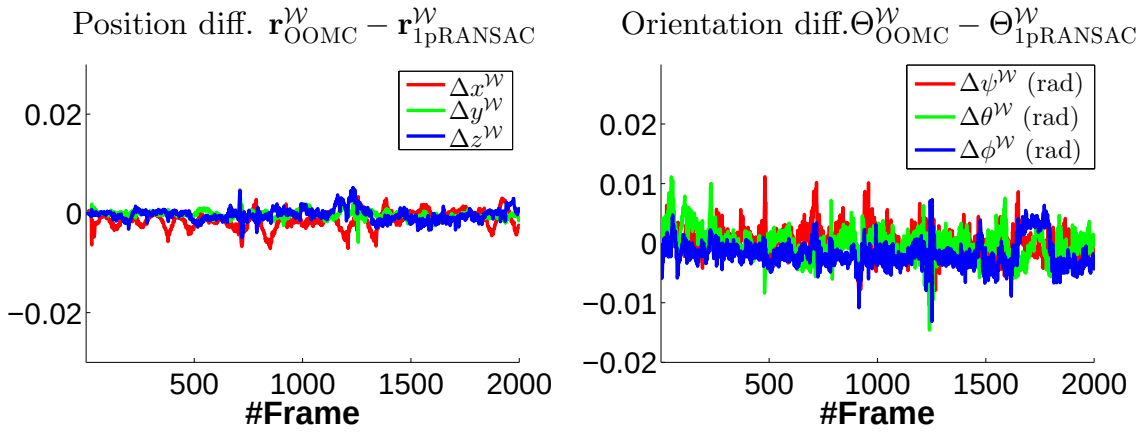


Figure 17: Differences in camera localization results of 1pRANSAC SLAM and OOMC SLAM

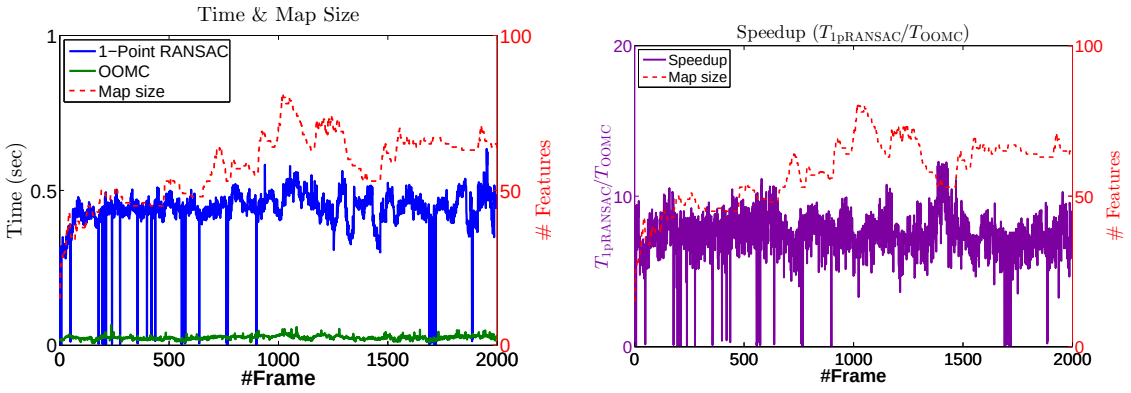


Figure 18: Left: execution time of 1pRANSAC and OOMC, and the map size. Right: speedup of OOMC over 1-Point RANSAC.

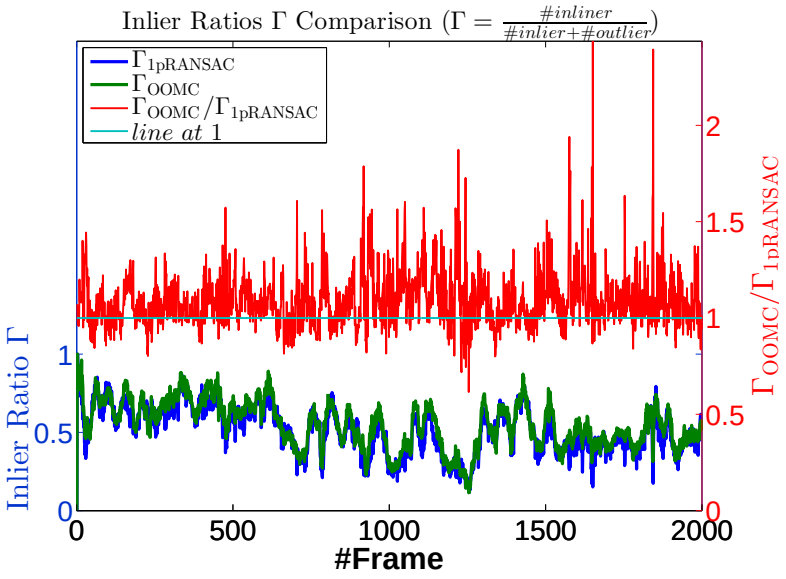


Figure 19: Inlier ratios of 1pRANSAC algorithm and OOMC algorithm, and their comparison. The inlier ratios are from the maximum support consensus sets for each frame.

features are. For OOMC, these features are the selected triplet; for 1pRANSAC, they are the features of the best supported hypothesis. In the result, Γ_{OOMC} is higher than Γ_{1pRANSAC} for 1469 out of 2000 frames. The statistics of their ratio is $(\Gamma_{\text{OOMC}}/\Gamma_{\text{1pRANSAC}}) = 1.0884 \pm 0.1436$.

3.4.2 Long distance experiment against GPS

Validation of the algorithm on a larger dataset use the public benchmark *RAWSEEDS* [17]. A sequence with a loop-closure scenario is selected for validation. The illustrative trajectory is plotted with a map image in Figure 20. The trajectory is about 620 meters long and captured outdoor under natural lighting. It consists of 21000 images with resolution 320×240, captured by a Unibrain camera with a wide-angle lens at 30 fps. Ground truth of the trajectory is collected with a Real Time Kinematics GPS.

For every frame, 50 triplets are generated for EKF partial update. Some example frames with the final selected triplet are shown in Figure 21. The results from OOMC SLAM are plotted against GPS ground truth in Figure 22. The estimated trajectory

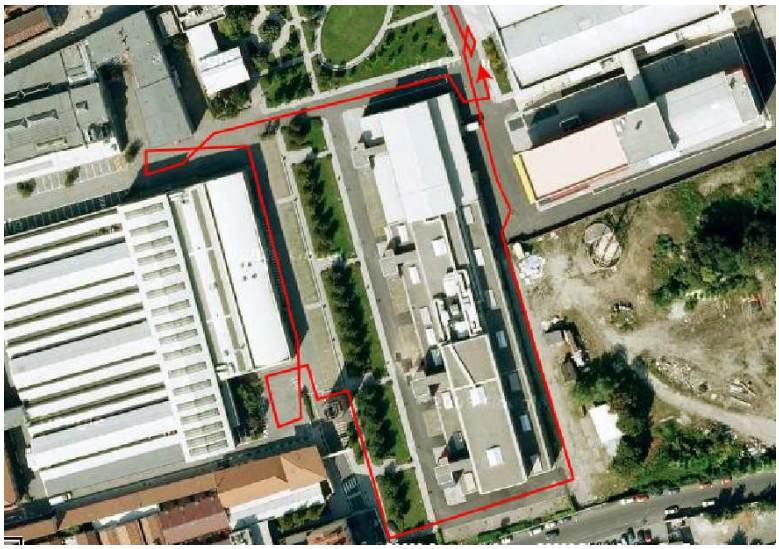


Figure 20: Illustration of the 620-meter trajectory (in red).

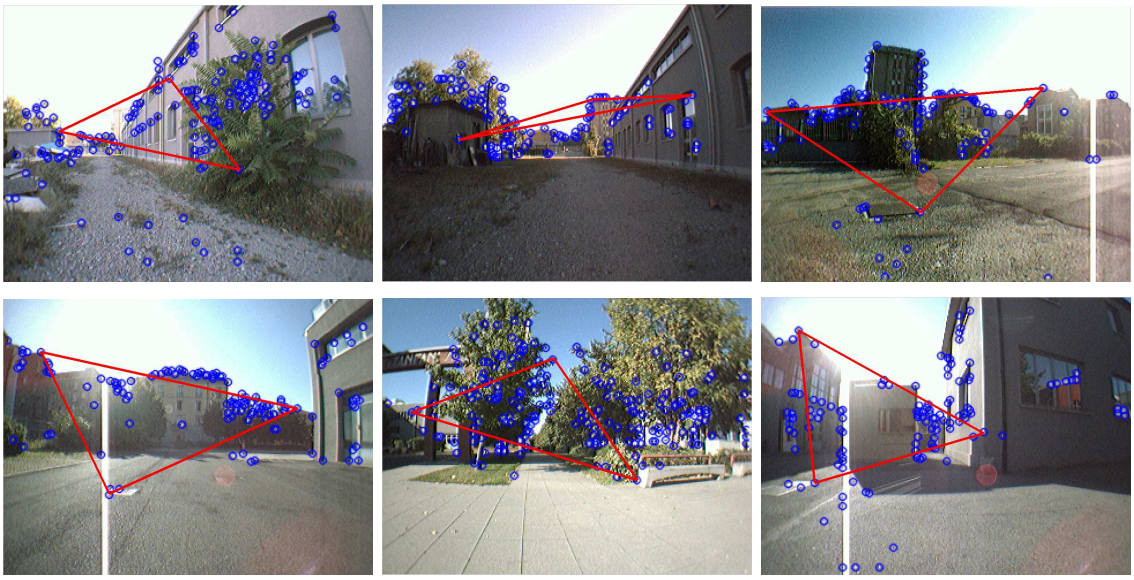


Figure 21: Example frames from 620m sequence. The detected features are plotted in blue circles and the selected triplets are plotted in red triangles.

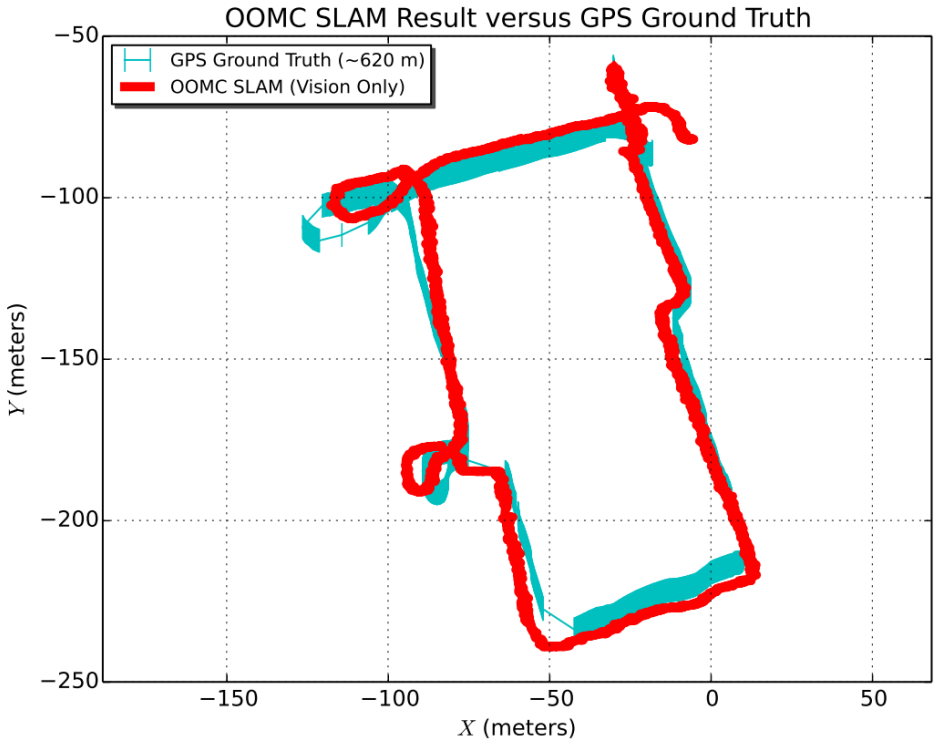


Figure 22: Estimated trajectory from the OOMC SLAM compared to trajectory from a RTK Differential GPS.

is benchmarked against GPS as in [22].

The estimation errors are computed by the $l2$-distance between the estimated position $\mathbf{r}^{\mathcal{W}}_{\text{OOMC}}$ from OOMC and the corresponding GPS reading. This error is only computed at frames with available GPS records. In Figure 22, points on GPS trajectory marked in thick marker are the points with GPS records. Figure 23 (top) shows the histogram of the instantaneous errors. Some error statistics are listed in Table 1. The OOMC SLAM achieves a high accuracy with 0.92% relative error (relative to trajectory length). This accuracy is comparable to the benchmarking results reported [22].

The cumulative error versus trajectory distance is demonstrated in Figure 23 (bottom). As the trajectory gets longer, the mean of cumulative error increases with the 1-σ range expanding, but the growth of the error is of sub-linear order.

Mean error (m)	**Error Std (m)**	**Maximum error (m)**	**Relative mean error over trajectory (%)**
5.6904	3.3449	14.6547	0.9178

Table 1: Errors of OOMC SLAM against GPS data on 620 meter sequence.

3.5 Conclusion and Future Work

Based on the system complete observable condition #1 in Theorem 2.3.3, this chapter investigates into the idea of selecting and using the optimally observable but minimal subsystem for camera localization and mapping. Using the minimal cardinality needed for observability and the observability score, the Optimally Observable and Minimal Cardinality (OOMC) SLAM procedure is presented. In each time segment, three features that form the optimally observable subsystem are augmented as the anchor triplet, then used for localization and mapping. For more robust and efficient triplet selection, a two-phase triplet selection algorithm is developed by incorporating

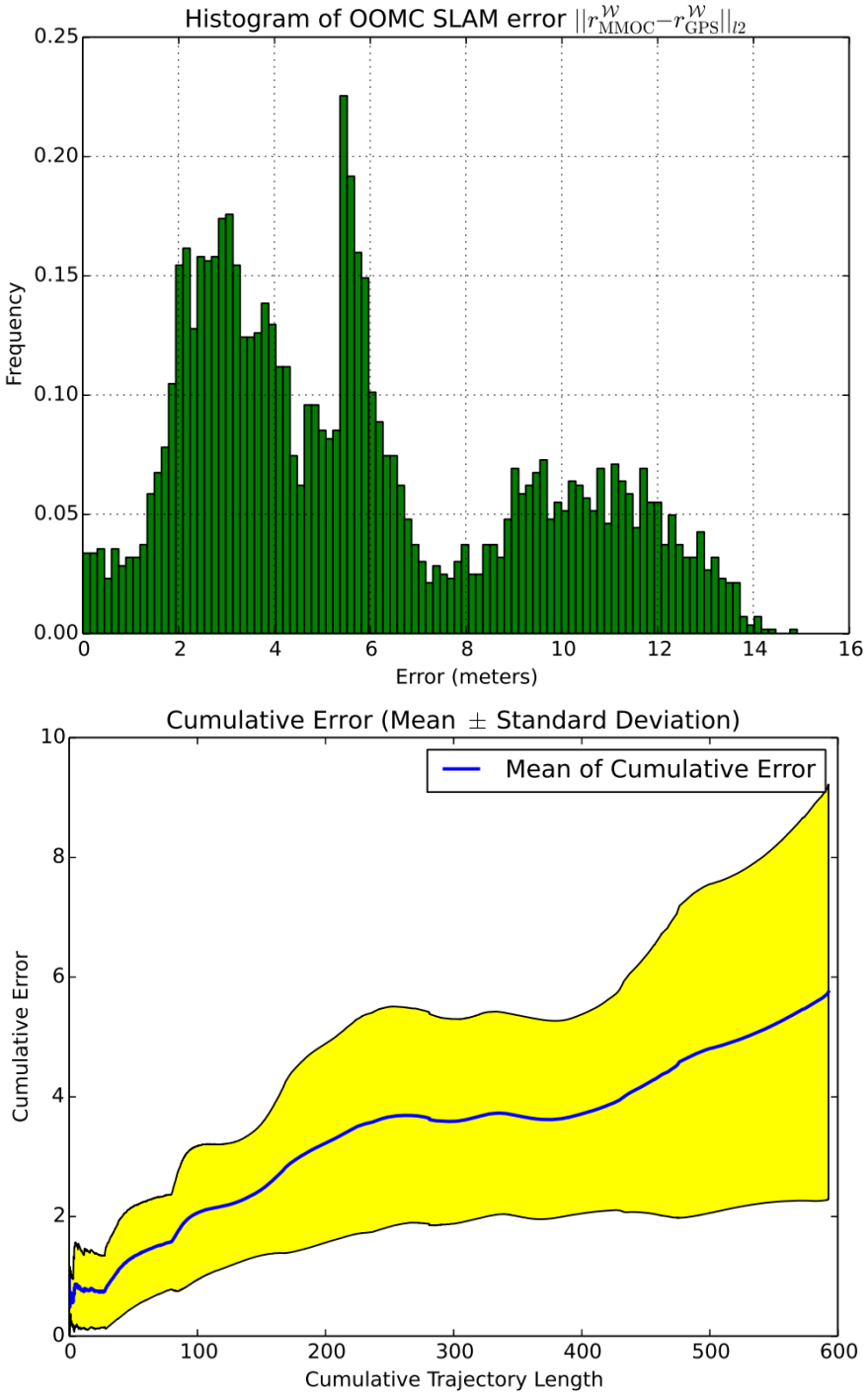


Figure 23: Errors of OOMC SLAM compared to the GPS ground truth. Top: histogram of instantaneous error. Bottom: cumulative errors with 1-σ range along the trajectory.

image geometric measures. The final OOMC SLAM is cast as an EKF. A 6-DOF experiment with controlled comparison to 1-Point RANSAC method is discussed. The OOMC demonstrates the same level of localization accuracy but with a significant runtime speedup. Finally, OOMC SLAM is applied to a 620-meter long outdoor image sequence with RTK Differential GPS ground truth. Using only a small set of triplets, the proposed algorithm achieves comparable accuracy to the state-of-the-art algorithms benchmarked with the dataset.

CHAPTER IV

GOOD FEATURES TO TRACK FOR VSLAM ALGORITHM

The work in this chapter utilizes system observability condition #2 derived in Chapter 2 to develop a feature selection algorithm for $SE\langle 3\rangle$ VSLAM. The condition #2 states that if an individual anchor is tracked across two time segments then the VSLAM system can be completely observable [115]. Selection of features which give the best conditioned SLAM system is expected to increase estimation accuracy with the same feature cardinality, or increase the efficiency by with more informative feature subset. This chapter describes a method for selecting a subset of features that are of high utility for localization in the SLAM/SfM estimation process. It is derived by examining the observability of SLAM and, being complimentary to the estimation process, it easily integrates into existing SLAM systems. The measure of estimation utility is formulated with temporal and instantaneous observability indices. Efficient computation strategies for the observability indices are described based on incremental singular value decomposition (SVD) and greedy selection for the temporal and instantaneous observability indices, respectively. The greedy selection is near-optimal since the observability index is (approximately) submodular.

I further demonstrate how to integrate the proposed algorithm into two prevalent types of VSLAM systems, along with extensive evaluations on both data association and localization accuracy. Experiments include: (1) controlled synthetic experiments using filtering-based VSLAM with ground truth, demonstrating the improved localization accuracy; (2) filtering-based VSLAM experiments on real-world datasets,

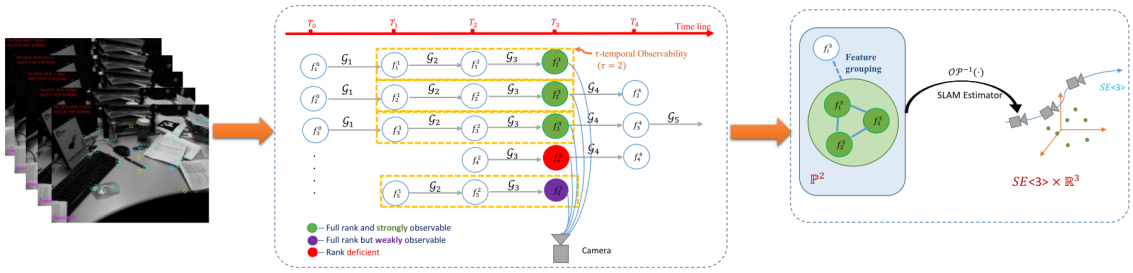


Figure 24: Overview of my approach. The proposed method can be plugged in as a sub-step in the SLAM process. In a time step (T_3 in the figure), for features which are initially matched, the algorithm first examines the rank conditions for them, i.e. whether the feature is **completely observable** to the SLAM system. If the rank condition of a feature is satisfied (depicted in green/purple), the τ-temporal observability score is evaluated by considering the relative motion of the feature in the past τ local frames. Features with high observability scores are selected as **good features** (depicted in green). If the number of highly observable features is too few, feature grouping with a submodular learning scheme is applied to collect more good features. These subset of good features provide the near-optimal value for SLAM estimation.

demonstrating the improved data association; (3) lifelong experiments with integration into a modern multi-thread VSLAM system (ORB-SLAM[70])on a benchmark dataset (KITTI dataset), demonstrating improvements in both localization/mapping accuracy and data association, especially improvements in combating drifts in sequences without loop-closures.

4.1 Introduction

The accuracy of the converged SLAM estimate is determined by the operator mapping the projective space of image observations to the space of camera motion and feature 3D positions, and its temporal dynamics, as indicated in the right block in Figure 24. Intuition then indicates that the better conditioned this operator is, the more tolerant the output space is to the perturbations in the input space. This operator encodes the camera motion across frames due to temporal coupling of SLAM estimates. To exploit nature of $SE\langle 3\rangle$ SLAM operator to feature ranking, I study the SLAM problem using system theory to define the **temporal observability scores** for feature selection.

Using systems theory, I develop a feature ranking criterion for selecting **individual** features which provide good conditioning for visual SLAM ego-motion estimation. The overview of my method is depicted in Figure 24. The contributions of this paper is three-fold: I

1. propose a feature ranking criterion based on **observability scores** using the second **completely observable** condition in Theorem 2.3.3 for $SE\langle 3\rangle$ SLAM;

2. describe an efficient algorithm for computing temporal observability based on incremental SVD;

3. describe an efficient algorithm for computing instantaneous observability via submodular learning.

The algorithm is called the "Good Features (GF)" algorithm and can be integrated into most existing VSLAM systems to arrive at "GF-SLAM".

The GF algorithm is extensively evaluated to demonstrate performance gains regarding ego-motion estimation and data-association in VSLAM. Section 4.6 includes three sets of experiments:

- Controlled synthetic experiments (in Sec.4.6.1) with EKF-VSLAM, which demonstrate the accuracy gain in ego-motion estimation over the information gain method.

- Real-world experiments with EKF-VSLAM (in Sec.4.6.2) on sequences with various dynamics, which demonstrate the improvements in data association and investigate into how the temporal-length affects the performance under different dynamics.

- Comprehensive long-term experiments(in Sec.4.6.3) by integrating with a multi-thread VSLAM system (ORB-SLAM), to demonstrate the utility of GF algorithm in the recent VSLAM system design. The experiments validate both

the performance in ego-motion estimation accuracy and data-association, on (1) the original KITTI visual odometry datasets with loop-closures, and (2) re-generated long sequences from the KITTI datasets without loop-closures in order to emphasize on improvements for drifts.

4.2 Background

In this section I first review the previous work on feature selection for VSLAM; then discuss VSLAM system designs is presented, because the designs concern how the GF algorithm can integrate into actual SLAM systems.

4.2.1 Individual Feature Selection in VSLAM

The earlier VSLAM systems are mostly filtering-based [27]. In such systems, information gain is widely used to select the features to be used in ego-motion estimation, since the information gain can be directly obtained from the filter such as the Extended Kalman Filter (EKF). Representative work includes [25, 57]. Information gain is also used to guide the feature matching by imposing the prior uncertainty information in matching range [68, 50]. Recently information gain is also used in landmark and pose reduction in graph-based SLAM [20]. The rationale behind information gain is that selecting the features which maximize the information gain in estimation will maximize the uncertainty reduction for both the camera pose and landmark positions. Thus, the convergence rate in terms of entropy is maximized, and within the same amount of iterations, the estimate with feature selection should be closer to the final converged estimate than estimate without the selection. Nevertheless, low uncertainty and fast convergence in estimation is not equivalent to high accuracy. For instance, if drift exists in the estimate from the selected feature set, the converged estimates with lowest uncertainty still suffer from the drift.

For bundle adjustment [101], which perform post-optimization assuming global information is available, various methods have been proposed for selecting the features

globally. Feature "visibility/co-visibility" is often exploited based on the assumption that features viewed by more cameras are more reliable. Such features also result in a better conditioned least-square system in the batch optimization. To exploit the co-visibility of features, [16] proposed to select the best subset of points from the complete structure of features-camera graph. ORB-SLAM [70] maintains a co-visibility graph in its keyframe BA thread to assist the points and keyframe selections using the "survival of the fittest" approach. While the co-visibility approach improves the BA, it only works for *post-optimization* of the camera trajectory.

Heuristic methods are also used in feature selection for VSLAM. LSD-SLAM [36], being a direct method which uses intensity discrepancy for ego-motion estimation, selects the features with large baselines between frames so that the $SIM\langle 3\rangle$ optimization will be better posed.

Besides ranking and selecting individual features, RANSAC-like estimation frameworks can also be viewed as feature selection methods. These methods select features by retrieving the inlier set with the maximum support or the most probable hypothesis. Traditional RANSAC methods [99, 39] are expensive for real-time VSLAM, and thus various approaches [103, 22] were later proposed to improve the computational efficiency, mainly by exploiting prior information of motion models. Such randomized methods are popular for ego-motion estimation in earlier VSLAM systems e.g. MonoSLAM [27] and 1-point RANSAC [22], and is still used in recent VSLAM, e.g. PTAM [60] ORB-SLAM [70], when no frame-to-frame information is available, especially during the map initialization or the recovery after tracking loss.

4.2.2 VSLAM System Designs

The major types of sequential VSLAM systems are reviewed in this section, in order to facilitate the discussion in Section 4.5 on how the GF algorithm can be integrated into these designs.

Filtering based VSLAM [27, 25, 68, 50, 22, 6, 116, 88] formulates the process dynamics and measurement with pinhole projection into a state-space model. The state vector consists of both the camera pose and velocity as well as the 3D position vectors of all feature points in the probabilistic map. Localization and mapping are performed simultaneously by a filter (such as EKF). Block updates can be used in the filter update step [22, 88, 116], so that the ego-motion sub-vector is updated with only a subset of features, and then mapping sub-vector is updated assuming the ego-motion is fixed.

Keyframe BA based VSLAM has recently become popular due to the computation and accuracy gains [92]. These systems, represented by PTAM [60] and ORB-SLAM [70], factor the VSLAM process into multiple parallel threads. PTAM operates two threads in parallel: a faster camera tracking thread responsible for estimating the camera pose locally, and a slower keyframe BA thread for global optimization. ORB-SLAM has a third thread responsible for loop-closure detection. Once a loop-closure is detected, constraints are imposed to the BA with feature fusion, which improves the posterior localization accuracy.

As compared to keyframe BA based systems, another type of SLAM systems which performs smoothing for **every** frame is represented by the work *(incremental) smoothing and mapping*, including $\sqrt{\text{SAM}}$ [28] and iSAM/iSAM2 [59, 58]. The recent work *concurrent filtering and smoothing* [107] also structures the local filtering and global smoothing into different threads. This body of work focuses more on multi-sensor navigation such as Vision and Inertial Navigation (VIN) and is out of the scope of this paper.

Another branch of typical VSLAM design is the "feature-less" monocular VSLAM, including semi-direct VSLAM [40] and direct VSLAM [36, 37]. These methods eliminate the feature extraction step, and optimize poses by minimizing *photometric errors*. The representative designs such as SVO [40] and LSD-SLAM [36] both divide

the local motion estimation and global mapping into separate threads. LSD-SLAM, the state-of-the-art direct method, is reported to have significant lower localization accuracy than "feature-based" including ORB-SLAM and PTAM [70].

In Section 4.6, I will demonstrate the results of integrating the GF algorithm into both filtering based VSLAM (particularly EKF-VSLAM) and keyframe BA based VSLAM (particularly ORB-SLAM).

4.3 Good Features to Track for Visual SLAM

Let $\mathcal{F}$ be the set of features being tracked during the monocular SLAM process. Much like [87] sought good features within an image for data association across frames, the *Good Features* algorithm here aims to find the subset of features which aids most the SLAM camera ego-motion estimates across time (in terms of accuracy and robustness to noise). This subset is selected by ranking features according to their contribution to system observability (higher system observability means better conditioned estimation). The score is formulated based on the observability of the subsystem composed of the camera and each individual feature.

4.3.1 Temporal Observability Score

Condition #2 in Theorem 2.3.3 states that, if a feature is tracked across two time segments (i.e. three frames), the VSLAM system with that feature can become completely observable. The degenerate conditions are typically of measure zero in the observation space. Tracking multiple features would guarantee observability for some subset of the tracked set. Under the observable condition for a feature, the value of a feature towards ego-motion estimation is reflected by the conditioning of the SOM.

Thus, I define the **τ-temporal observability score** of a feature across τ **local frames,** $\tau \geq 2$ with the minimum singular value of SOM:

$$\psi(f, \tau) = \sigma_{\min}(\mathcal{Q}_{\text{SOM}}(\tau|f)), \tag{66}$$

where at time k, $\mathcal{Q}_{\text{SOM}}(\tau|f)$ is defined on the time segments $(k-\tau), (k-\tau+1), ..., k$.

This temporal observability score measures how constrained the SLAM estimate is w.r.t. the feature observation in the projective space, when considering the relative poses of the feature and camera over a recent period of time. The temporal nature of the measure is important because the SLAM estimate, in both the filtering and smoothing versions, is performed across time, with the current estimate affected by the previous

4.3.2 Rank-k Temporal Update of Observability Score

Computation of the τ-temporal observability score is efficient.

Firstly, due to the sparse nature of the process matrix F, each subblock in $\mathcal{Q}$ can be computed iteratively with Equation 44.

Secondly, the running temporal observability score of a feature can be computed efficiently with incremental SVD. Computation of the τ-temporal observability score is divided into the following phases:

1. In the first two frames that a feature is tracked, the observability cannot be full-rank. Build the SOM;
2. In frame three, the full rank condition of SOM may be satisfied. Compute SVD of the SOM;
3. From frame 4 to frame $\tau + 1$ (in total τ time segments), for each new time segment a block of linear observability matrix is added to the SOM. Instead of computing SVD on the expanded SOM, perform a constant time rank-k update of the SVD [11], as per below.

   The SVD of $\mathcal{Q}_{\text{SOM}}(j)$ is $USV^{\intercal} = \mathcal{Q}_{\text{SOM}}(j)^{\intercal}$, where $S \in \mathbb{R}^{r\times r}$ with $r = 13$ (camera state). For the new row $\mathbf{a}^{\intercal}$, compute

$$\mathbf{m} \triangleq \mathbf{U}^{\intercal}\mathbf{a};\ \mathbf{p} \triangleq \mathbf{a} - \mathbf{U}\mathbf{m};\ \mathbf{P} \triangleq \mathbf{p}/||\mathbf{p}||. \tag{67}$$

Let

$$\mathbf{K} = \begin{pmatrix} \mathbf{S} & \mathbf{m} \\ 0 & ||\mathbf{p}|| \end{pmatrix}. \tag{68}$$

Diagonalize $\mathbf{K}$ as

$$\mathbf{U'}^{\mathsf{T}}\mathbf{K}\mathbf{V'} = \mathbf{S'} \tag{69}$$

and update

$$[\mathcal{Q}_{\mathrm{SOM}}(j)^{\mathsf{T}}|a] = ([\mathbf{U}\ \mathbf{P}]\mathbf{U'})\mathbf{S'}([\bar{\mathbf{V}}\ \mathbf{Q}]\mathbf{V'})^{\mathsf{T}} \tag{70}$$

where $\bar{\mathbf{V}}^{\mathsf{T}} = [\mathbf{V}^{\mathsf{T}}, \mathbf{0}]$, $\mathbf{Q} = [0, \cdots, 0, 1]^{\mathsf{T}}$. Diagonalization of $\mathbf{K}$ takes $\mathcal{O}(r^2)$ [47].

Expanding the SOM with more time segments results in adding $2r$ new rows into SOM. Each new row requires a rank-1 update, leading to rank-$2r$ update for the whole SOM.

4. After frame $\tau + 1$, for each new frame, update the SOM by replacing the subblock from the oldest time segment with the linear observability matrix of the current time segment. For example, let SOM at time k be $\mathcal{Q}^{(k)}_{\mathrm{SOM}}(\tau) = \left[\mathcal{Q}^{\mathsf{T}}_{k-\tau+1}|\mathcal{Q}^{\mathsf{T}}_{k-\tau+2}|\cdots|\mathcal{Q}^{\mathsf{T}}_{k}\right]^{\mathsf{T}}$, then at time k+1, $\mathcal{Q}^{(k+1)}_{\mathrm{SOM}}(\tau) = \left[\mathcal{Q}^{\mathsf{T}}_{\mathbf{k+1}}|\mathcal{Q}^{\mathsf{T}}_{k-\tau+2}|\cdots|\mathcal{Q}^{\mathsf{T}}_{k}\right]^{\mathsf{T}}$.

   Computing the SVD of $\mathcal{Q}^{(k+1)}_{\mathrm{SOM}}(\tau)$ given the SVD of $\mathcal{Q}^{(k)}_{\mathrm{SOM}}(\tau)$ can also be done with a rank-$2r$ update similar to phase 3. Let row b be replaced by row vector c in this case, by setting $a = (c - b)^{\mathsf{T}}$, the updated SVD is generated via (67)-(70).

After updating the τ-temporal observability scores of the features and ranking them, the top K_a features over a selected threshold are upgraded to be anchors. If the anchor set has less than $(K_a - 2)$ elements passing the threshold test, then additional features will need to be added to complete the anchor set.

4.4 Submodular Learning for Feature Grouping

A key remaining problem is how the system should handle when there are not enough *individually* observable features. In this case, grouping more than two feature may

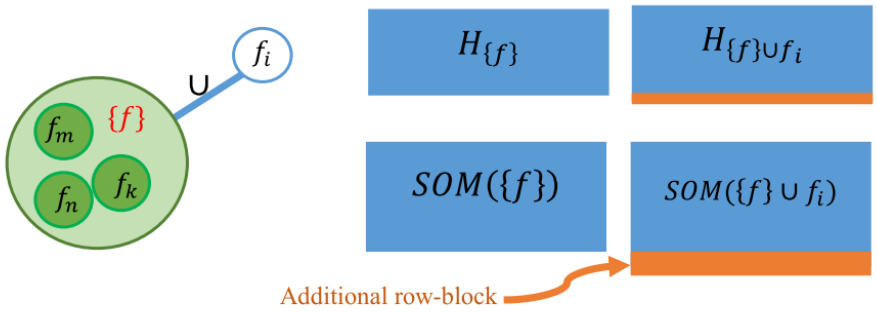


Figure 25: In spatial grouping, selecting one more feature as anchor results in an additional row-block in the measurement Jacobian, which further expands the SOM.

result in a completely observable VSLAM system, according to condition #1 in Theorem 2.3.3.

The group completion step selects more features as anchors by maximizing the minimum singular value of SOM over the selected features. Upgrading a feature to be an anchor will expand the dimension of F and H in Equation (8)-(17), resulting in additional rows in SOM.

The group completion problem can be formulated as follows: Let X be the SOM of the features with high observability score, $X \in \mathbb{R}^{n\times m}, n \geq m$. Adding a feature results in adding a row-block R_k to the SOM as in Figure 25. Denote the set of all candidate row-blocks as $\mathbf{R} = \{R_1, R_2, ..., R_K\}$, $R_k \in \mathbb{R}^{n'\times m}$. Finding K^* features which form the most observable SLAM subsystem is equivalent to finding a subset of the candidate rows that maximize the minimum singular value of the augmented matrix

$$\mathbf{R}^* = \underset{\mathbf{R}^*\subseteq\mathbf{R},|\mathbf{R}^*|=K^*}{\operatorname{argmax}} \sigma_{min}\left([X^\intercal|R_1^{*\intercal}|R_2^{*\intercal}|...|R_{K^*}^{*\intercal}]^\intercal\right) \tag{71}$$

Such a combinatorial optimization problem is NP-hard. However, the problem has nice submodular properties.

Definition 4.4.1. *[61]* ***(Approximate submodularity)***

A set function $F : 2^{\mathbf{V}} \mapsto \mathbb{R}$ *is approximately submodular if for* $\mathbf{D} \subset \mathbf{D}' \subset \mathbf{V}$ *and*

$v \in \mathbf{V} \setminus \mathbf{D}'$

$$F(\mathbf{D} \cup \{v\}) - F(\mathbf{D}) \geq F(\mathbf{D}' \cup \{v\}) - F(\mathbf{D}') - \varepsilon \tag{72}$$

Theorem 4.4.1. *When $\mathbf{X} \cap \mathbf{R} = \emptyset$, the set function $F_{\sigma_{min}}(\cdot) : 2^{\mathbf{X} \cup \mathbf{R}} \mapsto \mathbb{R}$ is approximately submodular,*

$$F_{\sigma_{min}}(\mathbf{X} \cup \mathbf{R}^*) = \sigma_{min}\left([X^\intercal | R_1^{*\intercal} | R_2^{*\intercal} | ... | R_{K^*}^{*\intercal}]^\intercal\right). \tag{73}$$

The proof requires the following two lemmas.

Lemma 4.4.1. *[10]* ***(Concavity of min eigenvalue function)*** *For any real symmetric matrix $G \in \mathbb{R}^{m \times m}$, let $f(G) \triangleq \lambda_{min}(G)$, $f(G)$ is a concave function of G.*

Lemma 4.4.2. *[42]* ***(Eigenvalues of sum of two matrices)*** *Let A, B, C be Hermitian n by n matrices, denote the eigenvalues of A by $\alpha : \alpha_1 \geq \alpha_2 \geq ... \geq \alpha_n$, and similarly write β and γ for eigenvalues of B and C, then:*

$$\gamma_{i+j-1} \leq \alpha_i + \beta_j \quad \textit{whenever } i + j - 1 \leq n. \tag{74}$$

Proof. **(Theorem 4.4.1)** WLOG consider the two row-blocks R_1 and R_2 from $\mathbf{R}$. Denote the Gram matrices $G_\circ$ as:

$$G_X = X^\intercal X,\ G_{R_1} = R_1^\intercal R_1, \text{ and } G_{R_2} = R_2^\intercal R_2.$$

Also define the augmented Gram matrices as

$$G_{XR} = \left(X^\intercal | R^\intercal\right) \cdot \begin{pmatrix} X \\ R \end{pmatrix}$$

It holds that $G_{XR_1} = G_X + G_{R_1}$, $G_{XR_2} = G_X + G_{R_2}$, $G_{XR_1R_2} = G_{XR_1} + G_{R_2}$, and $G_{XR_2R_1} = G_{XR_2} + G_{R_1}$. Let the minimum eigenvalue of G_X be $\lambda_{min}(G_X) \equiv \lambda_m(G_X)$, the maximum eigenvalue be $\lambda_{max}(G_X) \equiv \lambda_1(G_X)$. Since X is a real matrix,

$\lambda_{min}(G_X) = \sigma^2_{min}(X)$, and likewise for the augmented matrices. From Lemma 4.4.1,

$$\begin{aligned} \lambda_{min}\left(G_{XR1}\right) &= \lambda_{min}\left(G_X + G_{R1}\right) \\ &\geq \left(\lambda_{min}\left(G_X\right) + \lambda_{min}\left(G_{R1}\right)\right) \\ &\geq \lambda_{min}\left(G_X\right) \end{aligned} \tag{75}$$

Thus, $F_{\sigma_{min}}\left(\mathbf{X} \cup \{R_1\}\right) \geq F_{\sigma_{min}}\left(\mathbf{X}\right)$.

From Lemma 4.4.2, and the fact that the Gram matrices are real-symmetric and hence Hermitian, the following holds:

$$\lambda_{min}\left(G_{XR1R2}\right) = \lambda_{m+1-1}\left(G_{XR1R2}\right) \leq \lambda_m\left(G_{XR2}\right) + \lambda_1\left(G_{R1}\right) \tag{76}$$

Combining (75) and (76),

$$\lambda_{min}\left(G_X\right) + \lambda_{min}\left(G_{XR1R2}\right) \leq \lambda_{min}\left(G_{XR1}\right) + \lambda_{min}\left(G_{XR2}\right) + d_\rho(R_1),$$

where $d_\rho(R_1) = \lambda_{max}(R_1) - \lambda_{min}(R_1)$. Similarly,

$$\lambda_{min}\left(G_X\right) + \lambda_{min}\left(G_{XR1R2}\right) \leq \lambda_{min}\left(G_{XR1}\right) + \lambda_{min}\left(G_{XR2}\right) + d_\rho(R_2)$$

The tighter bound is:

$$\lambda_{min}\left(G_X\right) + \lambda_{min}\left(G_{XR1R2}\right) \leq \lambda_{min}\left(G_{XR1}\right) + \lambda_{min}\left(G_{XR2}\right) + \min\left(d_\rho(R_1), d_\rho(R_2)\right).$$

This leads to

$$\begin{aligned} F_{\sigma_{min}}\left(\mathbf{X} \cup \{R_1\}\right) + F_{\sigma_{min}}\left(\mathbf{X} \cup \{R_2\}\right) \geq F_{\sigma_{min}}\left(\mathbf{X}\right) + \\ F_{\sigma_{min}}\left(\mathbf{X} \cup \{R_1\} \cup \{R_2\}\right) - \min\left(d_\rho(R_1), d_\rho(R_2)\right). \end{aligned} \tag{77}$$

When $\mathbf{X} \cap \mathbf{R} = \emptyset$, $F_{\sigma_{min}}(\cdot)$ is approximately submodular, with the bound $\varepsilon = \max(d_\rho(R_k)),\ \forall R_k \in \mathbf{R}$. □

Theorem 4.4.1 means that a greedy algorithm will be near-optimal. The simplest greedy algorithm outline in Algorithm 3 identifies the group completion in the cardinality deficient case with a complexity of $\mathcal{O}(K^*Kn')$ (when using incremental SVD). The near-optimality bound is

Algorithm 3: Submodular learning for feature grouping.

Data: $X \in \mathbb{R}^{n \times m}, n \geq m$, $\mathbf{R} = \{R_1, R_2, ..., R_K\}$, $R_k \in \mathbb{R}^{1 \times m}$, K^*
Result: $\mathbf{R}^*$, $|\mathbf{R}^*| = K^*$

```
R* ← ∅;
while |R*| < K* do
    R* ← arg max_{R*∈R} F_{σ_min}(X ∪ {R*});
    R* ← R* ∪ {R*};
    R ← R \ {R*};
```

Theorem 4.4.1. *[61]. Let* $\mathbf{A_G}$ *be the set of the first* K^* *elements chosen by Algorithm 3, and let* $OPT = \max_{\mathbf{A} \subset \mathbf{R}, |\mathbf{A}| = K^*} F_{\sigma_{min}}(\mathbf{X} \cup \mathbf{A})$. *Then*

$$F_{\sigma_{min}}(\mathbf{A_G}) \geq \left(1 - \left(\frac{K^* - 1}{K^*}\right)^{K^*}\right)(OPT - K^*\varepsilon) \tag{78}$$

4.5 Integration into VSLAM Systems

The proposed GF algorithm provides a ranking of features which can be used in different phases:

1. **Ego-motion estimation.** After data-association but prior to post-optimization, the GF algorithm can be used to select a subset of features from the best matched measurements, so that both the data-association scores and the observability scores are considered. Localization is performed using only the subset, while the mapping is performed on the whole feature set based on the localization results (acting as external input).

2. **Data-association.** The observability scores can be used in some data-association processes. For example, in 1-Point RANSAC method, for each iteration the features with high observability scores are used to partially update the model, which is then used to retrieve the inlier set.

The GF algorithm is complementary to different VSLAM system designs:

- **Filtering based VSLAM.** The GF algorithm can be directly applied to filtering based VSLAM in ego-motion estimation or data-association. For ego-motion estimation, block update of the filter can be performed to partially update the camera pose sub-vector with only the highly ranked GF subset (i.e. anchors), and then the whole mapping sub-vector is updated based on the ego-motion estimate. For data-association, the GF algorithm can assist in generating RANSAC hypotheses, as discussed above.

- **Keyframe BA based VSLAM.** Since keyframe BA VSLAM typically has a local motion tracking thread operates in filtering manner, the GF algorithm can naturally be integrated into these systems by applying to the motion tracking thread. Particularly, ego-motion is estimated with only GFs selected from the initially matched features.

4.6 Evaluation

In this section, three sets of experiments are presented to demonstrate the performance of the proposed algorithm when integrated into different VSLAM systems, with focuses on localization and data-association. The synthetic experiments illustrate applications of GF algorithm for ego-motion estimation in filtering based VSLAM; the real-world EKF-VSLAM experiments illustrate integration of GF algorithm into the data-association step of filtering VSLAM; the real-world keyframe BA experiments demonstrate the GF algorithm performance with ORB-SLAM, with evaluations regarding both ego-motion estimation and data-association.

4.6.1 Synthetic Experiments with EKF-VSLAM for Ego-motion Estimation

Evaluation of the proposed method for the ego-motion estimation phase focuses on the estimation accuracy. Therefore this experiment will isolate the data association error from the localization error such that the SLAM accuracy is only affected by the

selection of anchors. Precisely benchmarking the SLAM accuracy is a difficult task, because most of the publicly available datasets do not provide exact ground truth and perfect data association. The usual SLAM baseline for evaluating accuracy is a global optimization, usually bundle adjustment [6, 119]. However, these data-driven baseline methods are not actual ground truth.

4.6.1.1 Experimental Scenarios

To perform controlled experiments for accuracy evaluation, I use camera motion and observation simulation modules from software in [88] which assumes perfect data association, but implements the SLAM estimation process. Two scenarios are simulated. The simulated environment is of dimension $12m \times 12m$ with 72 landmarks forming a square. Two scenarios are tested. In the first one, the robot performs circular trajectory as in Column 1, Figure 26. The second scenario simulates a more cluttered scene. The robot moves away from the landmarks while performing slight rotation as shown in Figure 27.

4.6.1.2 Experiment Setup and Comparison

In each time step, ego-motion estimation is performed with an Extended Kalman Filter **only with the anchors**, while the features are estimated based on the ego-motion estimate. Experiments are performed with different levels of observation noise and anchor set sizes. I tested the configurations with standard deviation of observation noise of 0.5, 1.0, 1.5, 2.0, 2.5 pixels under Gaussian noise, and maximum anchors sets of $K_a = 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 10,\ 12$. The temporal parameter $\tau = 5$ is used in my method. The threshold for observability score is 0.003. In cases with less than $K_a - 2$ strongly observable features, at most 2 more features are added via spatial grouping. The baseline state-of-the-art method uses information gain for feature selection [57]. The same ego-motion estimation and mapping scheme is applied on both methods. Due to the randomized effects from the noise simulation, 15 experiments are run per

configuration.

4.6.1.3 Metrics

Localization accuracy is evaluated by the cumulative translation errors and cumulative orientation errors. Let $\Delta\mathbf{r}^{\mathcal{W}}_{\mathcal{R}_k}$ be the translation error at time k, $\Delta\theta^{\mathcal{W}}_{\mathcal{R}_k}$ be the orientation error in Euler angles, $\sum_k ||\Delta\mathbf{r}^{\mathcal{W}}_{\mathcal{R}_k}||_2$ and $\sum_k ||\Delta\mathbf{r}^{\mathcal{W}}_{\mathcal{R}_k}||_\infty$ are used for evaluating cumulative translation errors, and accordingly $\sum_k ||\Delta\theta^{\mathcal{W}}_{\mathcal{R}_k}||_2$ and $\sum_k ||\Delta\theta^{\mathcal{W}}_{\mathcal{R}_k}||_\infty$ for cumulative orientation errors. The average value of the 15 runs are used as the final evaluation result for each configuration.

4.6.1.4 Results

The evaluation results are shown in Figure 28 and Figure 29. For the interest of space, configurations of #Anchors $\in \{3, 4, 5, 10\}$ are displayed for scenario #1 to highlight both the extreme cases and saturated cases, and #Anchors $\in \{3, 4, 8, 10\}$ for the more cluttered scenario #2. my method outperforms the information gain based method in 92.5% (37/40) cases for translation and 82.5% (33/40) for orientation in scenario #1; 85% (34/40) for translation and 95% (38/40) for orientation in scenario #2. These ratios are the same for both l_2-norm and l_∞-norm metrics.

4.6.2 Real-world Experiments with EKF-VSLAM for Data-association

The proposed method is tested in data-association with real scenes and via modification of the baseline SLAM system (1-Point RANSAC) from [22]. For data-association, the features are first matched with individual compatibility. Then in each iteration of 1-Point RANSAC, one feature measurement is selected randomly to partially update the localization, which further generates a hypothesis to retrieve the inlier set. The maximum supported hypothesis is used as the data-association results. The Good Features modification changes selection of the feature for hypothesis generation such that strongly observable features are selected.

4.6.2.1 Dataset

For the purpose of evaluating the effect of temporal parameter, I collected videos under smooth motion and highly dynamic motion respectively. I use three videos for each type of motion respectively. The videos are collected in 640×480 resolution and 40 fps frame rate. Each video clip has about 2300 frames.

4.6.2.2 Experiment Setup and Comparison

The code was written in C++ with OpenCV and Armadillo following the pipeline described in [22]. The experiments are run on a 2.7GHz 8-core PC with 16GB RAM. For my method, the strongly observable features quantity parameter is set to K_a, which are then used to generate the data-association hypothesis. I tested my method with temporal parameter $\tau \in \{3, 5, 7, 9, 11\}$. Some example frames under three motion segments are shown in Figure 30.

4.6.2.3 Metrics

I evaluate data-association results by comparing average inlier ratios of the maximum supported data-association hypothesis. The inlier ratio is defined as

$$\Gamma = \frac{\#inlier}{\#inliers + \#outliers}. \tag{79}$$

4.6.2.4 Results

The relative improvements of the inlier ratios from my good features for SLAM method (denoted as Γ_{GFSLAM}) over that from [22] (denoted as Γ_0) are shown in Figure 31. my method outperforms [22] in all the datasets by at least $\approx$5.5%. For the slow motion, the inlier ratio of my method has the peak value with $\tau \in [9, 11]$. For the fast motion, the peak value is at about $\tau \in [5, 7]$.

4.6.3 Experiments with Keyframe BA Based VSLAM

To demonstrate the performance of GF algorithm with recent keyframe BA based VSLAM, ORB-SLAM [70] is used as the baseline system due to its state-of-the-art performance demonstrated. The ORB-SLAM uses the efficient ORB features [82] and has three threads running in parallel: (1) the tracking thread which is responsible for frame-by-frame ego-motion estimation and keyframe insertion; (2) the mapping thread is in charge of keyframe based BA for posterior optimization of both the map and camera poses; (3) the loop closing thread is for detecting loop-closures, computing the similarity transformation when loop-closure detected, and global optimization over the similarity constraint.

4.6.3.1 Experiment Setup

To integrate the GF algorithm, a feature selection step is added in the tracking thread, such that only the selected GFs are used in the ego-motion estimation. In details, after the original matched feature set is found, the temporal observability score with temporal length three is computed for each tracked feature. In the actual implementation, the observability matrix of each tracked feature is maintained across the frames. Thus in each time instance the observability scores only need to be updated with the additional observability matrix block. Moreover, the computation for the whole feature set is multi-threading, and 50 threads are used in the experiments. The features are then ranked according to their observability scores. The highly ranked subset is selected to be used in the ego-motion estimation. The keyframe global mapping thread and the loop closing thread are kept the same as in the original ORB-SLAM for fair comparisons.

Since the numbers of matched features in each frame vary along the trajectory, no absolute number is imposed on the size of GF subsets. Instead, the GFs are selected from the **top K percentage** of features with **none-zero observability**

scores. Furthermore, to illustrate how the value of K affects the performance, $K = [20, 30, 40, 50, 60]$ percents are tested. The system is configured to detect 1500 keypoints in each frame as the real-time requirement allows. Tracking more keypoints also makes it more challenging for the GF algorithm since non-informative features are more likely to appear.

Each combination of the experiments is executed for five times. The experimental result with the median localization accuracy out of the five runs is selected for quantitative comparison. The experiments are run on the a Linux desktop with a CPU of Intel Core i5 quadcore 2.8GHz and memory of 8 GB, executed in ROS Indigo with the highest system priority.

4.6.3.2 Dataset

The KITTI visual odometry dataset [44] is used for experiments. The dataset sequences are of large-scale, captured from the city of Karlsruhe, Germany. Stereo images are collected at 10 Hz with typically 1226×370 resolution. Only the images from the left camera are used for monocular VSLAM experiments. Accurate localization ground truth is provided by a differential GPS/INS system. The sequences of 00, 02, 03, 04, 05, 06, 07, 08, 09, 10 are tested since the ground truth is publicly available, in which sequences 00, 02, 05, 06, 07, 09 contain loop-closures. This dataset is challenging because of its large scale (typically of lengths of kilo-meters), versatile motions (e.g. fast vehicle turning), and challenging scenes (e.g. scenes with few features or moving objects, scenes under illumination changes). Figure 32 shows three examples from the KITTI set. Figure 33 shows two examples of the system in action.

To investigate the drifts in long-distance trajectories without loop-closures, four new sequences are generated by the *longest none-loop-closure sub-sequence* in the original KITTI sequence 02, 06, 07, 09 (denoted as "xx Non-LC" as compared to

Table 2: Configurations of the new sequences without loop-closures. *Frame indices: corresponding indices in the original KITTI sequences.

Sequence	Dimension (m×m)	Frame indices*
02 Non-LC	599×946	$0 \sim 4160$
06 Non-LC	23×457	$0 \sim 800$
07 Non-LC	191×209	$0 \sim 980$
08 Non-LC	808×391	$0 \sim 4070$
09 Non-LC	465×559	$0 \sim 1520$

the original sequences). The dimensions and the corresponding frame indices in the original sequences are summarized in Table 2. Together with the original KITTI 08 sequences, these five sequences form the second set of experimental data without loop-closures, but with challenging motion as well as long enough distances. KITTI 03 and 10 are not included because they are not challengingly long, although they do not have loop-closures either.

4.6.3.3 Evaluation metrics

Localization accuracy and data-association inlier ratio are used for quantitative evaluation.

Localization accuracy is measured by the mean square root of the translation error. A $SIM\langle 3\rangle$ transformation is solved to match the estimated trajectory to the ground truth [93], with a fine-grain parameter sweep to find the optimal scaling factor.

Inlier ratio is defined as in Equation 79, but the number of inliers is counted in a different way from the last experiment. Since the ego-motion is estimated with the *g2o* framework [62], the inlier set is also retrieved with *g2o*. If the χ^2 error of a feature is smaller than 6.0 in the graph with the estimated pose, then the feature is considered as an inlier; otherwise it is considered as an outlier.

4.6.3.4 Experimental Results on Original KITTI Sequences with Loop-closures

The results on original KITTI set is summarized in Table 3 for GF percentages from 20% to 60%. Except for 04, 05, 10, the best result happens in GF-ORB-SLAM.

Notice that sequences 04, 05, 10 are all relatively simple sequences with the RMSE smaller than 10m, and the absolute RMSE differences between original ORB-SLAM and GF-ORB-SLAM are small.

The actual numbers of features used in ego-motion estimation are showed in Table 4. Note that because top $K\%$ GF are selected from the completely observable features with **none-zero** observability scores, the ratio of feature number in GF-ORB-SLAM over the original ORB-SLAM is usually smaller than K.

The average inlier ratios are plotted in Figure 34 for different sequences and GF-ORB-SLAM configurations. It can be observed that for KITTI dataset, the best inlier ratio typically happens with 40% or 50% GFs, except for sequence 10. The inlier ratios also show positive correlation with the localization accuracy.

The final estimated trajectories, ground truth, and the differences are visualized in Figure 35 to 37. With the loop-closures, the final estimated trajectories under different GF ratios appear to be very similar. However, during the actual tracking process, the estimated trajectories **before** loop-closure are more different due to the drifts. Figure 38 illustrates an example from the sequence 05 at 243 sec before a loop-closure happens. The GF-ORB-SLAM demonstrates a smaller drift than the original ORB-SLAM at the affinity of the loop-closure point. This observation also motivates the next experiment in which no loop-closure happens.

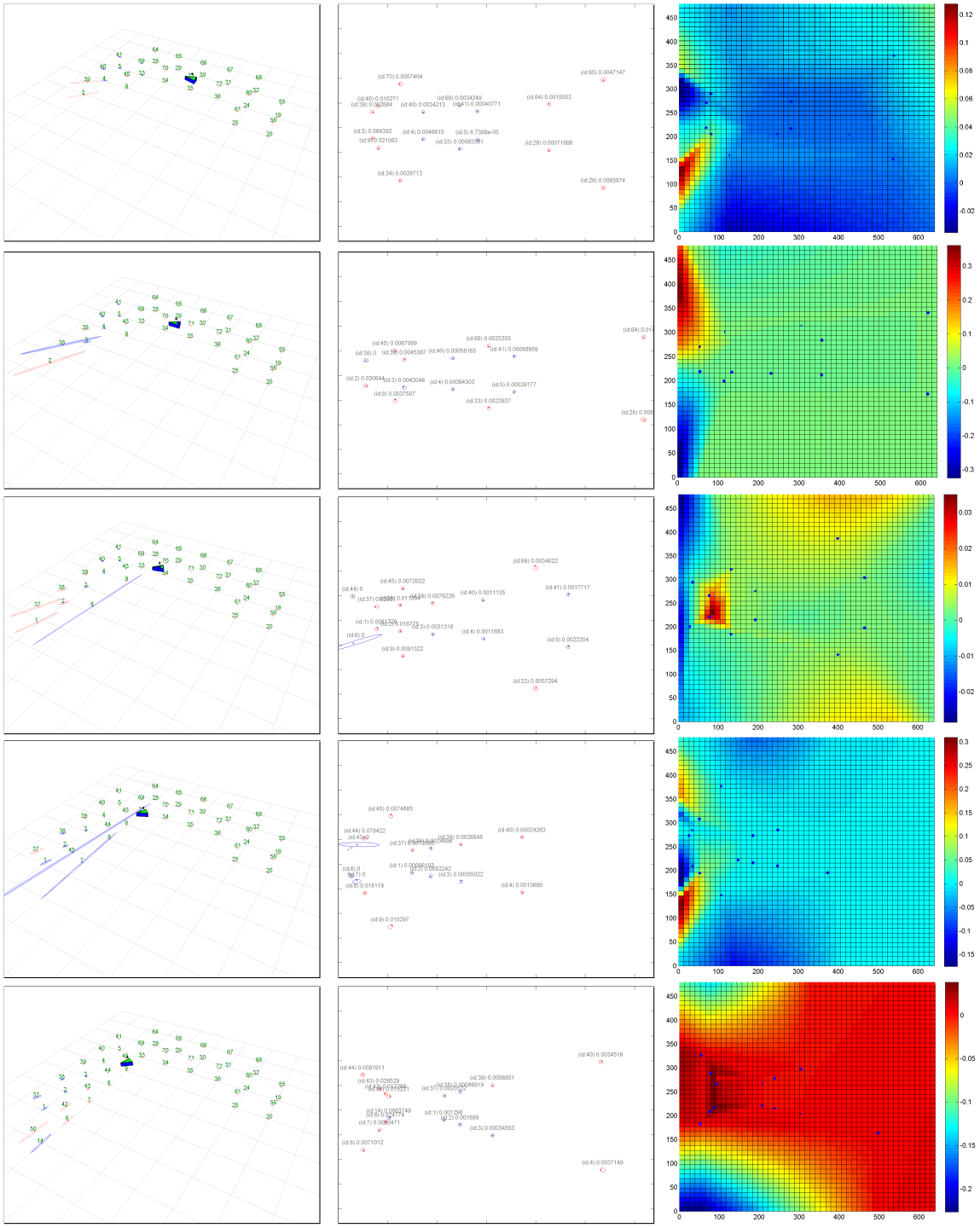


Figure 26: Simulated scenario #1 for ego-motion estimation experiment. Results shown have 1.0 pixel measurement standard deviation and $K_a = 10$. Column 1: reconstructed maps at time steps when camera is performing circular movement; features are depicted with estimated mean and covariance; points in red are selected as anchors. Column 2: corresponding camera frames with observability scores shown for all measurements. Column 3: interpolated maps of observability score on image plane showing how it changes during the motion.

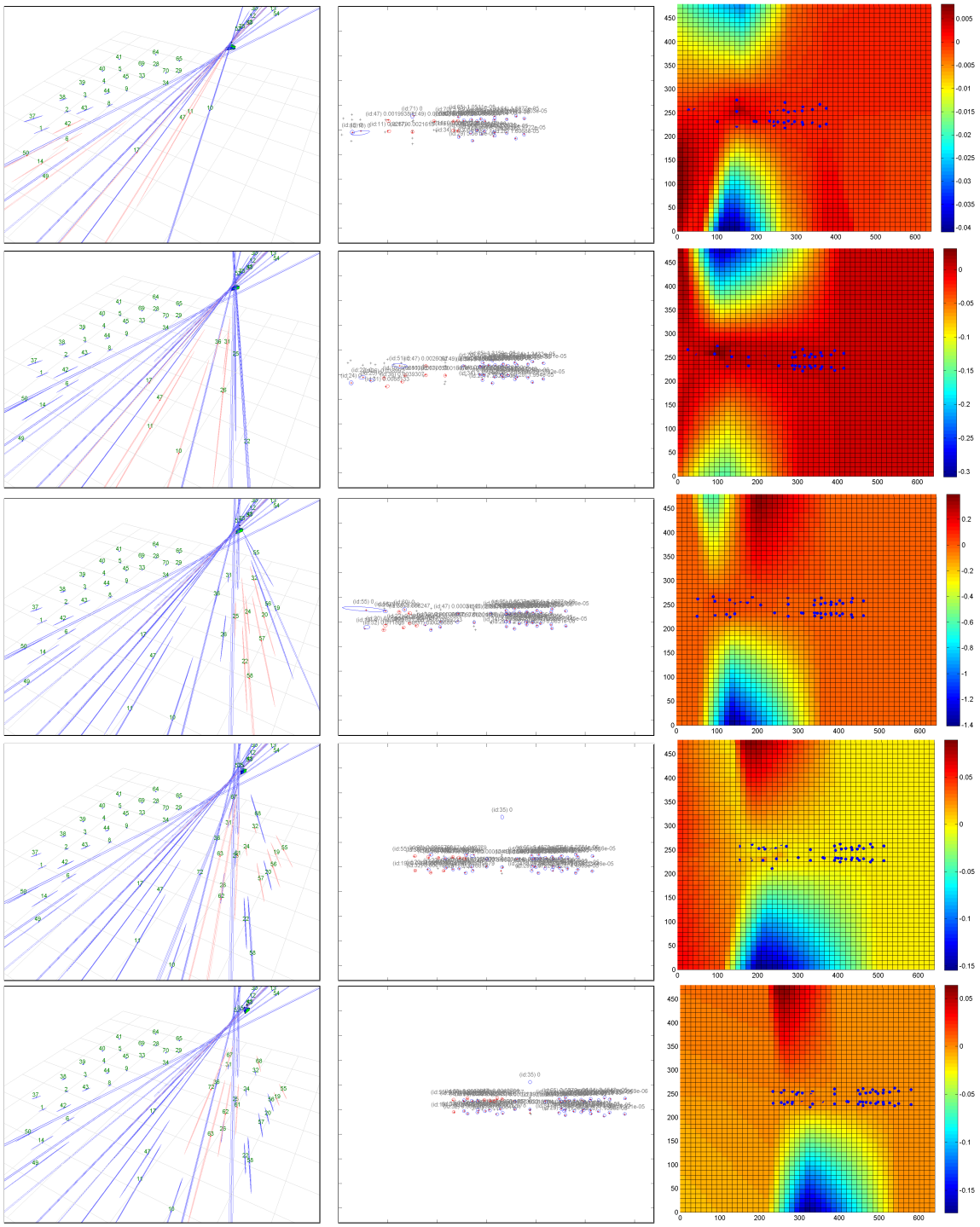


Figure 27: Simulated scenario #2 for ego-motion estimation experiment. Results shown have 1.0 pixel measurement standard deviation and $K_a = 10$. Column 1: reconstructed maps at time steps when camera is performing circular movement; features are depicted with estimated mean and covariance; points in red are selected as anchors. Column 2: corresponding camera frames with observability scores shown for all measurements. Column 3: interpolated maps of observability score on image plane showing how it changes during the motion.

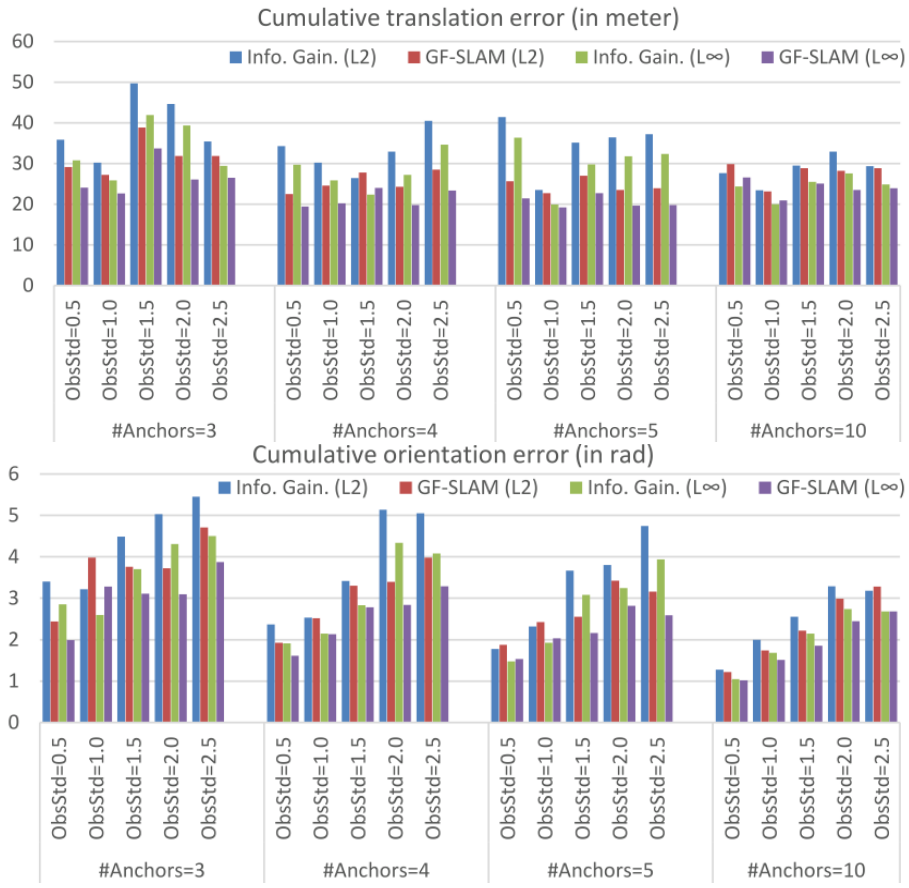


Figure 28: Results of simulation scenario #1 with cumulative translation errors and cumulative orientation errors. "ObsStd"stands for the standard deviation of observation noise in pixel units.

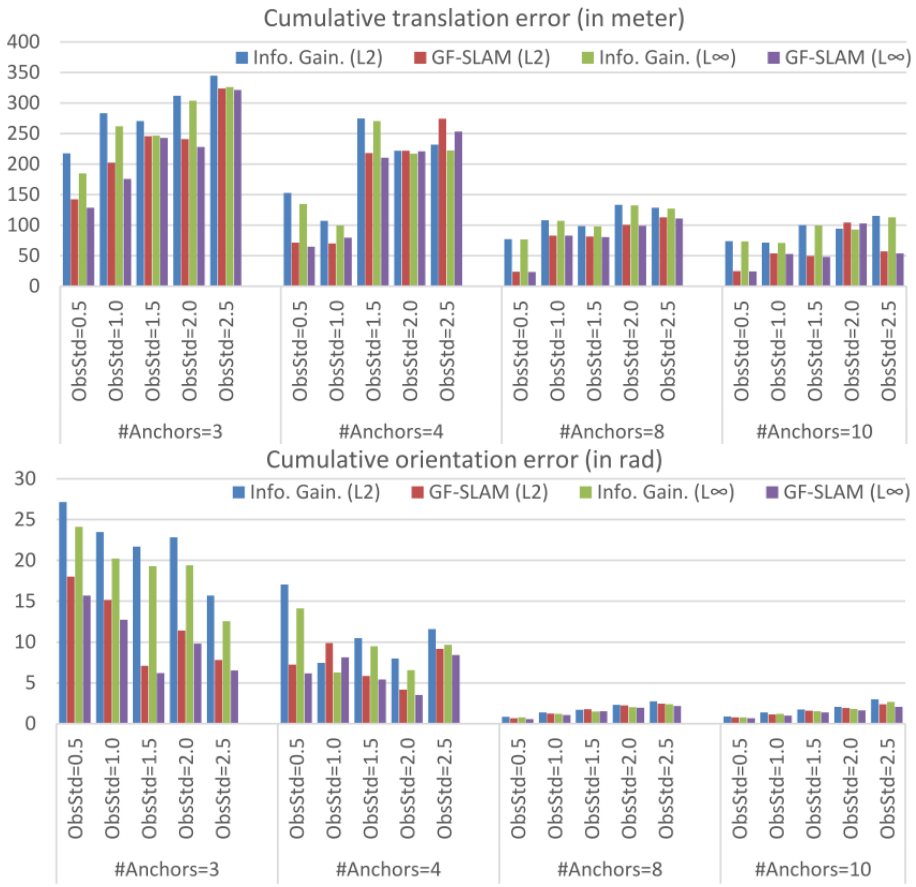


Figure 29: Results of simulation scenario #2.

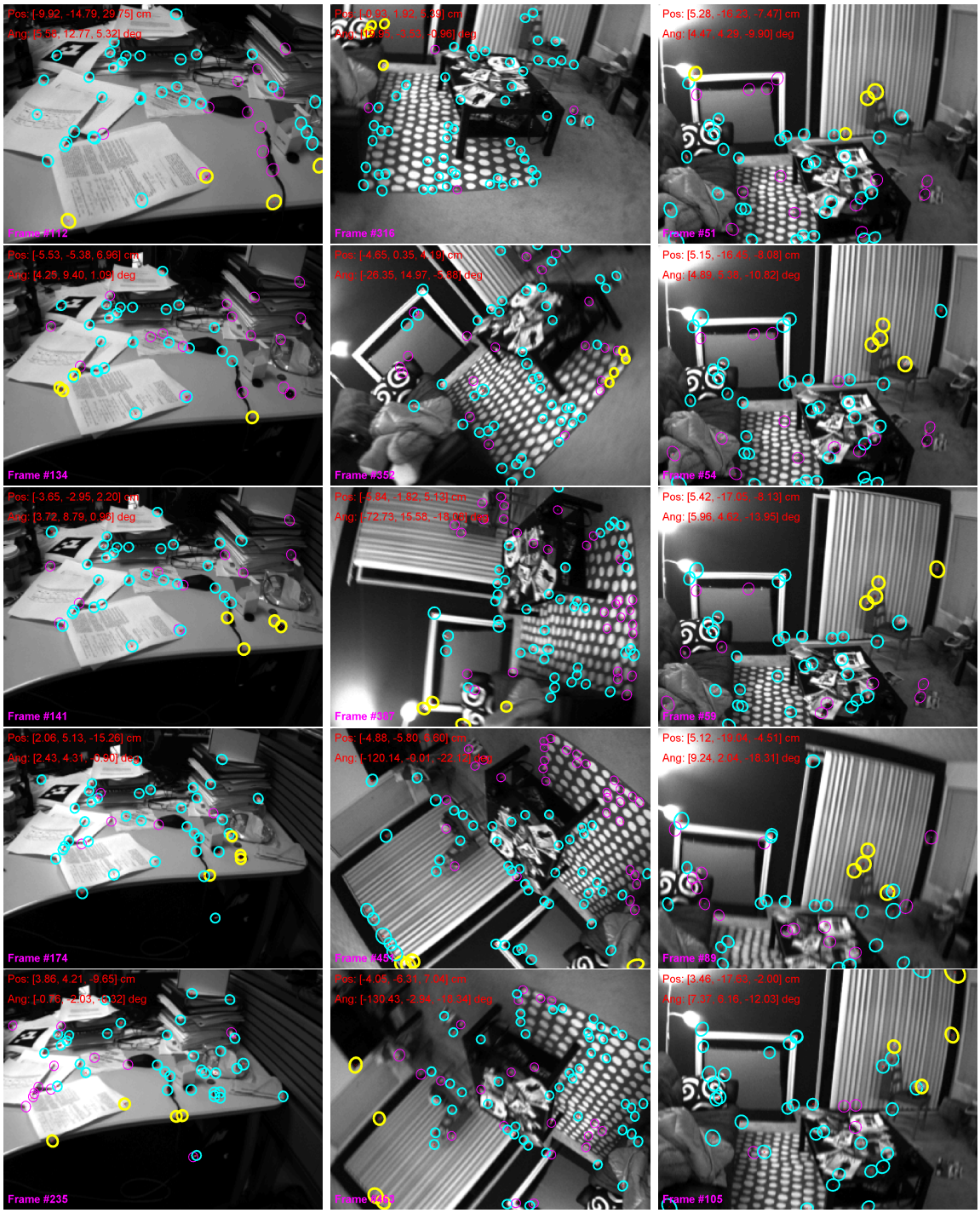


Figure 30: Example frames from data-association experiment. The strongly observable features are illustrated in yellow, retrieved inlier set is in cyan, and the outlier set is in purple. Column 1: camera is moving away from the desktop. Column 2: camera is rotating w.r.t. the optical axis. Column 3: camera is rotating w.r.t. the x axis of camera.

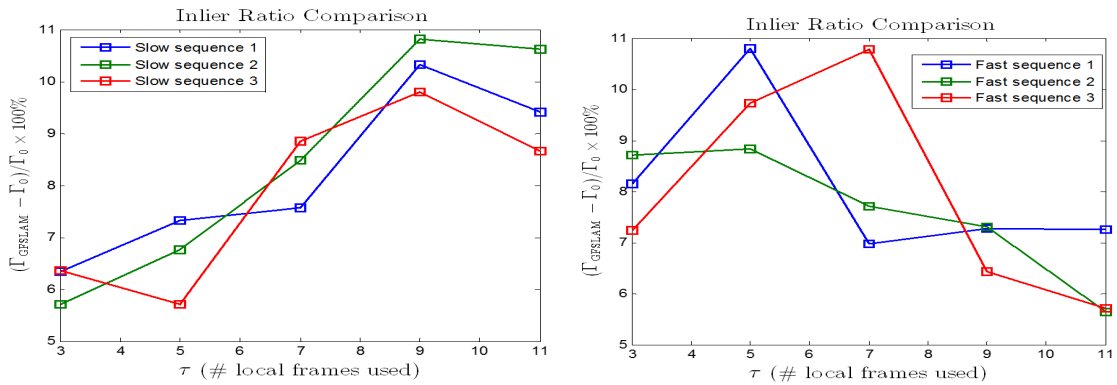


Figure 31: Relative improvements of inlier ratios versus [22].

Figure 32: Three example frames from the KITTI visual odometry sequences.

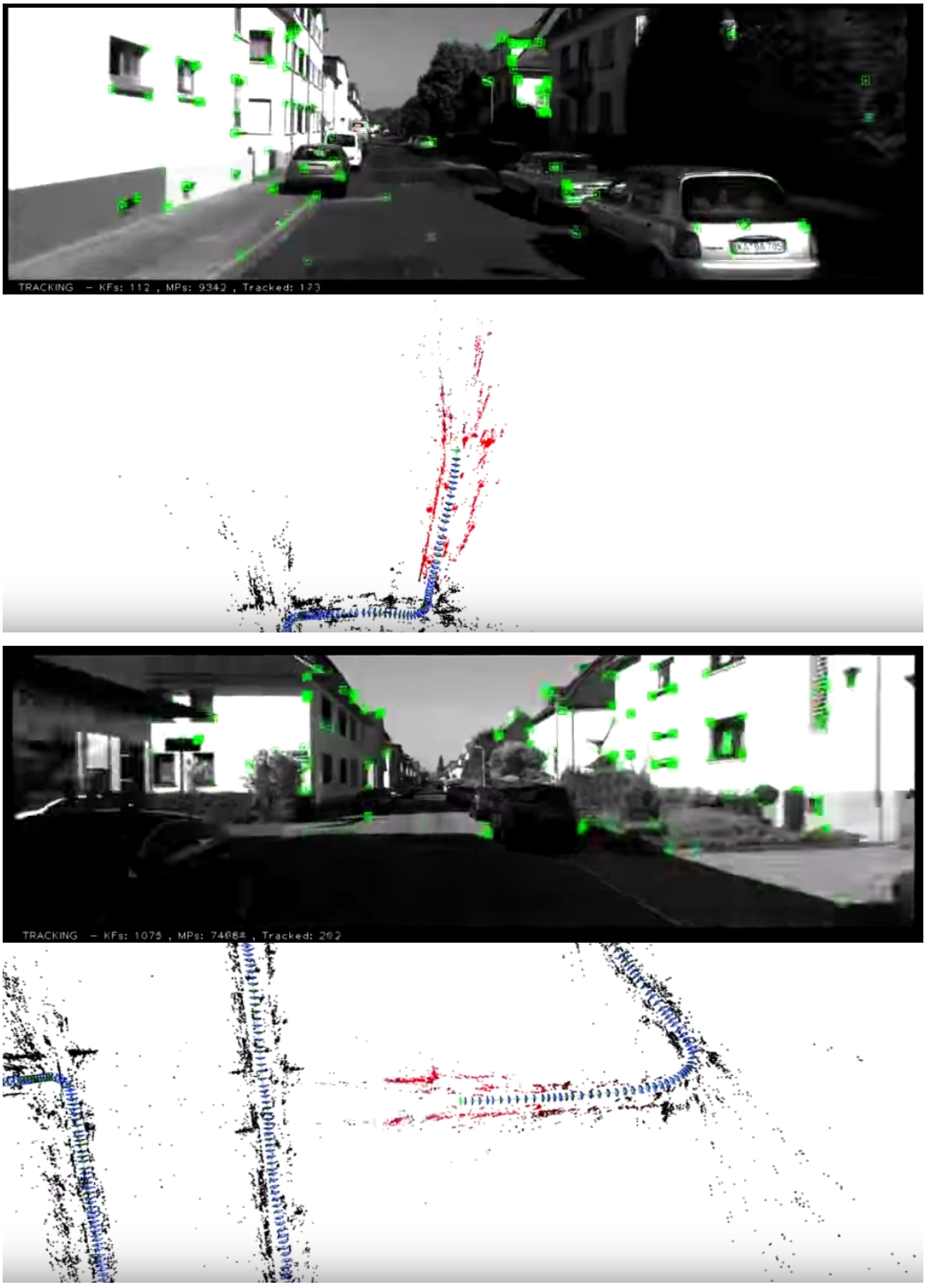


Figure 33: Two examples of the keyframe BA VSLAM system in action, on the KITTI dataset. In each sub-figure, top: current frame plotted with tracked visual features; bottom: estimated camera trajectory and 3D positions of the features.

Table 3: Results on original KITTI sequences. Keyframe numbers and translation RMSE are reported for both original ORB-SLAM and GF-ORB-SLAM with different GF selected.

Sequence	ORB-SLAM	GF-ORB-SLAM (top 20% gf)	GF-ORB-SLAM (top 30% gf)	GF-ORB-SLAM (top 40% gf)	GF-ORB-SLAM (top 50% gf)	GF-ORB-SLAM (top 60% gf)
	RMSE (m)	RMSE (m)	RMSE (m)	RMSE (m)	RMSE (m)	RMSE (m)
KITTI 00	8.53	6.48	7.20	7.48	7.56	7.69
KITTI 02	20.66	17.43	15.48	15.86	16.93	18.67
KITTI 03	1.40	1.25	1.35	1.12	1.35	1.46
KITTI 04	1.78	1.84	2.10	1.82	1.96	1.98
KITTI 05	5.58	6.06	5.83	6.02	5.91	5.86
KITTI 06	14.52	14.25	13.33	16.13	15.50	14.75
KITTI 07	3.08	2.78	2.58	2.55	2.61	2.70
KITTI 09	7.60	7.31	8.54	7.28	7.22	7.31
KITTI 10	8.51	9.30	9.26	9.06	9.07	8.54

Table 4: Numbers of features (mean $\pm$ std. dev.) used for pose optimization in the original KITTI sequences.

Sequence	ORB-SLAM	GF-ORB-SLAM (top 20% gf)	GF-ORB-SLAM (top 30% gf)	GF-ORB-SLAM (top 40% gf)	GF-ORB-SLAM (top 50% gf)	GF-ORB-SLAM (top 60% gf)
KITTI 00	183.88 ± 94.56	35.54 ± 18.72	53.79 ± 28.49	73.50 ± 37.50	91.89 ± 48.61	105.13 ± 63.36
KITTI 02	130.93 ± 45.51	25.24 ± 9.35	37.41 ± 13.40	52.12 ± 19.09	64.83 ± 26.01	65.18 ± 36.40
KITTI 03	187.90 ± 78.52	36.38 ± 15.76	55.23 ± 23.65	75.38 ± 31.46	93.55 ± 39.90	110.23 ± 51.80
KITTI 04	126.69 ± 30.23	24.30 ± 6.28	36.71 ± 9.09	50.12 ± 13.44	61.20 ± 18.94	59.94 ± 29.21
KITTI 05	180.45 ± 69.84	35.63 ± 14.15	52.51 ± 20.85	72.46 ± 27.96	88.81 ± 35.53	103.14 ± 49.16
KITTI 06	157.65 ± 65.14	31.11 ± 13.27	46.04 ± 19.37	61.38 ± 26.46	76.54 ± 36.75	81.34 ± 51.04
KITTI 07	185.25 ± 66.60	36.07 ± 13.20	54.65 ± 20.61	73.07 ± 25.47	92.14 ± 34.11	107.19 ± 48.16
KITTI 09	140.51 ± 50.96	27.24 ± 10.43	41.53 ± 15.15	53.67 ± 20.64	67.07 ± 29.40	74.22 ± 39.30
KITTI 10	157.76 ± 78.49	31.96 ± 16.06	43.96 ± 22.87	62.76 ± 32.27	76.62 ± 42.57	85.77 ± 55.00

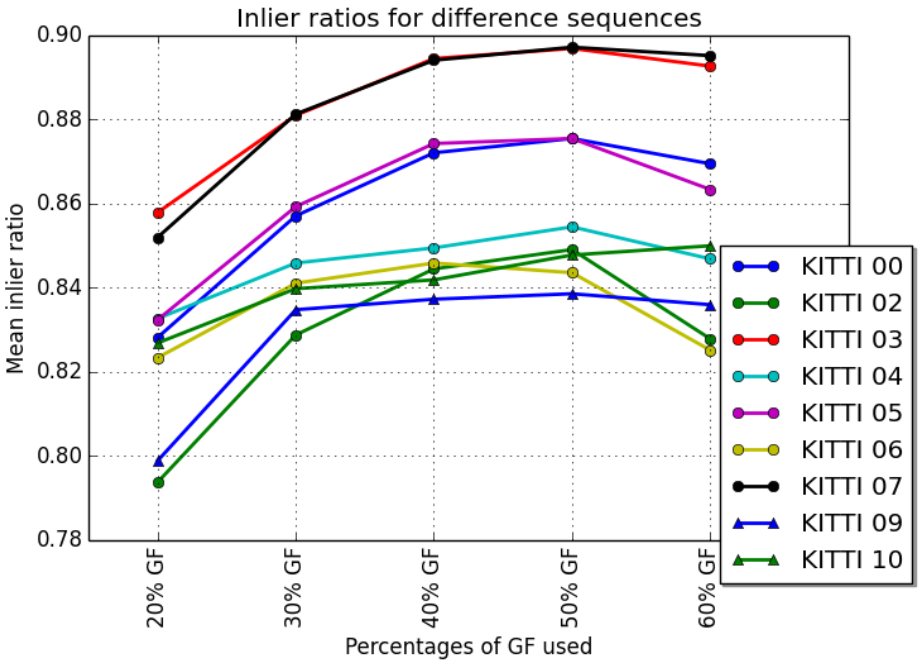


Figure 34: Inlier ratios of the GF-ORB-SLAM on original KITTI.

4.6.3.5 Experimental Results on None-Loop-Closure Sequences

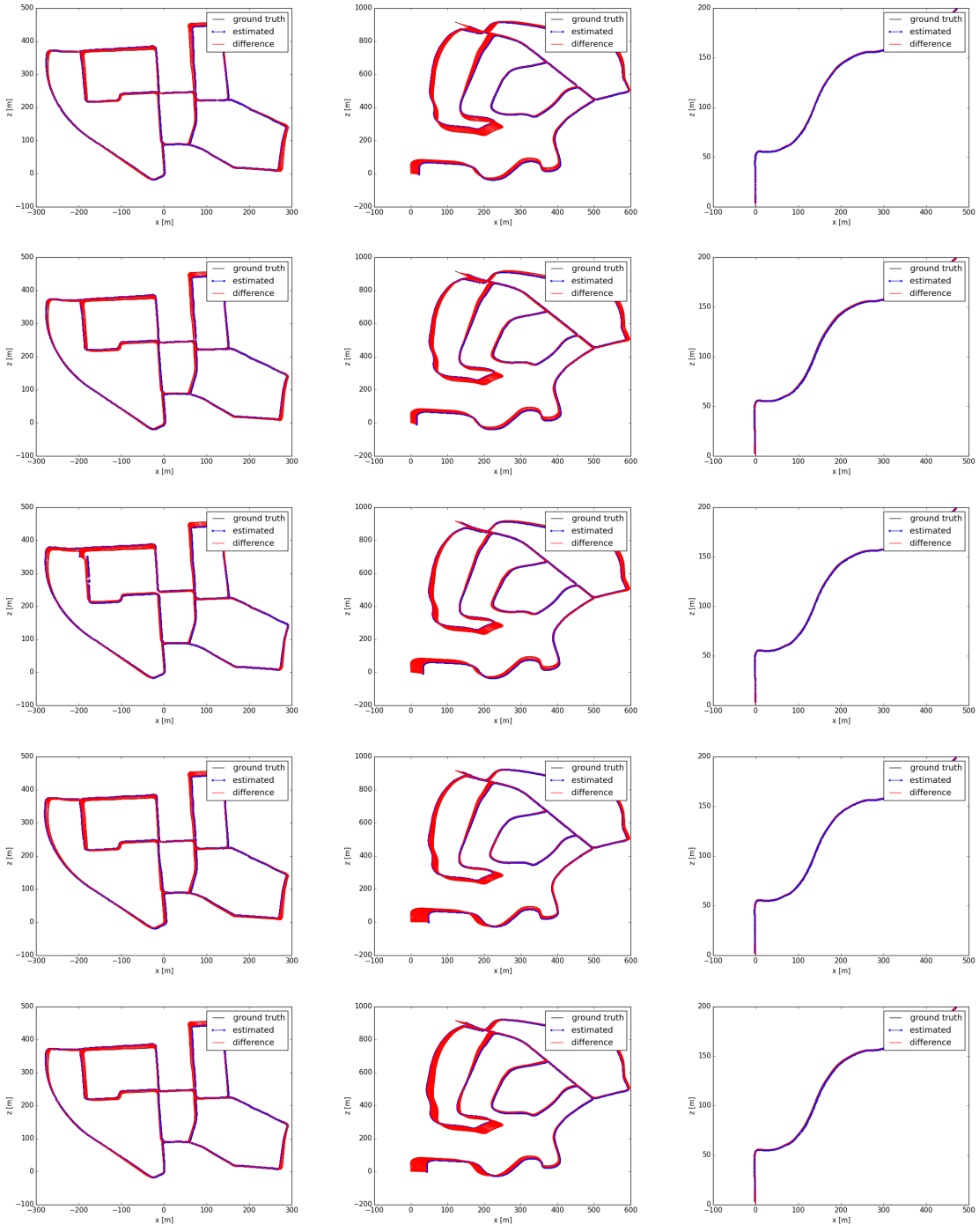


Figure 35: Results on original KITTI 00, 02, and 03: estimated trajectory, ground truth, and translation errors. Row 1: 20% GFs. Row 2: 30% GFs. Row 3: 40% GFs. Row 4: 50% GFs. Row 5: 60% GFs. Note that the X and Z axes are of different unit lengths, for better illustration of localization error.

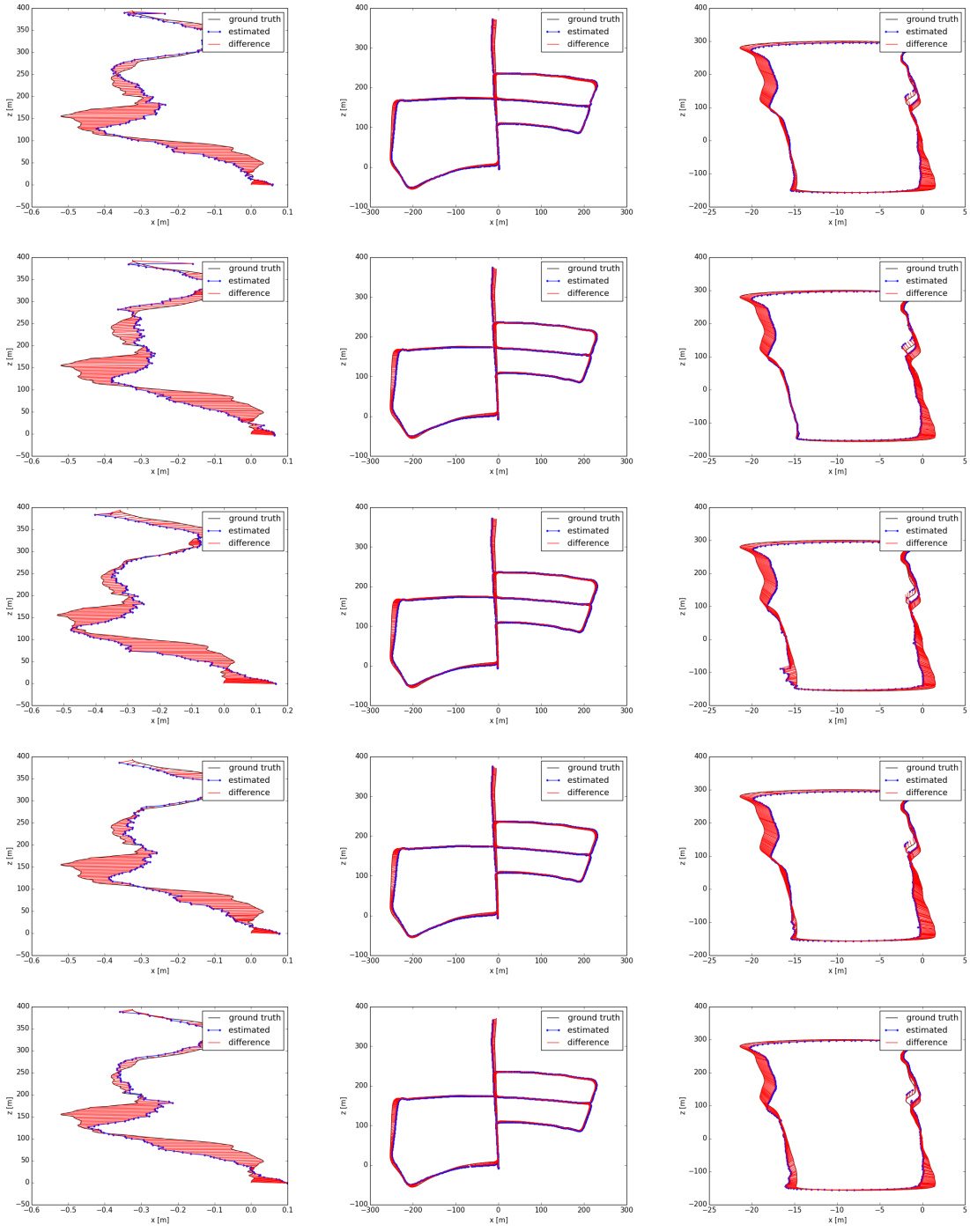


Figure 36: Results on original KITTI 04, 05, and 06: estimated trajectory, ground truth, and translation errors. Row 1: 20% GFs. Row 2: 30% GFs. Row 3: 40% GFs. Row 4: 50% GFs. Row 5: 60% GFs. Note that the X and Z axes are of different unit lengths, for better illustration of localization error.

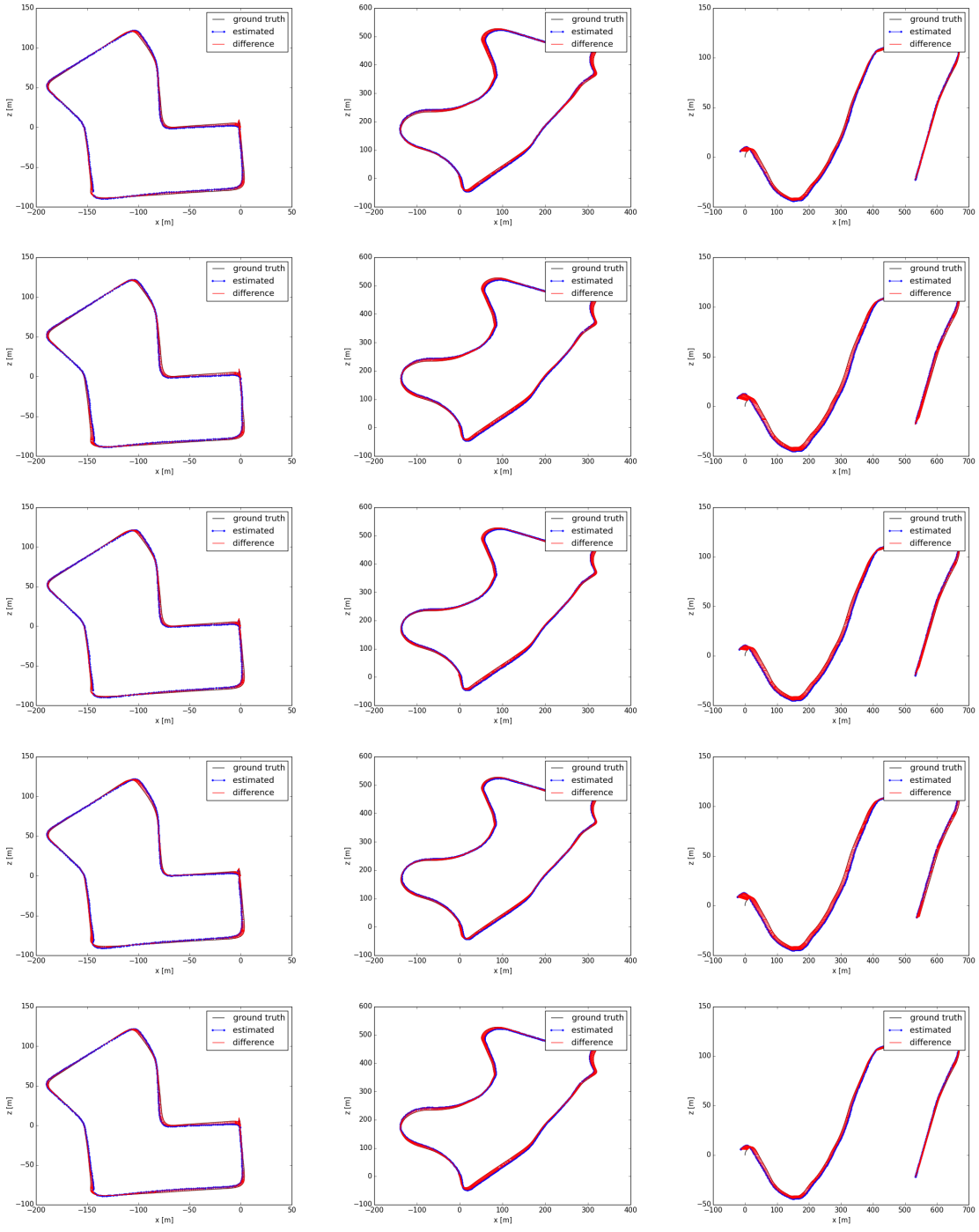
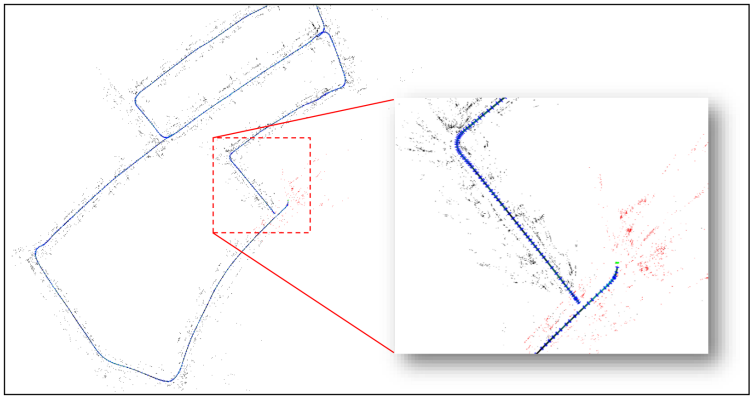
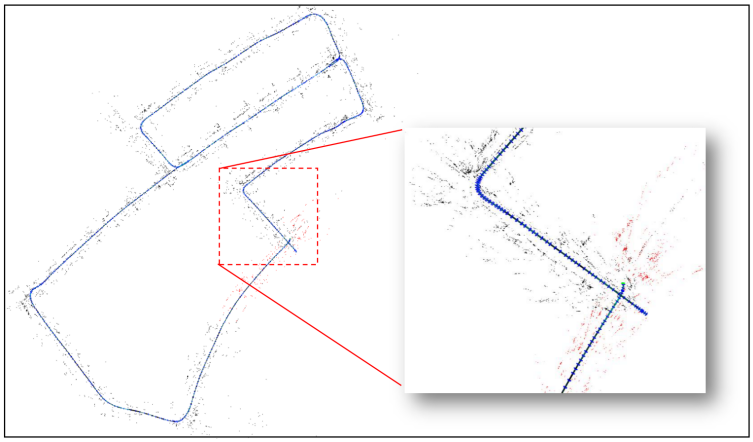


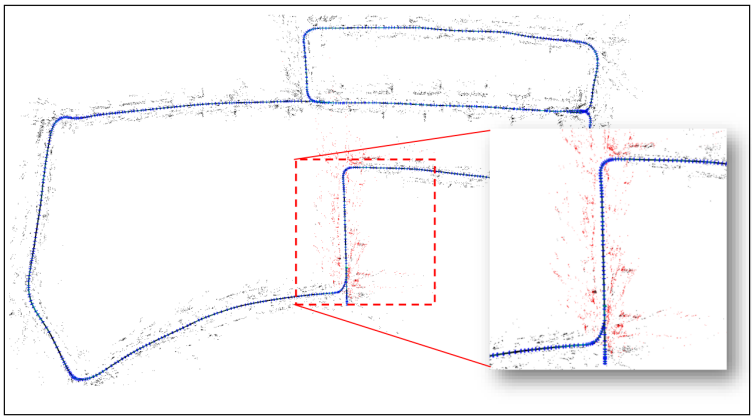
Figure 37: Results on original KITTI 07, 09, and 10: estimated trajectory, ground truth, and translation errors. Row 1: 20% GFs. Row 2: 30% GFs. Row 3: 40% GFs. Row 4: 50% GFs. Row 5: 60% GFs. Note that the X and Z axes are of different unit lengths, for better illustration of localization error.



(a) Original ORB-SLAM: Estimated trajectory of sequence 05 at 241 sec, before loop-closure.



(b) GF-ORB-SLAM (40% strongest observability): Estimated trajectory of sequence 05 at 241 sec, before loop-closure.



(c) Original ORB-SLAM: Estimated trajectory of sequence 05 at 243 sec, after loop-closure.

Figure 38: Comparison of drifts: an example from sequence 05. (a)(b): Estimated trajectory before loop-closure. (c) Estimated trajectory after loop-closure (which is almost the same for the original ORB-SLAM and GF-ORB-SLAM).

Table 5: Results on none-loop-closure sequences. Keyframe numbers and translation RMSE are reported for both original ORB-SLAM and GF-ORB-SLAM with different GF selected.

Sequence	ORB-SLAM	GF-ORB-SLAM (top 20% gf)	GF-ORB-SLAM (top 30% gf)	GF-ORB-SLAM (top 40% gf)	GF-ORB-SLAM (top 50% gf)	GF-ORB-SLAM (top 60% gf)
	RMSE (m)	RMSE (m)	RMSE (m)	RMSE (m)	RMSE (m)	RMSE (m)
02 Non-LC	93.05	74.97	76.38	70.02	76.00	78.23
06 Non-LC	26.42	25.21	25.65	25.64	24.93	24.53
07 Non-LC	14.25	12.52	13.14	13.60	13.53	13.98
08 Non-LC	43.37	36.95	34.86	34.07	37.41	37.65
09 Non-LC	44.13	36.09	36.89	36.65	39.07	42.76

Table 6: Numbers of features (mean ± std. dev.) used for pose optimization in the none-loop-closure sequences.

Sequence	ORB-SLAM	GF-ORB-SLAM (top 20% gf)	GF-ORB-SLAM (top 30% gf)	GF-ORB-SLAM (top 40% gf)	GF-ORB-SLAM (top 50% gf)	GF-ORB-SLAM (top 60% gf)
02 Non-LC	131.71 ± 43.67	25.24 ± 9.28	38.49 ± 13.34	51.67 ± 18.22	64.38 ± 24.95	67.76 ± 35.81
06 Non-LC	141.39 ± 56.43	27.47 ± 11.80	41.35 ± 17.01	41.82 ± 17.21	67.13 ± 32.09	67.82 ± 44.54
07 Non-LC	175.89 ± 62.71	34.74 ± 12.60	52.20 ± 20.94	71.76 ± 25.76	87.99 ± 32.65	101.81 ± 47.44
08 Non-LC	143.51 ± 55.86	28.01 ± 11.18	41.65 ± 16.48	56.47 ± 22.76	68.39 ± 29.35	69.35 ± 40.67
09 Non-LC	137.77 ± 46.68	26.85 ± 9.37	40.76 ± 13.85	53.38 ± 18.11	66.18 ± 24.70	71.13 ± 35.16

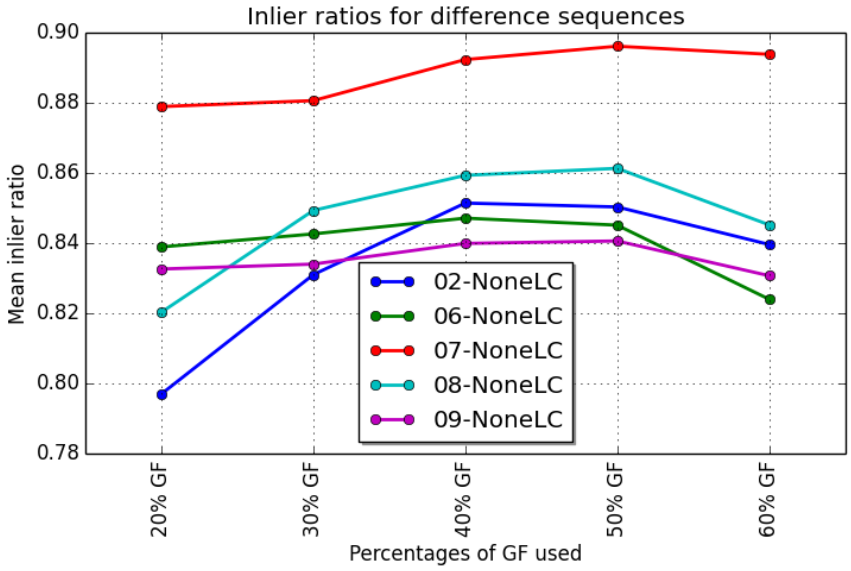


Figure 39: Inlier ratios of the GF-ORB-SLAM on none-loop-closure sequences.

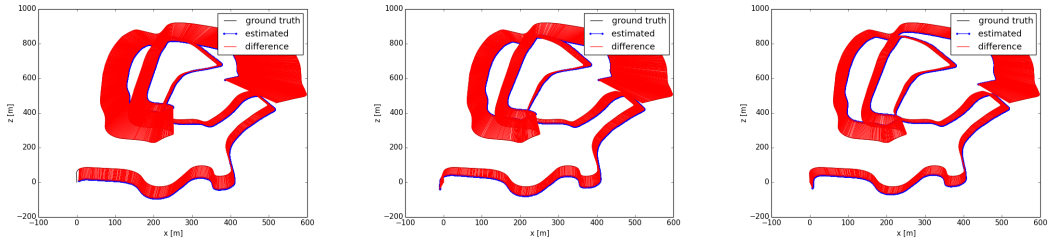
In many real-world scenarios, no loop-closures exist. Moreover, for many robotics application such as robot navigation and control, the optimized trajectory after loop-closure is not practically useful. Therefore, this experiment is designed to examine the performance of GF algorithm on long-distance sequences without loop-closures.

Similar to the previous experiment, the localization error and number of features used are summarized in Table 5 and Table 6 respectively. The GF-ORB-SLAM demonstrates better localization accuracies than the original ORB-SLAM in all sequences. The inlier ratios shown in Figure 39 depicts a similar trend as on the original KITTI sequences.

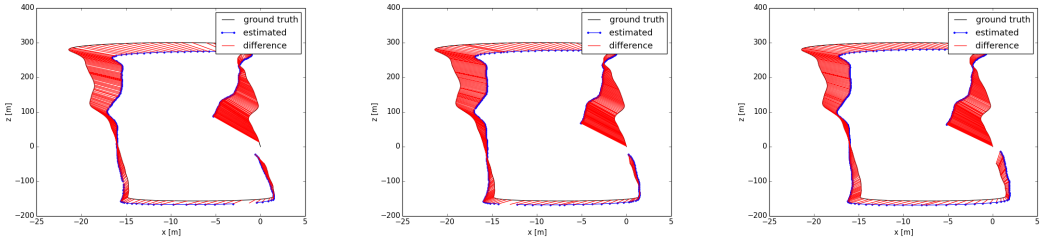
It can be observed from the final trajectories in Figure 40 that GF-ORB-SLAM has smaller drifts along the sequences, particularly in the long sequences such as 02 Non-LC, 08 Non-LC, and 09 Non-LC.

Table 7 summarizes the time used for computing the observability scores and the pose optimization with only GFs in some representative sequences. The observability score computation is implemented with simple multi-threading using C++ STL with 50 threads. As seen in Table 7, the time for observability score computation of the

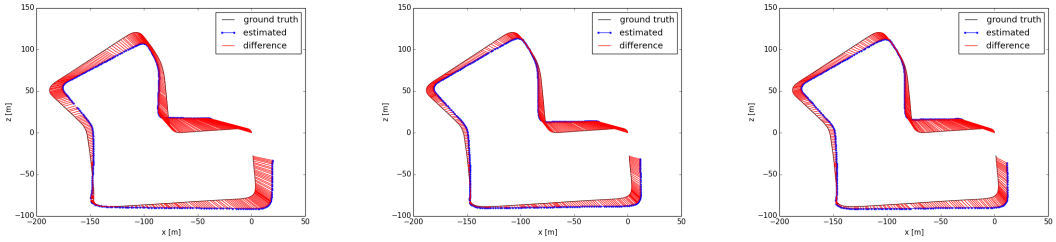
same sequence is relatively stable regardless the GF percentages used. This is because no matter how many GFs are selected, the observability scores need to be updated for all matched features, and the final runtime is therefore determined by the number of the initially matched features. The time for pose optimization, however, demonstrates a positive correlation with the number of GFs used. Overall, the implementation is fast enough for real-time VSLAM applications.



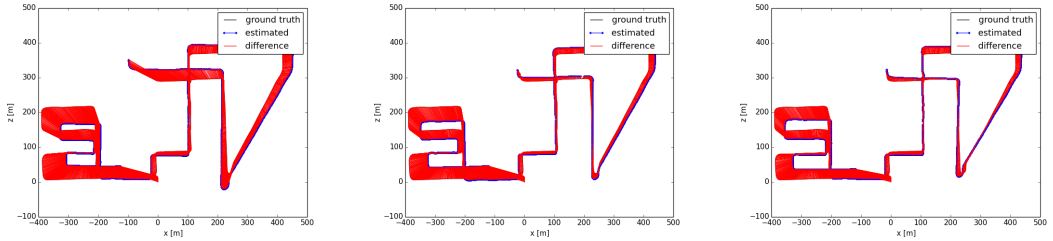
1.a Original, 02-Non-LC | 1.b GF (20%), 02-Non-LC | 1.c GF (30%), 02-Non-LC



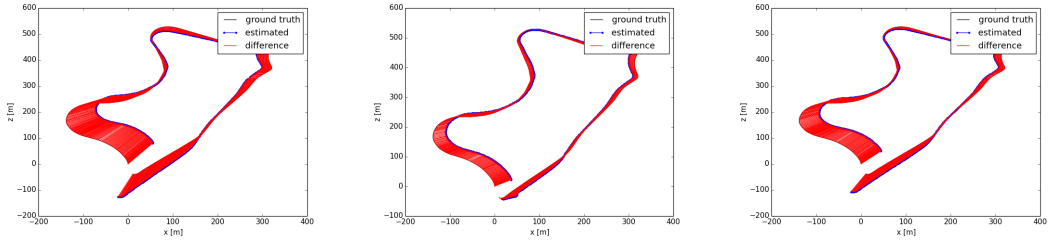
2.a Original, 06-Non-LC | 2.b GF (20%), 06-Non-LC | 2.c GF (30%), 06-Non-LC



3.a Original, 07-Non-LC | 3.b GF (20%), 07-Non-LC | 3.c GF (30%), 07-Non-LC

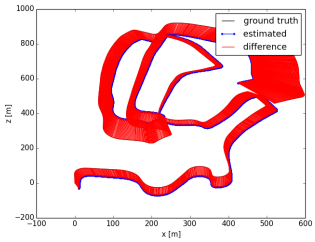


4.a Original, 08-Non-LC | 4.b GF (20%), 08-Non-LC | 4.c GF (30%), 08-Non-LC

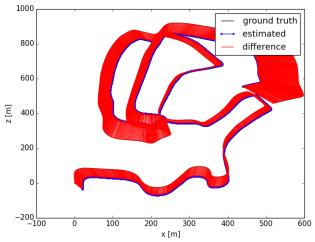


5.a Original, 09-Non-LC | 5.b GF (20%), 09-Non-LC | 5.c GF (30%), 09-Non-LC

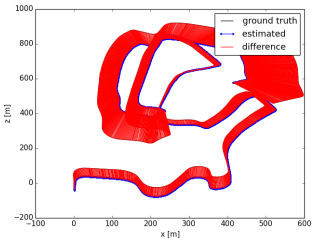
Figure 40: (Continued in Figure 41.) Estimated trajectory, ground truth, and translation errors on none-loop-closure sequences. Note that the X and Z axes are of different unit lengths, for better illustration of localization error.



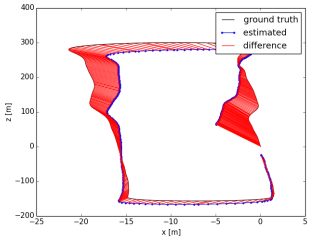
1.d GF (40%), 02-Non-LC



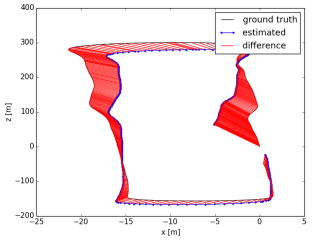
1.e GF (50%), 02-Non-LC



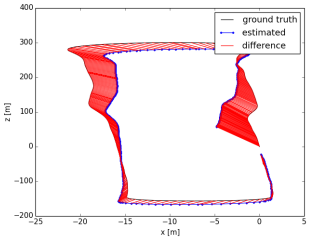
1.f GF (60%), 02-Non-LC



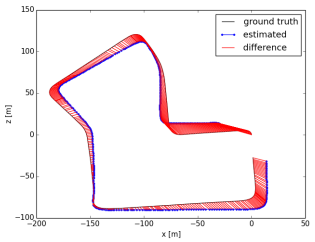
2.d GF (40%), 06-Non-LC



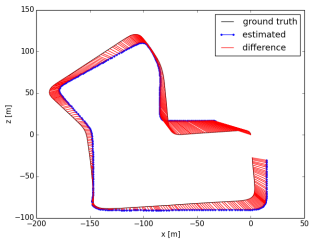
2.e GF (50%), 06-Non-LC



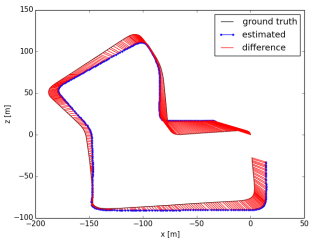
2.f GF (60%), 06-Non-LC



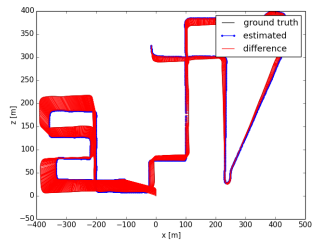
3.d GF (40%), 07-Non-LC



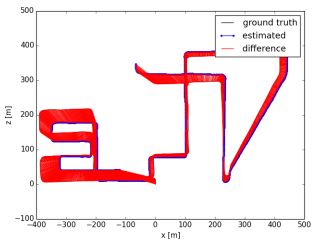
3.e GF (50%), 07-Non-LC



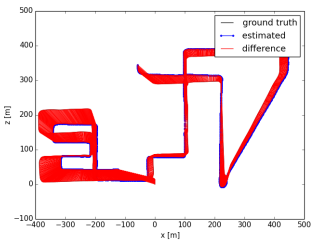
3.f GF (60%), 07-Non-LC



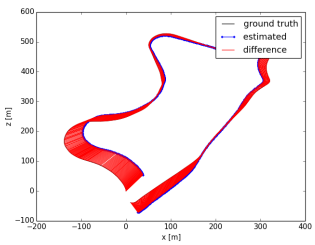
4.d GF (40%), 08-Non-LC



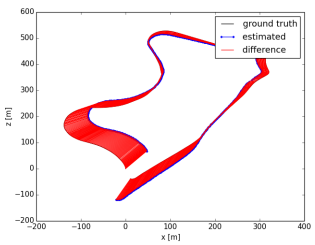
4.e GF (50%), 08-Non-LC



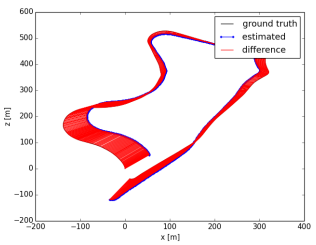
4.f GF (60%), 08-Non-LC



5.d GF (40%), 09-Non-LC



5.e GF (50%), 09-Non-LC



5.f GF (60%), 09-Non-LC

Figure 41: Continued from Figure 40.

Table 7: Timing results (mean ± std. dev). The time statistics of each experiment combination is collected from all five executions.

Sequences	GF 20%	GF 30%	GF 40%	GF 50%	GF 60%
Mean time (in milliseconds) for observability score computation					
02 Non-LC	10.5330 ± 7.1340	11.4630 ± 9.1537	10.4810 ± 6.3130	8.4712 ± 5.3057	10.5820 ± 5.5665
06 Non-LC	10.1520 ± 3.5280	10.0570 ± 4.1248	10.8090 ± 3.5996	10.6450 ± 3.7620	10.7070 ± 3.6860
07 Non-LC	11.6510 ± 4.1993	12.1110 ± 4.8791	11.7460 ± 3.9130	11.6680 ± 4.0951	11.7910 ± 4.0858
09 Non-LC	9.9296 ± 3.5732	13.1510 ± 8.4866	10.1430 ± 3.9305	11.1020 ± 4.6324	10.1220 ± 3.9207
Mean time (in milliseconds) for pose optimization w/ only GFs					
02 Non-LC	0.6561 ± 0.3380	0.6414 ± 0.3053	0.8751 ± 0.4567	0.7966 ± 0.4279	1.1049 ± 0.6866
06 Non-LC	0.4620 ± 0.2591	0.6510 ± 0.3620	0.8409 ± 0.4996	1.0289 ± 0.6515	1.0172 ± 0.7923
07 Non-LC	0.5693 ± 0.3029	0.8243 ± 0.5083	1.0719 ± 0.5711	1.3475 ± 0.7194	1.4951 ± 0.8965
09 Non-LC	0.4753 ± 0.2485	0.6642 ± 0.3049	0.8790 ± 0.4663	0.9533 ± 0.5904	1.0268 ± 0.6655

4.7 Conclusion

In this chapter, a new method is presented for selecting the features in visual SLAM process which provides the best values for SLAM estimation. The feature selection criterion of temporal observability is proposed based on the system complete observable condition #2 in Theorem 2.3.3. Efficient computation methods are further developed for temporally updating the score via incremental SVDs. A greedy algorithm for group completion, in the case of insufficient high-observability features, is also presented and justified. The Good Features method is extensively evaluated with integrations into two major types of visual SLAM systems, including filtering based VSLAM (EKF-VSLAM) and keyframe Bundle Adjustment based VSLAM. The proposed method performs competitively with respect to the state-of-the-art methods in terms of localization accuracy and data-association inlier ratios.

CHAPTER V

PCD MODELING BASED ON PLANAR PATCHES VIA SPARSITY-INDUCING OPTIMIZATION

As depicted in Figure 1, after generating the point cloud data (PCD) with VSLAM, the PCD modeling module is responsible to convert the PCD into an augmented model. The PCD modeling work in this thesis focus on the application of as-built modeling for civil infrastructures, which is further used in generating the as-built Building Information Models (BIMs).

The objective of this chapter is to develop an algorithm which takes raw point cloud data as input, and outputs a collection of **planar patch models** [111, 117]. The planar patch model description consists of the plane model parameters and the patch boundary. The planar patches found in the point cloud serve as a substitute structure for visualizing the infrastructure modeled by the point cloud, and serve as an intermediate representation in the PCD to BIM conversion pipeline. More specifically, the algorithm admits as input a civil infrastructure PCD, which is typically large scale, embedded with multiple shape components, and corrupted by noise. The algorithm is able to deal with the unstructured PCD without any topology information. In addition, the algorithm does not require advanced knowledge of the number of planar patches nor any assumptions concerning their geometry (such as alignment to specific axes). The output includes the parametric models and the boundaries of the planar patches.

5.1 Introduction

Traditional Building Information Models represent the conditions under which a facility is designed. However, the reality of the facility's construction can differ from the nominal design. Furthermore, changes in facility's conditions may happen during the life span of the facility. Hence, generating as-built BIMs, which aim to capture the as-built conditions of facilities, have been a recent topic of interest in the literature [55]. Generating as-built BIMs usually consists of two phases: (1) data collection; and (2) objects identification, extraction, and modeling. Current developments in technologies and techniques for remote spatial sensing, e.g. high density LiDAR (Deshpande 2013), image-based 3D reconstruction [86, 43] and video-based VSLAM/SfM [120, 27, 77], have largely simplified and facilitated the data collection process such that generating dense point clouds with color information of target objects is quickly becoming standard. Nevertheless, fully and simply automating the phase of objects identification, extraction and modeling remains an open problem.

The difficulty of automating "objects identification, extraction, and modeling" lies in that a raw Point Cloud Dataset (PCD) provides only Cartesian measurements and no knowledge of the elements contained therein (i.e. which parts of the PCD belong to which entities? which parts are from which geometric shapes?), nor does it immediately provide any other as-built information (i.e. changes in building conditions, etc.). To automate the process of generating as-built BIMs, recognition of infrastructure elements needs to be automated during the conversion from raw PCDs to 3D models, as shown in Fig 42.

To address the problem, this chapter presents a novel framework for automatically detecting and extracting planar patches from large-scale and noisy raw PCDs. The proposed method automatically detects planar structures, estimates the parametric plane models, and determines the boundaries of the planar patches. The first step recovers existing linear dependence relationships amongst points in the PCD

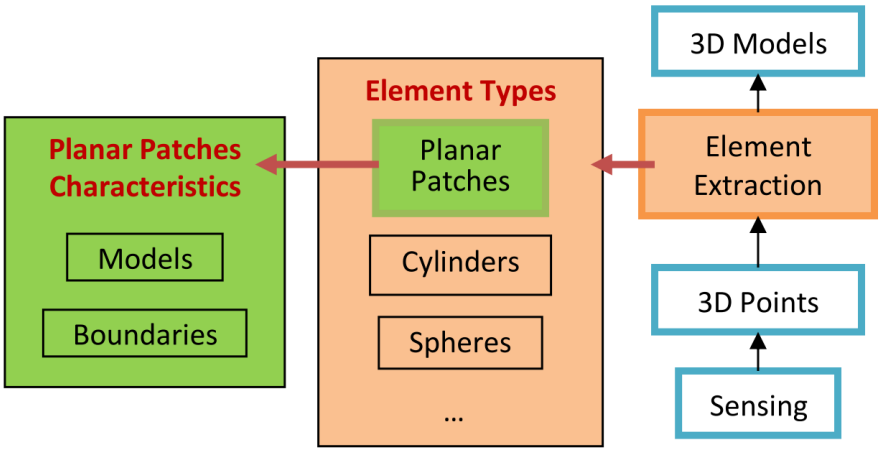


Figure 42: Role of planar patches extraction in the automatic conversion from raw PCDs to 3D models.

by solving a group-sparsity inducing optimization problem. Next, a spectral clustering procedure based on the recovered linear dependence relationships segments the PCD. Then, for each segmented group, model parameters of the extracted planes are estimated via Singular Value Decomposition (SVD) and Maximum Likelihood Estimation Sample Consensus (MLESAC). Finally, the α-shape algorithm detects the boundaries of planar structures based on a projection of the data to the planar model. The proposed approach is evaluated comprehensively by experiments on two types of PCDs from real-world infrastructures, one captured directly by laser scanners and the other reconstructed from video using structure-from-motion techniques. In order to evaluate the performance comprehensively, five evaluation metrics are proposed which measure different aspects of performance. Experimental results reveal that the proposed method outperforms the existing methods, in the sense that the method automatically and accurately extracts planar patches from large-scaled raw PCDs without any extra constraints nor user assistance.

5.2 Background

Techniques for 3D surface modeling from point cloud data can be found in computer graphics literature. Most of the algorithms are based on building meshes from point

clouds with different explicit representations of surfaces. The problem of representing surfaces was partially addressed by [38], who proposed triangular meshes. Although modeling through surfaces meshes gives an explicit description of the object's surfaces, it fails to give information about the parameters of the surfaces' geometric models and thus they are not suitable to be used in generating as-built BIMs. Different from mesh-based 3D surface reconstruction, model-based surfaces reconstruction requires the detection and extraction of embedded surface models in PCDs. Many techniques for shape models extraction are based on Random Sample Consensus (RANSAC) algorithm [85]. In civil engineering applications,[97] applied RANSAC to building roof detection. Unfortunately, fully automatic RANSAC based methods usually have very high computational complexity when applied to large-scale, complex PCDs with multiple embedded surface models. To overcome the high complexity of RANSAC, [9] presented a semi-automatic RANSAC based method requiring manual plane selection.

Another approach for extracting planar models from PCDs utilizes the Hough transform [96]. To improve the traditional Hough transform based method, [75, 55] combined it with 2D image histograms to automatically model as-built floor plans. However, the approach is not able to achieve very high accuracy because of the voxelization step used in generating the 2D histograms.Also, it requires proper alignment of the PCD with the coordinate axes.

Other planar surfaces extraction methods proposed in the recent years include the following. The plane-sweep search algorithm presented in [15], which utilizes the distribution of the 3D points along different directions to recognize the parts which contain planes and then further extract the planes within each part. The region-growing methods proposed in [48], which extracts planes by first picking a seed point and then growing the planar region from this point if criteria based on the normal deviation and mean square error are satisfied. Adan and Huber [108, 55] also presented a modified region growing method on a voxelized PCD which connects nearby points

with similar surface normals and that are well described by a planar model when aggregated. Another modified region-growing method is presented in [29], which is optimized for airborne laser scanned point clouds. This method is initialized by seed clusters in the feature space defined by local regression planes. [73] also utilized a region growing method but with the normal computed from adaptive-radius neighboring regions. Different from the above methods, other methods include: machine-learning based methods using the Expectation-Maximization (EM) algorithm [98] or hierarchical EM [100], and a geometry-based method using clustering with co-normality and co-planarity metrics [91], etc. However, the existing methods discussed above do not provide a complete and global solution to fulfill the requirements of automatically detecting planes, estimating plane models and determining the patches boundaries without requiring the number the patches as input. For example, plane-sweeping algorithms focus on plane detection, region growing methods focus on segmentation of the PCD, RANSAC based methods do detection and estimation but do not extract the boundaries and the RANSAC family are intrinsically randomized which cannot provide a complete solution.

5.3 Point Clouds Segmentation by Clustering Sparse Linear Subspaces

The proposed algorithm begins with the segmentation of a PCD according to the embedded linear subspaces of $\mathbb{R}^3$. The reason to segment PCDs as a first step is that robust parametric estimation methods, such as RANSAC, are designed for datasets with one dominant underlying model. These methods are ineffective for datasets with multiple models, i.e., when more than one model can be fit from the dataset, or datasets without dominant models. Meanwhile, randomized estimation methods like RANSAC are of high computational complexity and are impractical when the cardinality of the point-set is large. Therefore, segmenting PCDs is necessary before extracting and estimating the plane models. However, segmentation of PCD may

destroy the underlying planar structures embedded in the PCD. Hence, the segmentation step should preserve the underlying planar structures.

Segmenting PCDs while preserving underlying models is a subspace clustering problem (also known as unsupervised subspace learning). Given a point-set $\{y_i \in \mathbb{R}^D\}_{i=1}^N$ containing a union of n linear or affine subspaces in $\mathbb{R}^D$, let $\{S_l\}_{l=1}^n$ be an arrangement of the n subspaces of dimensions $\{d_l\}_{l=1}^n$. The subspaces can be expressed as:

$$S_l = \{y \in \mathbb{R}^D : y = \mu_l + U_l x\}, l = 1, \ldots, n \tag{80}$$

where $\mu \in \mathbb{R}^D$ is an arbitrary point in subspace S_l that can be chosen as $\mu_l = 0$ for linear subspaces, $U_l \in \mathbb{R}^(D \times d_l)$ is a basis for subspace S_l, and $x \in \mathbb{R}^(d_l)$ is a low-dimensional representation for point y. Subspace clustering refers to the process of finding the number of subspaces n, their dimensions $\{d_l\}_{l=1}^n$, the subspace bases $\{U_i\}_{i=1}^n$, the points $\{\mu_i\}_{i=1}^n$, and segmenting groups of points according to the subspaces. A number of subspace clustering algorithms have been proposed, broadly categorized into algebraic methods [24, 105], iterative methods [1], statistical methods [67, 79],and spectral clustering-based methods [35, 66]. In [105, 104], the author compared different subspace clustering methods, and reported that the Sparse Subspace Clustering (SSC) method proposed in [35] had the best performance in terms of misclassification error. In [89], a geometric analysis of SSC is given proving that SSC can correctly cluster data points even when subspaces intersect. SSC is based on an l_1 optimized sparse representation. In the case of PCDs, due to the geometric nature of point clouds, l_2-norm penalties also capture the linear dependence relationship, and thus the linear dependence problem is formulated as an optimization problem to minimize the combined l_1 and l_2 penalties, denoted as group-sparsity optimization.

5.3.1 Recovering PCD linear subspaces

his section covers the recovery of linear subspaces in a PCD based on sparse optimization programming. Sparse optimization programming exploits the self-expressiveness property of the data, which presumes that each point of the

$$\begin{aligned} \underset{Z}{\text{minimization}} \quad & ||Z||_1 + \beta||YZ - Y||_2 \\ \text{subject to} \quad & \sum_i z_{ij} = 1 \; j = 1, 2, 3, \ldots \\ & diag(Z) = \mathbf{0}, \; \beta > 0 \end{aligned} \tag{81}$$

PCD can be expressed by linear combinations of other points from its underlying linear subspace.

5.3.1.1 Retrieving linear dependence relationships in PCD

This section describes how to generate a sparse representation of the PCD when it contains several planar subspaces and the data has measurement error. Let $\{S_i\}_{i=1}^m$ be a union of m independent linear subspaces of dimensions $\{d_i\}_{i=1}^m$ embedded in a k dimensional space, and $\{y_i\}_{i=1}^N$ be a collection of N observations from the k dimensional space, $y_i \in \mathbb{R}^k$. If y_i belongs to subspace S_j, then y_i is a linear combination of the other data points in $\{S_i\}_{i=1}^m$. To compensate for measurement error, the basis-pursuit problem is modified to be a basis-pursuit denoising (BPDN) problem.

To collectively optimize all of the data points, form the matrix $Y = \begin{pmatrix} y_1 & y_2 & \ldots & y_N \end{pmatrix}$ and normalize the recovered coefficients. Let $Z \in \mathbb{R}^{N \times N}$ be the matrix of the sparse linear dependence coefficients whose i-th column corresponds to the sparse representation of y_i. Different columns of Z are independent. Here, the norm $||\cdot||_p$ of a matrix is the sum of the l_p vector norms of the columns.

Like the l_1-norm, the l_2-norm also captures the linear dependence relationship since points closer to each other in l_2 space are more likely to be linearly dependent.

Combining the two norms into the optimization leads to a group-sparsity optimization:

$$\begin{aligned} \underset{Z}{\text{minimization}} \quad & ||Z||_1 + \alpha||Z||_2 + \beta||YZ - Y||_2 \\ \text{subject to} \quad & \sum_i z_{ij} = 1 \; j = 1, 2, 3, \ldots \\ & diag(Z) = \mathbf{0}, \; \alpha > 0, \; \beta > 0. \end{aligned} \tag{82}$$

In the ideal case, solving the optimization program 82 recovers the sparse linear dependence coefficients corresponding to the embedded subspaces, which will be used for segmentation in the next step.

5.3.2 Subspace segmentation via spectral clustering

Once the data-driven representation for each data point is found, identification of the common underlying subspaces is the next step. This process of segmenting the linear subspaces from the recovered linear dependence coefficients involves constructing a weighted similarity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ capturing the linear dependence relationships. The N nodes in V of G correspond to the N input points; the set of edges E fully connect every two nodes v_i and v_j with the weight $w_{ij} = |z_{ij}| + |z_{ji}|$, where w_{ij} is an element of the adjacency matrix W and z_{ij} is an element of the sparse linear dependence coefficient matrix Z. For robustness to noise in the data, when building the similarity graph only the largest linear dependence coefficients should be kept for each point. Accordingly, the adjacency matrix $W_{(\gamma)}$ is expressed as $W_{(\gamma)} = |Z_{(\gamma)}| + |Z_{(\gamma)}^{\intercal}|$, where $Z_{(\gamma)}$ means the matrix with only the γ largest coefficients kept for each column with all others set to zero. Using the adjacency matrix $W_{(\gamma)}$, apply the normalized spectral clustering algorithm [74] to cluster the PCD with respect to the linear subspaces. Given the points set $Y = \left(y_1, y_2, \ldots, y_N \right) \in \mathbb{R}^k$ with adjacency matrix $W_{(\gamma)}$, define D to be the diagonal matrix whose (i, i)-element is the sum of $W_{(\gamma)}$'s i-th row.

Algorithm 4: Point cloud segmentation w.r.t. sparse linear subspaces.

Data: PCD $\mathcal{Y}$,, arranged as columns of $Y \in \mathbb{R}^{D\times N}$, which is a union of linear subspaces

Result: Partitions $\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_m$ lying in different subspaces

1 1. Solve the group-sparsity optimization program for the $N \times N$ matrix Z:

$$\begin{aligned} \underset{Z}{\text{minimization}} \quad & ||Z||_1 + \alpha||Z||_2 + \beta||YZ - Y||_2 \\ \text{subject to} \quad & \sum_i z_{ij} = 1 \; j = 1, 2, 3, \ldots \\ & diag(Z) = \mathbf{0}, \; \alpha > 0, \; \beta > 0. \end{aligned} \tag{85}$$

2 2. Use matrix Z to construct a balanced graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$. The vertices $\mathcal{V}$ are the N data points, and edges $(v_1, v_j) \in \mathcal{E}$ are with weight $w_{ij} \neq 0$. Compute the adjacency matrix

$$W_{(\gamma)} = |Z_{(\gamma)}| + |Z_{(\gamma)}^{\intercal}| \tag{86}$$

with the γ largest coefficients;

3 3. Perform spectral clustering on $\mathcal{G}$.

Construct the Laplacian

$$L = D^{-1/2} W_{(\gamma)} D^{-1/2} \tag{83}$$

Then perform eigen-decomposition on L and use the k (I choose k=3 for PCD in $\mathbb{R}^3$) largest eigenvectors $u_1, u_2, ..., u_k$ of L to form an eigenspace matrix $U = \Big(u_1|u_2|...|u_k\Big) \in \mathbb{R}^{N\times k}$ by stacking u_i in columns. Thirdly a matrix $T \in \mathbb{R}^{N\times k}$ is formed from U by normalizing each row to be of unit norm, such that

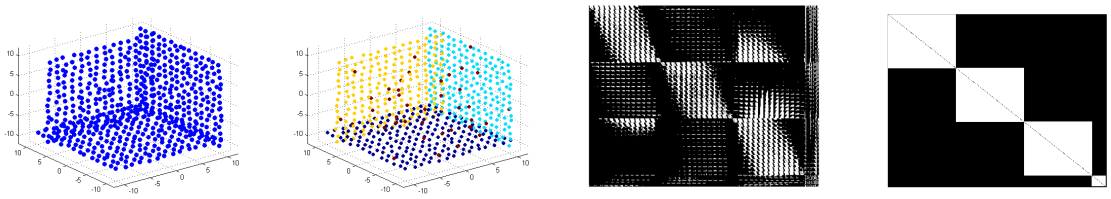
$$t_{ij} = \frac{u_{ij}}{(\sum_k u_{ik}^2)^{1/2}} \tag{84}$$

for $i = 1, 2, ..., n$. Let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of T. Lastly, perform meanshift clustering [23] on the points y_i to get a segmentation result.

5.3.3 Illustration of procedure on a synthetic PCD

To illustrate how the algorithm works, this section details the procedure for a synthetic PCD. The synthetic PCD has 628 points with 588 points from three intersecting planes (196 points per plane) and 40 randomly scattered points. Moreover, all points

are corrupted by Gaussian noise with 0.01 variance. The PCD is shown in Figure 43(a), and the ground truth for the PCD segmentation is shown in Figure 43(b) where points from the distinct embedded planes plotted with distinct colors. The PCD is processed using the proposed algorithm. First, solving the group-sparsity optimization program with $\alpha = \beta = 1$ results in the group-sparse linear dependence coefficients. The matrix containing the 628×628 linear dependence coefficients is visualized in Figure 43(c), in which the non-zero coefficients (meaning linear dependence) are plotted in white color while the zero coefficients (meaning linear independence) are plotted in black color. Each row in the coefficient matrix stands for one $\mathbb{R}^3$ point, and each 1×628-dimensional row vector consists of the linear dependence coefficients (diagonal elements of the matrix are zero). Compared to the ideal result shown in Figure 43(d), the recovered coefficients matrix is 77.87% accurate. The accuracy is computed by comparing the two coefficient matrices in Figure 43(c) and (d). The white elements are assigned to 1 and black elements to 0 for both matrices in 3(c) and 3(d), to give the matrices M_c and M_d respectively. Let $\tilde{M} = |M_c - M_d|$, then the accuracy is $\sum_{ij} \tilde{M}_{ij}/N^2$. Compared to the group-sparsity formulation, the BPDN formulation has a lower accuracy level of 68.71%. Rather than use the full matrix, the procedure indicates that the matrix with only the first $\gamma(\gamma < 628)$ largest coefficients should be used. In addition to increasing robustness to noise, the decimated matrix $W_{(\gamma)}$ reduces the computational complexity of the spectral clustering step. Figure 43(e) and Figure 43(f) show $W_{(\gamma)}$ with $\gamma = 10$ and $\gamma = 20$, respectively (visualized by displaying non-zero value elements as white and zero value elements as black). From Figure 43(e) and Figure 43(f), it can be observed that the larger coefficients are nearer to the diagonal elements, while the smaller coefficients are further from the diagonal elements. There is no universal criterion for how many coefficients should be used in constructing the adjacency matrix, but the observations are: if less coefficients are used, then less linear dependence relationships are captured but the

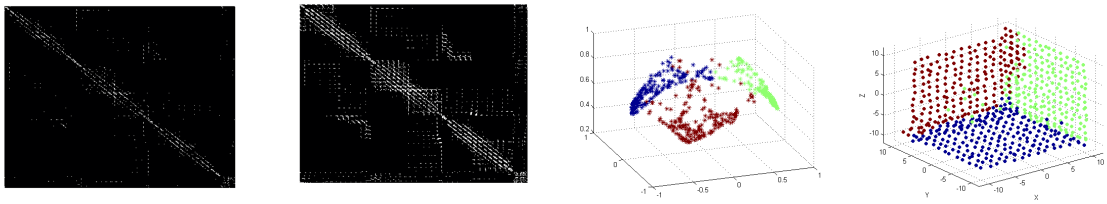


(a) Original PCD

(b) Ground truth for PCD segmentation

(c) Recovered sparse linear dependence coefficients (the Z matrix) (black: zero values; white: non-zero values)

(d) Ideal result of linear dependence coefficients (black: linearly independent; white: linearly dependent)



(e) Linear dependence adjacency matrix (with 10 largest coefficients for each point)

(f) Linear dependence adjacency matrix (with 20 largest coefficients for each point)

(g) Clustering result in eigenspace in spectral clustering step

(h) Retrieved segmentation result in $\mathbb{R}^3$

Figure 43: Illustration of linear subspace clustering on a synthetic PCD.

algorithm is more robust to noise and has lower computational complexity. For the following steps of the experiment, γ is set to be 10, because this is small enough to generate a sparse adjacency matrix but large enough to capture the linear dependence bases. Further discussion is included in Section 5.6.

By following the remaining steps, an eigenspace point-set can be obtained, which lies in a simplex structure, as shown in Figure 43(g). Cluster the eigenspace point-set using mean-shift. The clustering result is shown in Figure 43(g) with different clusters plotted in different colors. The PCD segmentation step is finished by assigning the cluster memberships of each point in the eigenspace to the original $\mathbb{R}^3$ points. The final segmentation result is plotted in Figure 43(h). The segmentation step achieves 89.46% accuracy for the points from the underlying planes. Most of the misclassifications occur around the intersecting areas of the planes. The classification accuracy will be further improved in the subsequent steps.

5.4 Plane Detection and Model Estiamtion via Maximum Likelihood Sample Consensus

The previous step gives a segmentation of the PCD but not the plane model, with some data points potentially misclassified. After the segmentation step, ideally within each segmented group there is at most one linear subspace, meaning that there is one or zero planes in each group. A robust detection and estimation step is needed to determine whether each segmented group arises from a planar subspace, and if so, to estimate the parametric planar model. Moreover, after model estimation, all of the plane models are used to correct potential false segmentations.

Because the data is noisy and the segmentation result from the previous step may not be accurate, the detection and estimation algorithm in this step should be robust to both noise and false segmentation, which is traditionally done with RANSAC. Traditional RANSAC verifies the estimated models by thresholding the number of inliers. However, in the case of extracting models from PCDs, the cardinality of each segmented point-set varies, meaning that a predefined threshold is not suitable for each group. Compared to RANSAC, the Maximum Likelihood Sample Consensus (MLESAC) algorithm [99] adopts the same sampling strategy as RANSAC but chooses the solution by minimizing the probabilistic loss rather than the number of inliers. Minimizing probabilistic loss renders the model verification threshold value invariant to the cardinality of the models data set. MLESAC is reported to be of higher accuracy and robustness than RANSAC [19]. Therefore, MLESAC is more suitable for model extraction from PCDs.

5.4.1 Planes detection and estimation from PCDs

MLESAC first randomly samples a subset of points with the minimum cardinality $\Gamma_{\min}$ needed for model estimation, then the sampled subset is used to fit a parametric model. For plane estimation, $\Gamma_{\min} = 3$. Denote the three points sampled by

$\{p_i\}_{i=1,2,3} \in \mathbb{R}^{3\times 1}$, then the plane parameters are obtained from the following steps. First express $\{p_i\}$ in homogeneous form as $q_i = \begin{pmatrix} p_i^\intercal & 1 \end{pmatrix}^\intercal$, then form the matrix

$$M = \begin{pmatrix} q_1 & q_2 & q_3 \end{pmatrix} \in \mathbb{R}^{4\times 3} \tag{87}$$

Perform singular value decomposition of M, which estimates both the normal and the offset of the plane:

$$USV^\intercal = M \tag{88}$$

The hypothesized parameter vector $\begin{pmatrix} \alpha & \beta & \gamma & -1 \end{pmatrix}^\intercal$ of the plane is obtained from the fourth column of V with normalization by dividing the additive inverse of the last element.

MLESAC evaluates the fitness of the hypothesis using a probabilistic model for the errors arising from inliers and outliers. The inlier error is modeled as unbiased Gaussian distribution while the outlier error as uniform distribution. Hence the probability of the error given the estimated model is:

$$P(e|\text{Model}) = \gamma \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{e^2}{2\sigma^2}\right) + \frac{1-\gamma}{\nu} \tag{89}$$

where e is the inlier error, γ is the prior probability of being an inlier (the ratio of inlier), ν is the size of available error space, σ is the standard deviation of Gaussian noise. If $P(e|\text{Model})$ is larger than the threshold, then the model will be re-estimated using only the inliers and MLESAC terminates. Otherwise, repeat the process with another random sample set, compute the loss, and determine if a further iteration is needed. The maximum number of iterations to perform is

$$T = \frac{\log \alpha}{\log(1-\gamma^{\Gamma_{\min}})} \tag{90}$$

where α is the estimated failure probability of picking up inlier samples at least once. The MLESAC loop terminates when the required iterations have been finished.

The plane model estimation step is summarized in Algorithm 5. The estimation results of this step correct erroneously segmented points from the previous step by

Algorithm 5: Plane models extraction from PCD via MLESAC

Data: One segment $\mathcal{Y}$ of PCD ;
Result: Estimated plane parametric models with the inlier set, or failure to find a fit for the model

while *iterations* $< \frac{\log \alpha}{\log(1-\gamma^{\Gamma_{\min}})}$ **do**
    1. Randomly sample 3 points $\{p_i\}_{i=1,2,3}$ with the corresponding homogeneous coordinates $\{q_i\}$. Form matrix

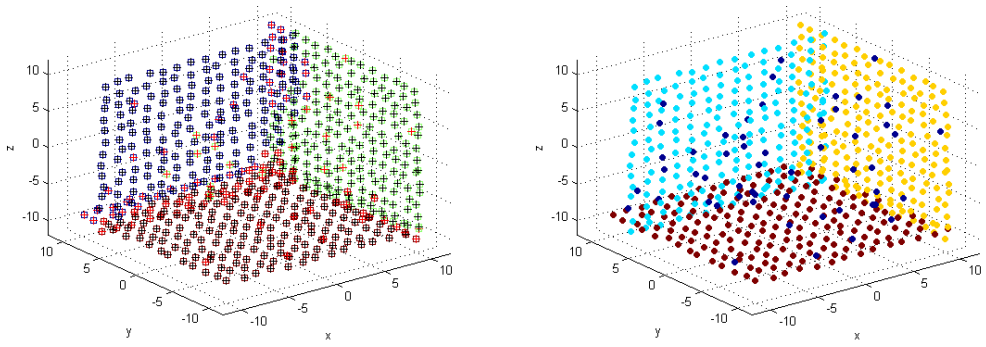
$$M = \big(q_1 \mid q_2 \mid q_3\big); \tag{91}$$

    2. Perform Singular Value Decomposition $USV^{\intercal} = M$, obtain the parameters vector for the plane model as from the last column of V;
    3. Determine the inlier and outlier sets, and the corresponding errors;
    4. Compute the loss of the model with Equation 89;
    if $P(e|Model) > Threshold$ **then**
        Re-fit the model with inlier set;
if *No verified model extracted* **then**
    return *failure* to find a fit for the model
else
    return the estimated parametric model with the smallest $P(e|\text{Model})$.

relabeling each point to the model with the minimum Euclidean distance between the point and the model. The estimated models are further merged together if the parametric models are close and the supports are adjacent. The estimated models are further merged together if the parametric models are closed and the supports are adjacent.

5.4.2 Illustration of Algorithm 5 on the synthetic PCD

Algorithm 5 is illustrated using the same synthetic PCD discussed in Sec 5.3.3. In the plane models extraction step, points from each group are processed using Algorithm 5. For MLESAC, the threshold for the probability P(e — Model) is set to be 0.5, which is optimized empirically. The inlier set and outlier set detected for each segmented group are plotted in Figure 44(a), in which black + signs stand for inliers and red + signs stand for outliers. The models extracted for the three groups are reported



(a) Inlier (black +) and outlier (red +) set obtained in MLESAC

(b) PCD segmentation after MLESAC re-correction

Figure 44: Illustration of Algorithm 5 on the synthetic PCD.

in Table 1 with the absolute errors computed by comparing to the ground truth. It can be concluded that the planar models extracted are of high accuracy. These extracted models are further used as feedback to improve the segmentation results by assigning all the points to the model to which the perpendicular Euclidean distance is the smallest among all the models and smaller than a predefined threshold (in this experiment the threshold is 0.1), and the points whose perpendicular distances are larger than the threshold are labeled as noise. The segmentation result after this assignment is illustrated in Figure 44(b), which has 93.79% accuracy for the whole PCD.

5.5 Determine Plane Boundaries via QR Decomposition based Projected α-Shape

After the previous two steps, the segmentation of the points from different planes and the corresponding planar models have been obtained. Generating the final representations for the planar patches requires identifying the boundary each extracted planar patch. The challenges of this step are: (1) the boundary point-sets may not be convex; instead they may be concave or even with openings inside the outer boundary ; (2) points in each point-set may not be uniformly distributed; and (3) the points are

corrupted with noise.

5.5.1 Maximum projected variance α-shape algorithm

Given a PCD point-set $\mathcal{Y} \in \mathbb{R}^{n\times 3}$ and its estimated normal $\boldsymbol{n} \in \mathbb{R}^{3\times 1}$, first a 3-by-3 matrix $A = \begin{pmatrix} \boldsymbol{n} \mid v_1 \mid v_2 \end{pmatrix}$ is formed, where $v_1, v_2 \in \mathbb{R}^{3\times 1}$ are random column vectors generated from the point-set. Then QR decomposition of A is:

$$A = QR \tag{92}$$

where $Q = \begin{pmatrix} Q_1 \mid Q_2 \mid Q_3 \end{pmatrix} \in \mathbb{R}^{3\times 3}$ is an orthogonal matrix. The natural coordinate vectors are given by the three column vectors of $Q_1, Q_2, Q_3 \in \mathbb{R}^{3\times 1}$. In this work, the Z-axis in the natural coordinate frame is defined with Q_1 (the plane normal), X-axis with Q_2 and Y-axis with Q_3. Then project $\mathcal{Y}$ onto the natural coordinates by

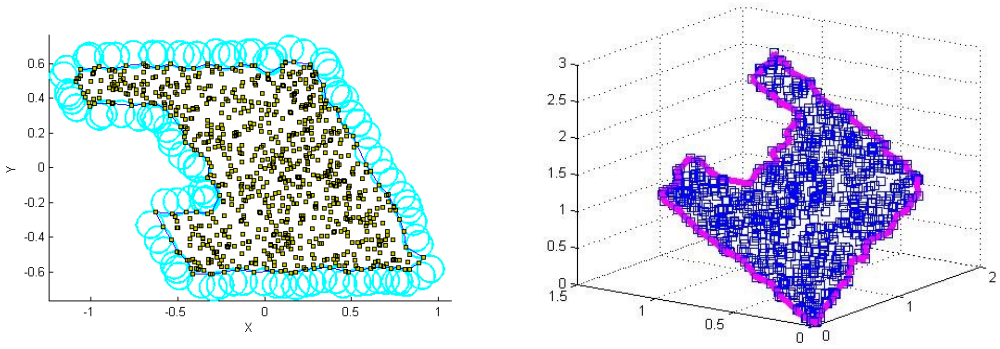
$$\hat{\mathcal{Y}}' = \begin{pmatrix} \mathbf{I}_{2\times 2} & \mathbf{0}_{2\times 1} \end{pmatrix} \cdot \begin{pmatrix} Q_2 \mid Q_3 \mid Q_1 \end{pmatrix}^{-1} \mathcal{Y}^{\mathsf{T}} \tag{93}$$

where the factor $\begin{pmatrix} \mathbf{I}_{2\times 2} & \mathbf{0}_{2\times 1} \end{pmatrix}$ projects 3D points to 2D points. The arrangement of columns in $\begin{pmatrix} Q_2 \mid Q_3 \mid Q_1 \end{pmatrix}$ performs a $\mathcal{SO}(3)$ transformation aligning the normal vector of plane in the original frame to the Z-axis in the projected frame. The projected point-set $\hat{\mathcal{Y}} \in \mathbb{R}^{n\times 2}$ is obtained by $\hat{\mathcal{Y}} = \hat{\mathcal{Y}}'^{\mathsf{T}}$.

The α-shape algorithm is then performed on $\hat{\mathcal{Y}}$. Since the boundary detected depends on the radius of the circles (or α value), here I set the α value as 3 times of the average single-link point-point distance, which is a conclusion assessed experimentally. The boundary detected using this α value is shown in Figure 45(a). The 2D boundaries are then projected back to the 3D space, shown in Figure 45(b). This outside concave boundary is detected without any additional boundaries for the openings inside the point set.

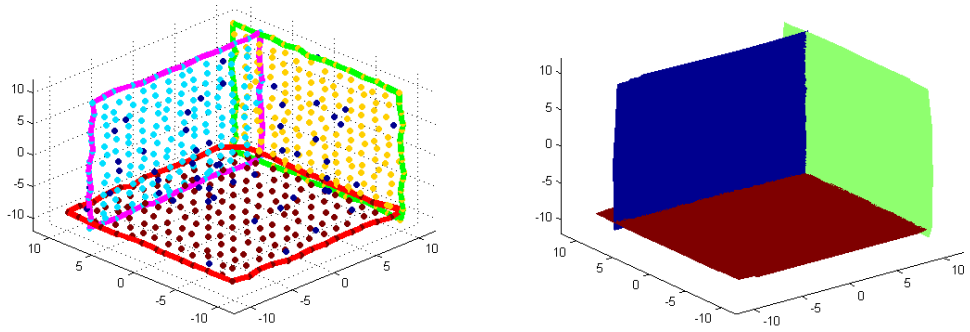
5.5.2 Illustration of Algorithm 6 on a synthetic PCD

As the final step, the boundaries of each extracted planes are detected by performing Algorithm 3 on each segmented group. The detected boundaries are plotted out in



(a) Boundary extracted on the projected 2D point set (b) Boundary back-projected to 3D space

Figure 45: Boundaries found using QR decomposition based projected α-Shape algorithm (Radius=$3D_{pp}$).



(a) Detected boundaries of extracted planes (b) Final planar patches representation

Figure 46: Illustration of Algorithm 6 on synthetic PCD.

Figure 46(a). Finally, the planar patches representation is generated as shown in Figure 46(b).

5.6 Evaluation

5.6.1 Evaluation metrics

To evaluate the complete plane identification and extraction algorithm, five different evaluation metrics will be computed. These metrics evaluate different aspects of the algorithm performance to give a comprehensive understanding of how well models detected and extracted. These metrics and their purpose are as follows: Root

Algorithm 6: Plane boundary detection via maximum projected variance α-shape algorithm.

Data: A PCD point-set $\mathcal{Y} \in \mathbb{R}^{n\times 3}$ on a detected plane and the estimated plane normal $\boldsymbol{n} \in \mathbb{R}^{3\times 1}$

Result: 3D boundary point-set P_S

1. 1. Form a matrix $A = \big(\boldsymbol{n} \mid v_1 \mid v_2\big) \in \mathbb{R}^{3\times 3}$, where v_1, v_2 are random column vectors;
2. 2. Perform QR decomposition on $\boldsymbol{A}$: $A = QR = \big(Q_1 \mid Q_2 \mid Q_3\big) \cdot R$;
3. 3.Define the natural coordinate frame with Q_2, Q_3, Q_1 and project $\mathcal{Y}$ onto the frame by:

$$\hat{\mathcal{Y}}' = \big(\mathbf{I}_{2\times 2} \quad \mathbf{0}_{2\times 1}\big) \cdot \big(Q_2 \mid Q_3 \mid Q_1\big)^{-1} \cdot \mathcal{Y}^{\intercal} \tag{94}$$

$$\hat{\mathcal{Y}} = \hat{\mathcal{Y}}'^{\intercal} \tag{95}$$

where $\hat{\mathcal{Y}} \in \mathbb{R}^{n\times 2}$ is the projected point-set;

4. 4. Get α-shape boundary of $\hat{\mathcal{Y}}$;
5. 5. Determining the 3D plane boundary point-set $\boldsymbol{P}_S$ by retrieving the membership of the 2D boundary point-set.

Mean Square error measures the model fitting accuracy; Normal Deviation measures the orientation accuracy; Unit Volume error measures both the orientation and the translation accuracy; Detection Percentage measures what percentage of the total number of patches were detected; Oversegmentation Factor gives the factor by which the planar models overrepresent the ground-truth models.

5.6.1.1 Root mean square error (RMSE)

The RMSE measures the consistency of the model to the data. For every point $\boldsymbol{X}_i \in \mathbb{R}^{3\times 1}$ that belongs to an extracted plane with the model $\hat{\boldsymbol{n}}^{\intercal} \cdot \boldsymbol{X} - d = 0$, where $\boldsymbol{n} \in \mathbb{R}^{3\times 1}$ is the normal of the plane with unit length and d is the offset of the plane. The point-plane distance is then measured by

$$distance = |\hat{\boldsymbol{n}}^{\intercal} \cdot \boldsymbol{X} - d| \tag{96}$$

The root mean square error (RMSE) for each extracted plane is defined as

$$RMSE = \sqrt{\frac{1}{I}|\hat{\boldsymbol{n}}^{\intercal} \cdot \boldsymbol{X} - d|} \tag{97}$$

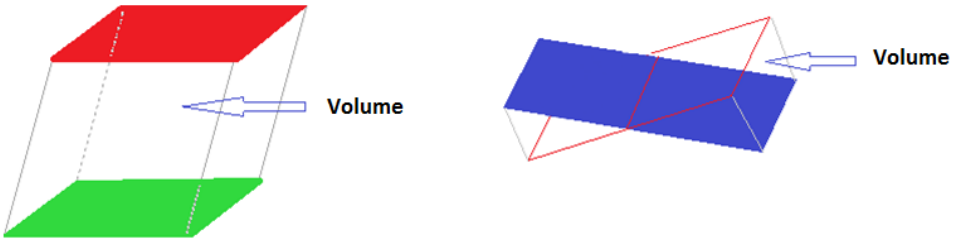


Figure 47: Volume between two planar patches.

where $i = 1, 2, \ldots, I$ is the index of the points that associated to the plane.

5.6.1.2 Normal deviations

The normal deviation measures the accuracy of orientation between the extracted plane compared to the ground-truth plane. Given the normal vector $\hat{\boldsymbol{n}}$ of an estimated plane and the corresponding ground-truth normal vector $\boldsymbol{n}$, the normal deviation is:

$$\text{Normal Deviation} = \text{acos}(\hat{\boldsymbol{n}} \cdot \boldsymbol{n}) \tag{98}$$

5.6.1.3 Unit volume error

Besides orientation accuracy, the localization accuracy of the plane is important. Accordingly, here I define an evaluation metric which captures both the orientation and translation accuracy, the unit volume error. It is the volume generated from the estimated patch and the ground-truth patch divided by the area of the ground-truth patch. The volume error is illustrated in Figure 47. The volume is defined in the direction orthogonal to the ground-truth patch. In the calculation of the volume, absolute distances are used instead of the signed distances. The units of this score are $m^3/m^2 = m$.

$$\text{UnitVolumeError} = \frac{\text{total volume error}}{\text{area of ground truth patch}} \tag{99}$$

5.6.1.4 Detection percentage

This metric evaluates how completely the algorithm is able to detect all existing planar patches in the PCD. It is the percentage of the number of extracted patches, relative to the quantity of patches in the ground truth model. The number of extracted patches is defined as the number of patches in the ground-truth data that are correctly found by the algorithm. For example, if there are 20 planar patches in the whole ground-truth PCD, and the algorithm is able to extract 12 out of the 20 planar patches, then the Detection Percentage is 60%. Moreover, if patch A in the ground-truth data is found but broken into two patches by the algorithm, patch A is counted as one patch extracted; or if only a part of patch A is found by the algorithm, it is still counted as one extracted patch. The ideal value is 100%.

5.6.1.5 Oversegmentation factor

This metric aims to evaluate for a detected ground-truth planar patch, how well the procedure models the patch. For each ground-truth planar patch, the number of the corresponding extracted planes is counted. Then the oversegmentation factor is defined to be the quantity of extracted plane models divided by the quantity of unique ground-truth models associated to them. For example, suppose that the procedure detected six planar patches, two belonging to one ground truth model, and four belonging to a second ground truth model. Then the oversegmentation factor is $(2+4)/2 = 3$. Combining Detection Factor, the ideal case is that the oversegmentation factor equals to 1 and the detection percentage equals to 100%. In this case, there is a one-to-one mapping from the estimated patches to the ground-truth patches.

5.6.2 Evaluation results on the synthetic PCD

Using the evaluation metrics, the proposed algorithm is compared to three state-of-art algorithms. The three baseline methods are the Hough transform based algorithm of [75], the plane-sweeping algorithm of [15], and the region-growing based method

Table 8: Evaluation results on the synthetic PCD.

Methods	**our method**	**[75]**	**[15]**	**[108]**
RMSE (cm)	2.17 ±0.61	6.91±1.57	7.95±2.12	**1.20±1.46**
Unit Volume Err. (m)	**0.13±0.02**	0.32±0.08	0.37±0.11	0.13±0.25
Normal Deviations (degree)	0.14±0.12	**0**	**0**	17.98±21.47
Detection Percentage	**100%**	**100%**	**100%**	**100%**
Oversegmentation Factor	**1**	**1**	**1**	**1**

of [108]. The final planar patch representations of these methods are different. For method [75] and method [15] the final results are in solid planar patches, while for method [108] the final results are segmentations of points. The results of these three methods on the synthetic PCD are shown in Figure 8 respectively. The evaluation results are shown in Table 8, which are presented in the format of "mean ± standard deviation", because there are multiple planes in the dataset and the statistics are computed over the planes. This simple, synthetic PCD example does not fully reflect real-world PCDs. For example, the real-world dataset may not be oriented precisely, which would introduce errors when using methods in [108] and [15].

In Table 8, the methods [75] and [15] have RMSE¿0 but the normal deviations are zeros because the extracted planes are of an offset compared to the ground-truth planes but they are also parallel to the ground-truth planes (and this is why the normal deviations are exactly zeros). Note that the normal deviations of[75] and [15] can be zeros because in this synthetic example all the planes are placed perfectly parallel to the coordinate planes. These two methods rely on the projection onto coordinate planes or plane-sweeping along the direction from rotational sweeping. Thus, they have zero normal deviations in this synthetic example. However, in reality, the planes in point-clouds may not perfectly align with the coordinate planes or the extracted direction. Therefore in the real-world PCDs example, these two methods do not have zero normal deviations. It is worthy to note that none of these compared methods is able to give estimated plane models or the detailed boundaries. Especially, the region-growing based methods are only able to give segmentation of point clouds that

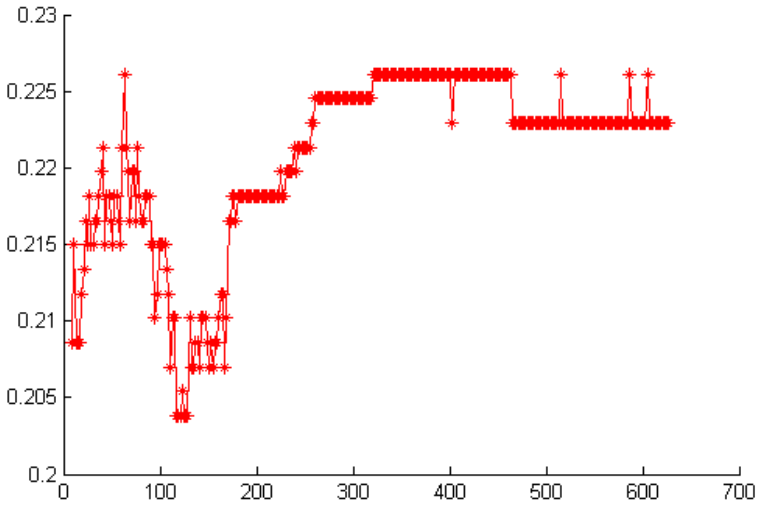


Figure 48: Misclassification rate w.r.t. different numbers of linear elements in constructing the similarity graph.

ideally belong to some planes.

I end the discussion for the synthetic PCD experiment by investigating the influence of the number (denoted as γ) of linear dependence coefficients used in constructing the similarity graph. The misclassification rates of the PCD segmentation step w.r.t. γ from 2 to 627 are plotted in Fig 48. As it can be observed, the misclassification rate varies between 20.38% and 22.61%. Given that the model fitting step corrects this error, the change in performance as a function of γ is not significant enough to warrant using large values of the parameter γ. Thus, it is recommended to use a relatively small γ, one which would correspond to selecting a small percentage of the total dataset.

5.6.3 Evaluation results on real-world PCD from VSLAM/SfM

5.6.3.1 Point cloud dataset

In this experiment, the PCD of a real building reconstructed from VSLAM/SfM is used. A frame from the video is shown in Figure 49. Due to the physical constraints of the environment, only three faces of the building were captured. Moreover, there are some occlusions in the scene, e.g., trees, decorations, etc. The reconstructed raw

Figure 49: A sample image used to reconstruct a building.

PCD is displayed in Figure 50. The point cloud consists of 1,681,634 points, with relatively large measurement uncertainty.

5.6.3.2 Experimental results

The building PCD is processed using the proposed algorithm, with parameter settings as listed in Table 9. To lower the computational complexity, the PCD is first partitioned into $8 \times 8 \times 4 = 256$ parts. The final result of the experiment, after merging the partition results, is shown in Figure 51(a),(b). The algorithm extracts 16 planar patches from the PCD.

The raw PCD is also plotted in Figure 51(a), (b) in magenta to provide intuitive comparison between the raw point cloud and the extracted patches. Note that some open parts (for instance, the intersecting part between two roof planes in Figure 51(b)) exist because the point cloud itself does not capture the corresponding part due to some occlusions. From the experiment it can be observed that the extracted patches fit with the point cloud very well.

5.6.3.3 Comparison to baseline methods

For [75] method, since proper orientation is required, the orientation of the PCD is corrected to align the walls to coordinate axes before conducting the experiment. The

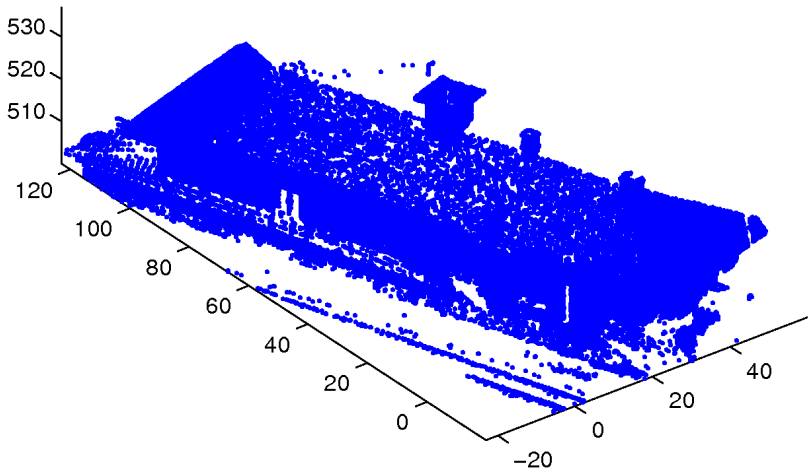


Figure 50: Raw PCD representation of a building.

parameters of the compared methods are as follows. For [75], I set the grid size of the 2D histogram as $0.1m \times 0.1m$. For [15] method, the number of the bins used to generate the histogram of point numbers for sweeping is 200 and the threshold to define a plane in the histogram is set to be half of the maximum value in the histogram. For [108] method, the PCD is voxelized into $2cm \times 2cm \times 2cm$ grids. The number of neighbor points for normal estimation is 50; the threshold of maximum angle between normal vectors is 2 degrees; the curvature threshold to guarantee points are well-described by plane models is set as 1. All of these parameter configurations are optimized empirically.

Ground-truth data of the building is collected using a professional total station (i.e., SOKKIA 30R). Points are measured for each facet of the infrastructures, especially the points that define the boundary of each facet of the infrastructure. The PCD is obtained by merging the point-sets from different scan domains using the software of the total station. After collecting the PCD, the measured points belonging to each specific facet are selected manually and used to generate the ground-truth data for each facet. The planes measured as ground-truth are shown in Figure 13. These

Table 9: Parameter configurations for the building PCD experiment

Parameters	**Values**
optimization parameter α	1
optimization parameter β	1
number of coefficients used in adjacency matrix	10
MLESAC verification probability threshold	0.5
MLESAC,false alarm rate (probability a good minimal sample set never picked)	1e-3
MLESAC assumed noise standard deviation	0.1
MLESAC minimum iterations	1000
Point-model distance threshold for Re-segmentation	0.1

Table 10: Evaluation results on the building PCD

Methods	**our method**	**[75]**	**[15]**	**[108]**
RMSE (cm)	2.05 ±0.57	7.20±1.66	**0.47±0.27**	7.85±5.33
Unit Volume Err. (m)	**0.45±0.63**	10.29±3.81	4.06±N/A	1.47±2.42
Normal Deviations (degree)	**0.9±0.6**	3.52±0.46	5.04±N/A	3.87±1.72
Detection Percentage	**93%**	65%	29%	71%
Oversegmentation Factor	**1.1**	3.5	2	1.3

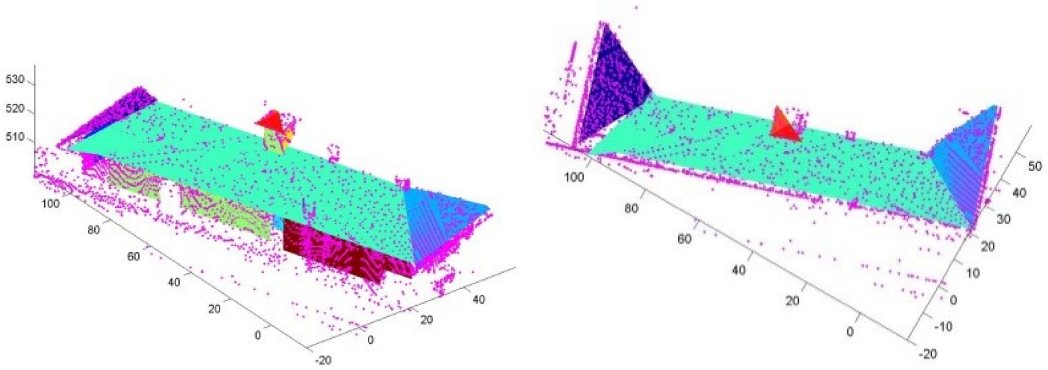
planes are used to evaluate RMSE, unit volume error and normal deviations. For detection percentage and oversegmentation factor, in total 14 planes are considered.

The evaluation results of the proposed procedure and the three baseline procedures are listed in Table 10. Note that in Table 10, method [15], no standard deviation is given because the method only extracts one patch that can be considered corresponding to a ground-truth plane, which is the largest wall of the building. Since only one extracted patch is considered to have a corresponding ground-truth patch, I only have one value for each metric and undefined standard deviation.

From Table 10 it can be concluded that the proposed method has the best performance among all the comparative methods.

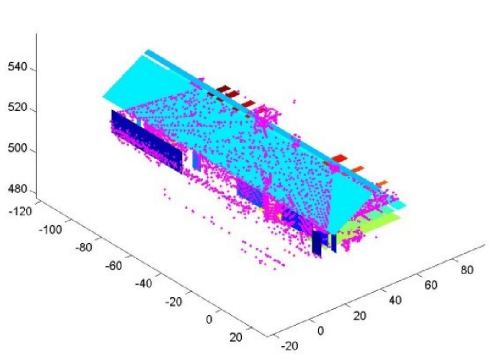
5.7 Conclustion

This work in this chapter focuses on the problem of planar model extraction from civil infrastructure PCDs, which requires three objectives including the detection of

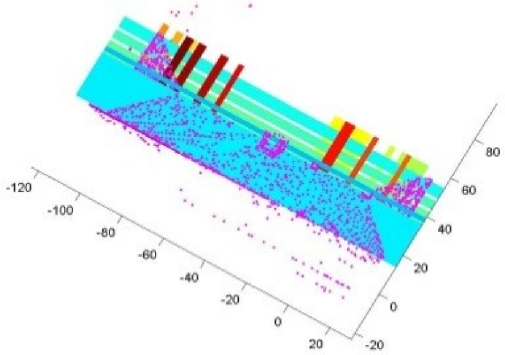


(a) Our method (view 1)

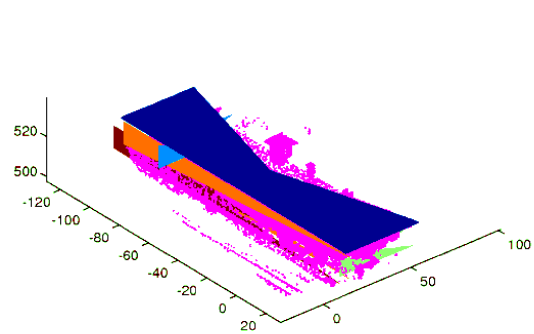
(b) Our method (view 2)



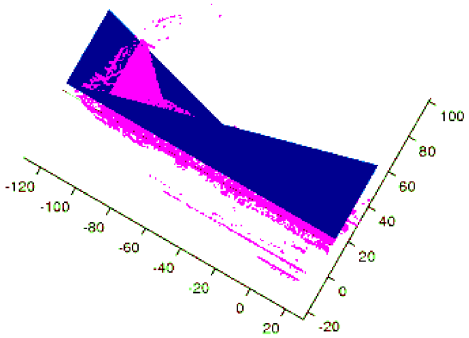
(c) Method in [75] (view 1)



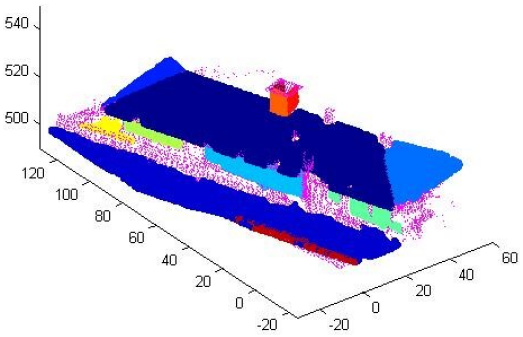
(d) Method in [75] (view 2)



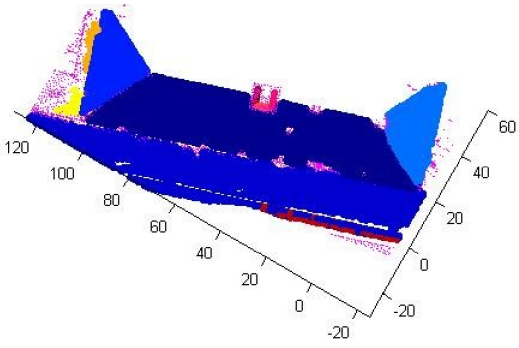
(e) Method in [15] (view 1)



(f) Method in [15] (view 2)



(g) Method in [108] (view 1)



(h) Method in [108] (view 2)

Figure 51: Extracted planar patches for the building PCD using different methods, plotted with the raw PCD (in magenta).

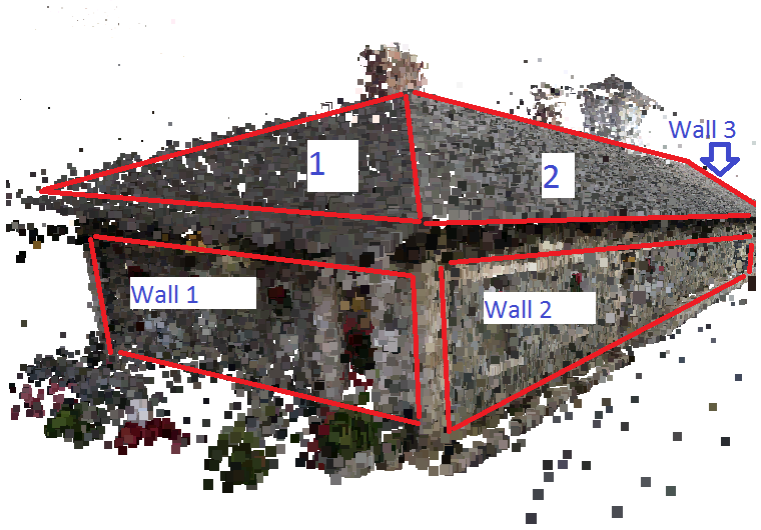


Figure 52: Planes measured using total stations to provide ground truth data.

planar structures, estimation of planar parametric models and determination of the planar model boundaries. The proposed procedure is demonstrated to be suitable for large-scale noisy infrastructure PCDs and able to address all the three objectives. One of the most important steps of this procedure is that it first recovers the linear dependence relationship between each point in the PCD, by solving a group-sparsity inducing optimization program. With the recovered linear dependence coefficients, the algorithm further segments the PCD by clustering the points according to the linear subspace. The clustering uses spectral clustering with a similarity graph formed from the linear dependence coefficients matrix. After PCD segmentation, planes are detected and estimated for each segmented group via an SVD based approach using MLESAC. Finally, the boundary of each plane is detected using the α-shape algorithm. The proposed algorithm is tested extensively using three types of PCDs: synthetic data and a PCD of a real building reconstructed from video. For the synthetic PCD experiment, detailed results are provided to illustrate every step of the procedure. To comprehensively evaluate the model extraction performance, five different evaluation metrics are applied. Furthermore, the proposed algorithm is also compared with three baseline methods. The experimental results and the evaluation

statistics on the real-world PCDs demonstrate that the proposed algorithm has the best overall performance among the comparative methods on the real-world PCDs.

CHAPTER VI

PCD MODELING WITH QUADRATIC SURFACE PRIMITIVES AND SEMANTIC INFORMATION

The approach in the last chapter has been demonstrated to be effective for real-world infrastructure PCDs. However, it only works for planar surfaces. In contrast, the work in this chapter seeks to extract more kinds of structures from infrastructure PCDs. The algorithm presented in this chapter is designed to detect, fit, and classify multiple surface primitives robustly and efficiently for civil infrastructure PCDs [112]. There are major advances over the previous one: (1) the segmentation of PCD is based on correlation relationships and performed in a fast manner; (2) the model estimation and the classification scheme is designed to make the algorithm work for much more geometric shape primitives; (3) a model merging procedure is performed with a novel model similarity measures to reduce the redundancy of the algorithm output. The algorithm is tested on a real-world infrastructure PCDs from VSLAM. The results are evaluated quantitatively with two evaluation metrics.

Beside the geometry driven modeling with surface primitive, semantic information is also very important in real-world applications, such as generating an as-built building information model from raw PCDs [113]. Although there is a solution to semantic learning for indoor environments [108], no solution exists for recognizing components and generating semantic models of infrastructures like bridges. In the latter part of this chapter, I address this problem by proposing a novel method to recognize both the semantic labels of facility components (e.g. beams, deck, columns, etc.) and geometric entity labels of computer-aided design (CAD) models (e.g. cuboids, cylinders, sheet, etc.). Note that there is not necessarily a one-to-one mapping between

these two kinds of labels. I tested the method on bridge PCDs, however the same principles and method should work for other structures as well (e.g. parking decks, building skeletons, etc.). The method takes the extracted surface primitives as input, then classifies the primitives into different classes of components or entities, and finally generates the CAD and Industry Foundation Classes (IFC) models with both the geometric information and semantic labels. The algorithm is designed for fast application: the classification step is of linear runtime. The algorithm is tested with PCDs modeling real-world bridges, and evaluated based on the classification error compared to the ground truth. Both evaluated results reveal the effectiveness of the algorithm.

6.1 Related Work

3D object detection is well-studied in information retrieval. One of the most widely used 3D object detection methods is the graph-based method which encodes both geometric and topological information. Examples include skeletal graphs [94] and augmented multi-resolution Reeb graphs [102]. A second class of methods is based on geometry, for instance, the principal plane analysis proposed in [63]. A survey is presented by Tangelder [95] on 3D object classification and retrieval methods. All of these methods take queried 3D models in specific formats as input and finally output the matching class. However, they are not specially designed or optimized for PCDs and require a large database for 3D models.

Within computer vision and robotics, object extraction algorithms specially designed for PCD often utilize classification and recognition algorithms. These methods, mostly being descriptor-based, are categorized into two groups: local methods and global methods. Local methods exploit descriptors based on locally invariant geometric properties around a surface point [56, 69, 85, 83]. These methods take a PCD, extract the descriptors, and then use the extracted descriptors as input to a classifier.

Local methods have the advantage of computational efficiency, but the performance heavily depends on the quality and resolution of the input PCD [30]. Moreover, these methods do not work globally and are designed for free-form object recognition. Thus they are not suitable for solving the problem we seek to address.

Of the global methods, the two main categories are generalized Hough transforms and RANSAC. Hough transform based methods first map input PCDs to parameter spaces and then classify shapes [78, 110]. Nevertheless these methods are expensive in both computation and memory. For RANSAC paradigm methods, Schnabel [85] proposed an efficient variant of RANSAC to detect and recognize primitives in PCDs, but it only works for five shapes and requires different estimation for different shapes.

For 3D entity models extraction from point cloud data, [4] proposed a method to recognize CAD models from range images by generating a strategy to select models' geometric features in sequence for identifying and localizing the model in the scene. The strategy is guided by objects' visibility, detectability, frequency of occurrence, etc, and the file output is stored in standard CAD models. Different from this, recent research in the computing in civil engineering community mainly studies the problem of entity model recognition for as-built BIMs. In [7, 8], a method based on an Iterative Closest Point (ICP)-based fine registration is proposed for recognizing CAD model objects from laser scans. Similarly, (Kim, C, et al 2011) also addresses the matching between the point clouds and the CAD models. They target the application of more complicated CAD models with registration combining principal component analysis coarse registration, and ICP fine registration. Another series of work aimed at as-built BIMs is presented in [108], which utilizes supervised stacked learning to learn indoor environments. The data-driven approach relies on high-quality and large training datasets, which themselves are difficult to obtain. Moreover, it is for indoor surroundings, not for the outdoor infrastructure that we target at.

6.2 Surface Primitive Extraction from Point Cloud Data

6.2.1 Over view of the algorithm

The proposed algorithm for detecting, fitting and classifying PCD surface primitives consists of several steps as outlined in Figure 53. The algorithm takes a raw PCD (with only Cartesian information) as input. Firstly, a fast segmentation is performed on the PCD. Each segmented group of points corresponds to a surface patch. The 6-DOF pose of the segmented points is recovered by finding a $\mathcal{SE}(3)$ matrix transforming the points to a canonical coordinate system. Then a full quadric model is fit to the segmented points using least-square estimation in a Maximum Likelihood Estimator SAmple Consensus (MLESAC) paradigm [99]. Using the estimated model parameters, the surface is classified as 1 of 12 types of quadric surfaces. After all groups have been processed, model merging is performed to join abutting groups from the same surface class, after which the models of the joined groups are re-estimated. Finally the algorithm outputs the estimated surface parametric models, surface primitive types, along with the PCD segmentation.

6.2.2 Fast Segmentation of PCDs

The fast segmentation strategy for large-scale PCDs firstly down-samples the dataset, then performs segmentation, and finally retrieves the labels for the original PCD. The original PCD $\mathcal{P}_0 \in \mathbb{R}^{N\times 3}$ is down-sampled by partitioning the PCD into discrete cubes in $\mathbb{R}^3$ and replacing the points in each cube with their centroid. The partitioning is done by a k-d tree search, which generates adaptive partitions is efficient for large-scale data and. A static k-d tree for a set $\boldsymbol{P}$ of n points can be built in $O(n \log n)$ time.

Segmentation is performed on the down-sampled point-set $\mathcal{P}' \in \mathbb{R}^{N'\times 3}$. The segmentation step exploits the local geometric relationship among points on the same quadric surface, i.e. a correlation relationship. The segmentation follows the idea in

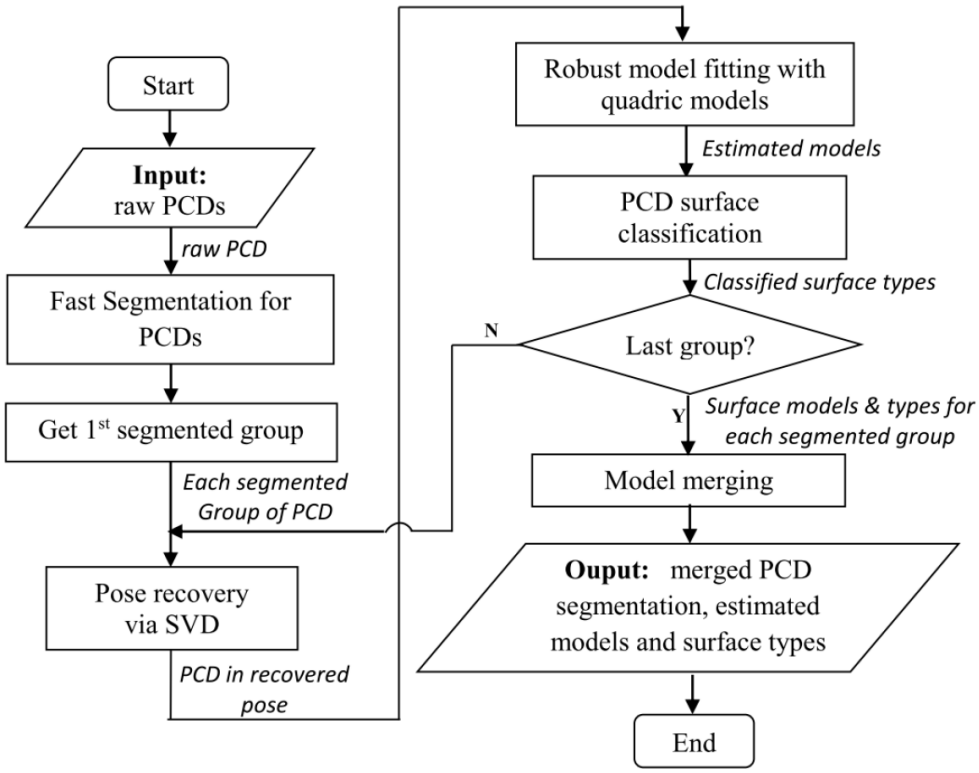


Figure 53: Flowchart of the surface primitive-based PCD modeling algorithm.

[67, 109], and clusters the PCD via lossy-compression. Assume $\mathcal{P}'$ is from a mixture of Gaussians and denote a segmentation into K clusters as $\mathcal{P}' = \{\mathcal{W}'_1 \cup \mathcal{W}'_2 \cup \ldots \cup \mathcal{W}'_K\}$, then the total number of bits to encode $\mathcal{P}'$ up to distortion λ is:

$$L^s(\mathcal{W}'_1,\ \mathcal{W}'_2,\ \ldots,\ \mathcal{W}'_K) = \sum_{i=1}^{K} \left[L(\mathcal{W}'_i) - |\mathcal{W}'_i| \log_2 \frac{|\mathcal{W}'_i|}{N'} \right] \tag{100}$$

for which $L(\mathcal{W}'_i)$ is the number of bits needed to encode each cluster $\mathcal{W}'_i$:

$$L(\mathcal{W}'_i) = \frac{N'+3}{2} \cdot \log_2 \det \left(I + \frac{3}{\lambda N'} \overline{\mathcal{W}'_i} \overline{\mathcal{W}'_i}^{\mathsf{T}} \right) + \frac{3}{2} \log_2 1 + \frac{\mu_i^{\mathsf{T}} \mu_i}{\lambda} \tag{101}$$

where μ is the mean of $\mathcal{W}'$ and $\overline{\mathcal{W}'} = \mathcal{W}' - \mu$.

It is proved in [67] that $\Delta L = L(\mathcal{W}'_i \cup \mathcal{W}'_j) - \left[L(\mathcal{W}'_i) + L(\mathcal{W}'_j) \right] \geqslant 0$, and that when $\mathcal{W}'_i$ and $\mathcal{W}'_j$ are more correlated, then ΔL is smaller. The segmentation minimizing the coding (bit) length will segment $\mathcal{P}'$ into clusters that maximize the intra-cluster correlation and minimize the inter-cluster correlation. This optimal segmentation is

found by a pairwise steepest descent procedure. After segmenting $\mathcal{P}'$, the segmentation for $\mathcal{P}_0$ is obtained by labeling each point $p_{0i} \in \mathcal{P}_0$ as belonging to the cluster $\mathcal{W}'_k \in \mathcal{P}'$ with the minimum point-to-cluster distance $D_p^{\mathcal{W}'_k} \in \mathcal{W}'_k$. Let , then the label for p_{0i} is:

$$k_{p_{0i}} = \underset{k}{\operatorname{argmin}}\ D_p^{\mathcal{W}'_k} = \underset{k}{\operatorname{argmin}} \left[\min \left(\sqrt{\|p_{0i} - p_i^{\mathcal{W}'_k}\|_{l2}} \right)_{j=1}^{|\mathcal{W}'_k|} \right]_{k=1}^{K} \tag{102}$$

6.2.3 PCDs Pose Recovery

Before model estimation, the PCD in each segmented group should be transformed to a canonical pose to improve the conditioning of the quadric model parameters estimation. To find the rigid transformation $T \in \mathcal{SE}(3)$ for transforming the PCD $\mathcal{P} \in \mathbb{R}^{N\times 3}$, $\mathcal{P}$ is first translated by subtracting the centroid $\mathcal{C}_\mathcal{P} \in \mathbb{R}^{1\times 3}$ to obtain $\overline{\mathcal{P}} = \mathcal{P} - \mathcal{C}_\mathcal{P} \cdot \boldsymbol{I}_{N\times 1}$. Then Singular Value Decomposition (SVD) is performed on $\overline{\mathcal{P}}^\intercal$:

$$\overline{\mathcal{P}}^\intercal_{3\times N} = R_{3\times 3} \cdot S_{N\times N} \cdot V_{N\times 3} \tag{103}$$

where $R_{3\times 3}$ is a unitary matrix which can be viewed as a matrix in $\mathcal{SO}(3)$, and $S_{N\times N}$ is a diagonal matrix encoding the scale. The rigid transformation T is then given by:

$$T = \begin{pmatrix} R & -\mathcal{C}_\mathcal{P}^\intercal \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{104}$$

The PCD $\tilde{\mathcal{P}} \in \mathbb{R}^{N\times 3}$ in the recovered pose is obtained from:

$$\begin{pmatrix} \tilde{\mathcal{P}} \\ 1 \end{pmatrix} = \begin{pmatrix} S_{N\times N} \cdot V_{N\times 3} \\ 1 \end{pmatrix} = T \cdot \begin{pmatrix} \mathcal{P} \\ 1 \end{pmatrix} \tag{105}$$

6.2.4 Robust Fitting with Quadric Models

After pose recovery, to every PCD $\tilde{\mathcal{P}}$ a fully quadric model is fit using least-square estimation. The quadric model is:

$$\tilde{\mathcal{P}}^\intercal \Theta \tilde{\mathcal{P}} = 0;\ \Theta \in \mathbb{R}^{10\times 10} \tag{106}$$

which can be expanded as:

$$ax^2 + by^2 + cz^2 + fyz + gzx + hxy + px + qy + rz + d = 0 \tag{107}$$

The parameter vector $\theta = [a, b, c, f, g, h, p, q, r, d]^\intercal$ can be estimated with direct linear transformation (DLT). Let $\tilde{\mathcal{P}} = [x_i, y_i, z_i]_{i=1}^N$, the algebraic error is shown in Equation (108).

$$\varepsilon = \begin{pmatrix} x_1^2 & y_1^2 & z_1^2 & y_1 z_1 & z_1 x_1 & x_1 y_1 & x_1 & y_1 & z_1 & 1 \\ x_2^2 & y_2^2 & z_2^2 & y_2 z_2 & z_2 x_2 & x_2 y_2 & x_2 & y_2 & z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^2 & y_N^2 & z_N^2 & y_N z_N & z_N x_N & x_N y_N & x_N & y_N & z_N & 1 \end{pmatrix} \cdot \theta = \mathbf{A} \cdot \theta \in \mathbb{R}^{N \times 1} \tag{108}$$

The estimate is obtained by minimizing $\|\varepsilon\|^2$, which can be done by performing SVD on the data matrix $\mathbf{A}$. Let $\mathbf{A} = USV^\intercal$, then θ is given by the tenth column of V. The least-square estimation of is performed using MLESAC. MLESAC is an accurate and robust method in the presence of measurement uncertainty and noise [19]. It is also invariant to the number of points, which means that the threshold need not vary with the sample size. The inlier error e is modeled by an unbiased Gaussian distribution, while the outlier error is modeled by a uniform distribution. The loss is defined as $Loss(e) = -\ln[Prob(e|\theta)]$, where $Prob(e|\theta)$ is the probability of the error given the estimated model. If the loss of the estimated model is smaller than a threshold, then the model will be re-estimated using only the inliers and the iterations terminate; otherwise repeat the random sampling and estimation step up to a maximum number of iterations. The maximum number of iterations is:

$$T = \frac{\log \alpha}{\log (1 - \gamma^1 0)} \tag{109}$$

where γ is the prior probability of being an inlier, and α is the probability of failing to pick a valid inlier set during the random sampling.

6.2.5 Classification of Surface Primitives

In classification step, 12 quadric surface primitives are considered, as shown in Table 11. The estimated model parameter vector θ is first truncated by setting the elements smaller than a threshold to be zero. Then form the following two matrices:

$$\mathbf{M_e} = \begin{pmatrix} a & h/2 & g/2 \\ h/2 & b & f/2 \\ g/2 & f/2 & c \end{pmatrix} \tag{110}$$

and

$$\mathbf{M_E} = \begin{pmatrix} a & h/2 & g/2 & p \\ h/2 & b & f/2 & q \\ g/2 & f/2 & c & r \\ p & q & r & d \end{pmatrix} \tag{111}$$

The classification parameters are:

$$\Delta = det(E) \tag{112}$$

$$\phi = rank(e) \tag{113}$$

$$\Phi = rank(E) \tag{114}$$

; "k-sign"and "K-sign"refer to the nonzero eigenvalues of e and E respectively

6.2.6 Model Merging

After model estimation, the PCDs from some groups may be from the same surface. Model merging is a procedure of merging the points from same surface primitives then re-estimating the quadric model. Model merging is performed in iterative manner. A binary adjacent matrix W is first generated. If i-th and j-th groups are adjacent then $w_{ij} = 1$; otherwise $w_{ij} = 0$. W is a symmetric matrix with the diagonal elements being one. Check all the adjacent pairs of groups in W. If the classified shapes of two adjacent groups are the same, then the Mahalanobis norm between these two models

No.	surface primitives		classification criteria				
	Name	canonical expression	ϕ	Φ	Δ	k-sign	K-sign
1	One real plane	$ax + by + cz + d = 0$	1	1			
2	Ellipsoid	$x^2/a + y^2/b + z^2/c = 1$	3	4	< 0		
3	elliptic cylinder	$x^2/a + y^2/b = 1$	2	3		same	opposite
4	hyperbolic cylinder	$x^2/a - y^2/b = 1$	2	3		opposite	
5	parabolic cylinder	$x^2 + 2y = 0$	1	3		opposite	
6	quadric cone	$x^2/a + y^2/b - z^2/c = 0$	3	3		opposite	
7	hyperboloid of one sheet	$x^2/a + y^2/b - z^2/c = 1$	3	3		opposite	
8	hyperboloid of two sheets	$x^2/a + y^2/b - z^2/c = -1$	3	4	> 0	opposite	
9	hyperbolic paraboloid	$x^2/a - y^2/b + 2z = 0$	2	4	< 0	opposite	
10	elliptic paraboloid	$x^2/a - y^2/b + 2z = 0$	2	4	> 0	same	
11	intersecting planes	$x^2/a - y^2/b = 0$	2	2	< 0	opposite	
12	parallel planes	$x^2 = 1$	1	2			opposite

Table 11: Descriptions and classification criteria for quadric surface primitives. $a, b, c \neq 0$.

is computed:

$$dist(\theta_i, \theta_j) = \|\theta_i - \theta_j\|_\Sigma = (\theta_i - \theta_j) \cdot \Sigma \cdot (\theta_i - \theta_j)^\intercal \tag{115}$$

where $\theta_i, \theta_j \in \mathbb{R}^{10\times 1}$ are estimated parameter vector of two adjacent groups; Σ is a diagonal matrix with the diagonal elements as [0.15, 0.15, 0.15, 0.1, 0.1, 0.1, 0.05, 0.05, 0.05, 0.1]. Note that this norm assigned different weights to different model parameters in the model comparison. If $dist(\theta_i, \theta_j)$ is smaller than a threshold, the points from these two groups are merged. Meanwhile, the adjacent matrix is updated accordingly and a quadric model is re-estimated using the new group. The model merging continues until no more adjacent groups need to be merged. Finally, the inlier sets for each merged model are determined from the whole PCD.

6.3 Semantic Modeling of Point Cloud Data

The proposed method consists of four major algorithms: (1) Pre-processing step: to extract the surface primitives and the support given an input query PCD, in order to generate classification features in later steps. (2) Feature extraction step: to generate the proposed feature vectors, which capture the distinct geometric properties of each

component to facilitate classification. (3) Classification step: to classify each surface primitive and generate the labels for both the facility component labels and geometric entity labels, with a pre-trained multiple-class adaboost decision tree. (4) Model generation and output step: based on the labels generated from classifier, this step generates the final CAD model files and IFC files for as-built BIMs.

Figure 54 shows the flowchart illustrating the structure of the proposed algorithm. Although the proposed algorithm is generic and can be

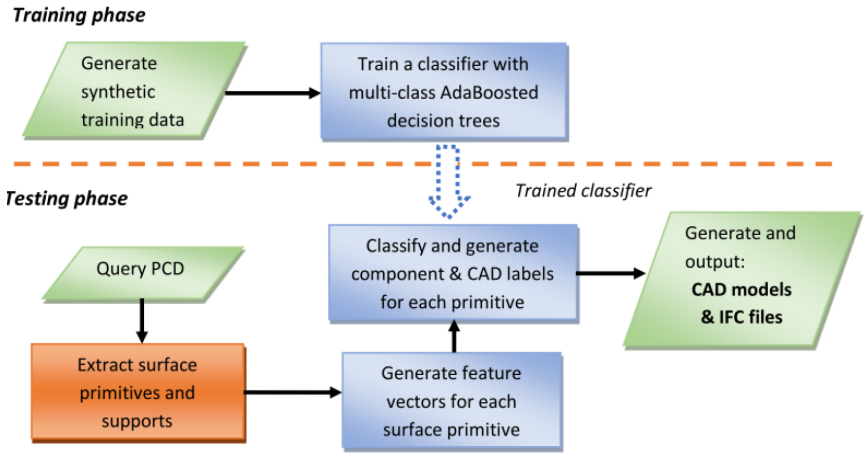


Figure 54: Pipeline of the algorithm for semantic recognition of PCDs based on Surface Primitives.

6.3.1 Features

Features for classification. Given the input data consisting of multiple surface primitives $\{M_i\}_{i=1}^{M}$ and the corresponding support set $\{S_i\}_{i=1}^{M}$ for the primitives, a set of feature vector $\{f_i\}_{i=1}^{M}$ is generated with one feature vector corresponding to one extracted primitive. The feature vector f_i is a $(9+2N_p)$ dimensional vector consisting of the coordinates:

$$f_i = [T_i, n_i, V_i, B_i, L_i], \tag{116}$$

where N_p is the number of the primitive types. This feature vector captures the following information of each input primitive: primitive type, principal direction,

normal vector, spatial scale and neighborhood statistics. The details of each sub-vector are elaborated as follows.

1. $T_i \in N^{N_p}$ is an index encoding the type of the surface primitive. For example, in the case of a bridge whose surface primitive set has planes and cylinders, then T_i can be encoded with indices $\{1, 2\}$ as:

$$T_i = \begin{cases} 1, \text{ if surface is a plane} \\ 2, \text{ if surface is a cylinder} \end{cases} \tag{117}$$

2. $n_i \in R^3$ is the normal of each surface primitive. This corresponds to the coefficients of the x-term, y-term and z-term in the estimated surface primitive model.

3. $V_i \in R^3$ is a unit vector which captures the principal direction of the support of the primitive M_i. Given the support of primitive M_i as $S_i \in R^{3\times k}$, S_i is first centered by it mean, yielding $\bar{S}_i$, then V_i is computed by finding the eigenvector corresponding to the largest eigenvalue from $\bar{S}_i{}^{\mathsf{T}}\bar{S}_i$. Note that since we only need the first principal eigenvector, performing a power iteration is sufficient instead of performing a full spectral decomposition, where the latter one is of much more computational cost.

4. $B_i \in R^3$ is a vector capturing the relative scale of bounding box size S_i compared to the bounding box size of the whole point set. Assuming the whole point set is $\{P_i = (x_i, y_i, z_i)\}$, then the bounding box size of $\{P_i\}$ is given by $[a, b, c] = [(\max(x_i) - \min(x_i)), (\max(y_i) - \min(y_i)), (\max(z_i) - \min(z_i))]$. If the bounding box of the primitive support S_i is $[a_i, b_i, c_i]$, then $B_i = [\frac{a^k}{a}, \frac{b^k}{b}, \frac{c^k}{c}]$.

5. $L_i \in R^{N'_p}$ is a vector capturing the neighborhood statistics indicating the number of each primitive types which are in 1-distance neighbor (directly connected)

of the current surface primitive. N_p' is the size of the unique type set of the neighboring primitives. For example, in the case of a bridge with only planes and cylinders, a primitive is directly connected to 3 cylinders and 2 other planes, then $L_i = [2, 3]$. The connectivity is determined by thresholding on the single-linkage pairwise Euclidean distance of each support.

6.3.2 Classification with Adaboost

Given the synthetic training dataset with ground truth infrastructure labels and CAD entity labels, I first compute the feature vector of each primitive. Then the feature vectors and the training labels are used to train the classifier (in total two classifiers are trained; one for classifying infrastructure component labels and one for CAD entity labels). Considering that the feature vector consists of different variable types (categorical, discrete, and continuous variables) and the classification problem is non-linear, I proposed to use AdaBoosted decision trees for the classification.

AdaBoost [41, 80, 49] is an iterative procedure which combines many weak classifiers with different weights to approximate the optimal Bayes classifier. In our work, maximum entropy decision trees are used as the weak classifiers. In the training phase, during each AdaBoost iteration, the feature vectors are first used to train one decision tree. Then according to the classification error, the classifier is weighted by a weight learnt by AdaBoost algorithm with an exponential loss function. Specifically, for the multi-class problem in our case, I use the SAMME algorithm [52], which utilizes the population minimizer of multi-class exponential loss. Given the input feature set $\{f_i\}$ containing K classes, the algorithm starts by initializing all the weights for weak classifiers as $w_k = \frac{1}{K}$. Then if N_C weak classifiers, i.e. the decision trees, will be boosted, in total N_C iterations need to be conducted. In each iteration, the algorithm first induces a decision tree $T^{(n_c)}(f)$ with the training set and the initial weight $\frac{1}{K}$.

Then the empirical error rate is computed as

$$Err^{n_c} = \frac{\sum_{k=1}^{K} w_k \mathbb{I}(l_k \neq T^{n_c}(f_k))}{\sum_{(} k=1)^{K} w_k}, \tag{118}$$

where $\mathbb{I}(\cdot)$ is an indicator function: if $l_k \neq T^{n_c}(f_k)$ is true, which means the classified label of T^{n_c} on f_i does not agree with the true label l_k, then $\mathbb{I}(l_k \neq T^{n_c}(f_k)) = 1$; otherwise $\mathbb{I}(l_k \neq T^{n_c}(f_k)) = 0$. With the error rates, the weights are updated per

$$C(f) = \arg\max_{l'} \sum_{n_c=1}^{N_C} \alpha^{(n_c)} \cdot \mathbb{I}(T^{(n_c)}(f) = l') \tag{119}$$

6.3.3 Decision Tree Induction

Decision tree is a nonmetric method for classification and regression [31, 12]. The decision tree used in our work follows binary-tree structures, with each node imposing one splitting criterion on one feature in the feature vector. The outcome of each node corresponds to a decision with respect to the feature of the node and yields a splitting of a subset in the training data. Hierarchically, the full training set is split by the root node, while each successive decision splits the proper subset of the data. The decision trees used as a weak classifier in our multi-class AdaBoost are linear decision trees, which generate decision boundaries aligned with the axis hyper-planes in the $(9 + 2N'_P)$ dimensional feature space. The testing runtime of a decision tree is linear, and the Adaboosted classifier is of linear runtime.

The training of the decision tree is done by top-down induction. In each iteration a decision attribute is selected and assigned for the next node. Then according to the attribute, the training subset is split, creating the descendants of the node. The induction terminates if all the training data is perfect classified.

The attribute selection criterion used in our work is to choose the attribute which

maximizes the information gain, or equivalently achieves the most reduction in entropy. Given a subset of training set $\mathcal{M}' = \{\mathfrak{m}'\}$ for current node and its corresponding label set $\mathcal{L}' = \{l'_i\}_{i=1}^{N'}$, I first measure the entropy before decision as

$$H(\mathcal{M}') = -\sum_{i=1}^{N'} \text{Prob}(m' = l'_i) \log_2 \text{Prob}(m' = l'_i) \tag{120}$$

Assume that after the binary decision A, set $\mathcal{M}'$ is partitioned into $\mathcal{M}'_1$ and $\mathcal{M}'_2$, and $|\mathcal{M}'| = |\mathcal{M}'_1| + |\mathcal{M}'_2|$, then the entropy reduction is measured by information gain :

$$\text{Gain}(A) = H(\mathcal{M}') - \sum_{j=1}^{2} \frac{|\mathcal{M}'_j|}{|\mathcal{M}'|} H(\mathcal{M}'_j) \tag{121}$$

Notice that when the decision A is to split between feature values $a_1, a_2, (a_1 \leq a_2)$, then any value between $a_1,\ a_2$ will lead to the same entropy reduction. In this case I select the threshold to be $\frac{a_1+a_2}{2}$.

6.3.4 Classification and Output Generation

Once the multi-class AdaBoosted decision tree classifier is trained with the synthetic training samples, it can be used to classify the query PCD directly.

Two classifiers are trained which correspond to two types of labels: infrastructure component labels and CAD entity labels. The infrastructure component labels the semantic description of the component. For example, in the case of bridge infrastructure, component labels include *deck*, *beam*, *barrier*, etc. The CAD entity labels include geometric solid models, e.g. *cuboid*, *cylinder*, *sheet*, etc.

After classification, the CAD models are generated by finding the bounding box of support set $\{S_j\}$ belonging to the same CAD entity. The CAD models are output in ply format. Similarly, IFC files are written according to the infrastructure component labels and the support set. The files are output with IFC4 specifications [65].

6.4 Evaluation

6.4.1 Results of surface primitive modeling.

The algorithm is tested on a PCD of a real-world bridge, shown in Figure 55 (Left). The PCD contains 407503 points. As part of the first step, the PCD is down-sampled to 7756 points, shown in Figure 55 (Middle). The segmentation step uses a coding length threshold of 15 and over-segments the PCD into eight groups. In model estimation, $\gamma = 0.9$, $\sigma = 0.5$. For the surface classification step, the normalized truncation threshold is 0.005. The final result after model merging is shown in Figure 55 (Right), with a detection percentage of 100%. . The experiment is performed on a PC with a 2.80GHz Intel Core i5 CPU and 8GB RAM, using C++ and MATLAB. The down-sampling step is in C++ and the other steps are in MATLAB. In total, the experiment took 452.13 seconds (the segmentation took 201.90 seconds). For each model, the algorithm returns the estimated model parameters. More quantitative results are listed in Table 12.

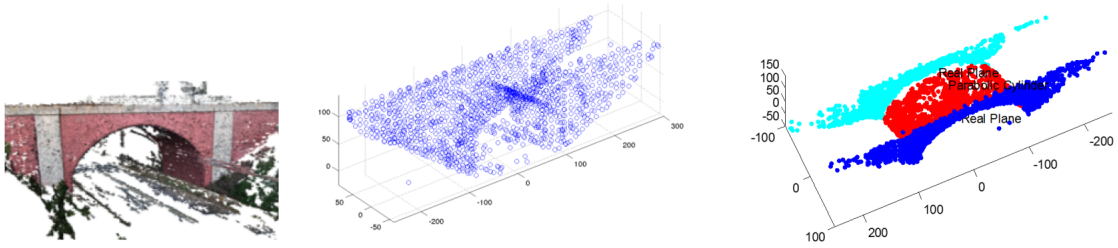


Figure 55: Results of surface primitive modeling. Left: input PCD. Middle: down-sampled PCD. Right: detection and classification results.

Surface Detected	Real Plane No.1	Parabolic Cylinder	Real Plane No.2
Algebraic fitting error	0.0235	5.3943e-04	0.0242
Point-to-surface average distance (cm)	1.2344	1.3248	1.9129

Table 12: Evaluation of experimental results on the bridge PCD.

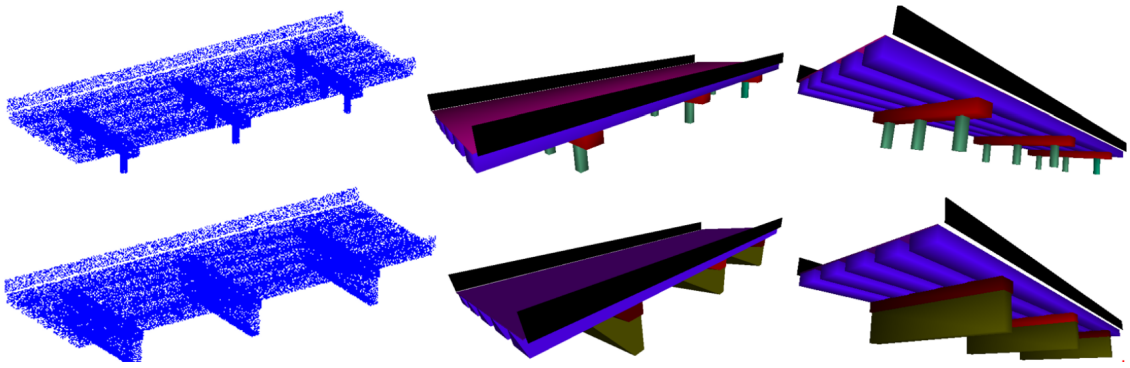


Figure 56: Left: query PCDs. Middle and right: augmented models consisting of a collection of CAD entities colored by semantic labels.

6.4.2 Results of semantic recognition and final augmented models.

The experiment uses a bridge infrastructure as the application scenario. A synthetic training set is generated according to the real configurations and dimensions of a bridge. The semantic labels include: "Pier column", "Pier cap", "Beam", "Barrier", "Deck", and "Wingwall". CAD entity labels include: "Cylinder", "Cuboid", and "Sheet". In the training phase, 10 PCDs with the ground truth labels are used for training.

With the trained classifier, two query PCDs of bridges are tested to recognize the semantic labels and CAD entity labels. The final output in CAD models combined with semantic labels are shown in Figure 56, in which different components are encoded with different colors. In this experiment, both the training and testing accuracy are 100%.

6.5 Conclusion

This chapter first presents an algorithm for detecting, fitting and classifying quadric surface primitives for infrastructure PCDs. The algorithm consists of several steps, including fast segmentation, pose recovery, quadric model estimation, quadric surface classification, and model merging. Among these steps, the pose recovery, model

estimation and model merging are new algorithms proposed while the other two are modified from existing state-of-the-art methods. Evaluation of the algorithm used three quantitative evaluation metrics, i.e. algebraic fitting error, mean point-to-surface (PS) distance and detection percentage. The algorithm was tested on a PCD of a real bridge, and shown to accurately recover the underlying surfaces.

Based on the surface primitive PCD modeling, an approach to recognize and classify the infrastructure components for as-built BIMs is proposed. The main contribution is two-fold: (1) I propose the feature vectors capturing the distinct geometric properties of the civil components; (2) based on the features I design a classifier for classifying both infrastructure component labels and geometric entity labels. The algorithm takes shape primitives and supports of a PCD as input, then generate the corresponding features and classify with a multi-class AdaBoost Decision Tree, which is trained by synthetic training set. The final result is output as CAD and IFC files with the infrastructure component labels. Experimental results applying to bridges reveal the effectiveness of the algorithm.

CHAPTER VII

CONCLUSION

This thesis investigates into the problem of generating augmented models from image sequences. This problem is divided into two key problems in robotics and computer vision. The first problem is monocular visual simultaneous localization and mapping (VSLAM), which estimates the camera poses and reconstructs the point cloud data (PCD) of the environment. For the VSLAM problem, I develop algorithms to improve the VSLAM accuracy, by leveraging system observability theories. The second problem is the PCD modeling problem, which builds the augmented models from low-level PCDs with only spatial information. To solve this problem, I design geometry-driven approaches to model the PCD with fundamental shape primitives, and develop a method to retrieve semantic information for the PCD model. The key contributions can be summarized as follows:

- **Modeling and analysis of VSLAM from the perspective of system observability.** I first model the $SE\langle 3\rangle$ VSLAM system as a piece-wise linear system (PWLS). To analyze the observability conditions of the PWLS, stripped observability matrix is used. Based on this, I further derive and prove two necessary conditions for the VSLAM system to be completely observable. These two conditions are further used to develop VSLAM algorithms guided by observability.

- **Optimally Observable and Minimal Cardinality (OOMC) VSLAM algorithm.** The OOMC algorithm is designed based on the instantaneous condition for VSLAM's complete observability. At each time instance, the algorithm aims to use only three features (a triplet), which is the minimum number of

features necessary to maintain a completely observable system. The triplets are selected via the proposed observability scores and image matching qualities. The OOMC algorithm is integrated with a Extended Kalman Filter based VSLAM system to improve the data association of EKF VSLAM. Experimental results demonstrate that the algorithm leads to a deterministic data association process with about seven times speed-up but maintains the same level of localization accuracy.

- **Good Features (GFs) to track for VSLAM algorithm.** The GF algorithm is designed based on the temporal condition for VSLAM's complete observability. Since a single feature can maintain the complete observability when it is tracked and estimated across two time segments, the GF algorithm is able to select the best features by ranking individual features. The τ-temporal observability score is further proposed as the ranking criterion. Efficient computation algorithm is designed to update the score via incremental singular value decomposition. To handle the case when not enough good individual features are available, a greedy feature grouping algorithm is developed, with theoretical supports of near-optimality via submodular learning. The GF algorithm is integrated respectively into filtering based VSLAM and multi-threading keyframe bundle adjustment VSLAM. Evaluation is performed on large-scale benchmark data sets with and without loop-closures in the scenes. Experimental results demonstrates that the algorithm improves the VSLAM accuracy with little loss of computation efficiency.

- **PCD modeling based on planar patches via sparsity-inducing optimization.** Given a reconstructed PCD, this work aims to build an augmented model driven by geometric shape primitives. This algorithm focuses on the

modeling PCDs with planar patches. A sparsity-inducing optimization is formulated with group-sparsity regularization for retrieving the linear subspace embedded in the PCD. Segmentation with respect to linear subspaces is further performed with spectral clustering. The model estimation is performed with singular value decomposition within Maximum Likelihood Estimation Sample Consensus (MLESAC) framework. Boundary of a patch is detected with a 3D variant of α-shape algorithm. Evaluation on real-world large-scale PCDs proves the advantages the algorithm with five quantitative metrics.

- **PCD modeling with quadratic surface primitives and semantic information.** To extend the planar patches based PCD modeling, an algorithm is proposed for modeling PCD with the quadratic surface primitives due to their flexibility. After the PCD segmentation via lossy-compression clustering, the algorithm transforms the segments to canonical poses by solving an $SE\langle 3\rangle$ transformation. The quadratic model is further estimated with direct linear transformation in MLESAC framework. The model coefficients are further used to recognize the types of shapes and perform model re-merging. To retrieve the semantic information, the multi-class Adaboost algorithm is applied.

There are several research directions for the future work. The first direction is to develop a real-time VSLAM algorithm which is able to reconstruct dense and accurate point clouds for large-scale scenes without loop-closures. The algorithms proposed in this thesis are able to combat long-term drifts for improving the accuracy, but to further improve the PCD density especially for uniform textures while performing VSLAM in real-time is still an open problem. The second direction is to directly incorporate the PCD modeling in the VSLAM process, to achieve "simultaneous localization and modeling". This kind of shape/object-oriented real-time VSLAM for large-scale scenes is till an open problem. Specifically, the point features need to be replaced with surface primitives. How to use observability to guide this type

of VSLAM is also an interesting topic. The third direction is to achieve reciprocal VSLAM and object recognition. Image-based object recognition incorporating the VSLAM localization results is largely solved in the literature. Nevertheless, it is still open problem on reciprocally incorporating the object recognition results to improve the VSLAM.

REFERENCES

[1] Agarwal, P. K. and Mustafa, N. H., "K-Means projective clustering," in *ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 155–165, ACM, 2004.

[2] Andrade-Cetto, J. and Sanfeliu, A., "The effects of partial observability in slam," in *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 397–402, IEEE, 2004.

[3] Andrade-Cetto, J. and Sanfeliu, A., "The effects of partial observability when building fully correlated maps," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 771–777, 2005.

[4] Arman, F. and Aggarwal, J., "CAD-based vision: object recognition in cluttered range images using recognition strategies," *CVGIP: Image Understanding*, vol. 58, no. 1, pp. 33–48, 1993.

[5] Bailey, T. and Durrant-Whyte, H., "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[6] Balzer, J. and Soatto, S., "CLAM: Coupled localization and mapping with efficient outlier handling," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1554–1561, 2013.

[7] Bosche, F. and Haas, C., "Automated retrieval of 3D CAD model objects in construction range images," *Automation in Construction*, vol. 17, no. 4, pp. 499–512, 2008.

[8] Bosché, F., "Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction," *Advanced engineering informatics*, vol. 24, no. 1, pp. 107–118, 2010.

[9] Bosché, F., "Plane-based registration of construction laser scans with 3D/4D building models," *Advanced Engineering Informatics*, vol. 26, no. 1, pp. 90–102, 2012.

[10] Boyd, S. and Vandenberghe, L., *Convex optimization*. Cambridge University Press, 2009.

[11] Brand, M., "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra and its Applications*, vol. 415, no. 1, pp. 20–30, 2006.

[12] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A., *Classification and regression trees*. CRC press, 1984.

[13] Bryson, M. and Sukkarieh, S., "Active airborne localisation and exploration in unknown environments using inertial SLAM," in *IEEE Aerospace Conference*, IEEE, 2006.

[14] Bryson, M. and Sukkarieh, S., "Observability analysis and active control for airborne SLAM," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 1, pp. 261–280, 2008.

[15] Budroni, A. and Böhm, J., "Toward automatic reconstruction of interiors from laser data," *Virtual Reconstruction and Visualization of Complex Architectures (3D-Arch)*, 2009.

[16] Carlone, L., Alcantarilla, P. F., Chiu, H.-P., Kira, Z., and Dellaert, F., "Mining structure fragments for smart bundle adjustment," in *British Machine Vision Conference*, 2014.

[17] Ceriani, S., Fontana, G., Giusti, A., Marzorati, D., Matteucci, M., Migliore, D., Rizzi, D., and Sorrenti, D. G., "RAWSEEDS ground truth collection systems for indoor self-localization and mapping," *Autonomous Robots*, vol. 27, no. 4, pp. 353–371, 2009.

[18] Cho, D.-M., Tsiotras, P., Zhang, G., and Holzinger, M. J., "Robust feature detection, acquisition and tracking for relative navigation in space with a known target," in *AIAA Guidance, Navigation, and Control Conference, Boston, MA*, 2013.

[19] Choi, S., Kim, T., and Yu, W., "Performance evaluation of RANSAC family," in *British Machine Vision Conference*, 2009.

[20] Choudhary, S., Indelman, V., Christensen, H., and Dellaert, F., "Information-based reduced landmark SLAM," in *IEEE International Conference on Robotics and Automation*, pp. 4620–4627, May 2015.

[21] Civera, J., Davison, A. J., and Montiel, J. M., "Inverse depth parametrization for monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[22] Civera, J., Grasa, O. G., Davison, A. J., and Montiel, J., "1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry," *Journal of Field Robotics*, vol. 27, no. 5, pp. 609–631, 2010.

[23] Comaniciu, D. and Meer, P., "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[24] Costeira, J. P. and Kanade, T., "A multibody factorization method for independently moving objects," *International Journal of Computer Vision*, vol. 29, no. 3, pp. 159–179, 1998.

[25] Davison, A., "Active search for real-time vision," in *IEEE International Conference on Computer Vision*, vol. 1, pp. 66–73, 2005.

[26] Davison, A. J., Cid, Y. G., and Kita, N., "Real-time 3D SLAM with wide-angle vision," in *IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[27] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O., "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[28] Dellaert, F. and Kaess, M., "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[29] Dorninger, P. and Pfeifer, N., "A comprehensive automated 3D approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds," *Sensors*, vol. 8, no. 11, pp. 7323–7343, 2008.

[30] Drost, B., Ulrich, M., Navab, N., and Ilic, S., "Model globally, match locally: Efficient and robust 3D object recognition," in *IEEE Conference Computer Vision and Pattern Recognition*, pp. 998–1005, 2010.

[31] Duda, R. O., Hart, P. E., and Stork, D. G., *Pattern classification*. John Wiley and Sons, 2012.

[32] Durrant-Whyte, H. and Bailey, T., "Simultaneous localization and mapping: part I," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[33] Eade, E. and Drummond, T., "Edge landmarks in monocular SLAM," in *British Machine Vision Conference*, 2006.

[34] Eade, E. and Drummond, T., "Scalable monocular SLAM," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 469–476, IEEE, 2006.

[35] Elhamifar, E. and Vidal, R., "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[36] Engel, J., Schöps, T., and Cremers, D., "LSD-SLAM: Large-Scale Direct monocular SLAM," in *European Conference Computer Vision*, pp. 834–849, 2014.

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

[38] Farin, G., *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier, 2014.

[39] Fischler, M. A. and Bolles, R. C., "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

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

[41] Freund, Y. and Schapire, R. E., "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[42] Fulton, W., "Eigenvalues, invariant factors, highest weights, and schubert calculus," *Bulletin of the American Mathematical Society*, vol. 37, no. 3, pp. 209–249, 2000.

[43] Furukawa, Y. and Ponce, J., "Accurate, dense, and robust multiview stereopsis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1362–1376, 2010.

[44] Geiger, A., Lenz, P., and Urtasun, R., "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[45] Gibbens, P. W., Dissanayake, G. M., and Durrant-Whyte, H. F., "A closed-form solution to the single degree of freedom simultaneous localisation and map building (SLAM) problem," in *IEEE Conference on Decision and Control*, vol. 1, pp. 191–196, 2000.

[46] Goshen-Meskin, D. and Bar-Itzhack, I., "Observability analysis of piece-wise constant systems. I. theory," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 28, no. 4, pp. 1056–1067, 1992.

[47] Gu, M. and Stanley C., E., "A stable and fast algorithm for updating the singular value decomposition," *Technical Report YALEU/DCS/RR-966, Department of Computer Science*, 1993.

[48] Hähnel, D., Burgard, W., and Thrun, S., "Learning compact 3D models of indoor and outdoor environments with a mobile robot," *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 15–27, 2003.

[49] Han, J., Kamber, M., and Pei, J., *Data mining: concepts and techniques: concepts and techniques*. Elsevier, 2011.

[50] Handa, A., Chli, M., Strasdat, H., and Davison, A. J., "Scalable active matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1546–1553, IEEE, 2010.

[51] Hartley, R. and Zisserman, A., *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[52] Hastie, T., Rosset, S., Zhu, J., and Zou, H., "Multi-class AdaBoost," *Statistics and Its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[53] Hesch, J., Kottas, D. G., Bowman, S. L., Roumeliotis, S., and others, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 158–176, 2014.

[54] Huang, G. P., Mourikis, A. I., and Roumeliotis, S. I., "Observability-based rules for designing consistent EKF SLAM estimators," *International Journal of Robotics Research*, vol. 29, no. 5, pp. 502–528, 2010.

[55] Huber, D., Akinci, B., Oliver, A. A., Anil, E., Okorn, B. E., and Xiong, X., "Methods for automatically modeling and representing as-built building information models," in *NSF CMMI Research Innovation Conference*, 2011.

[56] Johnson, A. E. and Hebert, M., "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[57] Kaess, M. and Dellaert, F., "Covariance recovery from a square root information matrix for data association," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1198–1210, 2009.

[58] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F., "iSAM2: Incremental smoothing and mapping using the bayes tree," *International Journal of Robotics Research*, vol. 31, pp. 217–236, 2012.

[59] Kaess, M., Ranganathan, A., and Dellaert, F., "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[60] Klein, G. and Murray, D., "Parallel tracking and mapping for small AR workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234, 2007.

[61] Krause, A., Singh, A., and Guestrin, C., "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, pp. 235–284, 2008.

[62] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W., "g 2 o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation*, pp. 3607–3613, IEEE, 2011.

[63] Kuo, C.-T. and Cheng, S.-C., "3D model retrieval using principal plane analysis and dynamic programming," *Pattern Recognition*, vol. 40, no. 2, pp. 742–755, 2007.

[64] Lee, K. W., Wijesoma, W. S., and Javier, I. G., "On the observability and observability analysis of SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3569–3574, IEEE, 2006.

[65] Liebich, T., "Ifc4–the new buildingsmart standard," 2013.

[66] Liu, G., Lin, Z., and Yu, Y., "Robust subspace segmentation by low-rank representation," in *International conference on machine learning*, pp. 663–670, 2010.

[67] Ma, Y., Derksen, H., Hong, W., and Wright, J., "Segmentation of multivariate mixed data via lossy data coding and compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1546–1562, 2007.

[68] Margarita, C. and Davison, A. J., "Active matching," in *European Conference on Computer Vision*, pp. 72–85, 2008.

[69] Mian, A. S., Bennamoun, M., and Owens, R., "Three-dimensional model-based object recognition and segmentation in cluttered scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1584–1601, 2006.

[70] Mur-Artal, R., Montiel, J., and Tardos, J. D., "ORB-SLAM: a versatile and accurate monocular SLAM system," *arXiv preprint arXiv:1502.00956*, 2015.

[71] Murray, R. M., Li, Z., Sastry, S. S., and Sastry, S. S., *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[72] Nemra, A. and Aouf, N., "Robust airborne 3D visual simultaneous localization and mapping with observability and consistency analysis," *Journal of Intelligent and Robotic Systems*, vol. 55, no. 4, pp. 345–376, 2009.

[73] Nevado, M. M., Garcia-Bermejo, J. G., and Casanova, E. Z., "Obtaining 3D models of indoor environments with a mobile robot by estimating local surface directions," *Robotics and Autonomous Systems*, vol. 48, no. 2, pp. 131–143, 2004.

[74] Ng, A. Y., Jordan, M. I., Weiss, Y., and others, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[75] Okorn, B., Xiong, X., Akinci, B., and Huber, D., "Toward automated modeling of floor plans," in *Symposium on 3D Data Processing, Visualization and Transmission*, vol. 2, 2010.

[76] Perera, L. D. L. and Nettleton, E., "On the nonlinear observability and the information form of the slam problem," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2061–2068, 2009.

[77] Pollefeys, M., Nistér, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.-J., Merrell, P., and others, "Detailed real-time urban 3D reconstruction from video," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 143–167, 2008.

[78] Rabbani, T. and Van Den Heuvel, F., "Efficient hough transform for automatic detection of cylinders in point clouds," *ISPRS WG III/3, III/4*, vol. 3, pp. 60–65, 2005.

[79] Rao, S. R., Tron, R., Vidal, R., and Ma, Y., "Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[80] Rätsch, G., Onoda, T., and Müller, K.-R., "Soft margins for adaboost," *Machine learning*, vol. 42, no. 3, pp. 287–320, 2001.

[81] Roussillon, C., Gonzalez, A., Solà, J., Codol, J.-M., Mansard, N., Lacroix, S., and Devy, M., "RT-SLAM: a generic and real-time visual SLAM implementation," in *Computer Vision Systems*, pp. 31–40, Springer, 2011.

[82] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G., "ORB: an efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision*, pp. 2564–2571, IEEE, 2011.

[83] Rusu, R. B., Blodow, N., and Beetz, M., "Fast point feature histograms (FPFH) for 3D registration," in *IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[84] Rusu, R. B. and Cousins, S., "3D is here: Point cloud library (pcl)," in *IEEE International Conference on Robotics and Automation*, pp. 1–4, IEEE, 2011.

[85] Schnabel, R., Wahl, R., and Klein, R., "Efficient RANSAC for point-cloud shape detection," in *Computer graphics forum*, vol. 26, pp. 214–226, Wiley Online Library, 2007.

[86] Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R., "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *IEEE Conference on Computer vision and pattern recognition*, vol. 1, pp. 519–528, IEEE, 2006.

[87] Shi, J. and Tomasi, C., "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[88] Sola, J., Vidal-Calleja, T., Civera, J., and Montiel, J. M. M., "Impact of landmark parametrization on monocular EKF-SLAM with points and lines," *International Journal of Computer Vision*, vol. 97, no. 3, pp. 339–368, 2012.

[89] Soltanolkotabi, M., Candes, E. J., and others, "A geometric analysis of subspace clustering with outliers," *The Annals of Statistics*, vol. 40, no. 4, pp. 2195–2238, 2012.

[90] Southall, B., Buxton, B. F., and Marchant, J. A., "Controllability and observability: Tools for Kalman filter design.," in *British Machine Vision Conference*, pp. 1–10, 1998.

[91] Stamos, I. and Allen, P., "3-D model construction using range and image data," in *IEEE Conference on Computer vision and pattern recognition*, vol. 1, pp. 531–536, IEEE, 2000.

[92] Strasdat, H., Montiel, J. M., and Davison, A. J., "Visual SLAM: why filter?," *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[93] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D., "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580, IEEE, 2012.

[94] Sundar, H., Silver, D., Gagvani, N., and Dickinson, S., "Skeleton based shape matching and retrieval," in *International Conference on Shape Modeling*, pp. 130–139, 2003.

[95] Tangelder, J. W. and Veltkamp, R. C., "A survey of content based 3D shape retrieval methods," *Multimedia tools and applications*, vol. 39, no. 3, pp. 441–471, 2008.

[96] Tarsha-Kurdi, F., Landes, T., and Grussenmeyer, P., "Hough-transform and extended RANSAC algorithms for automatic detection of 3D building roof planes from LiDAR data," in *ISPRS Workshop on Laser Scanning*, vol. 36, pp. 407–412, 2007.

[97] Tarsha-Kurdi, F., Landes, T., and Grussenmeyer, P., "Extended RANSAC algorithm for automatic detection of building roof planes from LiDAR data," *The photogrammetric journal of Finland*, vol. 21, no. 1, pp. 97–109, 2008.

[98] Thrun, S., Martin, C., Liu, Y., Hähnel, D., Emery-Montemerlo, R., Chakrabarti, D., and Burgard, W., "A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 433–443, 2004.

[99] Torr, P. H. and Zisserman, A., "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[100] Triebel, R., Burgard, W., and Dellaert, F., "Using hierarchical EM to extract planes from 3d range scans," in *IEEE International Conference on Robotics and Automation*, pp. 4437–4442, IEEE, 2005.

[101] Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W., "Bundle adjustmenta modern synthesis," in *Vision algorithms: theory and practice*, pp. 298–372, Springer, 2000.

[102] Tung, T. and Schmitt, F., "Augmented reeb graphs for content-based retrieval of 3D mesh models," in *Shape Modeling Applications*, pp. 157–166, 2004.

[103] Vedaldi, A., Jin, H., Favaro, P., and Soatto, S., "KALMANSAC: Robust filtering by consensus," in *IEEE International Conference on Computer Vision*, pp. 633–640, 2005.

[104] Vidal, R., "Subspace clustering," *Signal Processing Magazine, IEEE*, vol. 28, no. 2, pp. 52–68, 2011.

[105] Vidal, R., Ma, Y., and Sastry, S., "Generalized principal component analysis (GPCA)," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1945–1959, 2005.

[106] Vidal-Calleja, T., Bryson, M., Sukkarieh, S., Sanfeliu, A., and Andrade-Cetto, J., "On the observability of bearing-only SLAM," in *IEEE International Conference on Robotics and Automation*, pp. 4114–4119, 2007.

[107] Williams, S., Indelman, V., Kaess, M., Roberts, R., Leonard, J. J., and Dellaert, F., "Concurrent filtering and smoothing: A parallel architecture for real-time navigation and full smoothing," *The International Journal of Robotics Research*, vol. 33, no. 12, pp. 1544–1568, 2014.

[108] Xiong, X., Adan, A., Akinci, B., and Huber, D., "Automatic creation of semantically rich 3D building models from laser scanner data," *Automation in Construction*, vol. 31, pp. 325–337, 2013.

[109] Yang, A. Y., Wright, J., Ma, Y., and Sastry, S. S., "Unsupervised segmentation of natural images via lossy data compression," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 212–225, 2008.

[110] Zaharia, T. and Preteux, F. J., "Hough transform-based 3D mesh retrieval," in *International Symposium Optical Science and Technology*, pp. 175–185, 2001.

[111] Zhang, G., Karasev, P., Brilakis, I., and Vela, P., "A sparsity-inducing optimization algorithm for the extraction of planar structures in noisy point-cloud data," in *Computing in Civil Engineering*, pp. 317–324, ASCE, 2012.

[112] Zhang, G., Vela, P., and Brilakis, I., "Detecting, fitting, and classifying surface primitives for infrastructure point cloud data," in *Computing in Civil Engineering*, pp. 589–596, ASCE, 2013.

[113] Zhang, G., Vela, P., and Brilakis, I., "Automatic generation of as-built geometric civil infrastructure models from point cloud data," in *Computing in Civil and Building Engineering*, pp. 406–413, ASCE, 2014.

[114] Zhang, G., Kontitsis, M., Filipe, N., Tsiotras, P., and Vela, P. A., "Cooperative relative navigation for space rendezvous and proximity operations using controlled active vision," *Journal of Field Robotics*, 2015.

[115] Zhang, G. and Vela, P. A., "Good features to track for visual SLAM," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[116] Zhang, G. and Vela, P. A., "Optimally observable and minimal cardinality monocular SLAM," in *IEEE International Conference on Robotics and Automation*, pp. 5211–5218, 2015.

[117] Zhang, G., Vela, P. A., Karasev, P., and Brilakis, I., "A sparsity-inducing optimization-based algorithm for planar patches extraction from noisy point-cloud data," *Computer-Aided Civil and Infrastructure Engineering*, vol. 30, no. 2, pp. 85–102, 2015.

[118] Zhang, G., Vela, P. A., Tsiotras, P., and Cho, D.-M., "Efficient closed-loop detection and pose estimation for vision-only relative localization in space with a cooperative target," in *AIAA Space and Astronautics Forum and Exposition, San Diego, CA*, 2014.

[119] Zhang, G., Qin, X., Hua, W., Wong, T.-T., Heng, P.-A., and Bao, H., "Robust metric reconstruction from challenging video sequences," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[120] Zhang, T., Szlam, A., Wang, Y., and Lerman, G., "Hybrid linear modeling via local best-fit flats," *International journal of computer vision*, vol. 100, no. 3, pp. 217–240, 2012.

[121] Zhou, K., Doyle, J. C., Glover, K., and others, *Robust and optimal control*, vol. 40. Prentice hall New Jersey, 1996.