Technische Universität München

TUM School of Computation, Information and Technology

Tracking and Mapping with Structural Regularities

Yan Li

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology

der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Nils Thuerey

Prüfende der Dissertation:

1. Priv.-Doz. Dr. Federico Tombari
2. Prof. Luca Carlone
3. Prof. Juan Domingo Tardós

Die Dissertation wurde am 11.12.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 02.05.2024 angenommen.

Yan Li
Tracking and Mapping
with Structural Regularities
Dissertation, Version 1.0

Technische Universität München
Fakultät für Informatik
Lehrstuhl für Informatikanwendungen in der Medizin
Boltzmannstraße 3
85748 and Garching bei München

Abstract

Real-time tracking and mapping approaches support intelligent agents such as robots, AR/VR devices, and autonomous driving vehicles to interact with unknown environments. Visual-based tracking methods aim to estimate the six degrees of freedom (DoF) camera poses, while mapping algorithms aim at reconstructing unknown environments into sparse or dense models.

Commonly, camera poses tend to drift when errors accumulate during tracking processes. To limit the increase of pose errors, solutions, including local bundle adjustment, sliding window optimization, marginalization, and loop closure, are proposed to use correspondences to build co-visibility graphs. Those approaches achieve robust tracking performance after using optimization modules. However, the co-visibility strategy based on point features still needs to improve in low/non-textured regions since only some features are extracted during the tracking process. Furthermore, lines and planes, especially in indoor scenes, are explored under the co-visibility architecture to compensate for the reduction in the number of point correspondences. Given more features, the robustness of trackers will be continually improved. However, the shortness of co-visibility graphs that mainly rely on overlaps needs to be addressed, which leads to shorter constraint edges in the graphs.

Instead of only using re-projection errors of point-line-plane correspondences under the co-visibility graph pipeline, more structure information, such as Vanishing Point and Manhattan/Atalanta World Assumptions, is leveraged into our pose estimation modules by assuming scenes have some perpendicular and orthogonal cues. Since these structural cues are loosely organized by basic landmarks rather than represented as minimal parameterizations, it is difficult to use them in optimization modules. Even though they are often used in visual odometry systems, keeping these structural landmarks correct during the tracking process remains an open challenge.

How to exploit structural regularities in pose estimation and scene reconstruction is the most critical exploration goal of this dissertation. The methods presented here are incorporated into a completed tracking and mapping system. Specifically, our tracking module uses the structural regularities in the front-end and back-end modules. Moreover, we propose a new type of graph architecture, the Extensibility Graph, which is incorporated with co-visibility graphs to make up for the shortcomings of over-reliance on visual overlaps of traditional co-visibility ones.

Zusammenfassung

Echtzeit-Tracking- und Mapping-Ansätze helfen intelligenten Agenten wie Robotern, AR/VR-Geräten und autonomen Fahrzeugen, mit unbekannten Umgebungen zu interagieren. Visuell basierte Tracking-Methoden zielen darauf ab, die sechs Freiheitsgrade (DoF) von Kamerapositionen abzuschätzen, während Mapping-Algorithmen darauf abzielen, unbekannte Umgebungen in spärliche oder dichte Modelle zu rekonstruieren.

Im Allgemeinen neigen Kamerapositionen dazu, zu driften, da sich bei den Tracking-Vorgängen Fehler anhäufen. Um die Zunahme von Posenfehlern zu begrenzen, werden Lösungen einschließlich lokaler Bündelanpassung, Schiebefensteroptimierung, Marginalisierung und Schleifenschließung vorgeschlagen, um Korrespondenzen zur Erstellung von Ko-Sichtbarkeitsdiagrammen zu verwenden. Diese Ansätze erzielen nach Verwendung von Optimierungsmodulen eine robuste Tracking-Leistung. Allerdings muss die auf Punktmerkmalen basierende Co-Sichtbarkeitsstrategie in Regionen mit spärlicher/nicht texturierter Struktur noch verbessert werden, da während des Tracking-Prozesses nur einige Merkmale extrahiert werden. Darüber hinaus werden Linien und Ebenen, insbesondere in Innenszenen, im Rahmen der Co-Visibility-Architektur untersucht, um die Verringerung der Anzahl der Punktkorrespondenzen auszugleichen. Durch weitere Features wird die Robustheit der Tracker kontinuierlich verbessert. Ko-Sichtbarkeitsdiagramme basieren jedoch hauptsächlich auf Überlappungen, was zu kürzeren Einschränkungskanten in den Diagrammen führt.

Anstatt nur Reprojektionsfehler von Punkt-Linie-Ebene-Korrespondenzen als Teil der Co-Visibility Graph-Pipeline zu verwenden, werden mehr Strukturinformationen in unsere Posenschätzungsmodule integriert, indem davon ausgegangen wird, dass Szenen senkrechte und orthogonale Hinweise haben. Da diese strukturellen Hinweise lose nach grundlegenden Orientierungspunkten organisiert sind, ist es schwierig, sie in Optimierungsmodulen zu verwenden. Obwohl sie häufig in visuellen Odometriesystemen verwendet werden, bleibt die korrekte Beibehaltung dieser strukturellen Orientierungspunkte während des Verfolgungsprozesses eine offene Herausforderung.

Die Ausnutzung struktureller Gesetzmäßigkeiten bei der Posenschätzung und Szenenrekonstruktion ist das Hauptforschungsziel dieser Dissertation. Die hier vorgestellten Methoden werden in ein fertiges Tracking- und Mapping-System integriert. Konkret nutzt unser Tracking-Modul die Strukturgesetze in den Front-End- und Back-End-Modulen. Darüber hinaus schlagen wir eine neue Art von Grapharchitektur vor, den Erweiterbarkeitsgraphen, der in Co-Sichtbarkeitsgraphen integriert ist, um die Mängel einer übermäßigen Abhängigkeit von der visuellen Überlappung herkömmlicher Co-Sichtbarkeitsgraphen zu kompensieren.

Acknowledgments

Life is like a white colt bolting past a crevice. You never feel it until it flies by like a flash. Looking back on my entire PhD career, I feel fortunate and grateful for the support and encouragement I received from my supervisors, mentors, friends, and family along the journey. Without your strong support, I could not find my favorite research direction.

First and foremost, I am extremely grateful to my supervisors, Federico and Nassir, for their invaluable advice, continuous support, and patience during my PhD study. Their immense knowledge and plentiful experience have encouraged me throughout my academic research and daily life. I would also like to thank my mentor Keisuke and Niko, for their excellent technical support on my study. CNN-SLAM is the first bridge between me and the CAMP chair, and Niko gives me lots of valuable advice in coding and writing. Our research group has given me more than enough freedom to do my interesting research all the time.

I would like to thank all the members of the CAMP chair. Their kind help and support have made my study and life in the Deutschland a wonderful time. Thanks to you guys, Yida, Shun-Cheng, Fabi, Janhana, Helisa, Tolga, and Yan Di. Those all-nighters before submitting the paper, camping trips to the beautiful rivers and lakes, and, of course, great music and Munich beers are my precious memories.

Thanks to those friends in the SLAM and deep learning communities whom we have never met. It is precisely because of the existence of excellent open-source projects such as ORB-SLAM, DSO, VINS, KIMERA, etc., that our field can develop rapidly. At the same time, this open-source spirit also inspires me to release stable and more professional open-source projects.

Finally, I would like to express my gratitude to my parents, Mr. Jiaen Li and Ms. Mei Song, for teaching me to look up to the stars and keep my feet on the ground with your practical actions. I would like to thank my wife, Ms. Xiaomeng Xu. My heart flips whenever I think of the first time I met you ten years ago. With their tremendous understanding and encouragement in the past few years, I have the courage to pursue higher academic ideals.

Contents

II Methodology

III Conclusion and Future Work

IV Appendix

Chronological List of Authored and Co-authored Publications

2023

[1] **Y. Li**, Z. Guo, Z. Yang, Y. Sun, L. Zhao, F. Tombari. "Open-Structure: a Structural Benchmark Dataset for SLAM Algorithms". In: *arXiv 2310.10931*, 2023.

[2] Z. Yang*, **Y. Li***, Y. Sun, L. Lin, J. Zhu. "wMPS-SLAM: An Online and Accurate Monocular Visual-wMPS SLAM System". In: *IEEE Transactions on Instrumentation and Measurement (TIM)*, 2023.

[3] Z. Yang, **Y. Li**, L. Lin, Y. Sun, J. Zhu. "Tightly-coupled fusion of iGPS measurements in optimization-based visual SLAM". In: *Optics Express*, 2023.

2022

[4] **Y. Li** and F. Tombari. "E-Graph: Minimal Solution for Rigid Rotation with Extensibility Graphs". In: *European Conference on Computer Vision (ECCV)*, 2022.

2021

[5] **Y. Li**, R. Yunus, N. Brasch, N. Navab, F. Tombari. "RGB-D SLAM with Structural Regularities." In: *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[6] R. Yunus, **Y. Li**, F. Tombari. “ManhattanSLAM: Robust Planar Tracking and Mapping Leveraging Mixture of Manhattan Frames.” In: *International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[7] H. Du, **Y. Li**, Y. Sun, J. Zhu, F. Tombari. "SRH-Net: Stacked Recurrent Hourglass Network for Stereo Matching." In: *IEEE Robotics and Automation Letters (RA-L)*, 2021.

2020

[8] **Y. Li***, N. Brasch*, Y. Wang, N. Navab, F. Tombari. "Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments." In: *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[9] X. Li*, **Y. Li***, EP. Örnek, J. Lin, F. Tombari. "Co-planar parametrization for stereo-SLAM and visual-inertial odometry" In: *IEEE Robotics and Automation Letters (RA-L)*, 2020.

Notation

u → This font is used for quantities that are real scalars

$\mathbf{u}$ → This font is used for quantities that are real column vectors

$\mathbf{U}$ → This font is used for quantities that are real matrices

$\mathbf{P}_w$ → This font is used for quantities that are 3D point landmarks in the world coordinate

$\mathbf{p}^i$ → This font is used for quantities that are 2D point features on the image plane

$\mathbf{L}_w$ → This font is used for quantities that are Euclidean 3D line landmarks in the world coordinate

$\mathbf{l}^i$ → This font is used for quantities that are 2D line segments on the image plane

$\mathcal{L}_w$ → This font is used for quantities that are Plücker 3D line landmarks in the world coordinate

$\mathcal{O}_w$ → This font is used for quantities that are Orthonormal 3D line landmarks in the world coordinate

$\mathbf{R}_{w,c_i}$ → A rotation matrix which rotates from the i^{th} camera coodinate to the world coordinate

$\mathbf{T}_{w,c_i}$ → A transformation matrix from the i^{th} camera coodinate to the world coordinate

$SO(3)$ → The special orthogonal group

$SE(3)$ → The special Euclidean group

$\mathfrak{so}(3)$ → The Lie algebra associated with $SO(3)$

$\mathfrak{se}(3)$ → The Lie algebra associated with $SE(3)$

$\mathbb{R}^{m \times n}$ → The vectorspace of real $m \times n$ matrices

$[\cdot]_\times$ → The skew-symmetric operation

$P(\mathbf{u})$ → This font is used for time-invariant system quantities

$P(\mathbf{v}|\mathbf{u})$ → The probability density of $\mathbf{v}$ given $\mathbf{u}$

Part I

Introduction

1 Introduction

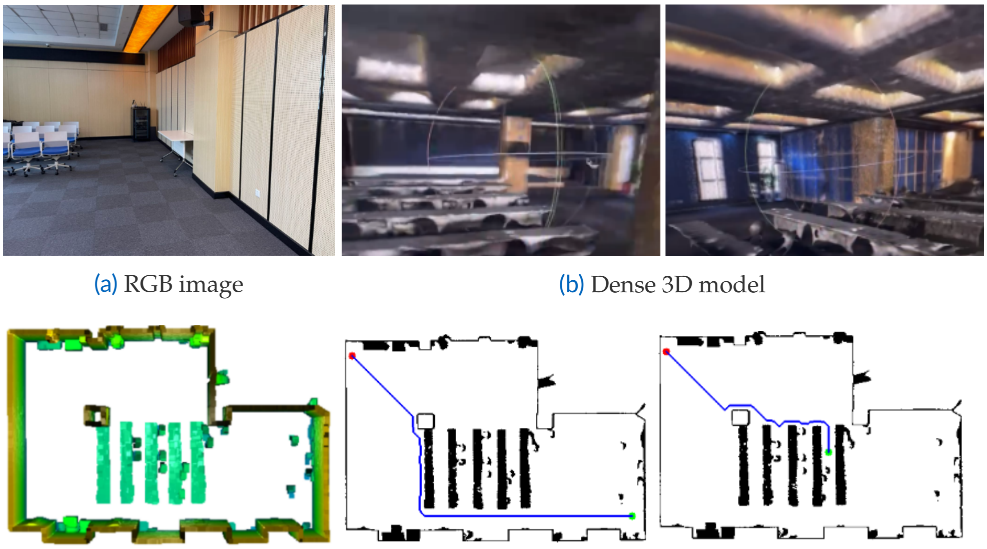
In the beginning chapter of the dissertation, Section 1.1 briefly introduces the motivations and background of localization and reconstruction tasks in 3D unknown environments. Based on a sequence of visual images as inputs, the goal of a general incremental tracking and mapping estimation system is introduced in Section 1.2. The special requirements for SLAM systems in man-made environments are described in Section 1.3. Finally, the structure of this dissertation is listed in Section 1.4.

1.1 Background

Organisms conduct positioning and navigation activities intentionally or unintentionally by searching for clues in unknown environments. Depending on those key patterns of scenes, the direction to a destination will become apparent in motion. In ancient times, people determined direction by observing natural features such as the sun, stars, and landmarks. For example, people use the position of the Big Dipper to determine direction when sailing at sea. Other natural phenomena, such as the flow of rivers, can help navigate land travel. Rivers always flow from high to low so that the general direction can be determined based on the river's flow and the sun's position.

With the development of science and technology, localization in marine scenarios was deduced using tools to measure the altitudes of stars, such as sextants and mariner's astrolabes. In the 20th century, aerospace technology achieved impressive development, and the global positioning system (GPS) supported by satellites went into daily life, which enabled people to conduct precise positioning and navigation more freely and conveniently. However, auxiliary algorithms are still needed to navigate agents in environments without GPS. Alternatives to these navigation systems include odometry-based mechanisms and inertial measurement unit (IMU) setups, which can provide tracking results directly and are commonly found in modern intelligent robots and vehicles. Nevertheless, those alternatives can only support robust tracking services for a short time.

When navigating in an unknown environment as shown in Figure 1.1, the mapping of that environment and the localization within the reconstructed map have proven to be interdependent. Furthermore, an accurate map makes localization robust, while accurate trajectories also help improve the quality of the map. Inspired by the iterative process, a complex solution is considered: simultaneous localization and mapping (SLAM), the concept



(a) RGB image

(b) Dense 3D model

(c) Height map (removing ceiling and floor)

(d) Navigation in the map. The start and end points are red and green respectively

Figure 1.1 An example of SLAM applications. Starting from a sequence of images, dense maps of unknown environments are reconstructed, which can be used for navigation applications of robots.

first proposed in the 1990s [10], introduced in the scientific community in the 1980s [11] and originated in the early stages.

1.2 Simultaneous Localization and Mapping

As its name suggests, SLAM aims to reconstruct unknown environments and position agents in the environments. Based on the outputs of SLAM systems, intelligent setups will have the fundamental ability to interact with those environments without needing prior knowledge of the location. Therefore, the method is of great importance to robots operating in unknown scenes, especially where GPS is not reliable anymore.

Given a sequence of images as illustrated in Figure 1.2, spatial patterns (features), such as points and lines, are extracted from each image. By matching the same patterns (correspondences) measured by different views, relative motion (rotation and translation) between two positions can be inferred. Meanwhile, those same patterns existing in different views can also be used in building 3D landmarks that are representations of the scenes. In the further process, reconstructing landmarks of environments and tracking new views are interdependent, and the process can be called Visual Odometry (VO). Even though there are different strategies for mapping and localization tasks, the core ideas are designed similarly. Since the performances of VO systems are sensitive to challenging scenes and motions, there are two widely used strategies, including new types of measurements and well-designed

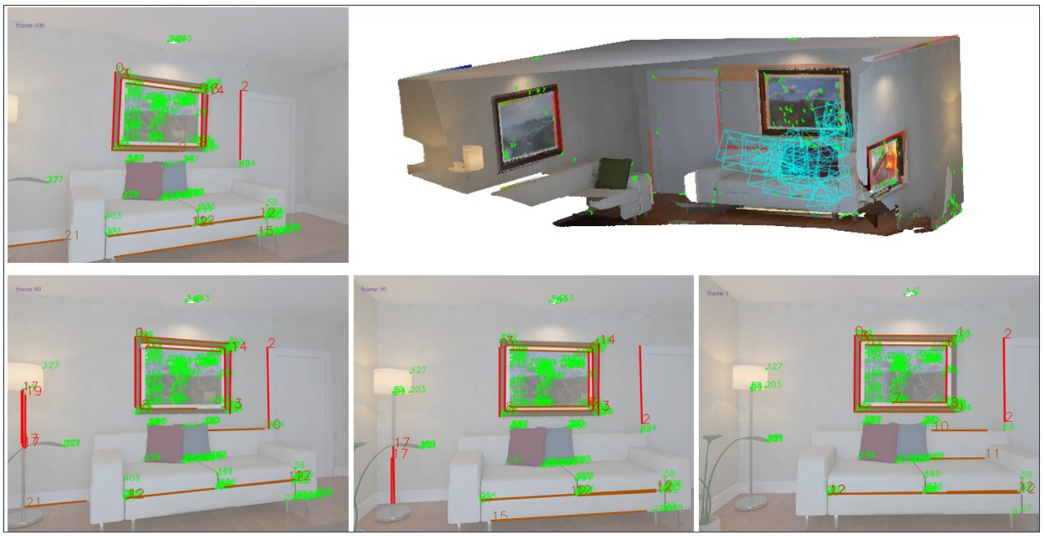


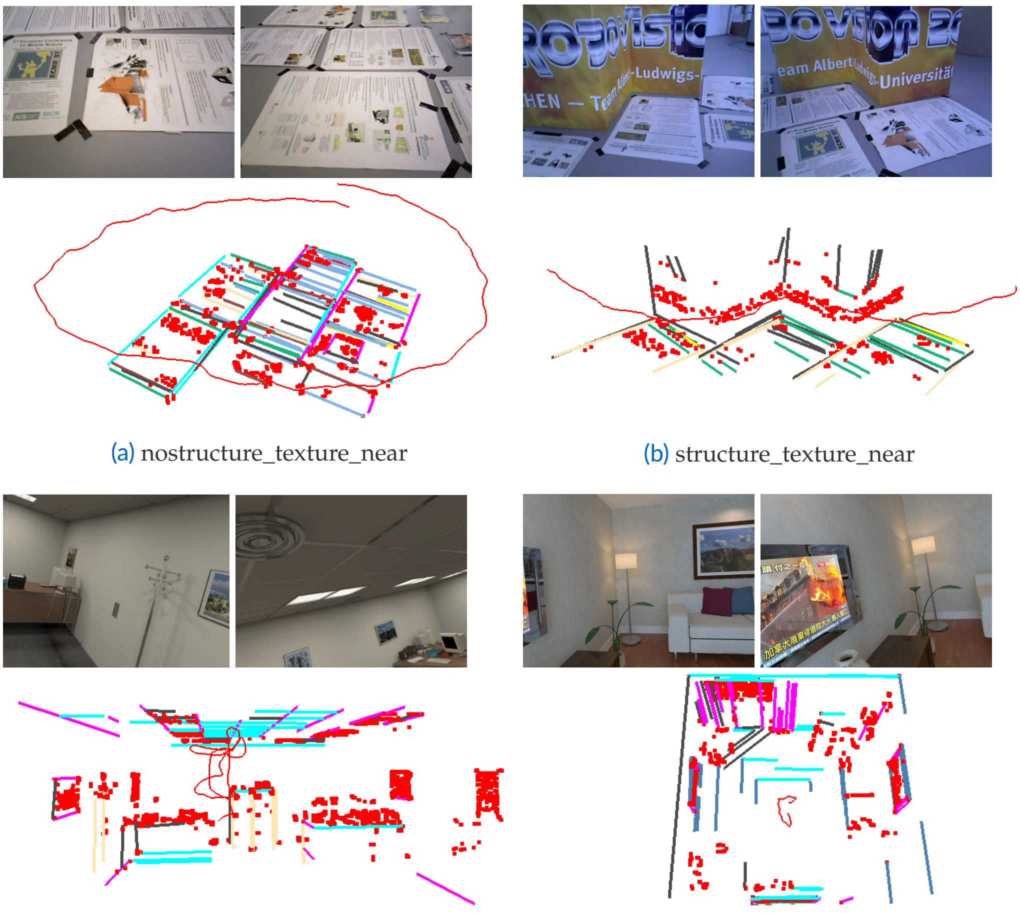
Figure 1.2 Illustration of 2D RGB images, point (green) and line (red) measurements, and trajectories and the 3D dense reconstruction model on the ICL-NUIM dataset [12].

refinement modules, to improve the robustness of systems. Generally, both of them can be found in impressive SLAM systems.

More reliable tracking and mapping performances could be expected when more powerful sensors enhance the process. Specifically, Stereo cameras provide real scale to the map and make the tracking process more robust. RGB-D sensors help depth maps generate dense 3D models easily on GPUs or CPUs. Moreover, visual-inertial setups are widely used in the improvement process, where IMU sensors can provide rotation and translation measurements directly based on post-processing steps.

Another direction for achieving robust tracking performances is to explore refinement, optimization, and drift-removing strategies for VO systems. The bundle adjustment approach is popularly used to refine a single relative pose based on the observation relationships between landmarks and measurements. Some solutions, including local bundle adjustment [13, 14], sliding window [15], marginalization [16, 17], and loop closure [18, 13], are widely used in camera tracking processes, and the idea behind of those solutions is using correspondences to build co-visibility factor graphs for structure and motion optimization.

Motivated by these refinement and optimization solutions, this dissertation focuses on incremental tracking and mapping in structural scenes, such as houses, office buildings, and parking lots, where structural scenes as illustrated in Section 1.3 provide new challenges and chances for SLAM systems.



(a) nostructure_texture_near

(b) structure_texture_near

(c) office_room_0

(d) living_room_0

Figure 1.3 Image views and 3D point-line sparse models of indoor scenes. Parallel 3D lines are in the same color. The process of structural model generation is introduced in Section 3.2.1.

1.3 Tracking and Mapping in Indoor Environements

Structural environments can be found in most artificial spaces, including general living scenarios, buildings, parking lots, etc, as shown in Figure 1.3. Compared with wild environments, indoor scenes have several apparent differences that provide new challenges to SLAM systems. For example, low/non-textured scenarios are common in such scenes (see Figure 1.3), which means that there is no guarantee to detect sufficient point features to support robust visual-based tracking performance.

From the perspectives of industry products, tracking and mapping in those scenes are involved in augmented/virtual reality, service robots, and autonomous driving. Therefore, achieving robust and accurate tracking, mapping, and understanding performance in structural scenes is of great value in both industry and academia. Although those SLAM systems have achieved existing results in general scenarios, in vertical domains, such as indoor environments, their performance shows some deterioration phenomena.

Before discussing indoor SLAM designing, we will first analyze indoor scene characteristics and then introduce how to leverage those characters in tracking and reconstruction tasks. Regular geometric elements are commonly detected in man-made scenes as lines and planes are used to build surfaces of floors, ceilings, walls, and objects. When we try to analyze the relationship of those lines (or planes), it is easy to find some parallel and perpendicular geometric relationships. For example, ceilings are generally parallel to floors, while walls are perpendicular to ceilings and floors. Not only does the architectural structure of the buildings have the above characteristics, but the furniture, office supplies, and other decorations also show the above characteristics to a greater or lesser extent.

How to take advantage of those new patterns is a critical problem for SLAM systems designed for indoor scenarios. Based on those different types of features, like lines and planes, a naive and direct idea is to extend point-based SLAM systems to read multiple features since multi-feature tracking can improve the robustness of general systems in indoor scenes. The direct idea improves the systems' robustness by leveraging more re-projection constraints in the front-end and back-end. Nevertheless, the journey of exploration goes beyond that. For example, scenes in Figure 1.3 have several sets of parallel lines, and some of those sets are perpendicular. Those geometric features and relationships can be used to reduce computation and improve accuracy in tracking and mapping simultaneously. When we redefine those new patterns and constraints in the feature detection, matching, reconstruction, minimal representation, and factor graph optimization modules, we will slowly uncover this field, which is the main topic of this dissertation.

1.4 Structure of the Dissertation

This chapter, **Chapter 1**, briefly introduces SLAM topics' motivation and illustrates the process from images to relative motions and maps. Given the introduced research background on the community and our interesting field, this section briefly outlines the structure of this cumulative thesis, starting from Chapter 2 to 8.

Chapter 2: Fundamental Theories. This chapter introduces the basic theory of multiple view geometry, including correspondence detection and initial pose estimation, in Section 2.2. Then, the optimization theory containing factor graph construction and solvers for non-linear least squares problems is introduced in Section 2.3. Finally, Section 2.4 introduces basic theories in deep neural networks, such as basic operations and the common architectures built from these operators.

Chapter 3: Tracking and Mapping in Structural Scenes. This chapter continues to direct our attention to the critical tracking and mapping involvements in structural environments, the main topic we aim to discuss in the following chapters. Furthermore, open challenges in indoor environments and our contributions are summarized in Section 3.1.

Chapter 4: Structure-slam: Low-drift monocular slam in indoor environments. Yanyan Li*, Nikolas Brasch*, Yida Wang, Nassir Navab and Federico Tombari; In. RA-L 2020 [8].

Chapter 5: RGB-D SLAM with structural regularities. Yanyan Li, Raza Yunus, Nikolas Brasch, Nassir Navab and Federico Tombari; In. ICRA 2021 [5].

Chapter 6, Co-Planar Parametrization for Stereo-SLAM and Visual-Inertial Odometry. Xin Li*, Yanyan Li*, Evin Pınar Örnek, Jinlong Lin, and Federico Tombari; In. RA-L 2020 [9].

Chapter 7, E-Graph: Minimal Solution for Rigid Rotation with Extensibility Graphs. Yanyan Li and Federico Tombari. In. ECCV2022 [4].

Chapter 8: Conclusion and Future work. This chapter provides conclusions in Section 8.1 and suggestions for Future work in Section 8.2.

Fundamental Theories 2

Fundamental theories related to tracking and mapping tasks are introduced in this chapter. First, Section 2.1 briefly outlines the development history of SLAM topics by revisiting the definition of the problem. Then, we introduce geometry-based approaches, including feature detection, matching, triangulation, pose initialization, and optimization, in Section 2.2, where those modules can build a complete visual odometry pipeline to estimate camera poses and build sparse maps simultaneously. Given initial poses and landmarks from visual odometry, another critical fundamental theory refers to optimization in Section 2.3. To compensate for traditional geometry approaches' limitations, we introduce basic operators and architectures, including multi-layer perceptron and convolutional neural blocks widely used in semantic segmentation, depth prediction, dense mapping, and scene completion networks in Section 2.4.

2.1 Problem Definition of SLAM

Standing at the beginning of this discipline decades ago, no one knew how to simultaneously track camera poses and reconstruct unknown scenes by taking several photos. With the community's joint efforts for such a long time, the original goal has been achieved by SLAM systems with different strategies. However, looking back, this vast tree's growth vein is clearer. Let us start with the definition of this problem and gradually expand the complexity of this domain.

For the discussion about the definition of SLAM topics, there are many different perspectives on how to present the core idea of the SLAM problem, however using probability representation is undoubtedly one of the most elegant ways.

An image sequence with n camera poses ($\mathcal{T} = \{\mathbf{T}_1, \dots, \mathbf{T}_n\}$) is recorded in an environment with m landmarks ($\mathcal{P} = \{\mathbf{P}_1, \dots, \mathbf{P}_m\}$). During the tracking process, measurements ($\mathcal{Q} = \{\mathbf{p}_1, \dots, \mathbf{p}_K\}$) are used to build a conditional probabilistic distribution model as follows:

$$P(\mathcal{T}, \mathcal{P} | \mathcal{Q}) \tag{2.1}$$

which shows how to update the camera poses and scene representations based on the batch of measurements $\mathcal{Q}$. The problem can further be transformed into a likelihood probabilistic model via the Bayesian probability operation

$$\begin{aligned} P(\mathcal{T},\mathcal{P}|\mathcal{Q}) &= \frac{P(\mathcal{Q}|\mathcal{T},\mathcal{P})P(\mathcal{T},\mathcal{P})}{P(\mathcal{Q})} \\ &\propto P(\mathcal{Q}|\mathcal{T},\mathcal{P})P(\mathcal{T},\mathcal{P}) \end{aligned} \tag{2.2}$$

here $P(\mathcal{Q}|\mathcal{T},\mathcal{P})$ is called Likelihood, which can be understood as the probability of producing those measurements under the 3D environments and camera poses. $P(\mathcal{T},\mathcal{P})$ is called Prior, which can be understood as the probability that the robot is in the current environment and pose. Compared to the posterior probability introduced in the formula 2.1, $P(\mathcal{Q}|\mathcal{T},\mathcal{P})$ provides a better perspective for solving incremental tracking and mapping topics.

Therefore, we can intuitively feel that the process of calculating the optimal pose and 3D landmarks is the process of finding the pose and landmark information that can best produce the current observation, which can be described by the following formula

$$(\mathcal{T},\mathcal{P})^* = \arg\max_{\mathcal{T},\mathcal{P}} P(\mathcal{Q}|\mathcal{T},\mathcal{P}). \tag{2.3}$$

For a measurement of the measurement batch $\mathcal{Q}$, $\mathbf{p}_{i,j}$ shows that the $\mathbf{P}_j$ is observed by $\mathbf{T}_j$, which can be represented based on the measurement model $f(\mathbf{T}_i,\mathbf{P}_j)+\mathbf{n}_{i,j}$, where $\mathbf{n}_{i,j}\in\mathcal{N}(\mathbf{0},\sum)$ means that the noise $\mathbf{n}_{i,j}$ obeys the zero-mean normal distribution. Then the probability distribution of the conditional probability $P(\mathbf{p}_{i,j})$ still obeys the Gaussian distribution in $\mathcal{N}(f(\mathbf{T}_i,\mathbf{P}_j),\sum)$, which can be represented as

$$P(\mathbf{p}_{i,j}) = \frac{1}{\sqrt{(2\pi)^N \det(\sum)}} \exp\left(-\frac{1}{2}(\mathbf{p}_{i,j}-f(\mathbf{T}_i,\mathbf{P}_j))^T \sum{}^{-1} (\mathbf{p}_{i,j}-f(\mathbf{T}_i,\mathbf{P}_j))\right) \tag{2.4}$$

here N shows the dimension of $\mathbf{p}_{i,j}$. Via the negative log operation, $-\ln(\cdot)$, the Equation 2.4 can be equivalently written as

$$\begin{aligned} (\mathbf{T}_i,\mathbf{P}_j)^* &= \arg\min_{\mathbf{T}_i,\mathbf{P}_j} (-\ln(P(\mathbf{p}_{i,j}|\mathbf{T}_i,\mathbf{P}_j))) \\ &= \arg\min_{\mathbf{T}_i,\mathbf{P}_j} \left(\frac{1}{2}\ln\left((2\pi)^N \det(\textstyle\sum)\right) + \frac{1}{2}(\mathbf{p}_{i,j}-f(\mathbf{T}_i,\mathbf{P}_j))^T \sum{}^{-1} (\mathbf{p}_{i,j}-f(\mathbf{T}_i,\mathbf{P}_j))\right) \\ &= \arg\min_{\mathbf{T}_i,\mathbf{P}_j} \left((\mathbf{p}_{i,j}-f(\mathbf{T}_i,\mathbf{P}_j))^T \sum{}^{-1} (\mathbf{p}_{i,j}-f(\mathbf{T}_i,\mathbf{P}_j))\right) \end{aligned} \tag{2.5}$$

here $\mathbf{p}_{i,j}-f(\mathbf{T}_i,\mathbf{P}_j)$ is denoted as $\mathbf{e}_z^{i,j}$ and $\sum{}^{-1}$ is denoted as $\mathbf{\Lambda}$. The single observation is written as ${\mathbf{e}_z^{i,j}}^T \Lambda \mathbf{e}_z^{i,j}$. Assuming that the measurements are independent of each other, therefore, Equation 2.3 can be rewritten as

$$P(\mathcal{Q}|\mathcal{T},\mathcal{P}) = \prod_{i,j} P(\mathbf{p}_{i,j}|\mathbf{T}_i,\mathbf{P}_j). \tag{2.6}$$

And the optimized camera poses and landmarks,

$$(\mathcal{T}, \mathcal{P})^* = \arg\min_{\mathbf{T}_i, \mathbf{P}_j} \left(\sum_i \sum_j {\mathbf{e}_z^{i,j}}^T \Lambda \mathbf{e}_z^{i,j} \right) \tag{2.7}$$

where the maximum likelihood problem is transfered to a nonlinear least square optimization probem that is widely used for building optimization modules. As shown in Equation 2.5, the problem mainly involves four parts, including **measurements**, **camera poses**, **scene representations**, and **error convergence models**, which are introduced in following sections.

2.1.1 Measurements

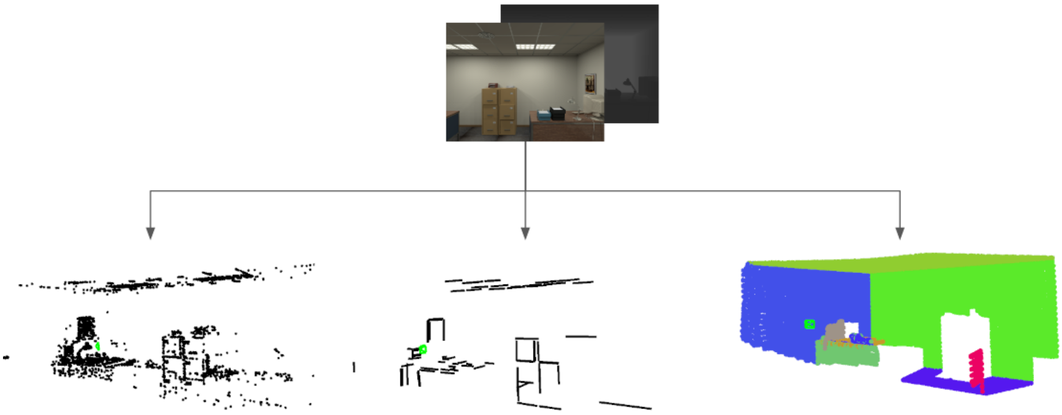


Figure 2.1 Feature reconstructed from a pair of RGB-D images. In the bottom raw, from left to right, there are point, line and plane primitives.

Generally, the method that only extracts distinct points can be called a detector, and the approach that generates abstract information by handcrafting rules to present the difference between 2D measurements is called a descriptor. The first important task is automatically detecting those measurements to achieve video-based tracking and mapping performances. As shown in Figure 2.1, point, line and plane measurements are obtained from a pair of RGB and depth images.

Points. For distinct point detection methods, FODPL [19], proposed in 1987, and Harris [20], proposed in 1988, are essential tools to open the curtain of modern computer vision. Given selected distinct points, the first strategy, the Kanade-Lucas-Tomasi (KLT) approach [21], is proposed to track points by defining a photometric error based on the constant brightness assumption of adjacent frames. Another more robust corner estimation method, FAST [22], was proposed in 2008.

Compared with the color intensity that can only be used to match correspondences in local areas, more abstract information is encoded into descriptor vectors by approaches, such as SIFT [23] (in 1999), SURF [24] (in 2006) and ORB [25] (in 2011), where those complicated vectors are more robust in being invariant to image scaling, translation, and rotation, and even illumination changes compared with color intensity. Therefore, it can support global matching searching strategies by computing the difference between two descriptors. SIFT [23] is one of the most famous traditional local feature point, which identifies candidate keypoints by looking for local maxima and minima in the Difference of Gaussian (DoG) scale-space representation of the image. Then, it refines these candidate keypoints by fitting a model that is invariant to scale, rotation, and affine distortion. Finally, it computes a descriptor for each keypoint by taking a histogram of gradient orientations in the local neighborhood around the keypoint. FAST [22] is a high-speed corner detection approach that is designed for real-time applications. It works by examining contiguous pixels on a circle around a central pixel to determine whether the central pixel is a corner or not. If a sufficient number of contiguous pixels differ greatly in intensity from the central pixel, then the central pixel is classified as a corner. BRIEF [26] is a binary feature descriptor that encodes the gradient information of detected keypoints into binary strings, which samples the intensity values at pairs of points in a local neighborhood surrounding the keypoint and compares them using a binary test. This process generates a binary string for each keypoint that describes its local appearance.

Lines and Planes. Lines and planes are existed widely in man-made environments, which provide more constraints than point features. The line extraction task is also interested in the community at an early stage. Burns et al. [27] proposed an approach to detecting straight lines in 1986. While a linear-time line detector, LSD [28], and a faster line detection method, EDLines [29], were proposed in 2008 and 2011, respectively. The descriptor LBD [30] proposed in 2013 is a widely used algorithm to match line segments between different views. With the development of RGB-D sensors, RANSAC-based methods, like [31], were used in plane detection. Furthermore, Hough transformation [32] was exploited to implement pre-segmentation steps for plane segmentation. Furthermore, a faster plane detection approach [33] based on an agglomerative hierarchical clustering algorithm was proposed in 2014. RANSAC-Based strategies are popular in fitting planes from point clouds by randomly selecting a subset of points that potentially belong to the same plane. AHC [34] start with some initial seed points and expand each region by adding neighboring points that satisfy certain criteria, such as distance and normal angle similarity. These methods compute the normal vectors of each point in a point cloud and accumulate them into bins based on their direction. The bin with the highest vote represents the dominant plane in the scene.

2.1.2 Scene Reconstruction

For scene reconstruction, sparse approaches triangulate those correspondences detected in tracking processes into sparse landmarks, while dense systems aim to use all pixels of 2D images. Methods between sparse and dense reconstruction are called semi-dense, where the number of tracked points is more significant than in sparse methods, but not every pixel can be used in modeling as dense reconstruction approaches are. In addition to visual sparseness, another fundamental principle is that those landmarks are regarded as independent in the sparse formulation. Although the reconstructed maps can be very dense and complete with

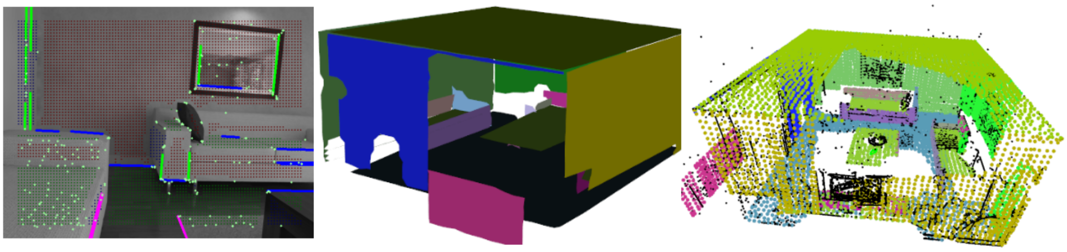


Figure 2.2 Example of dense and sparse models. Left: scene image with segmented planes marked in different colors; middle: dense mesh model based on the planar regions; right: sparse mapping based on the points, lines and planes.

the help of line and plane features, those maps of feature-based systems are still regarded as sparse methods.

Sparse Scene Parametrization. The Euclidean XYZ landmark parametrization is straightforward to represent point landmarks but faces challenges for low parallax correspondences. Inverse depth parametrization [35] uses the ray to go through the anchor frame position and the observed feature. The straightforward representations for line and plane landmarks are Euclidean XYZ endpoints and Hessian parameterization, but both of them have over-parametrization issues in optimization. Bartoli and Sturm [36] proposed Plücker and Orthonormal representations for using lines in triangulation and bundle adjustment. The minimal parametrization proposed in [37] uses azimuth and elevation angles to represent the normal vectors of plane landmarks. For feature-based SLAM systems, landmarks reconstructed incrementally are generally optimized via factor graphs. Therefore, a suitable parameterization method is necessary for landmarks refinement.

Dense Scene Representation. Compared with sparse models, dense reconstruction generally provides more information to support scene understanding tasks and new views rendering. In dense reconstruction, different types of methods are explored for different sensors. For monocular dense reconstruction, the core technology for obtaining dense depth maps of multiple views is per-pixel matching based on the Epipolar algorithm or patch-based matching methods [38]. Similar to those sparse landmarks that can be optimized in a bundle adjustment model, those estimated dense point clouds can also be filtered based on depth filters by using Gaussian Distribution models [39, 40].

For RGB-D sensors, depth maps can be obtained directly. Different from those parametric landmarks, the following introduced methods, the occupancy grid [41] and signed distance function (SDF) [42], build non-parametric environment models. A 3D volume of voxels represents environments, and the value recorded in each cell is a probability of occupancy [41] that will be updated when a new observation comes in. SDF [42] proposed in 1996 represents surface interfaces as zeros. From an SDF representation, two main algorithms render the surface regions, summarized by [43]. The first one is using the marching-cube algorithm [44]. Another strategy is using the raycasting method [45] to avoid visiting areas of the function that are outside the desired view field.

2.1.3 Error Convergence Model

As introduced in Section 2.1.1, a difference between measurements and estimated ones is computed based on obtained poses and landmarks. In this section, we introduce different routes for residuals computation and initial refinement via two directions: direct vs. indirect and filtering-based vs. optimization-based.

Direct vs. Indirect. In Equation 2.2, camera poses are the targets that we want to estimate using different methods. Based on the fed constraints about environments, tracking and mapping systems can be classified as direct and indirect methods. Specifically, the direct strategy makes use of raw sensor values as measurements in a probabilistic model, while an indirect method takes advantage of a part of the pixels with known matching relationships. In short, given two images, **direct methods will estimate the relative pose and depth of each pixel at the same time based on photometric constraints**. However, **indirect approaches first take action to detect correspondences that provide stronger geometric constraints in camera poses and features' depth information**.

Direct methods. These direct approaches do not require knowledge of robust correspondences between images. Therefore, the points used for tracking can be key points and corner points, and no descriptors need to be calculated. Some early systems [46, 47] are proposed to achieve robustness by detecting outliers via an iterative reweighting approach. Meanwhile, direct methods also can provide sparse, semi-sparse, and dense models when there are changes in light and dark in the scene.

Indirect methods. Based on correspondences generated by KLT feature tracking algorithms or descriptors, relative camera poses and depth information can be estimated in Epipolar geometry. With the help of those theories, filtering-based algorithms, from FastSLAM-2.0 [48] in 2000 and MonoSLAM [49] in 2007, are proposed to get real-time tracking capabilities. The first multi-thread system using non-linear least square optimization to deal with keypoints obtained from keyframes is PTAM [50] proposed in 2007.

Filtering vs. Optimization. Filtering and optimization are two directions for SLAM systems to refine initial estimates. Filtering-based SLAM is generally more computationally efficient, while optimization-based SLAM is often more accurate but computationally expensive.

Filtering-based SLAM methods [51, 52] use recursive Bayesian filtering techniques, such as Kalman Filter [53] or Extended Kalman Filter [52], to estimate the camera's pose and map the environment. These methods work by propagating a probabilistic state estimate forward in time and then updating it based on new measurements. Since filtering-based SLAM methods are computationally efficient and can operate in real time, they are ideal for applications with limited computational resources.

Optimization-based SLAM methods [13, 54, 15] use non-linear least square optimization techniques, such as bundle adjustment, to optimize the camera's pose and reconstruct environments. These methods minimize the difference between the predicted image measurements and the actual measurements from the sensors based on the photometric or geometric relationships. Optimization-based SLAM can be more accurate than filtering-based methods but is computationally expensive in global optimization.

2.1.4 Classical SLAM Frameworks

After introducing the important elements in SLAM systems, classical SLAM frameworks are briefly revisited in this section. With the gradual improvement of basic tools and theories, vision-based SLAM and odometry systems have shown impressive performance through different architectures. By extending these fundamentals from visual setups to IMU, LiDAR, and event camera sensors and creating new methods for these sensors, tracking and mapping services can be provided using increasingly diverse settings. In this section, we divide these state-of-the-art systems into two categories: visual SLAM and multi-sensor fusion, where the first category of methods is designed for monocular or stereo or RGB-D sensors, while the second category of methods uses IMU or LiDAR sensors in tracking and reconstruction tasks.

SVO: Semi-Direct Visual Odometry. Unlike traditional visual odometry approaches that rely on feature extraction and matching, SVO also takes advantage of photometric errors from corner points. Furthermore, the system proposes a depth filter based on the mixed-Gaussian distribution model. This approach allows SVO to be more robust to lighting changes and occlusions than feature-based methods.

LSD-SLAM: Large-Scale Direct Monocular SLAM. LSD-SLAM is fed by monocular input to reconstruct a 3D map and camera poses. LSD-SLAM directly estimates the camera pose by minimizing photometric errors using an efficient optimization algorithm. This approach allows LSD-SLAM to work well in large-scale, dynamic environments with few features and significant changes in illumination or perspective.

DSO: Direct Sparse Odometry. DSO is a visual odometry method that estimates the motion of a camera by directly minimizing the photometric errors based on all available pixel information and does not require explicit feature detection, matching, or tracking. This allows for better performance in challenging environments with few features or fast motion. Furthermore, DSO uses a sliding window optimization approach to improve estimation accuracy over time and reduce drift.

ORB-SLAM: Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. ORB-SLAM is a family of open-source SLAM libraries for monocular, stereo, RGB-D, and visual-inertial sensors based on feature-based techniques. Similar to PTAM [50], monocular ORB-SLAM [13] is a multi-thread tracking and mapping system, which relies heavily on feature tracking and optimization. ORB-SLAM2 [54] extends the functionality of ORB-SLAM by increasing robustness and improving accuracy. It also supports monocular, stereo, and RGB-D cameras. Compared with ORB-SLAM [13] and ORB-SLAM2 [54], atlas strategies are leveraged for different sensor setups, ORB-SLAM3 [55] further improves upon its predecessors by integrating the atlas map strategy and IMU sensors.

VINS: Visual-Inertial State Estimator. VINS is a system that estimates the motion of a camera in 3D space by fusing information from both visual and inertial sensors. It uses feature-based techniques to extract and match visual features in the images while also incorporating data from IMU to estimate the camera's orientation and velocity. VINS optimizes its estimates using a sliding window and nonlinear optimization to improve accuracy and robustness. Furthermore, it can operate in real-time on a variety of platforms, including smartphones, drones, and autonomous vehicles.

KIMERA: Open-Source Visual Inertial Odometry. Kimera is an open-source visual-inertial odometry system that combines visual and inertial data to estimate the motion of a camera in real time. It uses feature-based techniques to extract and match features in the images, as well as IMU measurements to estimate the camera's orientation and velocity. Kimera employs a sliding window optimization approach that improves accuracy over time by incorporating past information.

OpenVINS: A Research Platform for Visual-Inertial Estimation. OpenVINS is a research platform for visual-inertial estimation, designed to facilitate the development and evaluation of state-of-the-art algorithms for robotics and other applications. It provides a suite of open-source software tools that allow users to perform real-time visual inertial odometry, SLAM, and 3D reconstruction using a combination of visual and inertial sensor data. OpenVINS employs a modular approach to system design, allowing users to easily swap out different components, such as sensor models, optimization algorithms, and sensor fusion techniques. This flexibility enables researchers to quickly prototype and evaluate new ideas in a controlled environment. Additionally, OpenVINS includes a range of evaluation metrics and visualization tools that enable users to assess the performance of their algorithms in detail, including pose accuracy, drift, and computational efficiency.

2.2 Multiple View Geometry

Camera pose estimation and mapping problems are systematic and require many modules. In this section, we revisit the core blocks of this process that begin with image inputs and end at generating factor graphs as shown in Figure 2.3. The process can be summarized as follows. When we feed a sequence of images to the system, the 2D feature $\mathbf{f}^{j}_{c_i}$ is extracted from frame F_i first. We then obtain its correspondence $\mathbf{f}^{j'}_{c_{i+1}}$ in the next frame F_{i+1} via different feature tracking strategies, including descriptors [56, 57] and optical-flow [58] (see Section 2.2.1). Based on those correspondences, we can estimate the relative pose $\mathbf{T}_{i+1,i}$, from the camera coordinate C_i to C_{i+1}, and the 3D landmark $\mathbf{P}^{j}_{w}$ in the world coordinate (see Section 2.2.2). After an association step 2.2.4, we will get the observation relationship that shows the 3D landmark $\mathbf{P}_w^j$ is measured by frame F_i at the pixel position $\mathbf{p}_{c_i}^j$. Therefore, the graph (see Figure 2.3) can be generated finally.

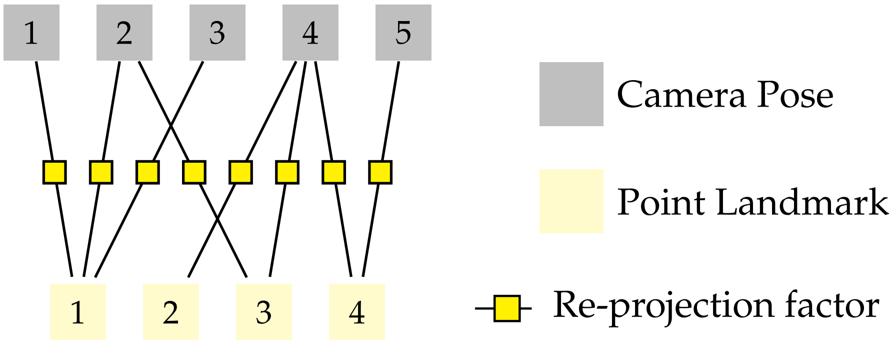


Figure 2.3 Factor graph representation. Camera poses and landmarks are vertices of the graph, where re-projection relationships are factors between vertices.

2.2.1 Feature Detection and Matching

Typically, features in SLAM are pixels (such as corners) or groups of pixels (such as lines and planes) that can be directly extracted from an RGB (depth) image because they are distinct from their neighbors. In this section, we first introduce popular features used in SLAM systems, including points, lines, and planes, while the corresponding matching methods are continued in subsequent paragraphs.

Feature Detection. The feature detection operation is the most essential step in a SLAM system, which aims to find unique clues for the current input. Therefore, this module is called frequently, almost on every input frame. Many aspects can be used to represent the specificity of an input frame. However, we refer to those cues that can be used in further pose estimation and reconstruction modules as features. This section details three popular features: points, lines, and surfaces. Details in point, line, and plane detection methods are illustrated in Section 2.1.1.

Feature Tracking. After extracting features from frame F_i and F_{i+1}, the next problem we have to solve is selecting the same features from two images. If two images observe the same position in the scene, those projections of the 3D point on those two images are termed correspondences. Therefore, the process of obtaining correspondence is feature tracking.

Descriptors are popular manners used in SLAM methods [13, 15] to track correspondences in different frames. The core idea of this method is to use abstract codes to represent each feature uniquely. The SIFT [23] algorithm generates a descriptor vector in 256×1, which is invariant to illumination changes, orientation, and uniform scaling. The BRIEF approach [59] used in the ORB algorithm provides binary descriptors for points, which can be measured by Hamming distance in the correspondence matching process.

Optical Flow is an effective method for feature matching during tracking based on the gray invariance assumption, as $\mathbf{I}(u + du, v + dv, t + dt) = \mathbf{I}(u, v, t)$, here $\mathbf{I}(u, v, t)$ is the gray value of pixel (u, v) at time t, and the new position of the pixel moves to $(u + du, v + dv)$ at time $t + dt$. The optical flow algorithm tracks these features over subsequent frames by computing each pixel's displacement vector (du, dv) from one frame to the next. The Lucas-Kanade method [60] is widely used in sparse optical flow estimation. Based on the gray invariance assumption, we can obtain that

$$\begin{bmatrix} \mathbf{I}_u & \mathbf{I}_v \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = -\mathbf{I}_t \tag{2.8}$$

here V_x and V_y are motion velocities of the pixel on the x-axis and y-axis respectively, and $\mathbf{I}_u$, $\mathbf{I}_v$, and $\mathbf{I}_t$ are the partial derivatives of the image I concerning position u, v and time t, evaluated at the point (u, v) at the current time. When pixels in a $w \times w$ window centered at (u, v) have the same motion, we can obtain $\begin{bmatrix} V_x & V_y \end{bmatrix}^T$ for pixels of the window. To improve the robustness of the algorithm, strategies about weighted windows and image pyramids are used in the optical tracking process.

2.2.2 Initial Pose Estimation

The task of camera pose estimation is to estimate 6-DoF camera poses $\mathbf{T}_{4\times4} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{t}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}$, where $\mathbf{T} \in SE(3)$, $\mathbf{R} \in SO(3)$, and $\mathbf{t} \in \mathbb{R}^3$. $SE(3)$ stands for the special Euclidean group, while $SO(3)$ is the special orthogonal group. Traditionally, the 6-DoF camera pose is regarded as a whole in our estimation model.

A 3D point $\mathbf{P}_i$ in the coordinate c_i can be transformed into the coordinate c_j via the following formulation

$$\bar{\mathbf{P}}_j = \mathbf{T}_{c_j,c_i} \bar{\mathbf{P}}_i \tag{2.9}$$

here $\bar{\cdot}$ shows the homogeneous operation. And the transformation $\mathbf{T}_{c_j,c_i}$ can be obtained by

$$\mathbf{T}_{c_j,c_i} = \mathbf{T}^{-1}_{w,c_j} \mathbf{T}_{w,c_i} \tag{2.10}$$

here $\mathbf{T}^{-1}_{w,c_j} = \mathbf{T}_{c_j,w} = \begin{pmatrix} \mathbf{R}w, c_j{}^T & -\mathbf{R}w, c_j{}^T t_{w,c_j} \\ \mathbf{0} & 1 \end{pmatrix}$.

According to the inputs, we clustered those methods into two strategies: **2D-2D** and **3D-2D**. For example, "**3D-2D**" means that the inputs of the pose computation task are 3D point clouds and 2D measurements. However, some structural cues help estimate orientations in manufactured environments. Therefore, another direction that decouples 4×4 transformation into two sequential steps. We will also introduce related theories in this section.

2D-2D Pose estimation. Given two RGB images F_i and F_j, and 2D correspondences $\mathbf{p}_i$ and $\mathbf{p}_j^{'}$, the relationship can be represented as

$$\mathbf{K}^{-1}\bar{\mathbf{p}}_i = R_{c_i,c_j}\mathbf{K}^{-1}\bar{\mathbf{p}}_j^{'} + \mathbf{t}_{c_i,c_j} \tag{2.11}$$

here $\mathbf{K}$ is the intrinsic matrix and $\bar{\mathbf{p}}$ is the homogenous vector of $\mathbf{p}$.

Then we use $[\mathbf{t}_{c_i,c_j}]_\times$ to deal with Equation 2.11, and obtain the following equation,

$$[\mathbf{t}_{c_i,c_j}]_\times \mathbf{K}^{-1}\bar{\mathbf{p}}_i = [\mathbf{t}_{c_i,c_j}]_\times \mathbf{R}_{c_i,c_j}\mathbf{K}^{-1}\bar{\mathbf{p}}_j^{'} \tag{2.12}$$

where $[\mathbf{t}_{c_i,c_j}]_\times$ is the skew-symmetric matrix of $\mathbf{t}_{c_i,c_j}$, and the nature of the $[\mathbf{t}_{c_i,c_j}]_\times \mathbf{K}^{-1}\bar{\mathbf{p}}_i$ operation is that the cross product operation between $\mathbf{t}_{c_i,c_j}$ and $\mathbf{K}^{-1}\bar{\mathbf{p}}_i$, namely $\mathbf{t}_{c_i,c_j} \times \mathbf{K}^{-1}\bar{\mathbf{p}}_i$. Therefore, the relationship can be further represented as

$$(\mathbf{K}^{-1}\bar{\mathbf{p}}_i)^T [\mathbf{t}_{c_i,c_j}]_\times \mathbf{R}_{c_i,c_j}\mathbf{K}^{-1}\bar{\mathbf{p}}_j^{'} = 0 \tag{2.13}$$

here $[\mathbf{t}_{c_i,c_j}]_\times \mathbf{R}_{c_i,c_j}$ is called **Essential Matrix E**, while $\mathbf{K}^{-T}[\mathbf{t}_{c_i,c_j}]_\times \mathbf{R}_{c_i,c_j}\mathbf{K}^{-1}$ is **Fundamental Matrix F**.

Therefore, the essential matrix or fundamental matrix can be computed by several correspondences. As we all know, $\mathbf{t}_{c_i,c_j}$ and $\mathbf{R}_{c_i,c_j}$ have three degrees of freedom, respectively.

Since the translation vector has a scale ambiguity problem, the degree of freedom of the essential matrix is only five. Therefore, five points [61] can be used to estimate the matrix. Furthermore, the 3×3 matrix can be estimated by the eight-match algorithm [62, 63] that builds a set of linear equations. Given the essential matrix, a SVD process is used to compute $\mathbf{t}_{c_i,c_j}$ and $\mathbf{R}_{c_i,c_j}$ which generates four matrices but only the result that makes the 3D landmark $\mathbf{P}$ is in front of both cameras is the correct one.

3D-2D Pose estimation. This method is commonly used in incremental tracking processes, where several 3D landmarks reconstruct a map. Benefitting feature matching methods, we obtain the matching relationships between incoming 2D features and 3D landmarks, and the **3D-2D** strategy is also called Perspective-n-Point (PnP). Similar to Equation 2.11, we also can make use of the following function to describe the problem,

$$s\bar{\mathbf{p}} = \mathbf{K}\mathbf{R}_{w,c}^T(\mathbf{P}_w - \mathbf{t}_{w,c}) \tag{2.14}$$

here s is a scale factor, and $\mathbf{P}_w = \begin{bmatrix} X_w & Y_w & Z_w \end{bmatrix}^T$. Then the matrix $\begin{bmatrix} \mathbf{KR}_{w,c}^T & -\mathbf{KR}_{w,c}^T \mathbf{t}_{w,c} \end{bmatrix}$ is denoted as $\mathbf{A}$, where $\mathbf{A} = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ t_5 & t_6 & t_7 & t_8 \\ t_9 & t_{10} & t_{11} & t_{12} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1^T \\ \mathbf{A}_2^T \\ \mathbf{A}_3^T \end{bmatrix}$. The Equation 2.14 can be rewritten as

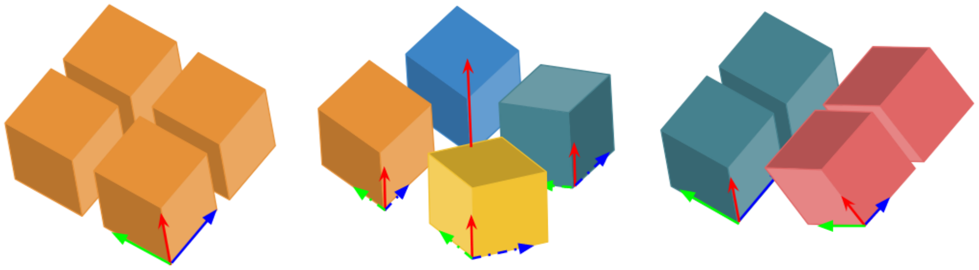
$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ t_5 & t_6 & t_7 & t_8 \\ t_9 & t_{10} & t_{11} & t_{12} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{pmatrix} \mathbf{A}_1^T \\ \mathbf{A}_2^T \\ \mathbf{A}_3^T \end{pmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \tag{2.15}$$

Since $s = t_9 X_w + t_{10} Y_w + t_{11} Z_w + t_{12} = \mathbf{A}_3^T \bar{\mathbf{P}}_\mathbf{w}$, we can remove the scale factor in our relationships to obtain the formula between unknown parameters and the given 3D-2D observation

$$\begin{cases} \mathbf{A}_1^T \bar{\mathbf{P}}_\mathbf{w} - \mathbf{A}_3^T \bar{\mathbf{P}}_\mathbf{w} u = 0 \\ \mathbf{A}_2^T \bar{\mathbf{P}}_\mathbf{w} - \mathbf{A}_3^T \bar{\mathbf{P}}_\mathbf{w} v = 0 \end{cases} \tag{2.16}$$

here at least 6 matches are needed to solve Equation 2.15 as a linear transformation problem. Extended from the basic PnP strategy, EPnP [64] and UPnP [65] are proposed to estimate camera poses.

Structure-based Pose estimation. Instead of using Equation 2.11 to estimate a transformation $\mathbf{T}_{4\times 4}$ in one time, here we decouple the transformation matrix into rotation and translation parts. The pose estimation problem can be transferred to a linear least-square problem with a known rotation part. Approaches [66, 67] make use of the Manhattan World and Atlanta World assumptions (see Figure 2.4) and explore a decoupled solution to deal with rotation $\mathbf{R} \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$ separately.



(a) Manhattan World

(b) Atalanta World

(c) Mixture of Manhattan Frame

Figure 2.4 Examples of different structure assumptions.

Manhattan World. Based on the assumption of Manhattan World, a Manhattan coordinate M is built, and the center of the coordinate is located on the same point as the camera coordinate. Therefore, we only have a rotation motion $\mathbf{R}_{m,c_i}$ between M and the camera coordinate c_i.

In the initialization module, besides building the relative rotation $\mathbf{R}_{w,c_0}$, we also have to compute the rotation between the Manhattan coordinate and the world coordinate $\mathbf{R}_{w,m}$ that is introduced in Section 3.3.3 in detail. Therefore, the traditional purpose of estimating $\mathbf{R}_{w,c_i}$ is transferred to estimate $\mathbf{R}_{c_j,m}$ via the following function

$$\mathbf{R}_{w,c_j} = \mathbf{R}_{w,m}\mathbf{R}^{T}_{c_j,m} \tag{2.17}$$

here $\mathbf{R}_{w,m}$ is obtained in the initialization process, while $\mathbf{R}^{T}_{c_j,m}$ is supposed to computed in the tracking process. In Equation 2.17, the rotation will be affected by frames between the start frame and the j^{th} frame, which has the core idea, **low-drift rotation estimation**, of the Manhattan World assumption.

Since the transformation from the first frame to the world coordinate is $\mathbf{T}_{w,c_0} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}$. Therefore, the $\mathbf{R}_{\mathbf{w},\mathbf{m}}$ is equal to $\mathbf{R}_{\mathbf{c_0},\mathbf{m}}$. During the tracking process, the rotation between the i^{th} frame and Manhattan world M is computed as $\mathbf{R}_{c_i,m}$. Therefore, the relative rotation motion $\mathbf{R}_{c_i,c_j}$ is represented as

After estimating rotation $\mathbf{R}_{w,c_j}$ of F_j, the unknown the parameters of Equation 2.14 is $\mathbf{t}_{w,c_j}$ which can be estimated by a closed-form method based on 2 pairs of 3D-2D matches via

$$\mathbf{t}_{w,c_j} = \mathbf{P}_w - s(\mathbf{K}\mathbf{R}^{T}_{w,c_j})^{-1}\bar{\mathbf{p}}_j. \tag{2.18}$$

Therefore, $\begin{bmatrix} t^{x}_{w,c_j} \\ t^{y}_{w,c_j} \\ t^{z}_{w,c_j} \end{bmatrix}$ can be computed directly. Therefore, one of the advantages of decoupling pose estimation methods is that fewer correspondences are required, which is helpful in low-textured scenes.

2.2.3 Initial Landmark Reconstruction

After illustrating feature extraction and pose estimation methods, we continue to introduce the fundamental theories in feature triangulation and sparse reconstruction here, which start from the topics that how to represent those landmarks in maps. Since a type of 3D landmark may have different representations, we can regard the representation is not minimal representation if the parameters is more than the degreeds of freedom of the landmark. The overparameterization issue is critical in the optimization modules, but for initial 3D reconstuction over-parameterized representation also can be used.

As shown in Figure 2.5, sparse maps are generally composed of points, lines, and planes, and these 3D landmarks are mainly concentrated in structured and textured areas. In this section, 3D points and lines are representation based on the simple Euclidean representation, where a

3D point is denoted as $\mathbf{P}_{3\times1} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ and a 3D line is denoted as two endpoints, $\mathbf{P}^s$ and $\mathbf{P}^e$, of the line, namely $\mathbf{L}_{3\times2} = \begin{bmatrix} \mathbf{P}^s & \mathbf{P}^e \end{bmatrix}$.

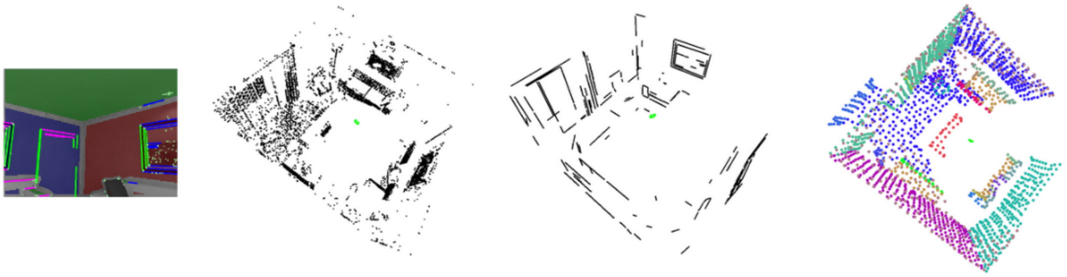


Figure 2.5 Examples of sparse reconstruction. From left to right, images are RGB inputs, point map, line map and plane map.

Point Triangulation. The point correspondences $\mathbf{p}^k$ and $\mathbf{p}^{k'}$ lying on the two images F_i and F_j are respectively extracted, which are re-projected from the same landmark point $\mathbf{P}_w$. Triangulation aims to use 2D feature points to compute spatial 3D landmarks. The triangulation process of a point is written as

$$\begin{cases} \hat{\mathbf{p}}^k = \mathbf{Q}_i \bar{\mathbf{P}}_w \\ \hat{\mathbf{p}}^{k'} = \mathbf{Q}_j \bar{\mathbf{P}}_w \end{cases} \tag{2.19}$$

where $\mathbf{Q}_i = [\mathbf{K}\mathbf{R}_{c_i,w}|\mathbf{t}_{c_i,w}]_{3\times4}$. Since $\hat{\mathbf{p}}^k \times \mathbf{Q}_i \bar{\mathbf{P}}_w = 0$, we can obtain the following function

$$\mathbf{A}\mathbf{P}_w = \begin{bmatrix} \hat{\mathbf{p}}^k \times \mathbf{Q}_i \bar{\mathbf{P}}_w \\ \hat{\mathbf{p}}^{k'} \times \mathbf{Q}_j \bar{\mathbf{P}}_w \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{p}}^k \times \mathbf{Q}_i \\ \hat{\mathbf{p}}^{k'} \times \mathbf{Q}_j \end{bmatrix} \bar{\mathbf{P}}_w = \mathbf{0} \tag{2.20}$$

here $\begin{bmatrix} \hat{\mathbf{p}}^k \times \mathbf{Q}_i \\ \hat{\mathbf{p}}^{k'} \times \mathbf{Q}_j \end{bmatrix}$ is represented as $\mathbf{A}$. Due to the noise of points' positions and estimated camera poses, the noise is more likely to be amplified when the parallax angle between correspondences is too small. Therefore, those cases are filtered out in map points reconstruction [13].

Line Triangulation. In order to be more intuitive in geometry, a 3D line landmark $\mathbf{L}$ is represented as two endpoints $\mathbf{P}^s$ and $\mathbf{P}^e$. Due to occlusion issues during the observation process, the two endpoints of the re-projected line on F_i cannot guarantee matching with corresponding endpoints on F_j when F_i and F_j observe the same line landmark. Therefore, the method to reconstruct a line landmark can be directly regarded as point reconstruction, which needs to perform some special processing.

First, we assume that pixel positions $\mathbf{p}_i^s$ and $\mathbf{p}_i^e$ lying on the i^{th} image plane are the re-projection results of those two 3D endpoints $\mathbf{P}^s$ and $\mathbf{P}^e$, respectively. Then, the observation relationship is described as

$$\mathbf{p}_i^s = \mathbf{Q}_i \mathbf{P}^s \tag{2.21}$$

here $\mathbf{Q}_i$ shows the measurement function of the i^{th} frame.

As we mentioned, we may do not know the exact position of $\mathbf{P}^s$ on the j^{th} image plane, but we make sure that the re-projected points $\mathbf{p}_i^s$ and $\mathbf{p}_j^s$ are lying on the 2D extracted lines $\mathbf{l}^i$ and $\mathbf{l}^j$, respectively. Therefore, another two relationships are built to describe the position of $\mathbf{P}^s$ as follows

$$\begin{cases} \mathbf{l}^i \mathbf{p}_i^s = \mathbf{l}^i \mathbf{Q}_i \mathbf{P}^s = 0 \\ \mathbf{l}^j \mathbf{p}_j^{s\prime} = \mathbf{l}^j \mathbf{Q}_j \mathbf{P}^s = 0 \end{cases} \tag{2.22}$$

here $\mathbf{l}^i$ is computed by 2D endpoints, $\mathbf{p}_i^s$ and $\mathbf{p}_i^e$, of a 2D line feature via a cross production $\frac{\bar{\mathbf{p}}_i^s \times \bar{\mathbf{p}}_i^e}{\sqrt{a^2+b^2}}$.

Combined by Equation 2.21 and 2.22, the relationships between 2D measurements and 3D line landmark $\mathbf{P}^s$ can be summarized as

$$\begin{cases} \mathbf{p}_i^s = \mathbf{Q}_i \mathbf{P}^s \\ \mathbf{l}^i \mathbf{p}_i^s = \mathbf{l}^i \mathbf{Q}_i \mathbf{P}^s = 0 \\ \mathbf{l}^j \mathbf{p}_j^{s\prime} = \mathbf{l}^j \mathbf{Q}_j \mathbf{P}^s = 0 \end{cases} \tag{2.23}$$

Similar to the triangulation process of point landmarks, we also make use of the SVD operation to solve Equation 2.23. Due to the noise of endpoints' positions, the length of 2D line features, and estimated camera poses, we also build a threshold for the parallax angle between correspondences, where those cases below the threshold are filtered out in the reconstruction process.

2.2.4 Keyframe and Observation Association

Another critical step in building factor graphs for optimization modules is data association, which is essential in bridging front and back ends. Generally, feature extraction, pose estimation, and landmarks reconstruction are finished in the front-end module. On the one hand, we need the association module to organize those observation relationships between frames and landmarks into structures. On the other hand, those redundant reconstruction results and unnecessary connections should be fused in this module.

Keyframe Selection. Only point features are considered in the keyframe selection process in ORB-SLAM [13] and VINS-based methods [15]. For methods [5] that uses lines, planes, and vanishing directions in the tracking process, the keyframe selection module will consider other features to achieve more precise information.

Landmark Fusion. During incremental tracking, for some reasons, including occlusion and noise, those IDs of frames that detect the same feature are not consecutive. As shown in Figure 2.6, feature 1 are detected in frames F_{i+1} and F_{i+2}, but they are failed in matching. Therefore, in local reconstruction, the same feature tends to be reconstructed into two landmarks $\mathbf{L}_1$ and $\mathbf{L}_1^{'}$. Therefore, every new reconstructed landmark should be examined via the following process.

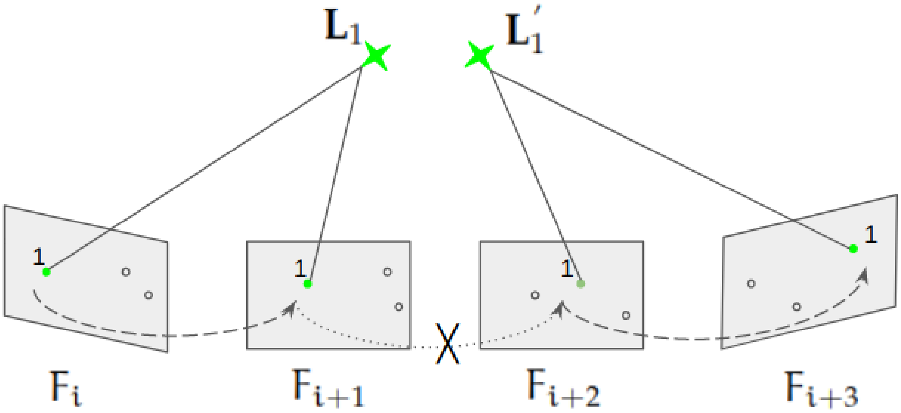


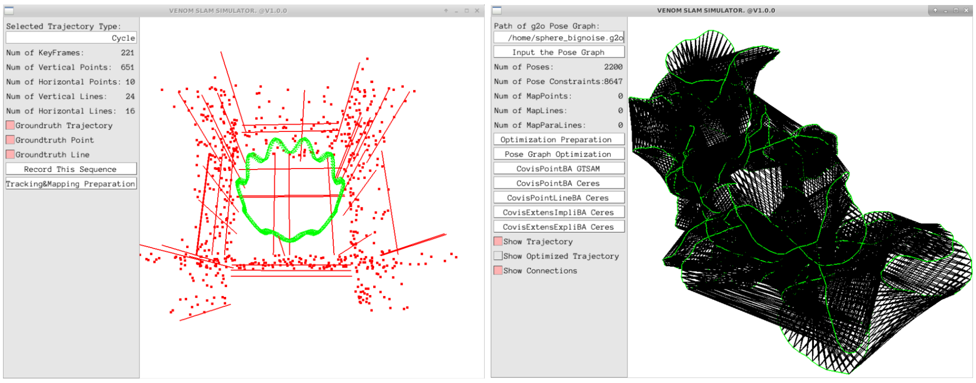
Figure 2.6 Example of landmarks fusion. From F_{i+1} to F_{i+2}, feature 1 is failed in tracking.

To remove redundant reconstruction results, we re-project map points $\mathbf{P}$ to the image plane of the newly generated keyframe kF_i based on the initial camera pose $\mathbf{T}_{w,c_i}$, where the image plane is divided into several grids. After obtaining the grid id g_n of the re-projection point, we select several interesting grids $\mathbf{G}_n$ that are neighboring of g_n. Then, point features located on the region $\mathbf{G}_n$ are regarded as correspondence candidates of $\mathbf{P}$.

Since each map point has its descriptor $\mathbf{D}$, we will compute the distance between candidates' descriptor vectors and $\mathbf{D}$. When a candidate is selected to connect to the map point, we merge related co-visibility relationships.

2.3 Optimization Theory

To limit the increase of pose errors, there are some optimization solutions, including local bundle adjustment [13, 14], sliding window [15], and loop closure [18, 13], where the core idea back of those solutions is abstracting observation relationships to co-visibility graphs. Therefore, optimizing camera poses and landmarks will be transformed into a problem of modifying those vertices to achieve the best result of a loss function introduced in Section 2.1. The ideas for building the loss function and updating vertices will be described in the following sections.



Selected Trajectory Type: Cycle

Num of KeyFrames: 221

Num of Vertical Points: 651

Num of Horizontal Points: 10

Num of Vertical Lines: 24

Num of Horizontal Lines: 16

☑ Groundruth Trajectory

☑ Groundtruth Point

☑ Groundtruth Line

Path of g2o Pose Graph: /home/sphere_bignoise.g2o

Num of Poses: 2200

Num of Pose Constraints:8647

Num of MapPoints: 0

Num of MapLines: 0

Num of MapParaLines: 0

☑ Show Trajectory

□ Show Optimized Trajectory

☑ Show Connections

Figure 2.7 Example of graphs. Pose vertices are in green and constraints between are in black.

2.3.1 Vertex Representation

In factor graphs, camera poses and landmarks are represented as pose vertices $\mathcal{V}_{\mathrm{pose}}$ and landmark vertices $\mathcal{V}_{\mathrm{p}}$. To limit the size of graphs, all those pose vertices are generated from keyframes rather than each frame fed from inputs in ORB-SLAM, while sliding-window-based optimization methods [15] further fix the size of pose vertices according to the size of the window. The keyframe selection and landmark reconstruction approaches are introduced in Section 2.2.4.

Given initial camera poses and landmarks in the world coordinate, another critical step to building a factor graph is parameterizing those landmarks for optimization. Generally, a line landmark in 3D space is in four DoFs, while point and borderless plane landmarks in 3D space are in three DoFs; therefore, if the optimized parameters of those landmarks are more than their degrees of freedom. It means having more model parameters than necessary. We call the situation over-parameterization, which leads to additional degrees of freedom and introduces errors in the process [68]. To address the problem, a minimal representation for the rotation is suggested to take place to those over-parameterization parameters.

Point landmarks representation. The straightforward minimal parameterization for a point landmark is Euclidean XYZ, which is represented as $\begin{bmatrix} x & y & z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3$. Even the representation has disadvantages in parameterizing low parallex features, the representation is also widely used in SLAM systems [13, 54] since bad cases can be removed in front-ends. To deal with features over a huge range of depths, the inverse depth parametrization [35] can be used to represent landmarks generated by little parallax correspondences.

In the inverse depth scheme, a 3D point landmark $\mathbf{P}_i$ in the scene is measured at pixel $\mathbf{p}_i = \begin{bmatrix} u_i & v_i \end{bmatrix}$,

$$\mathbf{P}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i} \begin{pmatrix} \cos\beta \sin\alpha \\ -\sin\beta \\ \cos\beta \cos\alpha \end{pmatrix} \tag{2.24}$$

here α and β are azimuth and elevation, respectively. ρ_i is the inverse depth.

Line landmarks representation. The 3D line $\mathcal{L} = [\mathbf{n}_{3\times1}\ \mathbf{d}_{3\times1}]$ used in the map is represented as $\mathtt{Plücker}$ Parametrization, where $\mathbf{n} \in \mathbb{R}^3$ and $\mathbf{d} \in \mathbb{R}^3$ show normal and direction vectors of the 3D line. To address the over-parametrization issue of the $\mathtt{Plücker}$ method, we make use of the orthonormal parametrization $\mathcal{O} = [\boldsymbol{\varphi}, \theta]$ in the optimization module. The transfermation between $\mathtt{Plücker}$ and Orthonormal is ilustrated as

$$\begin{aligned} [\mathbf{n}\ \mathbf{d}] &= \begin{bmatrix} \frac{\mathbf{n}}{||\mathbf{n}||} & \frac{\mathbf{d}}{||\mathbf{d}||} & \frac{\mathbf{n}}{||\mathbf{n}||} \times \frac{\mathbf{d}}{||\mathbf{d}||} \end{bmatrix} \begin{bmatrix} ||\mathbf{n}|| & 0 \\ 0 & ||\mathbf{b}|| \\ 0 & 0 \end{bmatrix} \\ &= \sqrt{||\mathbf{n}||^2 + ||\mathbf{b}||^2} R(\boldsymbol{\varphi}) \begin{bmatrix} \cos\theta & 0 \\ 0 & \sin\theta \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{2.25}$$

here $\boldsymbol{\varphi} = [\varphi_1\ \varphi_2\ \varphi_3]^T$ and $||\cdot||$ is the normalization operation.

Here we make use of $\mathbf{U} = R(\boldsymbol{\varphi}), \mathbf{U} \in SO(3), \mathbf{W} = \begin{bmatrix} w_1 & -w_2 \\ w_2 & w_1 \end{bmatrix} = \frac{1}{\sqrt{||\mathbf{n}||^2+||\mathbf{d}||^2}} \begin{bmatrix} ||\mathbf{n}|| & -||\mathbf{d}|| \\ ||\mathbf{d}|| & ||\mathbf{n}|| \end{bmatrix}$.

Plane landmarks representation. Similar to the Plücker method, the Hesse form $[\mathbf{n}^T\ d]^T$ is a good method to show geometry representation. But the method also has the over-parameterization issue due to the fact that the unit normal vector part has 3 parameters, but actually only has 2 degrees of freedom. To address the problem, the minimal parameterization $\boldsymbol{\tau} = [\theta\ \phi\ d]^T$ of a plane in optimization, here θ and ϕ are the azimuth and elevation angles of the normal vector respectively. Therefore, the relationship between $\boldsymbol{\tau}$ and $\boldsymbol{\pi}$ can be denoted as

$$\boldsymbol{\tau} = q(\boldsymbol{\pi}) = (\theta = \arctan\frac{n_y}{n_x}, \phi = \arcsin n_z, d) \tag{2.26}$$

here $\mathbf{n} = \begin{bmatrix} n_x & n_x & n_z \end{bmatrix}^T$, and the azimuth and elevation angles are restricted in $(-\pi, \pi]$ to avoid the singularities situation in optimization.

2.3.2 Co-visibility Factors

In this section, we introduce a widely used factor graph structure in detail, Co-visibility factor graphs, where those edges between vertices are built based on co-visible relationships. Co-visibility connections [69, 13, 54, 55] are built when two images detect the same landmark, like a mappoint or a mapline. Based on the re-projection models of point and line features, the co-visibility factors are defined in this section. Specifically, after feeding pose and landmark vertices into a graph optimization pipeline, a factor connection happens in two cases:

- During the tracking process, when an existed landmark $\mathbf{LM}^i$ reconstructed before is associated to the current frame F_c, a new observation will be built.
- During the loop closure modules, when two existing landmarks reconstructed repeatedly are fused into a landmark, old observations of those landmarks will be merged.

Observation in Tracking. In tracking process, the observation between 2D feature $\mathbf{lm}^j$ and 3D landmark $\mathbf{LM}^j$ is constructed, and the factor can be represented as

$$f(\mathbf{lm}^j, \mathbf{K}\mathbf{T}_{cw}\mathbf{LM}^j) \tag{2.27}$$

here $f(\cdot,\cdot)$ is the re-projection function between 2D measurements and corresponding 3D landmarks. Furthermore, $\mathbf{lm}^j$ used in this section represents points, lines, and planes. To be specific, points and lines can be matched by descriptors, while planes are associated based on the Gauss representation $\boldsymbol{\pi} = [\mathbf{n}\ d]$.

Observation fusion in Loop Closure. As introduced in Equation 2.27, the re-projection factor will be built incrementally. Caused by noise introduced in Section 2.1.1, an environment landmark measured by different frames may be reconstructed into several 3D primitives. Therefore, when we detect that a camera revisits the same place based on the DBoW [70] method, the correspondences between two vertices $\mathbf{T}_{c_n,w}$ and $\mathbf{T}_{c_m,w}$ are matched. Then those redundantly reconstructed landmarks, $\mathbf{lm}_j$ and $\mathbf{lm}_i$, are fused into one landmark associated with those two pose vertices. Based on the loop closure constraints, the factor between measurements and 3D primitives is represented as

$$\left.\begin{array}{r} f(\mathbf{lm}^j, \mathbf{K}\mathbf{T}_{c_n,w}\mathbf{LM}^j) \\ f(\mathbf{lm}^i, \mathbf{K}\mathbf{T}_{c_m,w}\mathbf{LM}^i) \end{array}\right\} \xrightarrow{\text{fusion}} \sum_{s\in(m,n)} \sum_{k\in(i,j)} f(\mathbf{lm}^k, \mathbf{K}\mathbf{T}_{c_s,w}\mathbf{LM}^i) \tag{2.28}$$

where $\xrightarrow{\text{fusion}}$ is the landmark fusion step in the loop closure module.

Point Re-projection Factor. Based on the point feature measurement model, the measurement of the i^{th} global point landmark $\mathbf{P}_i = [\ x_i \quad y_i \quad z_i\ ]^T$ detected by frame F_j is represented as $\mathbf{p}_k^j = [\ u_k^j \quad v_k^j\ ]^T$ in the normalized coordinate, and the re-projection error of this observation relationship is defined as $\mathbf{r}_\mathbf{p}(\mathbf{p}_k^j, \mathbf{P}_i, \mathbf{T}_{w,c_j})$ where

$$\mathbf{r}_\mathbf{p}(\mathbf{p}_k^j, \mathbf{P}_i, \mathbf{T}_{w,c_j}) = \mathbf{K}\overline{\mathbf{R}_{w,j}^T(\mathbf{P}_i - \mathbf{t}_{w,j})} - \mathbf{p}_k^j \tag{2.29}$$

here $\overline{(\cdot)}$ is to compute the normalized coordinate.

Line Re-projection factors. The mapline $\mathbf{L}_w$ in the world coordinate is represented by $\mathbf{L}_w = \begin{bmatrix} \mathbf{n}_w & \mathbf{d}_w \end{bmatrix}^T$, here $\mathbf{d}_w$ is the direction vector of $\mathbf{L}_w$ while $\mathbf{n}_w$ is normal vector of the plane constructed by landmark $\mathbf{L}_w$ and the original point of the world coordinate.

From the world coordinate to the i^{th} camera coordinate, the line landmark in the camera coordinate can be represented as

$$\mathbf{L}_{c_i} = \begin{bmatrix} \mathbf{n}_{c_i} \\ \mathbf{d}_{c_i} \end{bmatrix} = \mathbf{T}_{c_i,w} \begin{bmatrix} \mathbf{n}_w \\ \mathbf{d}_w \end{bmatrix} \tag{2.30}$$

here $\mathbf{T}_{c_i,w} = \begin{bmatrix} \mathbf{R}_{c_i,w} & [\mathbf{t}_{c_i,w}]_\times \mathbf{R}_{c_i,w} \\ \mathbf{0} & \mathbf{R}_{c_i,w} \end{bmatrix}$, and $[\cdot]_\times$ is the skew-symmetric operation.

Furthermore, we re-project the $\mathbf{L}_{c_i}$ onto the image plane. Assuming $\mathbf{p}_s$ and $\mathbf{p}_e$ are two endpoints re-projected by $\mathbf{L}_{c_i}$ via

$$\bar{\mathbf{p}}_k = \mathbf{K}\bar{\mathbf{P}}_k, k \in (s, e) \tag{2.31}$$

here $\bar{\mathbf{P}}_k$ is a normalized 3D endpoint in the camera coordinate.

Therefore, the re-projected line function $\mathbf{l}^j_k$ **[zhangline]**, can be obtained by

$$\mathbf{l}^j_k = \bar{\mathbf{p}}_s \times \bar{\mathbf{p}}_e = \mathrm{K}\bar{\mathbf{P}}_s \times \mathbf{K}\bar{\mathbf{P}}_e = \mathcal{K}(\bar{\mathbf{P}}_s \times \bar{\mathbf{P}}_e) = \mathcal{K}\mathbf{n}_j \tag{2.32}$$

here $\mathcal{K} = \begin{bmatrix} f_y & 0 & 0 \\ 0 & f_x & 0 \\ -f_y c_x & -f_x c_y & f_x f_y \end{bmatrix}$.

Finally, the re-projection error between $\mathbf{l}_j$ and its extracted line feature $\mathbf{l} = \begin{bmatrix} \mathbf{p}_s & \mathbf{p}_e \end{bmatrix}^T$ can be written as

$$\mathbf{r}_l(\mathbf{p}^j_{k,s}, \mathbf{p}^j_{k,e}, \mathbf{L}_w, \mathbf{T}) = \begin{bmatrix} d(\mathbf{p}^j_{k,s}, l^j) \\ d(\mathbf{p}^j_{k,e}, l^j) \end{bmatrix} \tag{2.33}$$

where $d(\mathbf{p}^j_{k,s}, l) = \frac{\mathbf{p}^{jT}_{k,s} \cdot l}{\sqrt{l_1^2 + l_2^2}}$, and $l^j = \begin{bmatrix} l_0 & l_1 & l_2 \end{bmatrix}^T$ is the 2D line re-projected from the j^{th} 3D mapline. $\mathbf{p}_k = \begin{bmatrix} \bar{x}_k & \bar{y}_k & 1 \end{bmatrix}^T$, $k \in (s, e)$, are the normalized coordinates of endpoints.

Factor Graph Construction. Given vertices and re-projection errors listed in Equation 2.29 and 2.33, the co-visibility graph $\mathcal{G}$ can be established, where the vertices in this graph $\mathcal{G}$ contain camera poses $\mathcal{V}_{pose}$, point landmarks $\mathcal{V}_p$, and line landmarks $\mathcal{V}_l$. Furthermore, those pose vertices are $\mathbf{T}_{w,c_i} = \begin{bmatrix} \mathbf{R}_{w,c_i} & \mathbf{t}_{w,c_i} \\ \mathbf{0} & 1 \end{bmatrix}$ where $\mathbf{T}_{w,c_i} \in SE(3)$, $\mathbf{R}_{w,c_i} \in SO(3)$ and $\mathbf{t}_{w,c_i} \in \mathbb{R}^3$. Points used in the optimization module is parameterized as $\mathbf{P}_w^k = \begin{bmatrix} x^k & y^k & z^k \end{bmatrix}^T$. The edge between $\mathcal{V}_{pose}$ and $\mathcal{V}_p$ is called as a factor $\mathcal{E}_{pose,p}$. Since the Plücker approach has an over-parameterization issue, we make use of Orthonormal representation $\mathcal{O}_w^k = \begin{bmatrix} \theta & \omega \end{bmatrix}$ in the line landmark updating process.

Based on the result of the frond-end, observation at different timestamps are considered together, then the co-visibility factor graph can be abstracted as the following loss function

$$\sum_{i=1}^{m}\sum_{j=1}^{n} ||\mathbf{r}_p(\mathbf{p}_k^j, \mathbf{P}_i, \mathbf{T})||^2_{\Lambda_p} + \sum_{i=1}^{m}\sum_{j=1}^{n} ||\mathbf{r}_l(\mathbf{p}_{k,s}^j, \mathbf{p}_{k,e}^j, \mathbf{L}_w, \mathbf{T})||^2_{\Lambda_l} \tag{2.34}$$

here j shows the ID of the camera $j \in (0, \dots, n)$ and i shows the ID of the map point $i \in (0, \dots, m)$. And $\boldsymbol{\Lambda}$ is the information matrix and $\mathbf{r}_p$ is shorten for $\mathbf{r}_p(\mathbf{p}_k^j, \mathbf{P}_i, \mathbf{T})$. where Λ_p and Λ_l are information matrices.

Pose Graph optimization. Instead of optimizing both pose and landmark vertices, sometimes pose graphs are more prevalent in real-time SLAM systems since the optimization problem size is much smaller than the factor graph introduced in the last paragraph. In pose graphs, there are only pose vertices as shown in Figure 2.7(b). Edges between frames represent the relative transformation that can be generated based on co-visible landmarks. Constraints between poses are used for optimization. In visual SLAM, the vertices are in 6-DoF, while visual-inertial odometry methods [15] perform 4-DoF pose graph optimization since the observation of roll and pitch angles can be fully supported by IMU.

When a loop closure is detected in a trajectory, new connections between the current frame and the previous ones that detect the same place will be established. For visual-SLAM systems, the relative translation can be estimated by methods like PnP [64]. Then, the sequential constraints and loop closure constraints are used to build the following function

$$\min_{\mathbf{T}_{c_i,w}} \sum_{(i,j)\in\mathcal{S}} ||\mathbf{r}_{i,j}||^2 + \sum_{(m,n)\in\mathcal{C}} ||\mathbf{r}_{m,n}||^2 \tag{2.35}$$

here $\mathcal{S}$ and $\mathcal{C}$ represent the set of sequential and all loop closure frames, respectively.

2.3.3 Matrix Lie Groups and Lie Algebras

Matrix Lie Groups

There are two specific matrix Lie Groups in pose optimization tasks, Special Orthogonal Group, denoted $SO(3)$, and the Special Euclidean Group, $SE(3)$, which can be represented rotation and transformations respectively.

Special Orthogonal Group. $SO(3)$ describes the set of orthogonal matrices as,

$$SO(3)\doteq\{\mathbf{R}\in\mathbb{R}^{3\times 3}|\mathbf{R}^{\mathsf{T}}\mathbf{R}=\mathbf{1},\det(\mathbf{R})=1\} \tag{2.36}$$

here $\det(\cdot)$ shows that the product of each one of its elements is of determinant 1.

In mathematics, a *group* is built by a set of elements together with an operation, and the third element generated via the operation and two of its elements is also in the set. Even the matrix $\mathbf{R}$ can be shown as the format of vectorspace, $SO(3)$ is not a valid subspace. But there are characteristics for $SO(3)$:

1. $SO(3)$ is closed under matrix multiplication, namely $\mathbf{R}_{ij}=\mathbf{R}_i\mathbf{R}_j, \mathbf{R}_{ij}\in SO(3)$;
2. the inverse, $\mathbf{R}_i^{-1}$, is the matrix transpose is $\mathbf{R}_i^{\mathsf{T}}$, matrix transposition, where $\mathbf{R}_i^{-1}\in SO(3)$;
3. given the identity matrix $\mathbf{I}\in SO(3)$, $\mathbf{R}_i\mathbf{I}=\mathbf{I}\mathbf{R}_i=\mathbf{R}_i, \mathbf{R}_i\in SO(3)$;
4. the associativity for the rotation motion $\mathbf{R}_k(\mathbf{R}_j\mathbf{R}_i)$ is equal to $(\mathbf{R}_k\mathbf{R}_i)\mathbf{R}_j$.

With those properties, also called as group axioms,the group is a differential manifold that can be seen as Euclidean on a local scale. and the optimization on manifold is possible.

Similar to SO(3), the Special Orthogonal group in dimension 2 is denoted as SO(2), which is used in planar orientation motions, as well as in 3D line optimization modules as introduced in Section 2.3.1.

Special Euclidean Group. $SE(3)$ describes the set of rigid motion in 3D space, which is denoted as

$$SE(3)\doteq\left\{\mathbf{T}=\begin{bmatrix}\mathbf{R} & \mathbf{t}\\ \mathbf{0} & 1\end{bmatrix}\in\mathbb{R}^{4\times 4}|\mathbf{R}\in SO(3),\mathbf{t}\in\mathbb{R}^3\right\} \tag{2.37}$$

here $SE(3)$ also satisfies four conditions, including closure, invertibility, identity and associativity. Specifically, the group operation of $SE(3)$ is matrix multiplication, $\mathbf{T}_i\mathbf{T}_j$, and the inverse is $\mathbf{T}^{-1}=\begin{bmatrix}\mathbf{R}^{\mathsf{T}} & -\mathbf{R}^{\mathsf{T}}\mathbf{t}\end{bmatrix}, \mathbf{T}^{-1}\in SE(3)$.

Lie Algebras

The tangent space to the manifold $SO(3)$ (at the identity) is denoted as $\mathfrak{so}(3)$, which is vectorspace of the *Lie algebra*. Therefore, $\mathfrak{so}(3)$ perfactly captures the local characteristics of $SO(3)$.

From a vector $\boldsymbol{\phi} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}$ in $\mathbb{R}^3$, the operator $[\cdot]_\times$ representing a skew symmetric matrix operation, transferms the vector as a matrix $[\boldsymbol{\phi}]_\times = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}$. So the vectorspace of $\mathfrak{so}(3)$ can be denoted as

$$\mathfrak{so}(3) \doteq \left\{ \boldsymbol{\Phi} = [\boldsymbol{\phi}]_\times \in \mathbb{R}^{3\times3} | \boldsymbol{\phi} \in \mathbb{R}^3 \right\}. \tag{2.38}$$

Furthermore, $\boldsymbol{\phi}$ can be represented by a normalized vector $\mathbf{a}$ and ϕ, $\boldsymbol{\phi} = \phi\mathbf{a}$. Then, $\exp([\boldsymbol{\phi}]_\times) = \exp(\phi\mathbf{a}^\wedge)$ is denoted by

$$\exp(\phi\mathbf{a}^\wedge) = \mathbf{1} + (1 - \cos\phi)(\mathbf{a}^\wedge)^2 + \sin\phi\mathbf{a}^\wedge \tag{2.39}$$

here the *exponential map* (at the identity) $\exp()$ can be used to associate Lie Algebra elements to rotation in $SO(3)$ by using $\mathbf{R} = \exp(\phi\mathbf{a}^\wedge)$. A first-order approximation of the exponential map is

$$\exp([\boldsymbol{\phi}]_\times) \approx \mathbf{1} + [\boldsymbol{\phi}]_\times. \tag{2.40}$$

In the tangent space, an increments $\delta\boldsymbol{\phi}$ in the additive operation is related to multiplicative increments of $SO(3)$ via the *left Jacobian* $\mathbf{J}_l(\boldsymbol{\phi})$,

$$\exp([\boldsymbol{\phi} + \delta\boldsymbol{\phi}]_\times) \approx \exp(([\mathbf{J}_l\Delta\boldsymbol{\phi}]_\times))\exp([\boldsymbol{\phi}]_\times) \tag{2.41}$$

which provides a new perspective to map the small movement in the tangent space to operations of $SO(3)$.

Similarly, the *Lie algebra* associated with $SE(3)$ is represented by

$$\mathfrak{se}(3) \doteq \left\{ \boldsymbol{\Xi} = [\boldsymbol{\xi}]_\times \in \mathbb{R}^{4\times4} | [\boldsymbol{\xi}]_\times \in \mathbb{R}^6 \right\} \tag{2.42}$$

here $[\boldsymbol{\xi}]_\times = \begin{bmatrix} [\boldsymbol{\phi}]_\times & \boldsymbol{\rho} \\ \mathrm{O}^T & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}$. $\boldsymbol{\phi}$ and $\boldsymbol{\rho}$ two 3-dimensional vectors.

2.3.4 Solver

In Section 2.3.2, different types of factor graphs are built to describe the tracking and mapping process as introduced in Equation 2.34. To obtain accurate camera poses and landmarks, we minimize the loss function in an iteration converge module

$$\mathbf{x}^* = \operatorname{argmin} \mathcal{H}(\mathbf{x}) \tag{2.43}$$

where the set of variables of the factor graph is denoted as §. Since those factors involve the non-linear rotation variables, the nature of the loss function is a non-linear least-squares problem. The function $f(\S)$ in Equation 2.43 can be approximated via the first order Taylor expansion at $\S_0$, written as

$$\mathcal{H}(\mathbf{x}_k + \delta\mathbf{x}) = \mathcal{H}(\mathbf{x}_k) + J(\mathbf{x}_k)\delta\mathbf{x} \tag{2.44}$$

$$z(x) = f(x_{k+1}) + g_k(x - x_{k+1}) + \frac{(x - x_{k+1})^T G(x - x_{k+1})}{2} \tag{2.45}$$

here $G = \Delta^2 f(x_{k+1})$ and $g_{k+1} = \Delta f(x_{k+1})$.

As introduced in Figure 2.9(a), there are n poses and m landmarks in the graph, where the re-projection factors only connect parts of vertices based on observation. Therefore, the structure of the problem is sparse as shown in Figure 2.9(b).

Gauss-Newton Solver The Gauss-Newton algorithm is a variance of the Newton method that has a faster convergence speed in unconstrained optimization problems, therefore before analyzing the problems of the Gauss-Newton method, the Newton optimization method is revisited first.

For the Gaussian solution, the extreme point can be obtained by taking the derivative of the above formula and making the derivative equal to zero.

$$\Delta z(x) = \Delta f(f(x_{k+1})) + G(x - x_{k+1}) = 0 \tag{2.46}$$

If G^{-1} is a non-singular matrix, then $x_{k+1} = x_k - G^{-1}g_{k+1}$, so that iterative update can be achieved. As mentioned before, the convergence speed of Newton's method is similar to the second order. However, when this method encounters the BA problem, solving the Hessian matrix requires a huge workload. At the same time, there is no guarantee that the Hessian matrix of the objective function exists at every iteration point. All can maintain righteousness. In order to take advantage of the advantages of Newton's method and avoid its disadvantages, BA solution methods such as the Gauss-Newton method have emerged.

The main idea of this method is to perform Taylor expansion at the iteration values, and then use the expansion to approximate the target equation. Therefore, the nonlinear problem can be transformed into a linear problem, and the new iteration point can be obtained by finding the minimum point of the quadratic model. Through multiple iterations, until a satisfactory accuracy is achieved. Perform first-order Taylor expansion on the formula 2.44, here we only expand the part of the feature points in the formula.

Normally, $\mathbf{H}\mathbf{d} = \mathbf{J}(x)^{\mathsf{T}}F(x)$, here $\mathbf{H}$ is Hessian Matrix, and $\mathbf{H} = \mathbf{J}(x)^{\mathsf{T}}\mathbf{J}(x) + \sum F_i(x)^2$. In the Gauss-Newton method, the second term is ignored, and the method only makes use of $\mathbf{J}(x)^{\mathsf{T}}\mathbf{J}(x)$ as Hessian Matrix. Therefore, it can be seen that the algorithm lacking high-order derivative information must satisfy the positive definite condition of the Hessian matrix in the iterative solution process and then requires the full rank of the Jacobian matrix. Otherwise, the algorithm fails. The Jacobian matrix is the representation between the camera and the three-dimensional feature points, which changes as the input parameters change. Therefore, the requirement for the Jacobian matrix becomes the requirement for the initial value of the observation, which is the fundamental reason why the Gauss-Newton method is sensitive to the initial value.

Leverberg-Marquart Solver. When we use $\mathbf{J}^{\mathsf{T}}\mathbf{J}$ to take the place of the Hessian Matrix, the bundle adjustment problem is often encountered from convergence issues. To solve the problem, Leverberg-Marquart was proposed to solve bundle adjustment models. The target function of Leverberg-Marquart is written as

$$(J^{\mathsf{T}}J + \lambda I)d_k^{LM} = -J^{\mathsf{T}}g_k \tag{2.47}$$

here $\mathbf{I}$ shows an identity matrix, and λ is a damped value. Therefore, the singularity issue of Gauss-Newton will be solved. There are two parts on the left of Equation 2.47. The method will automatically modify the ratio between $J^{\mathsf{T}}J$ and λI during the iterations. When λ is much bigger than $\mathbf{J}^{\mathsf{T}}\mathbf{J}$, Equation 2.47 can be regarded as a gradient descent algorithm, namely

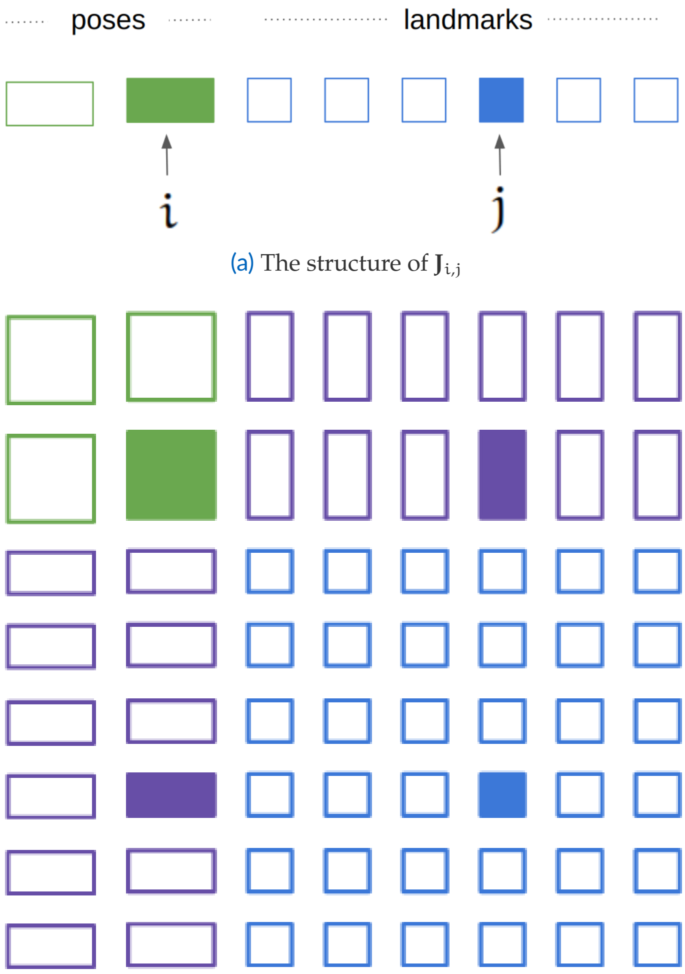
$$d_k^{LM} = -\frac{1}{\lambda}J^{\mathsf{T}}g_k \tag{2.48}$$

Similarly, if λ is much smaller, then the method can work as the Gauss-Newton method. Therefore, the LM method can be regarded as a combination of Steepest Descent and Gauss-Newton.

Sparsity in Jacobian and Hessian matrices. As introduced in the previous sections, the Jacobian matrix $\mathbf{J}$ computation is important in the iterative updating process. First, a connection between the j^{th} point landmark and the i^{th} camera pose will generate a residual error $\mathbf{e}_{ij}$, which means that the j^{th} point landmark is observed by the i^{th} view as shown in Figure 2.8. Therefore, the residual function only involves two vertices, namely $\mathbf{T}_{i,w}$ and $\mathbf{P}_w^j$. The Jacobian matrix generated by this connection can be denoted as

$$\mathbf{J}_{i,j}(\mathcal{X}) = \begin{bmatrix} \mathbf{0}_{2\times6} & \cdots & \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{T}_{i,w}} & \cdots & \mathbf{0}_{2\times3} & \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{P}_w^j} & \cdots & \mathbf{0}_{2\times3} \end{bmatrix} \tag{2.49}$$

here since other vertices have no relationship with this residual connection, those vectors besides $\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{T}_{i,w}}$ and $\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{P}_w^j}$ are zero matrices.



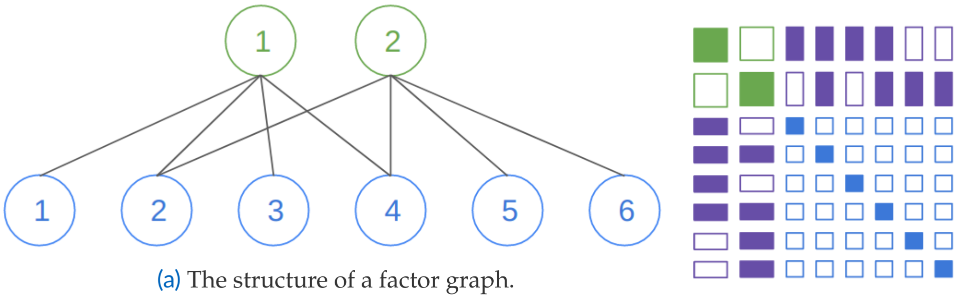
(a) The structure of $\mathbf{J}_{i,j}$

(b) The structure of $\mathbf{J}_{i,j}^T\mathbf{J}_{i,j}$

Figure 2.8 The sparsity of Jacibian and Hessian matrices generated by $\mathbf{e}_{ij}$. White cubes are zero blocks.

Following the Equation 2.47, $\mathbf{J}_{i,j}$ can be used to build a Hessian Matrix via the $\mathbf{J}_{i,j}^T\mathbf{J}_{i,j}$ operation, and the non-zero blocks are located at the position of (i,i), (j,j), (i,j) and (j,i) as shown in Figure 2.8. Since the sparsity of Jacobian matrices, the matrix $\mathbf{J}_{i,j}^T\mathbf{J}_{i,j}$ will remain the sparsity when we merge all residual connections.

Typically, factor graphs are constructed from hundreds of landmarks and dozens of poses, and it is not easy to directly solve such a large matrix in every iteration. However, based on



(a) The structure of a factor graph.

Figure 2.9 The structure of factor graph and sparsity of Hessian matrix generated by the graph. White cubes are zero blocks.

the sparse structure of the Hessian matrix, the computational burden will be greatly reduced. Let denote the $\begin{bmatrix} \mathbf{B} & \mathbf{A} \\ \mathbf{A} & \mathbf{C} \end{bmatrix}$, the Equation 2.47 can be rewritten as

$$\begin{bmatrix} \mathbf{B} & \mathbf{A} \\ \mathbf{A}^{\mathsf{T}} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{\mathrm{T}} \\ \delta\mathbf{x}_{\mathrm{P}} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_{\mathrm{T}} \\ \mathbf{g}_{\mathrm{P}} \end{bmatrix} \tag{2.50}$$

here $\begin{bmatrix} \delta\mathbf{x}_{\mathrm{T}} \\ \delta\mathbf{x}_{\mathrm{P}} \end{bmatrix}$ are required to update in iterations.

By using the Schur Elimination operation, Equation 2.50 can be transferred to

$$\begin{bmatrix} \mathbf{B} - \mathbf{A}\mathbf{C}^{-1}\mathbf{A}^{\mathsf{T}} & \mathbf{0} \\ \mathbf{A}^{\mathsf{T}} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{\mathrm{T}} \\ \delta\mathbf{x}_{\mathrm{P}} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_{\mathrm{T}} - \mathbf{A}\mathbf{C}^{-1}\mathbf{g}_{\mathrm{P}} \\ \mathbf{g}_{\mathrm{P}} \end{bmatrix} \tag{2.51}$$

Therefore, we can obtain the relationship for $\delta\mathbf{x}_{\mathrm{T}}$ by using

$$[\mathbf{B} - \mathbf{A}\mathbf{C}^{-1}\mathbf{A}^{\mathsf{T}}]\delta\mathbf{x}_{\mathrm{T}} = \mathbf{g}_{\mathrm{T}} - \mathbf{A}\mathbf{C}^{-1}\mathbf{g}_{\mathrm{P}}. \tag{2.52}$$

Since $\mathbf{C}$ is a diagonal matrix as illustrated in Figure 2.9, it is efficient to compute $\mathbf{C}^{-1}$. After obtaining $\delta\mathbf{x}_{\mathrm{T}}$, the $\delta\mathbf{x}_{\mathrm{P}}$ can be computed via the following function

$$\delta\mathbf{x}_{\mathrm{P}} = \mathbf{C}^{-1}(\mathbf{g}_{\mathrm{P}} - \mathbf{A}^{\mathsf{T}}\delta\mathbf{x}_{\mathrm{T}}). \tag{2.53}$$

In landmark optimization process, there is an over-parametrization issue that we have to face with. For example, a plane in 3D space $\mathbb{R}^3$ has only 3 degrees of freedom, which means that only 3 parameters can represent a plane, whrereas the Hesse form $\boldsymbol{\imath} = [\mathbf{n}, d]$ has 4 parameters.

The number of parameters is more than the actual representation of degrees of freedom, which is called the over-parameterization issue.

For the Gauss-Newton optimization solver, if it is an over-parameterized form, the Hessian matrix calculated during the optimization process will not be of full rank, so the proposed $\mathbf{J}^{\mathsf{T}}\mathbf{J}$ cannot be inverted for solving the problem. Even, the problem can be addressed by the Levenberg-Marquardt algorithm, we still have a convergence problem to deal with. Since the cost function is flat in some directions, there are infinitely many solutions, which will make the convergence rate of the algorithm become the linear convergence rate of the gradient descent method, rather than the quadratic convergence rate of the Gauss-Newton method, so it will lead to Convergence becomes slower. Therefore, we will introduce proper parametrization approaches for primitives implicated in this section.

Parameterization is an important step if we want to optimize a type of landmark in the factor graph. First of all, most of them focus on how to represent point, line, and plane features in the optimization modules. Inverse depth parameterization improves numerical stability compared with the XYZ method. Plücker and Orthonomal line representation are proposed to represent a line in the tracking and optimization process, respectively, where Orthonormal solves the over-parametrization issues in the optimization step by using 4 parameters to represent 3D lines. And then, for structural regularities, implicit methods [71, 14] add more structural constraints at the end of re-projection residual loss functions of individual landmarks, while the explicit strategies choose to parameterize those constraints as a type of primitive and feed them into optimization tasks directly. Generally, there are two advantages to explicit strategies. On the one hand, there are fewer parameters will be used in optimization modules. On the other hand, the binding force of explicit methods is stronger than implicit constraints.

2.4 Deep Neural Networks

Deep neural networks have shown impressive performances in the computer vision community. Compared with multiple view geometry, those networks are good at high-level tasks, including plane segmentation, single-RGB depth/normal prediction, and scene completion, by training large-scale parameterized pipelines with massive data. To break the bounds of traditional SLAM/VO methods, neural networks can be leveraged in tracking, mapping, and scene understanding modules. Based on the surface normal prediction [8] and plane instance segmentation [9] networks, new types of data can be leveraged into SLAM systems.

First, we introduce the basic operators and blocks, including convolutional and fully-connected layers, pooling, and dropout operations in Section 2.4.1, and then fundamental blocks in Section 2.4.2.

2.4.1 Basic Operators

Convolutional layer. A convolutional layer is typically composed of several 2D/3D filters, and the weights of each filter are learned via back-propagation during training processes. The filters then slide over the input data in a specified stride along different dimensions and convolve with a local patch of the input data at each position. Therefore, 2D and 3D filters form 2D feature maps and 3D feature volumes, respectively.

For 2D convolutional layers, the expression to generate a single convolution output for a single 2D position (u, v) with value $f(u, v)$ can be expressed as

$$g(u, v) = \sum_{d_u=-k_u}^{k_u} \sum_{d_v=-k_v}^{k_v} \omega(d_u, d_v) f(u - d_u, v - d_v) \tag{2.54}$$

here $g(u, v)$ is the result and ω is weight of the convolution kernel. Those elements are changed in the range of $[-k_u, k_u]$ and $[-k_v, k_v]$ in the filter kernel. The size of the receptive field is determined by the kernel size k_x. When the kernel size is larger than a single element, some deep features are extracted based on multiple local elements.

Pooling layer. Pooling operation is a typical post-process operation after convolutional neural layers, simulating the human visual system to reduce the data dimension and extract dominant features. For example, a max pooling layer [72] divides each input feature map into rectangular areas and filters out features that are not the maximum within a rectangular neighborhood. Based on this operation, the network parameters and calculation costs are reduced, while the over-fitting issue will reduced as well. The use of pooling layers, along with convolutional layers, has led to significant improvements in performance in many computer vision tasks such as object detection [73], segmentation [74], and classification [75].

Similarly, Average Pooling is the average of the values in the locally accepted domain. Stochastic pooling is proposed by [76] to the effect that it only needs to randomly select the elements of the feature area according to their probability values, and the probability of being selected with a significant element value is also considerable. Local Importance-based Pooling proposes automatically learning the importance through a sub-network based on input features. It can adaptively determine which features are more critical while automatically enhancing identifying features during sampling. The idea is to learn a map similar to attention on the original feature map, then perform a weighted average with the original image. The sampling interval here is actually fixed, which does not meet the first item of the above description. However, the author believes that the deformed receptive field can be realized since the importance is variable. Max pooling is a common type of pooling operation where the maximum value within each pooling region is taken as the output value for that region.

Fully-connected layer. Fully-connected layers, also known as dense layers, are a common type of layer used in neural networks. In a fully connected layer, every neuron of the layer is connected to every neuron in the previous layer, creating a dense matrix of weights that

$$x \longrightarrow \mathrm{fc}_{\mathrm{dim}^{\mathrm{in}},\mathrm{dim}^{1}} \rightarrow \text{bias} \longrightarrow h \rightarrow \mathrm{fc}_{\mathrm{dim}^{1},\mathrm{dim}^{2}} \rightarrow \text{bias} \longrightarrow h \rightarrow \mathrm{fc}_{\mathrm{dim}^{2},\mathrm{dim}^{3}} \rightarrow \text{bias} \longrightarrow h \xrightarrow{\qquad} y$$

Figure 2.10 A multi-layer perceptron pipeline composed by three fully-connected layers together with biases and non-linear activation.

can be trained during the learning process. In a neural network architecture, fully connected layers are typically used after one or more convolutional or pooling layers to analyze higher-level features extracted from input data such as images or audio signals. The previous layer's output is flattened into a 1D vector and fed as input to fully connected layers.

The neurons in a fully connected layer compute a weighted sum of their inputs, adding a bias term and an activation function. The activation function introduces non-linearity to the learned model and enables it to learn complex relationships between features in the input data. Fully-connected layers are often used as the final layer in a neural network model to produce classification or regression outputs. For example, in an image classification task, a fully connected layer can take the output of a convolutional layer and output a probability distribution over the possible classes.

Activation functions. Activation functions are non-linear mathematical operations applied to the output of a neuron, which introduces non-linearity into networks and enables them to model complex relationships between inputs and outputs. There are several types of activation functions used in neural networks, including ReLU [77], Sigmoid [78] and Softmax [79]. ReLU is a rectified linear unit function that sets all negative input values to zero and leaves positive input values unchanged. It is computationally efficient and has been shown to perform well in many deep-learning applications. A softmax function maps any input vector to a probability distribution over classes, making it useful for multi-class classification tasks, and a sigmoid function maps any input value to a value between 0 and 1, making it useful for binary classification tasks. However, it can suffer from vanishing gradients when the input values are too large or too small.

2.4.2 Architecture

By implementing some inference models in deep architectures, layer-by-layer operations are used, as mentioned in Section 2.4.1. The performance of deep architectures can surpass human expert performance usually because of its large-scale parametric model, where different operators should be carefully connected to form some standard modules. In the following paragraphs, we illustrate some basic architectures including

Multi-layer perceptron (MLP). A multilayer perceptron (MLP), such as the one shown in Fig. 2.10, consists of three fully connected layers, each with a nonlinear activation function at its output. Ignoring forward-pass data with batch sizes greater than 1, a single input to an MLP is always flattened into a vector, with each element in the vector connected to each element in the next layer with a certain weight.

The output dimension and the number of layers are the two most important factors in MLP. As shown in Figure 2.10, $\mathtt{dim}^{\text{in}}$ and $\mathtt{dim}^3$ are determined by the task. The sizes of $\mathtt{dim}^1$ and $\mathtt{dim}^2$ are hyperparameters that can be flexibly designed based on factors such as expected inference accuracy and efficiency. Overall, a multilayer perceptron (MLP) represents a function $y(x;\Theta)$, parameterized by the weights Θ and their biases in fully connected layers. Generally, deeper MLPs make the model more capable of adapting to the training data.

Convolutional neural networks. Convolutional neural networks (CNN) can be regarded as regularized versions of MLP, which connect each neuron in one layer to all neurons in the next. The complete connectivity between these layers makes MLPs prone to over-fitting data. However, the situation changes in CNNs where regularization is performed by using the hierarchical pattern in data and assembling those increasingly complex patterns. And those patterns embossed in their filters are smaller and simpler. One of the impressive works of CNN applications is LeNet-5 [80], which uses seven convolutional layers to classify digits from an image.

Tracking and Mapping in Structural Scenes | 3

Following Chapter 2 introducing fundamental theories related to general SLAM methods, this chapter continues to illustrate necessary theories and technologies used in tracking, mapping, and optimization modules in structural scenes. In this chapter, we first revisit the recent history of this field in Section 3.1, and introduce the data flow of a point-line SLAM from the front-end to the back-end modules on a simulated scene in Section 3.2, which helps describe how to organize fundamental theories in SLAM software. Furthermore, we continue introducing strategies for using structural regularities in pose estimation in Section 3.3, and 3D reconstruction in Section 3.4.

3.1 Recent History in Structural SLAM

In many scenarios, SLAM and VO methods introduced in Section 2.1.4 have shown impressive performance as general trackers. However, these methods still suffer from tracking loss in some man-made regions where insufficient point features are detected. Even though those areas are only a tiny part of the scene, the problem of tracking loss still dramatically reduces the robustness performance of the whole system.

Traditionally, monocular SLAM methods are considered extremely limited in information perception, especially in indoor environments with several low/non-textured regions. Although many non-textured regions exist in manufactured environments, we can also find other types of landmarks, such as lines and planes, which wildly exist on furniture surfaces, ceilings, floors, and walls. We agree that adding line features, such as PL-SLAM [81, 82], helps to increase information for tracking, but the ratio of information enhancement is minimal. Therefore, we can summarize the dilemma of monocular SLAM methods as the effective use of input information is low, which will be in trouble when the information usage gets lower in low-textured scenarios.

Compared with monocular-SLAM methods, stereo, and RGB-D approaches are more robust in the same evaluation environments since an accurate local/global map is easier to reconstruct to support map-to-frame tracking strategies [13, 54]. In some scenarios, such as corridors, where features are centered at a part of regions of images, those methods based on features still face enormous challenges.

Taking the fact that 3-DoF translation estimation is a linear problem with given known 3-DoF rotation [83], some approaches [67, 84] proposed to decouple the 6-DoF pose estimation into rotation and translation estimation via structural assumptions, like Manhattan/Atlanta World Assumptions [8, 85], based on perpendicular/orthognal lines and planes. At the same time, based on an estimated rotation matrix, fewer points are required to compute the translation motion, leading to more robustness in low-textured regions. A line of work exploits additional environmental constraints and regularities to improve pose estimation and mapping performance. When only planes are used for the rotation estimation as in OPVO [84], at least two orthogonal planes are required to be detected in each frame; the addition of vanishing points extracted from lines can be used alternatively, as done in LPVO [66]. Based on the assumption of environments, the situation of error accumulation will be improved. Since those methods do not consider parametrizing those structural regularities in optimization modules, it is challenging to optimize scene representations, camera poses, and structural regularities at the same time.

In contrast to sparse point clouds widely used in general SLAM systems, some dense mapping algorithms showing better visualization performance are designed for indoor scenes. By building a volume to an environment, elements of the scene are voxelized, which can be used to build an occupancy map [86] that is important in navigation for robots and autonomous driving vehicles. In addition, each voxel can also be used to record some implicit parameters, like sdf [87] and tsdf [45]. Based on those implicit voxels, we can mesh surfaces via raycasting [45] or marching cube [88] algorithms. Meanwhile, the geometric abstraction method [89] based on structure priors is proposed for lightweight model generation instead of reconstructing the visual scenes as the same as the physical ones.

3.2 Data Flow Analysis In Point-Line SLAM

In this section, we introduce how to connect those theories and modules in a complete tracking and mapping pipeline, PointLine-SLAM [1], as shown in Figure 3.1. Then, by analyzing the data flowing from one module to another, it will be easier to find the advantages and disadvantages of different modules, providing a systemic perspective in SLAM system design.

3.2.1 Data flow Overview

Inputs. As shown in Figure 3.1, the pipeline starts from point-line measurements directly by skipping the process of feature detection to remove the randomness of the detection and matching process. Each point and endpoint of lines have their corresponding depth, which means that 3D point and line landmarks in the camera coordinates can be obtained directly

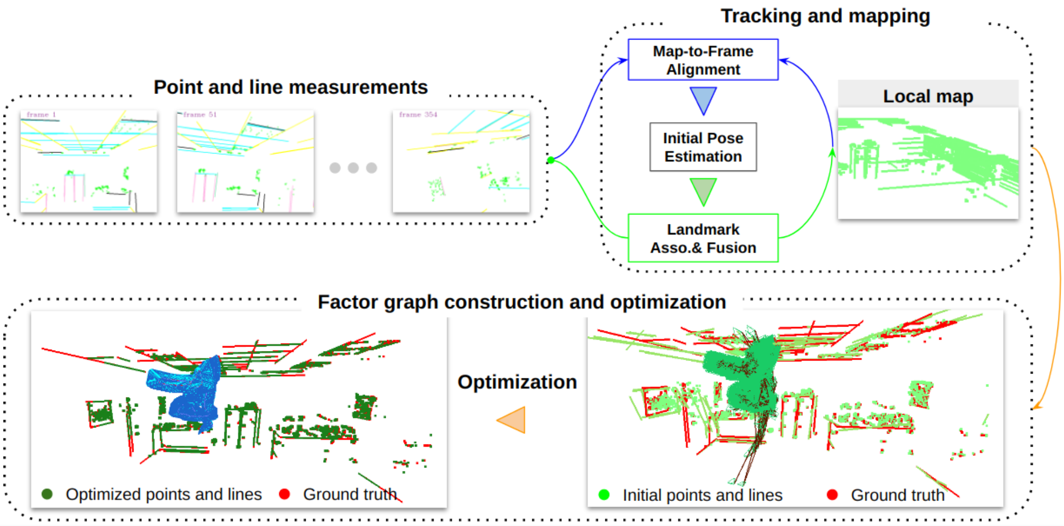


Figure 3.1 Architecture of tracking and sparse mapping system. In the "Point-Line Measurements" window, points and lines are green and red. In the co-visibility graph optimization window, dark blue, light blue, and red figures present optimized, initial, and ground truth, respectively.

via the intrinsic matrix **K**. Similar to those real measurements obtained from raw RGB-D images, the measurements also have noise. And the noise model follows the method in ICL-NUIM [12].

System Initialization. Since the 3D measurements are fed to the system, the initialization module is used to activate the world coordinate at $\begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}$, which is also the pose of the first frame. Furthermore, in the initialization module, we also have to activate the local map by feeding the point and line measurements, $\mathbf{p}$ and $\mathbf{l}$, of the first frame to the world coordinate.

Incremental tracking. After activating the coordinate systems and the local map, the pose $\mathbf{T}_{c_i,w}$ of the new frame F_{c_i} newly fed to the system can be estimated via Frame-to-Frame or Map-to-Frame strategies by build correspondences between two frames or landmarks-frames, which the map-based on method is more robust since the number of visual landmarks collected in the map are no less than last frame's.

The process of associating current frames' measurements with 3D landmarks is essential because the process affects two modules. First, the association's performance determines the performance of the initial pose estimation as introduced in Section 2.2.4. If there are some outliers in those matches, using the PnP method to compute the camera pose is challenging. Therefore, some RANSAC-based methods are proposed to remove those outliers in the iterative estimation process. Second, The association relationships will be used in co-visibility graphs for building re-projection factors, which are critical to the optimization process.

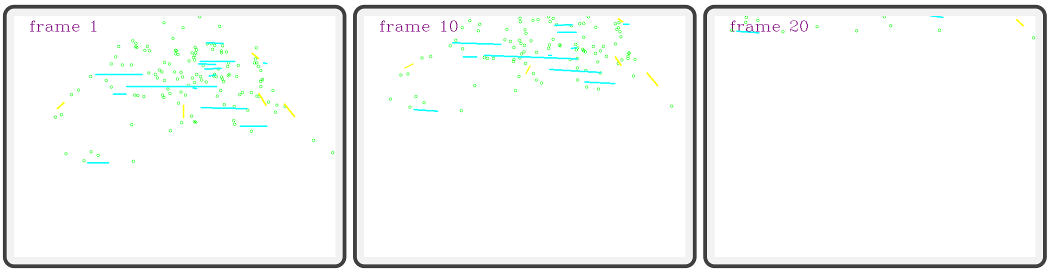


Figure 3.2 Measurements of points and structural line segments. Parallel lines are marked in the same color. Black shows individual lines.

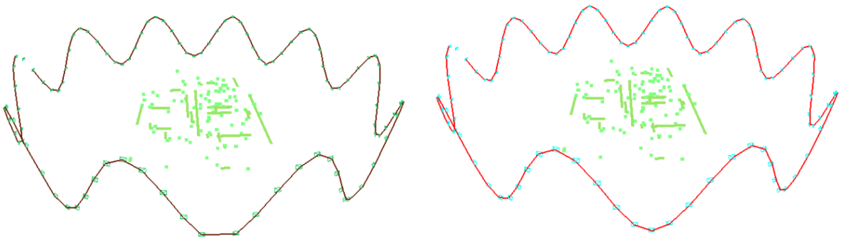


Figure 3.3 Initial trajectory estimates (left) and ground truth (right).

In this point-line SLAM system, the observation is simulated, where we give the same ID to those correspondences of different frames. The ID of each measurement looks like their descriptor vectors that can determine matches and co-visible connections.

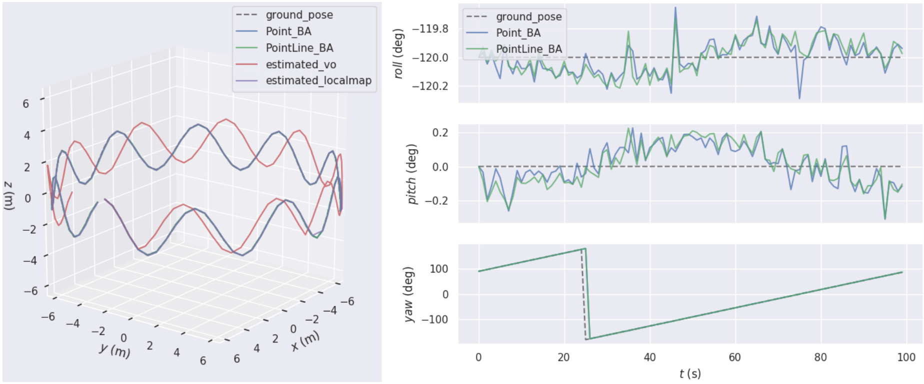
Sparse reconstruction and updating. Since former views already detect a part of the measurements detected in the new frame, the PnP algorithm can be used for camera pose estimation of this frame. One of the biggest problems resulting in tracking loss is caused by no co-visible relationships anymore. Another part of them is newly detected; otherwise, this new frame cannot provide new information anymore. Therefore, for most frames, those two types of measurements should be dealt with in two manners. New measurements are initialized and merged into the world coordinate, while measurements that have been detected before are passed through a landmark fusion module. In ORB-SLAM [54], each point landmark in the map saves a descriptor vector for the coming landmark fusion steps.

3.2.2 Front-end

After obtaining measurements and observing relationships between landmarks and camera poses, initial pose estimation and sparse mapping are implemented incrementally, as shown in Figure 3.2 and 3.3.

First, when a new group of measurements is fed to the system, initial 6-DoF transformation $\mathbf{T}_{w,c_i}$ is estimated based on the RANSAC-based PnP method [65]. Furthermore, the 3D points and lines are reconstructed and fused based on measurements and initial camera poses.

Given 2D measurements, including points and lines, and related depth information as shown in Figure 3.2, a sparse 3D map based on point and line landmarks can be initialized.



(a) Trajectory results

(b) Rotation results

Figure 3.4 Comparison in trajectory results of initials, Point-BA, and PointLine-BA methods in the sequence.

Furthermore, the initial map can be used to estimate the current camera pose when a new image is fed to the system based on the map-to-frame strategy, which is implemented based on 3D-2D alignment as introduced in Section 2.2.2. After getting the initial camera pose, measurements of the current frame are used to build the model. Those measurements are fused to the map for matched features, while those new measurements will be used to initialize landmarks.

As introduced in previous chapters, point-only visual odometry systems are sensitive to low-textured scenarios, like the 20^{th} frame, in Figure 3.2. The phononomen in Figure 3.3 shows that the Map-to-Frame pose estimation method cannot estimated accurate initial camera poses under those challenging scenes.

3.2.3 Back-end

General SLAM systems [55, 15, 14] have achieved impressive performances. However, they still need help in challenging scenarios and sharp camera motions. To solve those problems, algorithms are proposed to update parts of those core modules, such as structural regularities [8], parameterization [35], and new factor loss functions [14], of those popular SLAM systems. Inspired by those SLAM methods, the initial camera poses estimated based on the Map-to-Frame strategy are refined via point [13] and line [90] optimization methods.

Co-visibility factor Graph Construction

The vertices in this graph $\mathcal{G}$ contain camera poses $\mathcal{V}_{pose}$, point landmarks $\mathcal{V}_p$, and line landmarks $\mathcal{V}_l$. To be specific, camera pose $\mathbf{T}_{w,c_i} = \begin{bmatrix} \mathbf{R}_{w,c_i} & \mathbf{t}_{w,c_i} \\ \mathbf{0} & 1 \end{bmatrix}$, where $\mathbf{T}_{w,c_i} \in SE(3)$, $\mathbf{R}_{w,c_i} \in SO(3)$ and $\mathbf{t}_{w,c_i} \in \mathbb{R}^3$. Points used in the optimization module is parametrized as

$\mathbf{P}_w^k = \begin{bmatrix} x^k & y^k & z^k \end{bmatrix}^T$, and line landmarks are represented in Plücker Parameetrization [36] as $\mathcal{L}^k = \begin{bmatrix} w_n \mathbf{n}^k & w_d \mathbf{d}^k \end{bmatrix}$ where $\mathbf{n}^k$ and $\mathbf{d}^k$ are unit vectors, and the first one is the normal vector of the plane built by camera center c_i and endpoints of the k^{th} line, while the second one is a direction vector. In the iterative optimization process, we transfer the Plücker Parametrization to Orthonormal Representation.

Factor Graph optimization

In the architecture of the Co-visibility Factor Graph (CFG), we evaluate Point BA [54] and Point-Line BA [90]. For fair comparison, the input to those optimization modules is the same.

Point BA. Following ORB-SLAM2 [54], Point BA makes use of the Euclean parametrization to represent point landmarks and the loss function is based on re-projection errors between re-projected points and pixel measurements.

Based on the point feature measurement model, the measurement of the i^{th} global point landmark $\mathbf{P}_i = [\begin{matrix} x_i & y_i & z_i \end{matrix}]^T$ at frame c_j is represented as $\mathbf{p}_k^j = [\begin{matrix} u_k^j & v_k^j \end{matrix}]^T$ in the normalized coordinate, and the re-projection factor of a point feature is defined as $\mathbf{r}_\mathbf{p}(\mathbf{p}_k^j, \mathbf{P}_i, \mathcal{X})$ where

$$\mathbf{r}_\mathbf{p}(\mathbf{p}_k^j, \mathbf{P}_i, \mathcal{X}) = \mathbf{K}\overline{\mathbf{R}_{w,j}^T(\mathbf{P}_i - \mathbf{t}_{w,j})} - \mathbf{p}_k^j \tag{3.1}$$

here $\overline{(\cdot)}$ is to compute the normarlized coordinate. The corresponding Jacobian matrices $\mathbf{J}_\mathbf{p}$ and $\mathbf{J}_\mathcal{X}$ can be obtained respectively for updating landmarks and camera states according to Equation 3.1.

Point-Line BA. Following PL-VIO [90], the Plück presentation is used for re-projection error computation, and the Orthonormal method is used for iteration optimization steps.

Traditionally, the mapline $\mathcal{L}_w$ in the world coordinate is represented by

$$\mathcal{L}_{c_i} = \begin{bmatrix} \mathbf{n}_{c_i} \\ \mathbf{d}_{c_i} \end{bmatrix} = \mathcal{T}_{c_i,w} \begin{bmatrix} \mathbf{n}_w \\ \mathbf{d}_w \end{bmatrix} \tag{3.2}$$

here $\mathcal{T}_{c_i,w} = \begin{bmatrix} \mathbf{R}_{c_i,w} & [\mathbf{t}_{c_i,w}]_\times \mathbf{R}_{c_i,w} \\ \mathbf{0} & \mathbf{R}_{c_i,w} \end{bmatrix}$, and $[\cdot]_\times$ is the skew-symmetric operation.

On the image plane, the reprojected endpoints, $\mathbf{p}_s$ and $\mathbf{p}_e$, of a 3D line in the camera coordinate can be obtained by

$$\bar{\mathbf{p}}_k = \mathrm{K}\bar{\mathbf{P}}_k, k \in (s, e) \tag{3.3}$$

here $\bar{\mathbf{P}}_k$ is a normalized 3D endpoint in the camera coordinate. Therefore, the reprojected line

$$\mathbf{l}_j = \bar{\mathbf{p}}_s \times \bar{\mathbf{p}}_e = K\bar{\mathbf{P}}_s \times K\bar{\mathbf{P}}_e = \mathcal{K}(\bar{\mathbf{P}}_s \times \bar{\mathbf{P}}_e) = \mathcal{K}\mathbf{n}^j, \text{ and } \mathcal{K} = \begin{bmatrix} f_y & 0 & 0 \\ 0 & f_x & 0 \\ -f_y c_x & -f_x c_y & f_x f_y \end{bmatrix}.$$

When the mapline $\mathcal{L}_w$ is detected by frame F_i and F_j, we can find the 2D correspondences lying on related images. The error between the re-projected line $\mathbf{l}^j$ and two endpoints, $\mathbf{p}_s$ and $\mathbf{p}_e$, of the extracted 2D line in the frame, can be written as

$$\mathbf{r}_l(\mathbf{p}^j_{k,s}, \mathbf{p}^j_{k,e}, \mathcal{L}_w, \mathcal{X}) = \begin{bmatrix} d(\mathbf{p}^j_{k,s}, \mathbf{l}^j) \\ d(\mathbf{p}^j_{k,e}, \mathbf{l}^j) \end{bmatrix} \tag{3.4}$$

where $d(\mathbf{p}^j_{k,s}, \mathbf{l}) = \frac{\mathbf{p}^{j\,T}_{k,s} \cdot \mathbf{l}}{\sqrt{l_1^2 + l_2^2}}$, and $\mathbf{l}^j = \begin{bmatrix} l_0 & l_1 & l_2 \end{bmatrix}^T$ is the 2D line re-projected from the j^{th} 3D mapline. $\mathbf{p}_k = \begin{bmatrix} \bar{x}_k & \bar{y}_k & 1 \end{bmatrix}^T$, $k \in (s, e)$, are the normalized coordinates of endpoints.

As shown in Figure 3.4a, benefitting from the local map, we can obtain more accurate camera poses compared with the frame-to-frame tracking strategy. Furthermore, the optimization modules based on points and lines achieve more smooth trajectories than visual odometry systems. However, when we focus on those two optimization modules, it is easy to find that constraints from line features improve the robustness of the point-based refinement step, but the distance in accuracy is not very large, as shown in Figure 3.4b.

3.3 Structural Regularities in Tracking

Before discussing structure-based tracking, let us first revisit the co-visibility factor graph introduced in Section 2.3.2. As shown in Equation 2.34, point and line landmarks are connected to views that detect those landmarks. Based on those observation relationships, the co-visibility graph optimizes landmarks and camera pose simultaneously. Obviously, in these graphs, the only types of relationships we need to know are observations between individual landmarks and their corresponding measurements.

For textured scenes, such as outdoors, where landmarks are evenly distributed in the scene, the optimization module tends to converge to good enough results from initial values. But the fact changes in indoor environments. On the one hand, features are measured in uneven situations. For example, extracting useful point features from textureless walls is problematic, resulting in insufficient observations for these factor graphs. On the other hand, these indoor environments commonly provide particular patterns rarely detected in the wild. Specifically, those widely discovered structural regularities can be summarized as follows:

- Collinear points: multiple points located on the same line;

- Coplanar points and lines: points and lines located on the same plane;
- Vanishing point: the intersection point of two-dimensional parallel lines;
- Atlanta World: all scene elements share a dominant direction;
- Manhattan world: those where the relationship between landmark elements is limited to parallel or perpendicular.

This section introduces three manners of usage in those structural regularities. The first is to compute additional residuals based on structural constraints (see section 3.3.1). Instead of using constraints in optimization modules, Section 3.3.2 provides the second strategy that directly codes structural priors into landmarks by proposing new representations for the optimization process. Finally, the third approach, illustrated in Section 3.3.3, makes a structure containing several landmarks into a primitive and optimizes the primitive directly rather than optimizing its landmarks.

3.3.1 Structural constraints.

Collinear points. In direct methods, like the direct visual odometry DSO [17], point-based photometric errors are critical constraints for the refinement process in a sliding window, which results in less robust depth estimation issues since those point landmarks are dealt with individually. In the DPLVO [71] method extended from DSO, line segments are extracted using the LSD algorithm. Then, points are sampled from those 2D line segments, while the length of each line decides the size of sampled points.

In the tracking process, the depths of those collinear points are estimated as those ordinary points. And then, these depths are exploited in 3D line initialization. Based on the Plücker representation $\mathcal{L} = \begin{bmatrix} \mathbf{n} & \mathbf{d} \end{bmatrix}$, the residual vector $\mathbf{r}_{col}$ of the collinear constraint on point $\mathbf{P}$ is built by

$$\mathbf{r}_{col} = \mathbf{n} - \mathbf{P} \times \mathbf{d} \tag{3.5}$$

here the shape of $\mathbf{r}_{col}$ is 3×1.

Furthermore, the constraints are used in the windowed optimization module with traditional photometric constraints.

Parallel and perpendicular planes. Given depth maps, plane features can be extracted and generally represented in the Hessian form $\boldsymbol{\pi} = [\mathbf{n}, d]$. However, the parameterization approach has an over-parameterization issue and cannot be used directly in optimization modules. To address the problem, PS-SLAM [91] makes use of minimal parameterization $\boldsymbol{\tau} = [\theta, \phi, d]$ of planes in optimization, here θ and ϕ are the azimuth and elevation angles of the normal vector respectively. Therefore, the relationship between $\boldsymbol{\tau}$ and $\boldsymbol{\pi}$ can be denoted as

$$\boldsymbol{\tau} = q(\boldsymbol{\pi}) = (\theta = \arctan\frac{n_y}{n_x}, \phi = \arcsin n_z, d) \tag{3.6}$$

here $\mathbf{n} = \begin{bmatrix} n_x & n_x & n_z \end{bmatrix}$, and the azimuth and elevation angles are restricted in $(-\pi, \pi]$ to avoid the singularities situation in optimization.

Since PS-SLAM is extended from ORB-SLAM2 [54], the PS-SLAM system manages all extracted planes in the local map thread inherited from ORB-SLAM2's architecture. Therefore, the method continues to analyze the parallel or perpendicular constraints between those planes saved in the map based on the structural priors in indoor scenes. Based on the structure regularities, those planes closing in parallel or perpendicular relationships provide residuals for pose optimization.

Those constraints constructed based on plane normal vectors are written as follows

$$\begin{cases} \mathbf{r}_{pp} & = q_n(\mathbf{R}_{\perp}\mathbf{n}_{c_i}) - q_n(\mathbf{R}_{c_i,w}\mathbf{n}_w) \\ \mathbf{r}_{po} & \quad = q_n(\mathbf{n}_{c_i}) - q_n(\mathbf{R}_{cw}\mathbf{n}_w) \end{cases} \tag{3.7}$$

here the rotation matrix $\mathbf{R}_{\perp}$ is used to rotate the normal to the same direction of $\mathbf{R}_{c_i,w}\mathbf{n}_w$, $\mathbf{r}_{pp}$ and $\mathbf{r}_{po}$ are residuals for perpendicular planes and parallel planes, respectively. Given depth maps, plane features can be extracted and generally represented in the Hessian form $\boldsymbol{\pi} = (\mathbf{n}, d)$. However, the parameterization approach has an over-parameterization issue and cannot be used directly in optimization modules. To address the problem, PS-SLAM [91] makes use of minimal parameterization of planes in optimization $\boldsymbol{\tau} = (\theta, \phi, d)$, where θ and ϕ are the azimuth and elevation angle of the normal respectively.

co-planarity constraints. In the last paragraph, parallel and perpendicular constraints between planes are explored in pose optimization. Here, we introduce co-planarity constraints that are generated from those points and lines located on the same plane. In indoor scenes, points and lines are commonly detected on the surfaces of floors, ceilings, and the surface of furniture, where most of those regions are in planar shapes.

For the co-planar regularities in points, $\mathbf{r}_{cop}$ can be built based on a landmark $\mathbf{P} \in \mathbb{R}^3$ and a plane $\boldsymbol{\pi} = [\mathbf{n} \ d]$. The residuals $\mathbf{r}_{cop}$ can be obtained based on the distance between the point and the plane via the following function,

$$r_{cop} = \mathbf{n}^T \cdot \mathbf{P} - d \tag{3.8}$$

here the distance residual r_{cop} is scalar.

In Kimera-VIO [92], the parameters of planes are refined and updated in the optimization process. To avoid the over-parametrization problems of planes, the method optimizes them in the tangent space $T_{\mathbf{n}}S^2$, based on the retraction operation, $\mathcal{R}_{\mathbf{n}}()$, changes in the tangent space are mapped to changes of the normals in $\mathcal{S}^2$. Therefore, the method optimize $[\mathbf{v}, d] \in \mathbb{R}^3$ directly.

In addition, Kimera-VIO also proposes a novel approach to reconstruct planar regions without using depth maps. First, the method uses 2D Delaunay triangulation to connect those keypoints tracked successfully in the latest frame. And then those 2D triangulations in the camera coordinate. Furthermore, geometric filters [93] are used to remove misrepresentative faces of the mesh. Following the idea of Kimera-VIO, PL-VIO [94] extended the point-based pipeline to lines and proposed co-planar constraints for line segments.

3.3.2 Landmark Structuralization

Compared with constraints introduced in Section 3.3.1, the big difference between those constraints and methods discussed in this section is that the structural regularities used in the following methods are explicit and stranger. In other words, **when we use those constraints, like coplanarity constraints, combined with other types of errors in joint optimization models, there is no guarantee that those points will lay on the corresponding planes during the optimization. But if the landmark is using a new structural representation to replace constraints. The coplanarity constraint will be satisfied during the whole refinement stage.** Under the assumption of structural regularities, the process of parameterizing traditional landmarks by encoding structural cues is called **landmark structuring** in this section.

StructLine. For line landmarks, the widely used minimal representation in bundle adjustment modules is the Othoronormal representation, which does not consider structural regularities. By assuming that dominant directions exist in most indoor scenes, Struct-SLAM [95], proposes a new parametrization for lines aligning with those three dominant directions.

In StructLine, three orthogonal planes crossed at the origin of the world coordinate, the blue line is parallel to the normal vector of the ZX plane, and the interaction point between the line and the plane is denoted as A. Then, we also could re-project the camera center O on the plane, and the position of the re-projected point O' is (C_a, C_b). Therefore, the blue line can be represented based on one of the dominant directions. And the minimal parameterization is $\mathbf{L} = \begin{pmatrix} C_a & C_b & \theta & h \end{pmatrix}^T$ where $\begin{bmatrix} C_a & C_b \end{bmatrix}^T$ where the θ is the angle of the ray through point A and O', while h shows the inverse depth of the distance between those two points. Based on the representation, those structural line landmarks are parallel with the related dominant directions in the whole process and do not need to add additional constraints in bundle adjustment modules.

After proposing a parametrization, a measurement and an update models are required to compute residuals and update parameters for solvers, as introduced in Section 2.4.2. First, since the proposed parameterization can be retracted to the Pücker representation, it is easy to build re-projection errors as traditional line landmarks. So, the core step for this structural parametrization is introducing the updating model.

Collinear Point constriant As introduced in Section 3.3.1, DPLVO [71] takes advantage of colllinear constraints in the windowed optimization module based on the architecture of DSO. In this section, the DPLVO method is updated by EDPLVO [96], which parameterizes each 3D point of the collinear group based on the line and inverse depth of the point. The representation process can be denoted as

$$\mathbf{P}_x = (\mathbf{A}_l^T \mathbf{A}_l)^{-1} \mathbf{A}_l^T \mathbf{b}_l \tag{3.9}$$

here $[\mathbf{d}_l]_\times$ and $[\mathbf{d}_x]_\times$ are the skew-symmetrics matrices of $\mathbf{d}_l$ and $\mathbf{d}_x$, respectively. Furthermore, those incorporating those lines in a photometric error model.

Since the number of collinear points is very large, when we use the shared parameters of the line and the unique inverse depth of each point, the number of variables in the Hessian Matrix can be significantly reduced. Meanwhile, the method satisfies the collinear constraint during the optimization, improving the accuracy.

3.3.3 Structural Primitives

In this thesis, **primitive** is the collective term for organized structures, which can be built by some basic landmarks like points, lines, and planes. Specifically, parallel lines lying on an image plane will cross at a point named Vanishing Point, and perpendicular/orthogonal planes/lines construct a well-known structure, Manhattan World. It is evident that compared with landmarks, primitives contain more structures of the environment. Therefore, those primitives tend to be observed by lots of frames, while few neighboring images generally observe individual landmarks.

Vanishing Point and Vanishing Direction Vanishing Point is a pixel position on image planes, representing relationships of a parallel line group. For example, there are a group of 3D parallel lines $\mathcal{L} = [\mathbf{L}_0, \ldots, \mathbf{L}_i, \ldots, \mathbf{L}_n]$ in the world coordinate. If point $\mathbf{p}_i$ is the cross point of those 2D parallel line measurements, the point can be associated with any lines of this group rather than considering its descriptor. The relationship between a vanishing point and a vanishing direction can be seen in Figure 3.5, namely, when we project the vanishing point $\mathbf{p}$ to the normalized camera coordinate and obtain $\hat{\mathbf{P}}$, the vanishing direction of $\mathbf{p}$ is $\frac{\hat{\mathbf{P}}}{||\hat{\mathbf{P}}||}$.

There are two main strategies to estimate vanishing direction. First, after reconstructing 3D line landmarks, clustering methods can be used to find parallel lines based on the constraints of direction angles since this method is affected by the quality of reconstructions. Another method is to extract parallel lines from 2D line segments having more robust performance. The J-Linkage algorithm [97] is widely used in vanishing point detection, which clusters the lines into several parallel line groups.

Since those clustering algorithms using only 2D information tend to generate incorrect line classification, in case of having depth inputs, a single-line RANSAC can be performed for

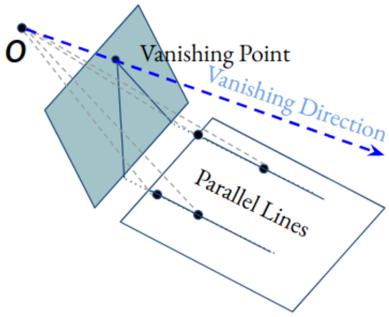


Figure 3.5 The vanishing point is a cross point when we re-project two parallel lines to the image plane.

each cluster to remove the outliers. Each cluster $\diamond = \{\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_n\}$ corresponds to a vanishing point $\mathbf{v}$. Mathematically, the formula is satisfied, *i.e.* ,

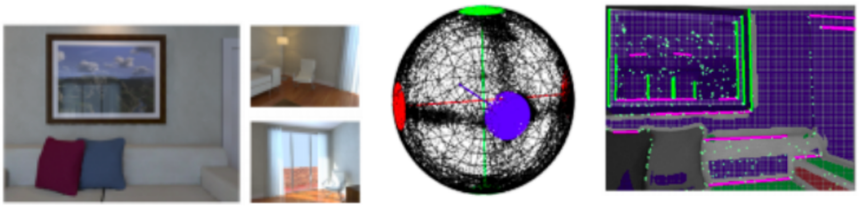
$$\mathbf{S}^T\mathbf{v} = 0 \tag{3.10}$$

where $\mathbf{S}$ is a $3 \times n$ matrix of which the columns represent the equation of the lines in the cluster $\mathbf{\Pi}$.

The linear system, Equation 3.10, can be easily solved via SVD decomposition. A two-step refinement method is performed to improve the vanishing points' accuracy further and remove outliers. For each cluster, we can get a nonlinear least-squares problem as follows:

$$\hat{v} = \arg\min \sum_{l_i \in L} \mathrm{dist}^2([e_i]_\times v, e_i^1) \tag{3.11}$$

where $\mathrm{dist}(\cdot,\cdot)$ represents the vertical distance between a line and a point, e_i and e_i^1 are the middle point and start point of the line l_i, respectively.



(a) Indoor Scenarios (b) Unit Sphere projection (c) Manhattan World Segmentation.

Figure 3.6 Process of Manhattan Frame detection based on RGB-D maps.

Manhattan Frame Detection As shown in Figure 3.6(a), an orthogonal cue, the Manhattan world, exists in the scene, which shows a global structure of the building. Generally, not all objects and scenarios satisfy this constraint. However, if parts of the views in a sequence detect this structure, methods [66] assume that the entire scene is a Manhattan world model, and the frame that detects at least a Manhattan World is regarded as a Manhattan Frame. The process of Manhattan Frame detection will be introduced in the following paragraphs.

1.Manhattan Frame Detection based on RGB-D images. Given a depth map, it is easy to compute a dense normal map for every pixel. In OPVO [98], the depth image is first filtered by a simple box filter for removing noise regions. Then, surface normal vectors are computed based on the cross-product operation on the two tangential vectors that are tangential to the local area of each 3D point. Furthermore, all those surface normals are mapped to the unit sphere as shown in Figure 3.6(b), where all normals are laying on the sphere's surface since every normal vector is normalized. According to the assumption, the distribution of those normals will group at several centers, each located on the axis of the orthogonal coordinate.

Since some local regions are not in good Manhattan World (like pillows on the sofa), several normals are lying in the other regions outside those areas. To solve the problems, a sphere-based mean shift method to detect three dominant axes is used in methods [67, 98], which is briefly introduced in the following paragraphs.

Sphere Mean Shift. A random rotation $\mathbf{R}_{3\times3}$ can be regarded as three 3×1 vectors, namely $\mathbf{R} = [\mathbf{s}_1\ \mathbf{s}_1\ \mathbf{s}_1]$, that are lying on the unit sphere. Each vector is regarded as a seed to find the corresponding dominant axis. In those methods, they run the Manhattan Frame detection procedure 100 times. In each iteration, given the seed $\mathbf{s}_i$ on the sphere $\mathbb{S}^2$, then all neighboring normal vectors within a conic window of $\mathbf{s}_i$ are considered for computing the mean value and shift size. The neighbouring vector $\mathbf{n}_k$ is associated to the conic window of $\mathbf{s}_i$ if $\mathbf{n}_k$ satisfies

$$||\mathbf{n}_k \cdot \mathbf{s}_i|| > \cos\theta_{conic} \tag{3.12}$$

here $\theta_{conic} \in [0, \pi/2]$ is the threshold as shown in Figure 3.6(b), and the selected region on the sphere is colored green, red, and purple, respectively. Furthermore, two perpendicular bases, $\mathbf{b}_x^i$ and $\mathbf{b}_y^i$, are selected on the tangent space of the vector $\mathbf{s}_i$. The process of basis selection can be summarized as three steps:

1. Randomly finding two non-parallel unit vectors $\boldsymbol{v}_x^i$ and $\boldsymbol{v}_y^i$, i.g. $\boldsymbol{v}_x^i = [1, 0, 0]$ and $\boldsymbol{v}_y^i = [0, 1, 0]$.

2. If $\mathbf{s}_i$ is parallel to $\boldsymbol{v}_x^i$, then $\mathbf{b}_x^i = \mathbf{s}_i \times \boldsymbol{v}_y^i$. Otherwise $\mathbf{b}_x^i = \mathbf{s}_i \times \boldsymbol{v}_x^i$.

3. $\mathbf{b}_y^i = \mathbf{s}_i \times \mathbf{b}_x^i$.

Furthermore, x_k and y_k are corresponding disturbments towards $\mathbf{b}_x^i$ and $\mathbf{b}_y^i$, respectively. Then the target $\mathbf{s}_i$ can be represented as

$$\mathbf{n}_k^{'} = \mathbf{s}_i + x_k \mathbf{b}_x^i + y_k \mathbf{b}_y^i. \tag{3.13}$$

The position (x_k, y_k) on the tangent space is represented as $\mathbf{m}_k$.

Therefore, the updated direction $\mathbf{s}_i^{'}$ is obtained by

$$\mathbf{s}_i^{'} = \mathbf{s}_i + \delta \mathbf{s}_{i,x} \mathbf{b}_x^i + \delta \mathbf{s}_{i,y} \mathbf{b}_y^i. \tag{3.14}$$

Maintaining Orthogonality. After computing a mean shift for each vector $\mathbf{s}_i^{'}$, the updated matrix is $\mathbf{R}^{'} = \begin{bmatrix} \mathbf{s}_1^{'} & \mathbf{s}_2^{'} & \mathbf{s}_3^{'} \end{bmatrix}$, which may lose the orthogonality of a rotation matrix.

To maintain the orthogonality, the updated $\mathbf{R}^{'}$ is fed to a SVD progress $\begin{bmatrix} \mathbf{U} & \mathbf{D} & \mathbf{V} \end{bmatrix} = SVD(\mathbf{R}^{'})$, and the final rotation between the camera and Manhattan World can be obtained by $\mathbf{R} = \mathbf{U}\mathbf{V}^{T}$.

3.3.4 Orientation Estimation via Multi Manhattan World

To solve the problem, a multi-Manhattan World model is proposed to estimate the camera pose robustly. It collects all Manhattan Frames detected in the tracking process rather than retains a dominant one in their environments. When a new Manhattan Frame is detected, we will associate it with other Manhattan Frames saved in the map. If the relationship is built successfully, a similar rotation estimation method introduced in Section 2.2.2 will be used here to estimate rotation in the world coordinate.

In this thesis, those Manhattan Frames detected from scenes are collected in the Manhattan map $\mathcal{G}_m = M_k, k \in (0, \ldots, m)$, where $\mathcal{G}_m$ stores both full and partial Manhattan Frame observations. Specifically, the full Manhattan Frame means that three orthogonal elements (we use planes) are detected, while a partial one only has two perpendicular elements. Furthermore, the matching process between a new Manhattan Frame MF_j and the Manhattan map $\mathcal{G}_m$ is based on the plane matches because each plane in the map is given a unique ID during the tracking process. To be specific, $\mathbf{MF}_j$ is represented as $\mathbf{\imath}_i$ and $\mathbf{\imath}_j$, where i and j are ids of those planes, respectively. If other Manhattan Frames are also constructed by those two planes, then the match process for Manhattan Frames can be implemented.

Only perpendicular and orthogonal planes are considered in the Manhattan maps $\mathcal{G}_l$ introduced in Section 3.3.4. Therefore, the numbers of Manhattan Frame are decreased since parts of images cannot detect two planes but measure several lines that satisfy similar conditions.

To break the limitation existing in Section 2.2.2 and 3.3.4, a more flexible algorithm is proposed and introduced in this section. First, we introduce the process that transfers a 2D vanishing point to a 3D vanishing direction vector. As shown in Figure 3.5, the ray $\mathbf{l}$ (blue dash line) constructed by Vanishing Point and camera center O is parallel to those two 3D parallel lines in the camera coordinate. Therefore, the

$$\mathbf{p}_{c_i} = f(\mathcal{K}\mathbf{L}_{c_i}^{m}, \mathcal{K}\mathbf{L}_{c_i}^{n}) \tag{3.15}$$

here the function $f(\cdot,\cdot)$ is to compute the cross point $\mathbf{p}$ between two 2D lines that are re-projected by two 3D parallel lines, respectively. And the operation $\mathbf{KL}_{c_i}^{m}$ re-projects 3D line $\mathbf{L}_{c_i}^{m}$ (in the camera c_i coordinate) to the image plane. And the ray $\mathcal{K}^{-1}\mathbf{p}_{c_i}$ can be built from the pixel $\mathbf{p}_{c_i}$.

Therefore, when those two parallel lines are detected in camera coordinate $\mathrm{c_j}$, the similar equation, like 3.15, can be built to compute another ray $\mathcal{K}^{-1}\mathbf{p}_{\mathrm{c_j}}$ in the camera $\mathrm{c_j}$ coordinate. Based on the relative rotation $\mathbf{R}_{\mathrm{c_i,c_j}}$, those two rays can be connected via

$$\mathcal{K}^{-1}\mathbf{p}_{\mathrm{c_j}} = \mathbf{R}_{\mathrm{c_i,c_j}}\mathcal{K}^{-1}\mathbf{p}_{\mathrm{c_i}}. \tag{3.16}$$

However, in Equation 3.16, the relationship cannot fully observe the relative rotation. Therefore, another type of vanishing point or a plane surface normal is required to compensate for the observation ignored by $\mathcal{K}^{-1}\mathbf{p}_{\mathrm{c_j}}$.

Therefore, two non-parallel directions generated from lines and planes are used to generate a perpendicular group to supervise the relative motion via the Gram-Schmidt orthogonalization process.

3.4 3D Reconstruction in Indoor Scenes

Sparse point clouds [54, 39] reconstructed by feature points and corners are essential in removing camera drift via a frame-to-map pose estimation strategy. However, dense models have more advantages than sparse ones in scene understanding, interaction, and robot navigation applications. In 2011, the impressive dense reconstruction method, KinectFusion [45], demonstrated real-time dense mapping using a consumer depth camera, but the original KinectFusion system was limited to small-scale scenes. This limitation in model size is removed based on technologies like the voxel hashing scheme [99]. More and more efficient dense reconstruction systems are proposed for indoor reconstruction tasks.

In this section, we first introduce the volumetric-based dense mapping methods in Section 3.4.1. Then, lightweight reconstruction methods based on planar priors will be illustrated in Section 3.4.2.

3.4.1 Volumetric Representation

The volumetric data structure implicitly stores environments as an alternative reconstruction method for point-based methods. For example, Those volume voxels can be used to save the occupied probability and generate an occupancy map for intelligent agent navigation applications. However, those voxels can also record the signed distance fields that are converted from depth maps, which can be exploited for further surface reconstruction by extracting the zero-level set of the implicit function using raycasting. This section will focus on surface reconstruction methods based on volumetric representations.

TSDF. The truncated signed distance function (TSDF) is a core component of the KinectFusion system, where each voxel stores the signed distance to the closet surface. In KinectFusion, the volumetric model is also responsible for pose estimation via an ICP registration approach by aligning the current frame's surface model with the global map. Given camera poses, the TSDF fusion module will be more efficient, and the fusion process will be introduced in the following paragraphs.

As a classic algorithm for real-time 3D reconstruction, the TSDF algorithm is simple and easily parallelizable. TSDF is an improvement of SDF. It limits the value to the range of $[-1, 1]$ and only performs calculations within a limited distance range from the object surface, reducing the amount of calculation.

Given a volume built based on the size of the scene and the resolution of each voxel, we can build a retraction function $\mathtt{pic}(\dot{})$ for transferring a voxel x_i to the world coordinate via $\mathbf{P}_w^i = \mathtt{pic}(x_i)$. Furthermore, we re-project those valid voxels into the camera coordinate of C_k and get depth information by using

$$d_{x_i} = D(\mathbf{K}\mathbf{R}_{c_k,w}\mathbf{P}_w^i + \mathbf{t}_{c_k,w}) \tag{3.17}$$

here $D(\mathbf{p})$ is depth value at the location of $\mathbf{p}$. Then, the SDF value is computed by computing the distance $sdf(x) = d_{x_i} - cam(x)$.

According to the requirements of TSDF, the SDF value is truncated in the range of $[-1, 1]$ based on

$$\mathtt{tsdf}(x) = \max(-1, \min(1, \frac{\mathtt{sdf}(x)}{t})) \tag{3.18}$$

where t is the truncate threshold.

Finally, the new tsdf value is fused to the volume by using

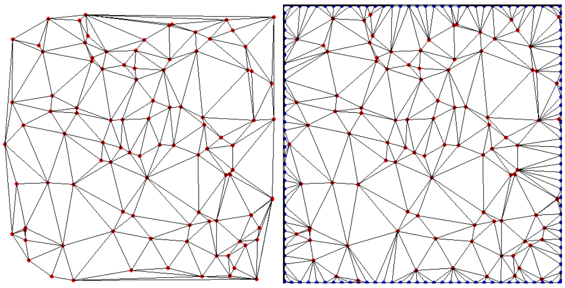
$$\mathtt{TSDF}(x) = \frac{W_{i-1}(x)\mathtt{TSDF}_{i-1}(x) + w_i(x)\mathtt{tsdf}_i(x)}{W_{i-1}(x) + w_i(x)} \tag{3.19}$$

where W is the weight of voxel x in the volume.

Occupancy grid map. Occupancy Grid reconstruction is widely used in probabilistic robotics for mobile robots to generate maps from noisy and uncertain sensor measurement data, assuming that the robot pose is known. First, an environment is devided into several grids, and the basic idea of the occupancy grid is recording the possibility of obstacle occupied at that location.

3.4.2 Mesh with Structural Regularities

Each voxel is dealt with individually for dense models introduced in Section 3.4.1. Although some methods are proposed to speed up the reconstruction method, the computation burden still needs to be more significant to implement in embedded devices. Generally, an environment can be divided into planar and non-planar areas. Plane areas often can be detected from floors, walls, ceilings, and furniture surfaces. They occupy a large area but are easy



(a) Original mesh plane (b) boundary completion

Figure 3.7 Planar reconstruction. (a) original mesh with flaws in boundaries. (b) complete mesh after boundary completion.

to parameterize. As shown in Figure 3.7(a), planar regions can be reconstructed by several triangular faces, which are constructed by several sparse points lying on the plane surfaces, which is much more efficient than TSDF-based mesh approaches.

Scene Abstraction

Instead of building a one-to-one model of the real world, some applications require lightweight, low-polygonal, and high-resolution models. Scene Abstraction [100, 101, 102, 5] is the task from the scanned geometry to the lightweight model.

This topic has been explored by geometric [100] and data-driven [101, 102] methods that make use of a single RGB-D pair or a reconstructed 3D model. Based on surroundings, those single-frame approaches try to recover the shape of some objects and structures, which rely heavily on learned priors from training datasets. Moreover, those methods suffer from inconsistency issues since they deal with each pair as independent scenes.

Unlike those data-driven methods, offline methods [100] are designed to deal with dense 3D models generated by other dense reconstruction architectures. After extracting planar regions, they can reduce the size of models sharply. In general, those approaches need more time to process each whole model, which can satisfy the need for real-time applications.

During the tracking process, we built a sparse map using points, lines, and planes, where different planar regions are already segmented into different plane instances. The growing process of a plane instance starts when a small plane is detected from a new coming depth map. If the small plane is matched with the planes in the map, it will be merged into the corresponding planar regions. Otherwise, the small plane is a new planar region we did not detect before. We will initialize a new plane instance in the map. Generally, the planar point clouds directly segmented from a depth map are very noisy. When we merge those planes, the planar regions will be very noisy. Based on feature-based methods [5] exploit re-project errors from plane features to optimize poses, the normal vector of the fused plane is optimized. Then, those points that do not ideally satisfy a plane function can be fine-tuned to achieve flat mesh planar models.

Part II

Methodology

4 Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments

Traditionally, limited pixels can be used by monocular SLAM methods since features only take a small part of pixels on an image. For some low-textured scenes, even semi-direct or direct approaches also suffer from the issue since most of the pixels in the low-textured area are very similar. A direct strategy is using more different types of features, but more is needed to solve the issue effectively.

To improve pose estimation accuracy, we propose a novel architecture for monocular sensors. First, the 6DoF camera pose is decoupled into rotation and translation estimation processes. The rotation estimation module is implemented based on dense surface normal maps. Based on the assumption of Manhattan World, those normals lying on the unit sphere are clustered into several groups, and the centers of those groups are supposed to construct an orthogonal coordinate. And other normals are regarded as outliers if they do not lie in those areas. Therefore, the relative rotation between two frames can be computed by tracking the center movement of each group. Benefiting from the estimated rotation, the 3DoF translation part can be solved robustly based on fewer point and line features. More results of the architecture tested on untrained scenes can be found in Appendix IV.

Contributions. Federico Tombari's suggestion inspires the idea of providing the Manhattan World assumption to monocular RGB sensors. I proposed and implemented a rotation and translation estimation SLAM architecture based on point-line features and a surface normal predictor, Where the neural network is modified and fine-tuned by Nikolas Brasch and Yida Wang. Federico Tombari and Nikolas Brasch helped structure and polish the paper description.

Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments

Yanyan Li[1], Nikolas Brasch[1], Yida Wang[1], Nassir Navab[1,2] and Federico Tombari[1,3]

[1] Technical University of Munich
[2] Johns Hopkins University
[3] Google

Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments

Yanyan Li, Nikolas Brasch, Yida Wang, Nassir Navab, and Federico Tombari

***Abstract*—In this letter a low-drift monocular SLAM method is proposed targeting indoor scenarios, where monocular SLAM often fails due to the lack of textured surfaces. Our approach decouples rotation and translation estimation of the tracking process to reduce the long-term drift in indoor environments. In order to take full advantage of the available geometric information in the scene, surface normals are predicted by a convolutional neural network from each input RGB image in real-time. First, a drift-free rotation is estimated based on lines and surface normals using spherical mean-shift clustering, leveraging the weak Manhattan World assumption. Then translation is computed from point and line features. Finally, the estimated poses are refined with a map-to-frame optimization strategy. The proposed method outperforms the state of the art on common SLAM benchmarks such as ICL-NUIM and TUM RGB-D.**

***Index Terms*—SLAM, visual learning.**

I. Introduction

Visual Simultaneous Localization and Mapping (V-SLAM) systems are important for autonomous robots and augmented reality, as they are used to estimate poses and reconstruct unknown environments. In numerous SLAM use cases and applications, monocular cameras are the most common sensors in indoor scenarios. Indoor environments are often characterized by a lack of textured surfaces, and by irregularly distributed feature points. In particular, low-textured walls, floor and ceiling are difficult to deal with by both state-of-the-art feature-based methods [1] as well as direct methods [2], [3]. For low-textured scenes, SLAM systems combining point and line features have been proposed to target low-textured scenes, e.g. Stereo-PLSLAM [4], PLVO [5], Mono-PLSLAM [6] and [7], extending the working scenarios to low-textured environments with visible structural edges. Since the map is built from a sequence of input frames, small errors accumulate over time, resulting in drift which affects dense reconstruction by leading to misaligned surfaces and artifacts.

There are two main strategies to overcome these errors. Loop closure detection [1], [8] combined with pose graph optimization detects previously seen landmarks and optimizes the pose graph based on the new constraints, thus correcting the accumulated drift. Loop closure, however, brings in an extra computational burden and removes the drift only when revisiting the same place. Another strategy consists of assuming an underlying (global) structure in the world frame, then each tracked frame can be directly aligned to this world structure instead of the last frame or keyframes. The most common formulation of a structured scene is the Manhattan World (MW) [9], [10] where the environment shown in Fig. 1(a) consists of geometric structures (planes and lines) oriented in one of three orthogonal orientations. It is particularly useful in indoor environments where structures such as walls, floor and ceilings often show consistent alignment over multiple rooms, enabling a global alignment.

The MW approach is an efficient method to keep the accumulated drift low by providing a drift-free strategy for rotation estimation, as the rotational component is the main source of overall drift [11], [12].

The state of the art of monocular approaches relying on a MW [9], [10] are based on parallel and orthogonal lines alone, as it is difficult to extract 3D information, except for vanishing points, from a monocular RGB image, which is a quite strong limitation for most scenarios. Furthermore, indoor environments often consist of large planar regions with few features for pose estimation. RGB-D methods [12], [13], directly measure the structure of the scene in the form of depth maps, this allows them to compute dense surface normals for each pixel.

Inspired by recent works based on convolutional neural networks (CNN) and scene geometry prediction approaches from a single view [14], [15], we propose a monocular SLAM framework which leverages the underlying scene structure to carry out low-drift SLAM even in presence of low-textured environments, in the form of densely predicted normal maps from a CNN, analogously to existing works based on dense RGB-D sensors.

Specifically, we propose the following contributions:

- A low drift real-time monocular SLAM framework for structured environments, with decoupled rotation and translation
- Dense monocular normal estimation for rotation estimation leveraging the MW assumption

Manuscript received February 24, 2020; accepted July 23, 2020. Date of publication August 11, 2020; date of current version August 21, 2020. This work was supported by China Scholarship Council. This letter was recommended for publication by Associate Editor J. Andrade-Cetto and Editor S. Behnke upon evaluation of the reviewers' comments. *(Yanyan Li and Nikolas Brasch contributed equally to this work.) (Corresponding author: Yanyan Li.)*

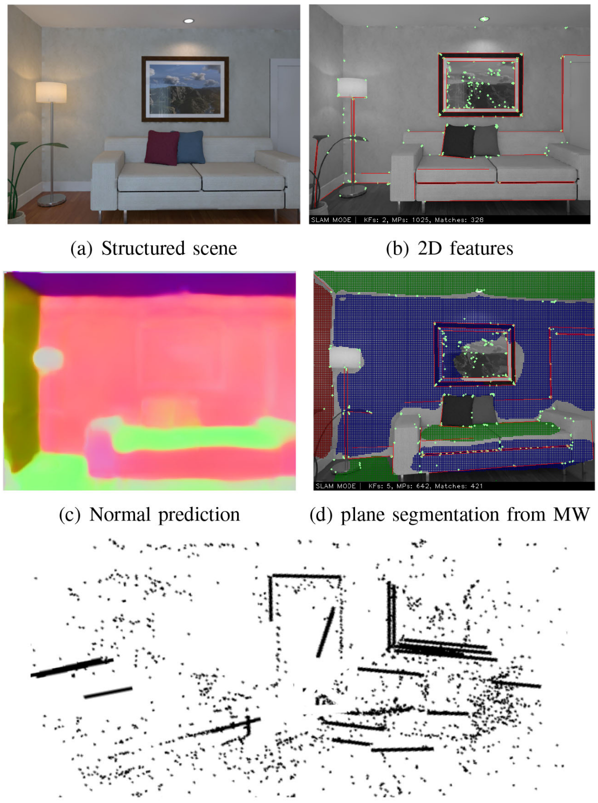
Yanyan Li, Nikolas Brasch, and Yida Wang are with the Technical University of Munich, 80333 München, Germany (e-mail: yanyan.li@tum.de; nikolas.brasch@tum.de; yida.wang@tum.de).

Nassir Navab is with the Technical University of Munich, 80333 München, Germany, and also with the Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: nassir.navab@tum.de).

Federico Tombari is with the Technical University of Munich, 80333 München, Germany, and also with the Google, 8002 Zürich, Switzerland (e-mail: tombari@in.tum.de).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3015456


SLAM MODE | KFs: 2, MPs: 1025, Matches: 328
SLAM MODE | KFs: 5, MPs: 642, Matches: 421

(a) Structured scene (b) 2D features

(c) Normal prediction (d) plane segmentation from MW

(e) Sparse output map

Fig. 1. The proposed approach targets low-textured indoor scenes to carry out low-drift monocular SLAM based on dense normal prediction and leveraging the Manhattan World assumption.

- A method for translation estimation relying on point and line features

We evaluate numerically on common SLAM benchmarks such as ICL-NUIM [16] and TUM RGB-D [17] showing that the proposed approach outperforms the state of the art in monocular SLAM.

II. Related Work

A. Monocular SLAM

PTAM [18] is a monocular, keyframe-based SLAM system which was the first work to introduce the idea of splitting camera tracking and mapping into parallel threads, and demonstrate to be successful for real time augmented reality applications in small-scale environments. Strasdat *et al.* [8] present a large scale monocular SLAM system in which the front-end bases on optical flow implemented on a GPU, followed by FAST feature matching and motion-only BA, and a back-end based on sliding-window BA. As a complete SLAM pipeline, ORB-SLAM [1] combines feature based tracking, sparse point mapping, descriptor based re-localization and loop closure altogether. In addition to point features several works propose the use of lines [4], [5] for low-textured environments, we propose to use additional dense structural information in the form of predicted normal maps.

Inspired by the recent success of deep learning based depth prediction, CNN-SLAM [19] incorporates a neural network which estimates depth information within the popular LSD-SLAM [2] framework to create dense scene reconstructions in metric scale, where depth predictions are used to initialize the SLAM system and merged continuously with the semi-dense depth maps optimized by the SLAM system. Instead of estimating depth maps only for key-frames in CNN-SLAM, our approach predicts surface normals from every RGB frame in real-time. In CodeSLAM [20], a neural network learns a compact latent representation for the structure of a scene conditioned on the RGB image, showing that the joint optimization of both structure and pose can improve monocular pose estimation. By predicting normal maps instead of depth maps we avoid the necessary differentiation operation which could introduce noise. Predicting normal maps also seems to generalize better between datasets as depth does.

B. RGB-D SLAM

Probabilistic-VO [7] combines points together with lines and planes for pose estimation while modeling their uncertainties. Due to the combination of 2D-3D point and line correspondences and 3D-3D plane matches, a weighting between re-projection and euclidean errors must be chosen empirically. CPA-SLAM [21] extended DVO-SLAM with global plane landmarks. Pose estimation and soft assignment of depth measurements to planes are computed in an Expectation-Maximization framework. KDP-SLAM [22] combines photometric and geometric loss based on plane segments instead of points for frame-to-frame pose estimation and additionally aligns plane segments with global planes in a Smoothing and Mapping (SAM) framework.

C. Manhattan World

Straub *et al.* [11] and Zhou *et al.* [23] show that the main source of drift in traditional feature-based systems is caused by the rotation estimation.

Even if the MW assumption is a good constraint for indoor SLAM, it is difficult to enforce it in monocular methods because only limited 3D information can be obtained. Zhou *et al.* [10] applies J-linkage [24] to classify parallel line segments into different groups and estimate the dominant direction from the vanishing points. If depth maps are available, surface normals can be computed directly. Joo *et al.* [25] provide a branch-and-bound framework for Manhattan Frame estimation. MVO [23] propose a unit sphere mean shift method to find the rotation matrix between the Manhattan World and the camera system. For the translational part, they compute and align density distributions of points in each orthogonal direction, avoiding the costly matching of points. OPVO [26] use planes to estimate the Manhattan Frame rotation, limiting its application to environments with at least 2 orthogonal planes. LPVO [12] adds vanishing points of lines for the rotation estimation. Both use point based methods for translation estimation. L-SLAM [13] replaces the graph based translation estimation from LPVO with a Kalman filter based SLAM update, using the LPVO translation estimation in the prediction step. Compared with [12], [13], we build an initialization module based on points, lines and predicted normals. Further more, a refinement module is added to optimize the pose after the decoupled initialization.

III. Scene Structure Analysis

The structural information used in the system is analyzed in this section. First, we describe the methods for extraction and triangulation of points and lines; Then, an architecture for surface normal prediction is introduced.

A. Points and Lines Analysis

Point features, due to their descriptiveness, compactness and robustness to illumination changes, are the most common features used in visual SLAM systems. In our method, ORB features [27] are adopted which are fast enough to extract and robust enough to get matched. Since it's hard to extract sufficient feature points for robust pose estimation in low-textured environments, we further supplement them with line segments extracted and encoded using the LSD [28] and LBD [29] accordingly.

Similar to ORB-SLAM [1], once the 2D point features $p_n = (u_n, v_n)$ and line segments $l_m = (p_{m,s}, p_{m,e})$ are extracted in the new keyframe F_i, new features are triangulated to 3D points P_n and lines L_m with correspondences located on other connected keyframes.

Due to the factorization of rotation and translation estimation, it is possible to estimate the pose even in cases with pure rotation and no translation or with small parallax, which would not be possible with pure monocular feature based approaches. The rotation can be estimated from the Manhattan World Frame, this means fewer landmarks are needed to obtain the remaining 3 degrees of freedom for the translation.

B. Surface Normal Prediction

We use learned knowledge to reason about the 3D environment, instead of measuring dense depth values directly. Therefore, a 2D convolutional architecture(CNN) is trained to segment planar regions and predict pixel-wise surface normals. The proposed CNN is composed of a ResNet101-FPN [14] encoder for feature extraction and a two-branch decoder for planar area segmentation and normal estimation. As the planar and non-planar regions are unbalanced in indoor scenarios, we use the balanced cross entropy loss for training

$$\mathscr{L}_p = -1(1-w)\sum_{i\in P}\log p_i - w\sum_{i\in P_{neg}}\log(1-p_i), \tag{1}$$

where P and P_{neg} represent planar and non-planar regions, respectively. p_i represents the probability of the ith pixel being located in a planar region. We use w to balance the contributions of planar and non-planar pixels. Then the loss function for the normal estimation is filtered by the planar mask.

$$\mathscr{L}_n = -\frac{1}{n}\sum_{i\in P} n_i \cdot n_i^*, \tag{2}$$

where n_i and n_i^* are the predicted normal and ground truth normal for the ith pixel.

IV. Initialization

In this section, we describe the strategy of computing the relative poses between two frames and reconstructing an initial map. In order to be robust to different motions, we decouple pose estimation into rotation and translation which is explained further in the following paragraphs.

Rotation. First, we assume that there is a Manhattan coordinate system M shown in Fig. 3, we compute the relative rotation R_{C_1M} from Manhattan coordinate frame M to the first frame C_1 by clustering the normal map v_i^s of C_1 on the unit Gaussian sphere [12], [23] centered on the M. Following [12], [23], we project the normals onto the tangent plane of each Manhattan world axis r_n, where $n \in [1, 2, 3]$, for the current estimation. Instead of testing several random matrices, we found that setting R_{C_0M} to identity and running multiple mean-shift iterations is enough to obtain a good estimate. In order to remove noise from normal maps, we only consider the vectors $v_{in}^{s'}$ which are close to the axis r_n.

Then, the refined surface normal vectors $v_{in}^{s'}$ are projected to two-dimensional vectors $m_{in}^{'}$ in the nth tangential plane. We compute the cluster mean $s_n^{'}$ for the nth tangential plane under a Gaussian kernel by

$$s_n^{'} = \frac{\sum_{in} e^{-c\|m_{in}^{'}\|^2} m_{in}^{'}}{\sum_{in} e^{-c\|m_{in}^{'}\|^2}} \tag{3}$$

where c is a hyper parameter that defines the width of the kernel, which is set to 2 in our experiments. Then, we transform the cluster centers back onto the Gaussian sphere as s_n, which are used to update the angle between the camera and the MW axis $\hat{r}_n$ combining with the current rotation Q_n,

$$\hat{r}_n = Q_n s_n, \tag{4}$$

here $Q_n = [r_{mod(n,3)}, r_{mod(n+1,3)}, r_{mod(n+2,3)}]$ and $mod()$ is a modulus operation. The tangent plane and the cluster centers are iteratively computed until the rotation estimate is converged. Then we obtain $R_{C_1M} = [\hat{r}_1, \hat{r}_2, \hat{r}_3]^T$.

Translation. As for the translation estimation, 2D correspondences of points $[p_i^1, p_i^2]$ between two frames and their relative rotation $R_{C_1C_2}$ are used

$$X_i^2 = \begin{bmatrix} x_i^2 \\ y_i^2 \\ z_i^2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} X_i^1 + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{5}$$

where X_i^j represents a 3D point in the jth camera. By eliminating the scale z_i^2, we obtain

$$\begin{bmatrix} \tilde{x}_i^2 \\ \tilde{y}_i^2 \\ 1 \end{bmatrix} = \begin{bmatrix} (r_1 \cdot X_i^1 + t_1)/(r_3 \cdot X_i^1 + t_3) \\ (r_2 \cdot X_i^1 + t_2)/(r_3 \cdot X_i^1 + t_3) \\ 1 \end{bmatrix} \tag{6}$$

where $[\tilde{x}_i^j \ \tilde{x}_i^j \ 1]^T$ represents the ith normalized 3D point in the jth camera frame. Since X_i^1 is also a 3D point, we need to eliminate z_i^1 and build

$$\begin{bmatrix} -\tilde{y}_i^1 t_3 + t_2 \\ \tilde{x}_i^1 t_3 - t_1 \\ -\tilde{x}_i^1 t_2 + \tilde{y}_i^1 t_1 \end{bmatrix}^T \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \begin{bmatrix} \tilde{x}_i^1 \\ \tilde{y}_i^1 \\ 1 \end{bmatrix} = 0 \tag{7}$$

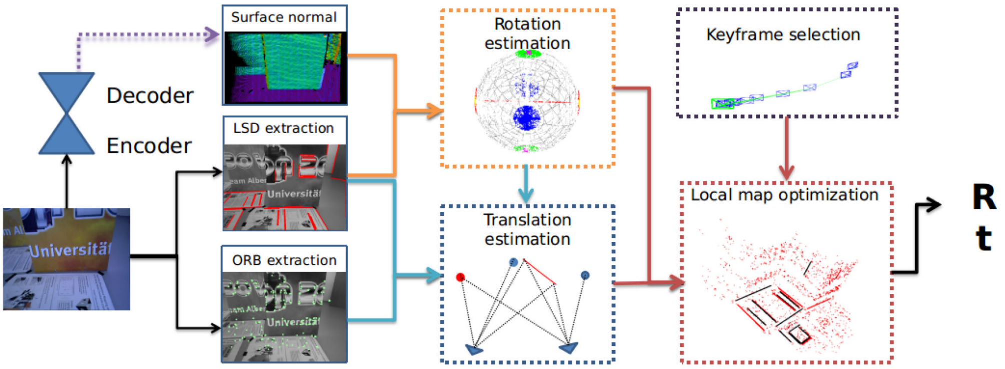


Fig. 2. Proposed SLAM framework (StructureSLAM). In the front-end, the encoder-decoder network predicts dense surface normals. In parallel, point and line features are extracted from the RGB image. In the back-end, first the scene structure in the form of normals and lines is used to estimate the global rotation of the camera. Then, the remaining 3-DoF for the translation are obtained using point and line features. The initial pose estimate is validated and refined using the local map. Keyframes are selected based on the availability of point and line features.

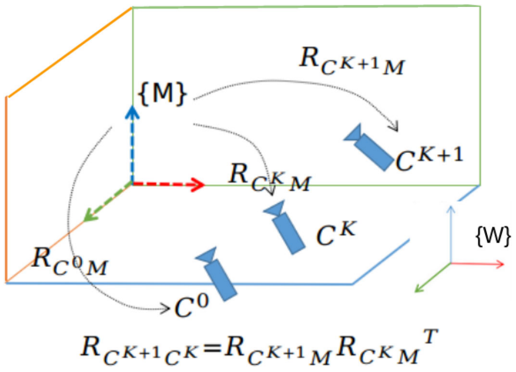


Fig. 3. Rotation estimation between multiple frames via the Manhattan world.

where $[\tilde{x}_i^j\ \tilde{x}_i^j\ 1]^T = K^T(u_i^j\ v_i^j\ 1)$ and K is the intrinsic matrix of the camera [12]. $(u_i^j\ v_i^j)$ is the ith pixel in the jth frame. Based on eq. (6) and eq.(7), we construct a translation relationship between those 2D correspondences. Then, we solve the system in eq. (7) using SVD to obtain the translation.

V. Tracking

Instead of estimating rotation and translation between two frames, we estimate the rotation between each frame and the underlying Manhattan World. The residual rotation errors are independent of the sequence length and cannot be propagated between frames. Point and line correspondences are used to estimate translation (3 DoFs) by a combination of frame-to-frame and frame-to-map methods.

A. Manhattan Rotation Estimation

This section describes the rotation estimation between camera and Manhattan system.

Given the surface normals and mask of planar regions from the network, we follow the mean-shift clustering approach, as desribed in IV, to find the dominant axes on the euclidean sphere and estimate the rotation $R_{C_K M}$. Since normal maps might contain errors due to the networks inference process, the clustering approach is used to remove outliers first. Furthermore, the initial rotation will be refined in following sections.

B. Translation Estimation

After obtaining the rotation matrix, we use the points and line segments to estimate the 3-DoF translational motion, which requires less features than the full 6-DoF estimation. We reproject the 3D points from the last frame to the current one and define the error function, based on the re-projection error, as follows,

$$e^p_{k,j} = p_k - \pi(R_{k,j}P_j + t_{k,j}) \tag{8}$$

here $\pi()$ is the projection function. Since the rotation matrix $R_{k,j}$ has already been estimated in the last step, we fix the rotation and only optimize the translation using the right half of the Jacobian matrix for eq. (8),

$$\frac{\partial e^p_{k,j}}{\partial \xi} = \begin{bmatrix} \frac{xyf_x}{z^2} & -\frac{z^2+x^2}{z^2}f_x & \frac{yf_x}{z} & -\frac{f_x}{z} & 0 & \frac{xf_x}{z^2} \\ \frac{z^2+y^2}{z^2}f_y & -\frac{xyf_y}{z^2} & -\frac{xf_y}{z} & 0 & -\frac{f_y}{z} & \frac{yf_y}{z^2} \end{bmatrix} \tag{9}$$

For the lines we obtain the normalized line function from the 2D endpoints p_{start} and p_{end} as follows,

$$l = \frac{p_{start} \times p_{end}}{\|p_{start}\|\|p_{end}\|} = (a, b, c) \tag{10}$$

We formulate the error function based on the point-to-line distance between l and the projected 3D endpoints P_{start} and P_{end} from the matched 3D line in the keyframe. For each endpoint P_x, the error function can be noted as,

$$e^l_{k,j} = l\pi(R_{k,j}P_x + t_{k,j}) \tag{11}$$

The Jacobian matrix for the line error eq. (11) is given by

$$\frac{\partial e^l_{k,j}}{\partial \xi} = \begin{bmatrix} -\frac{f_y l_y z^2 + f_x l_x xy + f_y l_y y^2}{z^2}, & \frac{f_x l_x z^2 + f_x l_x x^2 + f_y l_y xy}{z^2}, \\ -\frac{f_x l_x y - f_y l_y x}{z}, & \frac{f_x l_x}{z}, \\ \frac{f_y l_y}{z}, & -\frac{f_x l_x x + f_y l_y y}{z^2} \end{bmatrix} \tag{12}$$

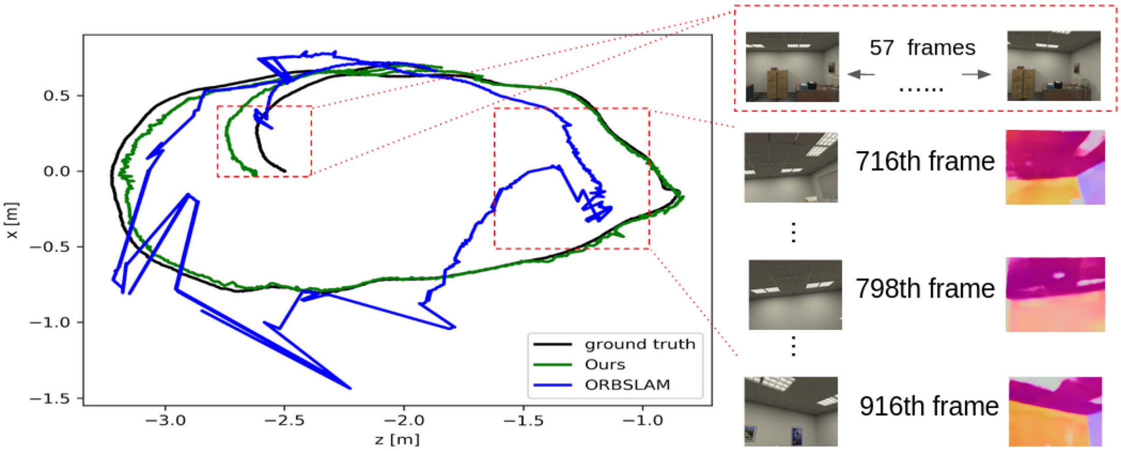


Fig. 4. Trajectory analysis, comparing the proposed method, ORB-SLAM and the ground-truth on the "of-k3" sequence in the ICL NUIM dataset.

The combined least squares cost for points and lines can be written as

$$t* = argmin \sum_{j\in(k-2,k-1)}^{M} \left(e^{p}_{k,j}{}^{T} e^{p}_{k,j} + e^{l}_{k,P_x}{}^{T} e^{l}_{k,P_x}\right) \quad (13)$$

The system is solved using the Levenberg-Marquardt algorithm.

C. Fallback and Pose Refinement

The pose estimate is based on the MW assumption. In cases of Non-Manhattan Worlds or where the Manhattan Frame is not visible in the current frame the estimated pose will be incorrect. To check whether the pose estimate obtained from the previous steps is correct we project all features from the last n keyframes onto the current frame and compute the re-projection error. By applying a threshold to filter the features we require a minimum number of inliers to accept the pose.

When not enough inliers are found we fall back to a frame-to-frame tracking method until we estimate a pose that agrees again with the Manhattan World. As a fallback we first track the new frame based on the last frame using an efficient re-projection search scheme [30] for points and lines, using the same least squares method as for the translation, this time using the full Jacobian matrix. In the case we do not get a good solution, measured based on the number of inliers, we try to estimate the pose based on the last keyframe using descriptor matching for the points [30] and re-projection based search for the lines. To reduce the drift, in the final step we optimize the pose of the new frame based on a local map constructed from the last n keyframes [30]. Here we do not use the MW assumption anymore, as we found that the initial rotation estimation is enough to reduce the drift and errors in the predicted normal maps can lead to inconsistent pose estimates.

In contrast to other work, based on Manhattan frames for rotation estimation this heuristic allows us to fall back to a purely feature based pose estimation in case the estimate from the MW pose estimation is wrong or not available.

VI. Experiments

Implementation details: We train the network implemented for normal estimation based on the ScanNet [31] dataset with a batch size of 32 for 8 epochs. The backbone is pretraind on ImageNet [32] for feature extraction and PlaneReconstruction [14] for understanding plane regions. We use the Adam optimizer with a learning rate of 10^{-4} and a weight decay of 10^{-5}. Our model is trained in an end-to-end manner and can predict normal maps in real-time. As a baseline we use the original GeoNet [15] model trained by the authors for 400 k iterations on NYU-DepthV2 [33]. Models used in the experiments are not fine-tuned on other datasets. All experiments were carried out with an Intel Core i7-8700 CPU (with @3.20 GHz) and a NVIDIA 2080 Ti GPU. We run each sequence 5 times and show median results for the accuracy of the estimated trajectory. We evaluate our proposed SLAM system on public datasets and compare its performances with other state-of-the-art methods. The evaluation metrics used in the experiments are the absolute trajectory error (ATE) and the relative pose error (RPE) [17], which measure the absolute and relative pose differences between the estimated and the ground truth motion.

Evaluation and datasets: In order to evaluate our method, on the one hand, we compare against several monocular SLAM frameworks, as CNN-SLAM [19] that connects SLAM with predicted depth maps based on keyframes, LSD-SLAM [2] that is popular direct method and ORB-SLAM [1]. We align the trajectories for ORB-SLAM, LSD and the proposed method to the ground truth trajectories using a similarity transformation [1] due to the unknown real scale. On the other hand, we run our SLAM architecture with different normal maps to evaluate the importance of accurate normals, by switching our normals with the ones from the state-of-the-art, but not real-time capable network, GeoNet [15] and normal maps computed from the depth maps provided by the dataset using [34].

- ICL-NUIM dataset [16] is a synthetic indoor datasets that provide RGB images, depth maps and ground-truth camera poses. There are two scenes, named "living room" and "office" which are noted as "lr" and "of" in our experiments.

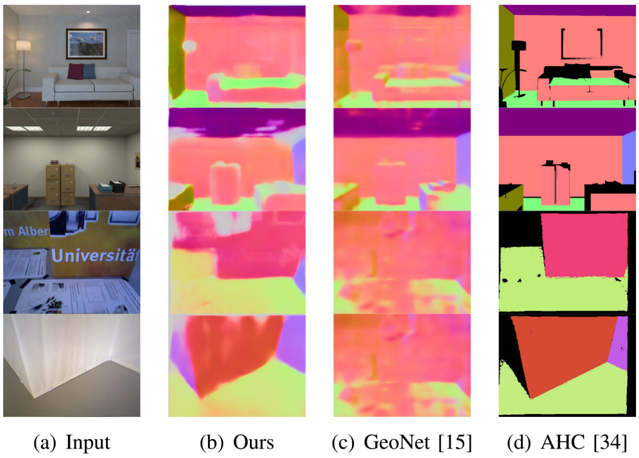


Fig. 5. Results of normal prediction model on ICL-NUIM (top) and TUM-RGBD (bottom) scenes for different approaches.

TABLE I
PERFORMANCE OF THE SURFACE NORMAL PREDICTION ON THE SCANNET [31] TEST SET

Error			Accuracy		
mean	median	rmse	<11.25°	<22.5°	<30°
15.2°	7.8°	24.4°	0.672	0.797	0.841

- TUM RGB-D dataset [17] was collected using a real RGB-D sensor in real scenes as well as specially designed scenes to challenge current SLAM algorithms, featuring challenging scenes with good structure, but without texture.
- HRBB4 dataset [35] which has 12,000 frames of 640 × 320 pixels recorded by a monocular camera in a corridor.

A. Normal Prediction

Fig. 5 presents qualitative results on unseen images of different normal estimation methods. In our method, we mask out the lampshade (first row) and small boxes (second row), as these regions are classified as non-planar. The first two rows, show common examples for indoor environments. Both of them show good results, GeoNet shows smaller inaccuracies. For the last two rows, which are very uncommon scenes, the planar region detection and normal estimation of our model are still generating reasonable results, while the quality of the normal predictions from GeoNet decreased severely.

The agglomerative hierarchical clustering (AHC) algorithm [34] is an efficient method to detect planes in a depth map. However it difficult to detect planes (like in the third an forth row) where the quality of the depth maps decrease due to a highly slanted surface. In the Table I, the performance of the network is evaluated on the ScanNet [31] dataset generated by [36] against the ground truth.

B. Pose Estimation

In order to evaluate our method in different environments, we select structured image sequences from the ICL-NUIM dataset [16] and the TUM RGB-D dataset [17]. Table II shows the RMSE for all methods on several sequences, 'lr' and 'of' stand for the living room and office room sequences in the ICL-NUIM dataset. 's-t-near' and 's-not-near' are the structure-texture-near and structure-notexture-near sequences in the TUM RGB-D dataset, respectively. 's-t-near' and 's-t-far' are showing the same environment consisting of multiple textured planes, 's-not-near' and 's-not-far' consist of a similar structure, but without texture.

From the six row to the eight row, different normal maps are given to the same backbone. It is obvious that using AHC-based normal maps (obtained from ground truth depth map) obtain the best results compared to other methods. It also shows the potential of our SLAM architecture, given precise normal maps. Performances from $-w$ Ours (combination of our normal prediction network and the backbone) is more robust than $-w$ GeoNet (combination of GeoNet and the backbone), especially in the 's-not-far' sequence. For those non-textured images, it is difficult for GeoNet to predict accurate normals. In the backbone, conic areas around each axis are used during the sphere mean-shift method to filter the normal maps, this allows the handling of normal outliers up to a certain point. In cases were the number of outliers is too high, it is difficult to obtain a good rotation from the back-end of the architecture. Different to the monocular methods, LPVO [12] works directly with RGB-D images, which prevents scale drift and allows tracking directly on the depthmap. In comparison our method achieves comparable performance without the use of a depth sensor.

Our method obtains good results and shows robust performance in all five sequences. In the first two sequences, the difference between the point based ORB-SLAM and our method, that connects structure and geometric information, is not significant. However ORB-SLAM is not able to find enough point matches over a sequence of frames and looses tracking in some of the sequences, these are marked with a cross (×). Our method, which additionally uses lines for the translation estimation achieves even better results.

When we compare -w Ours, -w GeoNet with ORB-SLAM in textured sequences, they obtain similar results because those sequences have a sufficient number of features distributed evenly on each frame. However for indoor environments, like Fig. 4, it is difficult to obtain enough point features because of large non-textured planar regions. In the 'of-kt3' sequence, there is little change in the first 57 frames, so ORB-SLAM cannot initialize successfully, because it needs enough points for homography/fundamental model selection. After initialization, it is also challenging for ORB-SLAM to track via the point-based motion model. For our case, the initial rotation matrix is estimated by the mean-shift method instead of estimating the essential or homography matrices. This means we can deal with pure rotational motion. Furthermore, points and line segments are used for 3 DoFs translation only, which is more robust even in large non-textured scene.

In order to present the robustness of our method, we compute the RPE for those sequences, which can be processed robustly by ORB-SLAM and our method. For 's-t-far' and 's-t-near' that are textured sequences, ORB-SLAM and the proposed method have similar performences. The relative translation errors for

TABLE II
COMPARISON OF TRANSLATION RMSE (M) FOR ICL-NUIM [16] AND TUM RGB-D [17] SEQUENCES USING MONOCULAR CAMERA. WE USE **BOLD** NUMBERS TO MARK THE BEST RESULT PER SEQUENCE. $-w$ MEANS THAT THE PROPOSED FRAMEWORK USES THE CORRESPONDING SURFACE NORMALS. × INDICATES THAT THE ALGORITHM FAILS DUE TO LOST TRACKING

Methods	lr-kt1	lr-kt2	lr-kt3	of-kt1	of-kt2	of-kt3	s-t-near	s-t-far	s-not-near	s-not-far
LSD-SLAM [2]	0.059	0.323	-	**0.157**	0.213	-	-	0.214	-	-
CNN-SLAM [19]	0.540	0.211	-	0.790	0.172	-	-	0.037	-	-
ORB-SLAM [1]	0.024	0.061	0.035	×	**0.031**	0.326	0.016	0.015	×	×
LPVO [12]	*0.04*	*0.03*	*0.10*	*0.05*	*0.04*	*0.03*	*0.11*	*0.17*	*0.08*	*0.07*
-*w* Ours	**0.016**	**0.045**	0.046	×	**0.031**	0.065	**0.014**	**0.014**	**0.065**	**0.281**
-*w* GeoNet [15]	×	0.047	**0.026**	×	0.048	**0.043**	0.107	**0.014**	0.068	×
-*w* AHC [34]	*0.016*	*0.028*	*0.020*	*0.763*	*0.021*	*0.020*	*0.015*	*0.013*	*0.015*	*0.220*

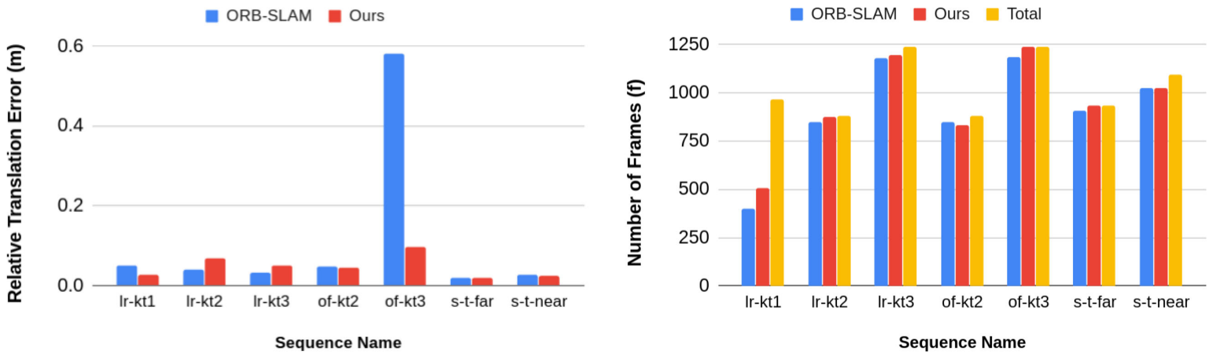


Fig. 6. Relative translational error comparison between ORB-SLAM and our method on different sequences (left) and a comparison of the average runtime length on each sequence before tracking is lost (right).

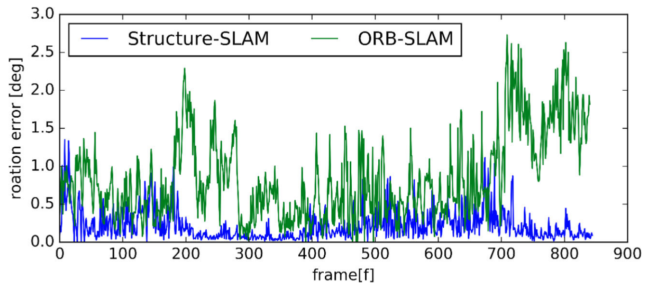


Fig. 7. rotation error comparison between ORB-SLAM and our method on sequence lr-k2.

the sequence 'of-kt3' in Fig. 6 (left) is significantly larger for ORB-SLAM, which corresponds to the result presented in Fig. 4. As shown in Fig. 7, the proposed method, Structure-SLAM, is more stable in rotation estimation compared with ORB-SLAM.

We also compare the number of frames tracked by different methods. Compared with ORB-SLAM, our method retrieves the camera pose more reliable. Especially in 'lr-kt2,' 'of-kt3' and 's-t-far,' our method initializes fast and tracks all frames in the sequences, as can be seen in sequence 'of-kt3' in Fig. 6 on the right. Similar results can be found for HRBB4 in Fig. 8. Compared with ORB-SLAM which only initializes after the 628th frame, our method is able to initialization much earlier around frame 110. Furthermore, the proposed method shows a more stable behaviour in the upper right corner of the corridor where the environment changes drastically.

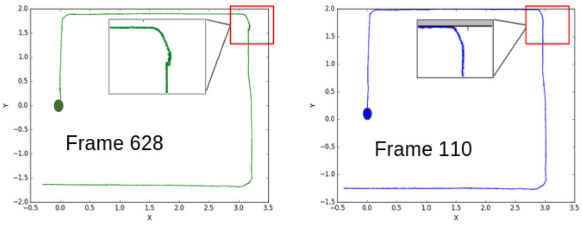


Fig. 8. The estimated trajectories of the camera on the HRBB4 [35] dataset. Left: ORB-SLAM, Right: Structure-SLAM.

VII. CONCLUSION

We have proposed a SLAM system for monocular cameras based on points, lines and surface normals. Using the Manhattan World assumption for rotation estimation and point and line features for windowed translation estimation we achieve state-of-the-art performance. We have shown that normals, learned from a single RGB image, can be used to estimate the rotation between frames leveraging the MW assumption. Compared to other state-of-the-art methods based on global rotation estimation, in our method there exists a fallback level using points and lines to estimate the full pose, in case no Manhattan frame can be found. This enables the tracking over short sequences to later re-localize within the Manhattan world. In the future, global bundle adjustment could be used to correct the frames during these sequences without global frames. Furthermore, we would like to leverage the learned structure information for the translation estimation as well.

REFERENCES

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: A versatile and accurate monocular SLAM system,” *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[2] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in *Proc. Springer Eur. Conf. Comput. Vision*, 2014, pp. 834–849.

[3] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[4] R. Gomez-Ojeda, F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, “PL-SLAM: A stereo SLAM system through the combination of points and line segments,” *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 734–746, Jun. 2019.

[5] Y. Lu and D. Song, “Robust RGB-D odometry using point and line features,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 3934–3942.

[6] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Morenonoguer, “PL-SLAM: Real-time monocular visual SLAM with points and lines,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4503–4508.

[7] P. F. Proença and Y. Gao, “Probabilistic RGB-D odometry based on points, lines and planes under depth uncertainty,” *Robot. Auton. Syst.*, vol. 104, pp. 25–39, 2018.

[8] H. Strasdat, J. Montiel, and A. J. Davison, “Scale drift-aware large scale monocular SLAM,” *Robot.: Sci. Syst. VI*, vol. 2, no. 3, pp. 73–80, 2010.

[9] H. Li, J. Yao, J.-C. Bazin, X. Lu, Y. Xing, and K. Liu, “A monocular SLAM system leveraging structural regularity in manhattan world,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2518–2525.

[10] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, “StructSLAM: Visual SLAM with building structure lines,” *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1364–1375, Apr. 2015.

[11] J. Straub, N. Bhandari, J. J. Leonard, and J. W. Fisher, “Real-time manhattan world rotation estimation in 3D,” in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2015, pp. 1913–1920.

[12] P. Kim, B. Coltin, and H. J. Kim, “Low-drift visual odometry in structured environments by decoupling rotational and translational motion,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7247–7253.

[13] P. Kim, B. Coltin, and H. J. Kim, “Linear RGB-D SLAM for planar environments,” in *Proc. Springer Eur. Conf. Comput. Vision*, 2018, pp. 333–348.

[14] Z. Yu, J. Zheng, D. Lian, Z. Zhou, and S. Gao, “Single-image piece-wise planar 3D reconstruction via associative embedding,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 1029–1037.

[15] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia, “GeoNet: Geometric neural network for joint depth and surface normal estimation,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 283–291.

[16] A. Handa, T. Whelan, J. McDonald, and A. Davison, “A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 1524–1531.

[17] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.

[18] G. Klein and D. Murray, “Parallel tracking and mapping on a camera phone,” in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2009, pp. 83–86.

[19] K. Tateno, F. Tombari, I. Laina, and N. Navab, “CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 6565–6574.

[20] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, “CodeSLAM–learning a compact, optimisable representation for dense visual SLAM,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 2560–2568.

[21] L. Ma, C. Kerl, J. Stückler, and D. Cremers, “CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1285–1291.

[22] M. Hsiao, E. Westman, G. Zhang, and M. Kaess, “Keyframe-based dense planar SLAM,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5110–5117.

[23] Y. Zhou, L. Kneip, C. Rodriguez, and H. Li, “Divide and conquer: Efficient density-based tracking of 3D sensors in Manhattan worlds,” in *Proc. Asian Conf. Comput. Vision*, 2016, pp. 3–9.

[24] R. Toldo and A. Fusiello, “Robust multiple structures estimation with J-linkage,” in *Proc. Springer Eur. Conf. Comput. Vision*, 2008, pp. 537–547.

[25] K. Joo, T.-H. Oh, J. Kim, and I. S. Kweon, “Globally optimal manhattan frame estimation in real-time,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 1763–1771.

[26] P. Kim, B. Coltin, and H. J. Kim, “Visual odometry with drift-free rotation estimation using indoor scene regularities,” in *Proc. Brit. Mach. Vision Conf.*, 2017, pp. 7–19.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2011, pp. 2564–2571.

[28] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, “LSD: A fast line segment detector with a false detection control,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, Apr. 2010.

[29] L. Zhang and R. Koch, “An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency,” *J. Vis. Commun. Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[30] R. Murartal and J. D. Tardos, “ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras,” *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[31] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “ScanNet: Richly-annotated 3D reconstructions of indoor scenes,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 2432–2443.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2009, pp. 248–255.

[33] P. K. N. Silberman, D. Hoiem, and R. Fergus, “Indoor segmentation and support inference from RGBD images,” in *Proc. Springer Eur. Conf. Comput. Vision*, 2012, pp. 746–760.

[34] C. Feng, Y. Taguchi, and V. R. Kamat, “Fast plane extraction in organized point clouds using agglomerative hierarchical clustering,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 6218–6225.

[35] Y. Lu, D. Song, and J. Yi, “High level landmark-based visual navigation using unsupervised geometric constraints in local bundle adjustment,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 1540–1545.

[36] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa, “PlaneNet: Piece-wise planar reconstruction from a single RGB image,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 2579–2588.

5 RGB-D SLAM with Structural Regularities

This work proposes an open-source SLAM system for RGB-D sensors, which uses points, lines, and planes of structured environments in tracking and mapping modules. Plane features are merged into our Manhattan-based framework, which estimates the initial translation vector and retains Manhattan relationships as constraints in the refinement module. Furthermore, an efficient meshing module is proposed that reconstructs the scene structure based on the obtained planar regions in the sparse map.

Based on the MW-based decoupled pose estimation theory, we improve the translation estimation by combining point and line features with planes and an additional pose refinement step with Manhattan relationships.

We propose a lightweight planar instance-wise mesh-based reconstruction method generating a compact representation of the environment from a sparse point cloud. A general framework for real-time RGB-D SLAM where these components are used to localize and map under structured environments with high accuracy.

Contributions. Yan Li proposed the architecture, and Yan Li and Raza Yunus implemented the system. Nokolas Brasch helped to correct the combined cost functions. Nikolas Brasch and Federico Tombari further helped to polish the methodology description.

RGB-D SLAM with Structural Regularities

Yanyan Li[1], Raza Yunus[1], Nikolas Brasch[1], Nassir Navab[1,2] and Federico Tombari[1,3]

[1] Technical University of Munich
[2] Johns Hopkins University
[3] Google

2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561560

RGB-D SLAM with Structural Regularities

Yanyan Li[1], Raza Yunus[1], Nikolas Brasch[1], Nassir Navab[1,2] and Federico Tombari[1,3]

Abstract— **This work proposes a RGB-D SLAM system specifically designed for structured environments and aimed at improved tracking and mapping accuracy by relying on geometric features that are extracted from the surrounding. Structured environments offer, in addition to points, also an abundance of geometrical features such as lines and planes, which we exploit to design both the tracking and mapping components of our SLAM system. For the tracking part, we explore geometric relationships between these features based on the assumption of a Manhattan World (MW). We propose a decoupling-refinement method based on points, lines, and planes, as well as the use of Manhattan relationships in an additional pose refinement module. For the mapping part, different levels of maps from sparse to dense are reconstructed at a low computational cost. We propose an instance-wise meshing strategy to build a dense map by meshing plane instances independently. The overall performance in terms of pose estimation and reconstruction is evaluated on public benchmarks and shows improved performance compared to state-of-the-art methods. The code is released at `https://github.com/yanyan-li/PlanarSLAM`.**

I. Introduction

Visual Simultaneous Localization and Mapping (SLAM) algorithms are used to estimate the 6D camera pose while reconstructing the surrounding unknown environment. They have shown to be useful in a wide range of applications, such as autonomous robots, self-driving cars and augmented/virtual reality, where camera pose estimation enables cars, robots and mobile devices to localize themselves, while the dense map provides a representation of the environment, *e.g.* for robot-environment or human-environment interaction.

Many SLAM applications have to deal with structured scenes, *i.e.* man-made environments that are usually characterized by low-textured surfaces - a typical example is an indoor scene, or an outdoor parking place. This induces a lack of visual features, that visual SLAM systems typically leverage to improve camera pose estimation and/or 3D reconstruction, *e.g.* by carrying out loop closure and bundle adjustment to reduce drift. In order to deal with structured scenes, specific SLAM methods based on points and line segments, like S-SLAM [1], Stereo-PLSLAM [2], PLVO [3], Mono-PLSLAM [4] and Probabilistic-VO [5] have been proposed, extending the working environment to scenes where more lines than points can be detected. SP-SLAM [6] merges plane features into ORB-SLAM2 [7], achieving robust results in low-textured scenes.

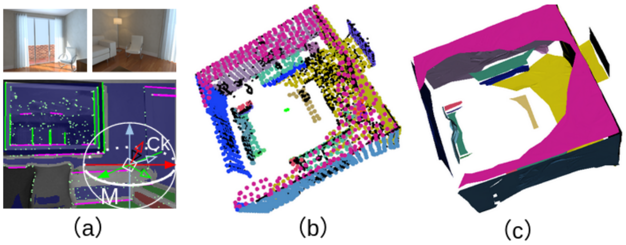


Fig. 1. RGB-D SLAM system. (a) Examples of a typical structured scene, and 2D features and orthogonal lines and planes segmentation. (b) Point cloud including points, lines and planes. (c) Real-time mesh on a CPU.

For the reconstruction, there are sparse, semi-dense and dense methods. Compared to the first two classes, which only provide incomplete maps, dense reconstruction is required to provide sufficient information for applications such as robot-environment interaction and 3D scene understanding. Many algorithms have been proposed to reconstruct indoor scenes via RGB-D sensors. KinectFusion [8] is a pioneering work relying on the truncated signed distance field (TSDF) representation of the map. In order to reconstruct large scale scenarios, surfel-based methods, like ElasticFusion [9], were proposed. Instead of reconstructing each pixel, Wang et al. [10] extracts superpixels from RGB images and depth maps, which is more efficient but still has redundant information especially in indoor scenarios where large planes can be commonly found.

In this paper, we build on our monocular Structure-SLAM [1] and propose a robust RGB-D SLAM system specifically designed to deal with structured environments, which improves tracking and mapping at the same time. Figure 1 illustrates the components of such structured scenes, which contains points, lines and plane segments. Following the decoupling strategy of Structure-SLAM, we estimate a drift-free rotation matrix first, and then the 3-DoF translation. The initial rotation and translation are refined via a map-to-frame strategy. Different to [1], [11], [12], plane features are merged into our Manhattan-based framework, which is used to estimation the initial translation vector and retain Manhattan relationships as constrains in the refinement module. Furthermore, an efficient meshing module is proposed that reconstructs the scene structure based on the obtained planar regions in the sparse map. In summary our contributions are:

- Based on the concept of MW-based decoupled pose estimation, we improve the translation estimation by combining point and line features with planes and an additional pose refinement step with Manhattan relationships.

[1]:Technical University of Munich, Germany; {`yanyan.li, raza.yunus, nikolas.brasch, nassir.navab, federico.tombari`}`@tum.de`; [2]:Johns Hopkins University, USA;[3]:Google Inc.

- We propose a planar instance-wise mesh based reconstruction method generating a compact representation of the environment from a sparse point cloud.
- A general framework for real-time RGB-D SLAM where these components are used to localize and map under structured environments with high accuracy.

We evaluate the performance of our approach in terms of both camera pose estimation and reconstruction on public benchmarks, showing improved performance compared to state-of-the-art methods.

II. Related work

In the following we review the literature related to RGB-D based SLAM systems as well as methods leveraging structural regularity as the MW assumption.

a) RGB-D SLAM.: In [13], [14] it was proposed to use planes over point features whenever possible, as the averaging over multiple depth measurements reduces the noise significantly. In Dense Planar SLAM [15] surfels belonging to the same planar areas are smoothed by fitting a plane to them and back-projecting the surfels onto the plane. Le *et al.* [16] rely on a scene layout consisting of a ground plane and several walls, and use dynamic programming to infer a sequentially consistent assignment of pixels to planes. In Probabilistic-VO [5], the uncertainties of points, lines and planes are modelled explicitly and used during pose estimation, where points, lines and planes are represented in a uniform framework in [17]. A direct SLAM system combining photo-metric and geometric terms is proposed in DVO-SLAM[18] and extended in CPA-SLAM [19] with global planes, where depth measurements are assigned to the global planes with weights.

b) Dense Reconstruction.: While the aforementioned methods have the goal to estimate precise poses and therefore only maintain a map with the most reliable information, several works have been proposed with the goal to create a complete dense reconstruction of the environment. KinetFusion [8] and ElasticFusion [9] explore dense reconstruction for RGB-D sensors. The first method fuses all depth data into a volumetric dense representation, which is used to track the camera pose using ICP. The size of the map is usually limited in volumetric methods due to memory constraints. Different from KinectFusion, ElasticFusion is a map-centric system that reconstructs surfel-based maps of the environment. In order to decrease the number of surfels in the map, superpixel-based surfels are proposed by [10], which reduces the number of surfels compared with ElasticFusion. Recently BAD-SLAM [20] proposed a direct bundle-adjustment approach for RGB-D SLAM. In [21] a textured mesh is extracted from a dense surfel cloud. A direct mesh based reconstruction approach for RGB-D sensors was proposed in [22].

c) Structural Regularity.: A line of works exploits additional constrains and regularities in the world, to improve the reconstruction performance. In [23] and [24] the authors showed that the rotation estimation error is the main reason for long-term drift.

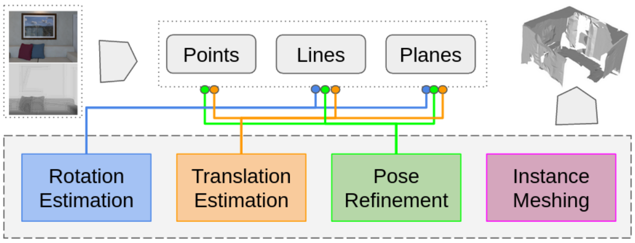


Fig. 2. Overview of the proposed framework. Point, line and plane features are extracted from the RGB-D frame. Rotation and translation are estimated in a decoupled fashion first and refined afterwards. The planar segments are used to create a mesh-based reconstruction of the environment.

A branch-and-bound framework for Manhattan Frame estimation is proposed in [25]. In MVO [24] a method using mean shift on the unit sphere is used to find the transformation between the MW and the current frame. When only planes are used for the rotation estimation as in OPVO [26] at least 2 orthogonal planes must be detected in each frame, the addition of vanishing points extracted from lines can be used alternatively, as done in LPVO [11]. The methods mentioned above use point features to estimate the translation. Structure-SLAM [1] is a monocular system that predicts normals via a convolutional neural network leverages normals with points and lines in a decoupling strategy. Since predicted normals are not as accurate as those computed from a depth map, the system provide a refinement/fallback module based on points and lines. Compared with Structure-SLAM, optimized vanishing points of lines and plane features are used for rotation and translation in this work. Then, the fallback part is removed and the refinement part incorporates geometric relationship of planes. Instead of the sparse point-line map, a dense mesh as output is more useful for robotics applications. L-SLAM [12] is also based on the MW assumption, which obtains translation, rotation and pixels of potential planar regions from LPVO. Then it refines 3D translational and 1-D plane offsets with a linear Kalman Filter. However, we use a more robust front-end for initial translation estimation. Furthermore, the 6D pose refinement step is used to optimize rotation and translation simultaneously and allows an offset to the initial rotation from MW, which is more robust to non-MW (curved surfaces and few planar regions) compared with L-SLAM and LPVO (see Figure 6).

III. Proposed framework

Given a sequence of RGB-D frames from a structured environment, the goal of our method is to reconstruct the 3D scene while simultaneously estimating the 6D camera pose at each frame. Section IV provides an overview of the proposed tracking pipeline, which decouples rotation and translation, while section V describes different types of mapping presentations generated by the system. We now describe the system's underlying features and structural components.

A. Extended feature set

In our method we use ORB features [27], which are fast to extract and match. In low-textured environments, it is hard to extract sufficient points for robust pose estimation, therefore we extend the feature set with lines, which are extracted using the LSD [28] approach, as described by LBD [29]. Furthermore, it is common to find non-textured planar regions in indoor environments, where plane instances extracted from the depth maps are valuable cues to extend points and lines. Planes are detected using the connected component analysis method [30]. They are represented by the Hessian normal form $\pi = (\hat{n}, d)$, where $\hat{n} = (n_x, n_y, n_z)$ is the normal of the plane, representing its orientation and d is the distance from the camera origin to the plane.

a) Points and lines: After the extraction of 2D point features $x_j = (u_j, v_j)$ and line segments $l_j = (x_{j,start}, x_{j,end})$ in frame F_i, we can back-project points and lines using the camera intrinsic parameters and the depth map to obtain 3D points X_j and 3D lines L_j. The depth map is not always correct, especially at depth discontinuities *e.g.* object boundaries. Therefore a robust fitting method for 3D lines is needed. First, we count the number of pixels with non-zero depth values intersected by the detected line segment. If the number exceeds a certain threshold, the 3D line L_j will be estimated via RANSAC to remove potential outliers.

b) Normals and planes.: Smooth normals are computed by averaging the tangential vectors from the depth image inside a patch of 10×10 pixels using integral images. After plane detection, we use the strategy of [6] to associate the observed planes with those present in the map. To match an observed plane with one from the map, we first check the angle between their normals. If it is below the threshold θ_n, we check the point-to-plane distance between them. The plane which has the minimum distance to the observed plane, and also lies below the distance threshold θ_P, is matched to the observed plane. In the experiments, θ_n and θ_P are set at 10 degrees and 0.1 m respectively. Furthermore, we also keep parallel and perpendicular relationships [6] between the map planes to leverage additional constraints during the tracking process. These are determined by the angle between the plane normals. Since they only provide constraints for orientation, we do not consider their distance.

B. Decoupling pose estimation and refinement.

To reduce error propagation between frames, we build on our monocular architecture [1] that computes rotational motion based on the MW assumption. Then the corresponding translational motion is estimated by features, with the fixed rotation computed from last step. In the font-end of this work, we use optimized lines for rotation estimation and planes for translation estimation.

Differently to Structure-SLAM [1] that uses a point-line local map to optimize translation and rotation together, we leverage planes in the local map and also make use of the geometric relationship (parallel and perpendicular) of those planes as constrains, which improves the accuracy of the system as it will be shown in Figure 4 and Table I.

IV. Tracking

Differently from traditional pose estimation methods, we decouple the 6D camera pose into rotation and translation. Based on the MW assumption, we obtain the rotational motion $R_{c_i m}$ between the MW and camera c_i. In this way, the rotation estimation will not be affected by the pose of the last frame or last keyframe, which reduces drift effectively. Afterwards, point, line and plane features as well as the initial rotation matrix are used for translation estimation, which consists of just 3 Degrees-of-Freedom (DoFs).

A. Rotation estimation

Instead of tracking the camera from frame-to-frame directly, the drift-free rotation estimation method estimates the rotation R_{cm} between each frame and the Manhattan coordinate frame, by modeling the indoor environments as a MW, thus reducing the drift generated from frame-to-frame tracking. As shown in Figure 1, Manhattan coordinate frames can be aligned to the starting frame of the camera via $R_{k+1,m}$. Generally, the coordinate of the first frame is regarded as the world frame, *i.e.* $R_{1,m} = R_{m,w}^T$. So we can obtain pose in the world coordinate by using,

$$R_{k+1,w} = R_{k+1,m} R_{m,w} \tag{1}$$

Here $R_{m,w}$ represents the relation from the world to MW, which is obtained by the MW initialization step and $R_{k+1,m}$ is the relation from MW to the $(k+1)^th$ frame. These two matrices are computed via a sphere mean-shift method [24], where the normals and normalized vanishing directions are projected onto the tangent planes of the current rotation estimate. Then a mean shift step is performed on the tangent planes, which generates new centers and back-projects them to the sphere as new estimates. We refer the reader to [24] and [26] for more details on the sphere mean-shift method. To handle difficult scenes where only one or no plane at all is detected, we feed the unit sphere with both vanishing directions of the refined 3D lines and surface normals of planes, which is a more robust approach than [26], [1] under these challenging conditions.

B. Translation estimation

After estimating the rotation, points, lines and planes are used to estimate the translation. We re-project 3D points from the last frame into the current one and define the error function, based on the re-projection error, as follows,

$$e^p_{k,j} = p_k - \Pi(R_{k,j} P_j + t_{k,j}) \tag{2}$$

where $\Pi(\cdot)$ is the projection function. Since the rotation matrix $R_{k,j}$ has been obtained in the last step, we fix the rotation and only estimate the translation using the Jacobian matrix corresponding to (2).

As for lines, we obtain the normalized line function from the 2D endpoints p_{start} and p_{end} as follows

$$l = [p_{start} \times p_{end}] / [\|p_{start}\| \|p_{end}\|] = (a, b, c). \tag{3}$$

Then, we formulate the error function based on the point-to-line distance [4] between l and the projected 3D endpoints P_{start} and P_{end} from the matched 3D line in the keyframe. For each endpoint P_x, the error function can be noted as,

$$e^l_{k,P_x} = l\Pi(R_{k,j} P_x + t_{k,j}). \tag{4}$$

To get a minimal parameterization of a plane π for optimization, we represent it as $q(\pi) = (\phi, \psi, d)$ where ϕ and ψ are the azimuth and elevation angles of the normal and d is the distance from the Hessian form

$$q(\pi) = (\phi = arctan(\frac{n_y}{n_x}), \psi = arcsin(n_z), d). \quad (5)$$

So, the error function between the observed plane π_k in the frame and corresponding map plane π_x is

$$e^{\pi}_{k,\pi_x} = q(\pi_k) - q(T_{cw}^{-T}\pi_x) \quad (6)$$

where T_{cw}^{-T} is the transformation from world to camera coordinates.Assuming that the observations follow a Gaussian distribution, the final non-linear least squares cost function $t*$ can be written as in (7), where $\Lambda_{p_{k,j}}$, $\Lambda_{p_{k,P_x}}$ and Λ_{k,π_x} are the inverse covariance matrices of points, lines and planes, and ρ_p, ρ_l and ρ_π are robust Huber cost functions, respectively.

$$\begin{aligned} t* =& argmin \sum_{j}^{M} \rho_p \left({e^p_{k,j}}^T \Lambda_{p_{k,j}} e^p_{k,j} \right) \\ &+ \rho_l \left({e^l_{k,P_x}}^T \Lambda_{p_{k,P_x}} e^l_{k,P_x} \right) + \rho_\pi \left({e^\pi_{k,\pi_x}}^T \Lambda_{k,\pi_x} e^\pi_{k,\pi_x} \right) \end{aligned} \quad (7)$$

Here, a solution is determined using the Levenberg-Marquardt algorithm.

C. Pose refinement

The last two steps assume that the scene is a good Manhattan model, nevertheless several general indoor environments are not strictly adhering to the MW assumption, leading to degradation in accuracy. So, after obtaining the initial pose via the decoupled rotation and translation strategy, the refinement module [1] fine-tunes the pose to compensate for deviations from the MW or unstable initial estimates. In the refinement step, to reduce the drift from frame-to-frame pose estimation, the local map constructed by previous keyframes is used to optimize the pose based on a map-to-frame strategy [7].

Similar to [6], [7], [31], we also use keyframes to build a local map, although our map has point, line and plane landmarks, which are projected into the current frame to search for matches. Furthermore, we explore the relationship between planes in the local map and planes detected in the current frame. The parallel and perpendicular constraints between those planes are described as (8),

$$\begin{cases} e^{\pi_{\parallel}}_{k,n_x} = & ||q_n(n_k) - q_n(R_{cw}n_x)|| \\ e^{\pi_{\perp}}_{k,n_x} = & ||q_n(R_{\perp}n_k) - q_n(R_{cw}n_x)|| \end{cases} \quad (8)$$

where $q_n(\pi) = (\phi, \psi)$ and R_{cw} is the transformation from world to camera coordinates. For perpendicular planes, their plane normal is rotated by 90 degrees ($R_{\perp}$) to construct the error function. These two error functions are merged to (7) to build a joint optimization function in the refinement module.

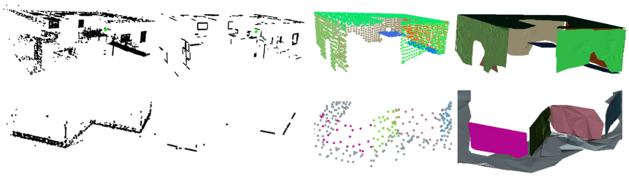


Fig. 3. Different levels of maps provided by the system.Top row: office room of the ICL-NUIM; bottom row: structure-nontexture-near of TUM RGB-D;

V. Mapping

This section describes the keyframe-based 3D mapping strategy used in our SLAM framework. Keyframes and 3D features build up a co-visibility graph, where nodes and edges are updated whenever a new keyframe and new features are available.

A. Sparse Mapping

As shown in Figure 3, the sparse map module is reconstructed by point-line-plane features extracted from keyframes. The first frame is set as the first keyframe and the global map is initialized by the landmarks thereby detected. When new points, lines and planes are detected in a new keyframe, which are not in the global map, they will be saved to a local map first. Then we check the quality of the landmarks in the local map, and then push reliable landmarks into a global map after culling bad ones. Different to the matching methods for points and lines, for each detected plane in a new keyframe, we first check whether it is associated with a plane in the map using the strategy described in section III. If we find an association, we add the 3D points of the new plane to the associated plane in the global map and filter out redundancies using a voxel grid to get a compact point cloud again. If the incoming plane is not associated to any plane in the global map, we add it to the map as a new plane.

B. Planar instance-wise meshing

The sparse map obtained in the previous section is still not adequate for applications involving robot-environment interactions, but it provides information about planar and non-planar instances. Therefore, we construct a denser map using an instance-wise meshing strategy. Indoor scenes can be divided into planar and non-planar regions. Planar areas like floors, walls and ceiling have often a large extent, however a dense pixel-wise information does not add to the quality and is highly redundant. So instead of using surfel or TSDF, we regard plane regions as instances that include a small and fixed number of elements independently of their size.

In particular, we input plane instances to the meshing module, which meshes them independently. First, the points belonging to a plane are organized as a kd-tree data-structure. Different to unstructured inputs, our method needs less time for searching several nearest neighbors. Then, we use Greedy Surface Triangulation (GST) [33] to build an instance-wise mesh, which is designed to deal with planar surfaces. Note

Sequence	Ours	Ours/-wo	ORB [7]	PS-SLAM [6]	LPVO [12]	L-SLAM [11]	DVO [18]	InfiniTAM [32]
lr-kt0	**0.006**	0.025	0.025	0.016	0.015	0.012	0.108	×
lr-kt1	0.015	0.036	0.008	0.018	0.039	0.027	0.059	**0.006**
lr-kt2	0.020	0.053	0.023	0.017	0.034	0.053	0.375	**0.013**
lr-kt3	**0.012**	0.059	0.021	0.025	0.102	0.143	0.433	×
of-kt0	0.041	0.068	0.037	0.032	0.061	**0.020**	0.244	0.042
of-kt1	0.020	0.028	0.029	0.019	0.052	**0.015**	0.178	0.025
of-kt2	**0.011**	0.060	0.039	0.026	0.039	0.026	0.099	×
of-kt3	0.014	0.012	0.065	0.012	0.030	0.011	0.079	**0.010**
snot-far	0.022	0.026	×	**0.020**	0.075	0.141	0.213	0.037
snot-near	0.025	×	×	**0.013**	0.080	0.066	0.076	0.022
cabinet	**0.035**	0.057	0.075	0.067	0.520	0.291	0.690	**0.035**
large-cabinet	**0.071**	0.813	0.124	0.079	0.279	0.140	0.979	0.512

TABLE I

COMPARISON OF TRANSLATION RMSE (M) FOR ICL-NUIM AND TUM-RGB-D SEQUENCES. × MEANS THE METHOD FAILS IN THE TRACKING PROCESS. -WO MEANS ONLY USING DECOUPLED TRACKING WITHOUT THE REFINEMENT STEP.

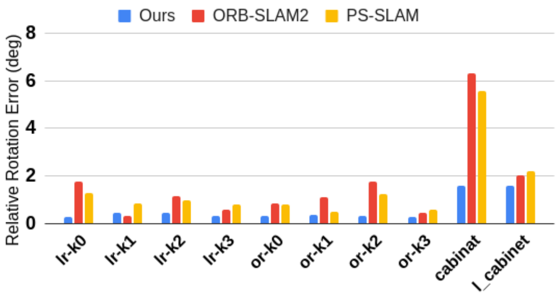


Fig. 4. Comparison of relative pose error (RPE) for rotation on the ICL-NUIM and TUM RGB-D sequences.

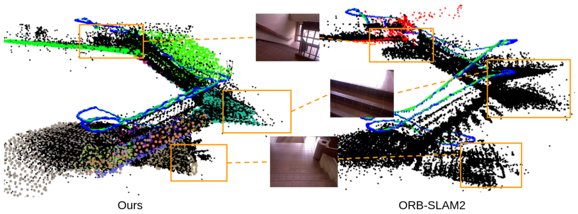


Fig. 5. Qualitative results of sparse reconstruction and trajectory between the proposed method and ORB-SLAM2 in the TAMU dataset.

time	Feat. extr.	Rotat.	Transla	Refinement	Total
Median	19.9	2.1	4.8	13.0	42.5
Mean	20.5	3.0	5.4	13.1	43.7
Std.	3.6	0.4	2.8	4.8	9.4

TABLE II

MEASURED TRACKING TIMES (MS) ON THE TUM RGB-D SEQUENCES

Sequence	RGB-D	ElasticFu	InfiniTAM	SPFu	Ours
kt0	4.4	0.7	1.3	0.7	**0.4**
kt1	3.2	0.7	1.1	0.9	**0.6**
kt2	3.1	0.8	**0.1**	1.1	0.6
kt3	16.7	2.8	2.8	1.0	**0.8**

TABLE III

RMSE RECONSTRUCTION ERROR (CM) ON THE ICL-NUIM DATASET IN CENTIMETERS.

that in our experiments, the initial search radius for selecting neighbors for triangulation is set to $5m$ and the multiplier is set as 5 to modify the final search radius to adapt to different point densities on the plane regions.

VI. EXPERIMENTS

We evaluate the proposed SLAM system on two well known public datasets, the ICL-NUIM [34] and TUM RGB-D [35] benchmarks, comparing its performance with other state-of-the-art methods such as ORB-SLAM2 [7], PS-SLAM [6] that are feature-based methods, but removed the global bundle adjustment modules in the following experiments. Methods based on the MW assumption such as LPVO [11] and L-SLAM [12]. DVO-SLAM [18] is a direct method and InfiniTAM [32] uses a GPU for real-time tracking and mapping based on RGB and depth images. Additionally, we provide the reconstruction accuracy of our reconstructed model on the ICL-NUIM dataset and compare it with other popular methods for dense reconstruction. Lastly, to demonstrate that our system is robust over time, we also test on a sequence from the TAMU [3] dataset containing long sequences covering a large indoor area. All experiments are carried out with an Intel Core i7-8700 CPU (with @3.20GHz) and without any use of GPU. The ICL-NUIM dataset [34] provides synthetic scenes for two indoor environments, one living room and one office room scenario. These scenes contain large areas of low textured surfaces such as walls, ceilings, floors, etc. There are four sequences for each scene. We evaluate our method on all sequences.

A. *ICL-NUIM RGB-D Dataset*

Table I shows that our method obtains the best performance on three out of the eight sequences, based on the translation RMSE (ATE). InfiniTAM also performs well on lr-kt1, lr-kt2 and of-kt3 sequences, but the method also loses tracking in other sequences. As the dataset contains large structured areas, the Manhattan-based methods LPVO and L-SLAM are able to get a good estimate of the orientation and provide good results throughout. However, they usually

Sequences	Ours	ORB-SLAM2	length
Corridor-A	1.62	3.13	88
Stair-A	0.94	1.44	66
Entry-Hall	1.33	2.22	80

TABLE IV

COMPARISON OF THE ACCUMULATED DRIFT (M) IN DIFFERENT LARGE SCALE SEQUENCES.

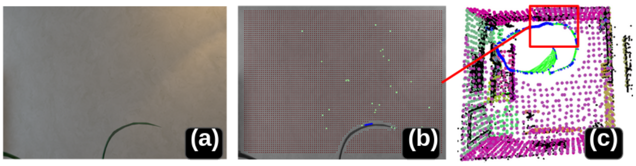


Fig. 6. Results in lr-k3. (a) input image; (b) point-line features and the segmented plane; (c) reconstructed 3D map and trajectory.

need two planes, or alternatively, one plane and a vanishing direction to be visible at all times to estimate a good Manhattan frame. As shown in Figure 6, there are several challenging scenes in lr-kt3, where only a white wall and two leaves from a plant are captured when the camera is close to the wall. In this situation, OPVO and L-SLAM are unable to yield a good performance. When a bad initial pose is obtained in our system due to the scene not being a rigid MW, the refinement step based on point-line-plane features allows us to recover the pose nevertheless, while L-SLAM ignores optimizing rotation in the LKF module. Moreover, while DVO, being a dense method, may struggle because of the large areas of walls, floor etc. not containing enough gradient for the photometric error, ORB-SLAM2 and PS-SLAM perform well, as both environments contain sufficient ORB features extracted from furniture, objects etc. As our method takes advantage of all geometric elements, it is able to perform robustly in most sequences. In addition, Figure 4 shows the relative pose error for ORB-SLAM, PS-SLAM and our method. Our method obtains notably better results than the other two in relative translation and rotation. Especially the rotation error is much lower for our method, due to the use of the decoupled MW rotation estimation.

B. TUM RGB-D Dataset

The TUM RGB-D benchmark [35] is one of the most popular datasets for RGB-D SLAM systems, which provides indoor sequences under different texture and structure conditions. This allows us to separately test sequences which have structure, texture or both. In order to evaluate our method in challenging environments, we select four structured image sequences, the first three with low texture and the last one with a large scale environment. As all sequences listed in Table I have structure, but the large-cabinet sequence is not a rigid Manhattan scenario. Manhattan-based methods are able to provide good pose estimates on snot-far sequence, but the results degenerate in large-cabinet and cabinet sequences. The first two sequences include the same environment consisting of multiple non-textured planes. Here ORB-SLAM2 is not able to find enough point correspondences along the sequence and loses tracking. Our method, which additionally uses lines and planes for translation estimation, achieves better results. As shown in Figure 4, cabinet and large-cabinet are challenging sequences because of several low-texture frames. Our method's tracking strategy limits the relative rotation error to under 2 degrees, which is better than ORB-SLAM2 and PS-SLAM. The statistics of the time spent for each operation are shown in Table II, where we use different CPU threads to deal with points, lines and planes in the feature extraction and refinement modules.

C. Large scale sequence

The TAMU dataset [36] provides large-scale indoor sequences (constant lighting). While it does not provide ground-truth camera poses, the start and end point are the same, which can be used to evaluate the overall drift by computing the final position errors. As shown in Figure 5, the trajectory in the sequence Stair-C is a loop between two floors, where the improvement of our method over the whole trajectory length is 34.7% in drift compared to ORB-SLAM2. Similar situations can also be found in Corridor-A and Entry-Hall. More qualitative results are provided in the supplementary material.

D. Reconstruction Accuracy

We reconstruct models from ICL-NUIM and compare the results with state-of-the-art mapping methods, as shown in Table III. The accuracy of the reconstruction results is defined as the mean difference between the predicted model and the ground-truth model [34]. We compare the proposed mapping module against RGB-D SLAM [37], ElasticFusion [9], InfiniTAM [38], and SuperpixelFusion [10].

The SuperpixelFusion method is constrained by using ORB-SLAM for pose estimation, whereas our method also works well in low-textured environments. InfiniTAM obtains the best results in kt2, but shows worse performance on the kt0 and kt3 sequences, potential due to the large low-textured regions. ElasticFusion shows a similar behavior. Our method reconstructs more accurate maps than the others, but InfiniTAM and ElasticFusion provide more complete models than our map since we ignore small objects even though features based on points, lines and planes cover most of the pixels. Remarkably, all fusion methods, except for SuperpixelFusion and ours, rely on GPU based acceleration.

VII. CONCLUSIONS

We have proposed a RGB-D SLAM system based on points, lines and planes. Using the MW assumption for rotation estimation, and point, line and plane features for translation estimation, we achieve state-of-the-art performance. Also, a novel instance-wise meshing approach can reconstruct planar regions in the environment efficiently. The resulting dense map allows for interactions with the environment in robotic and AR/VR applications. In the future we would like to extend the planar reconstruction with a meshing of the non-planar parts in the environment to allow the complete reconstruction of more complex scenes.

References

[1] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari, "Structure-slam: Low-drift Monocular SLAM in Indoor Environments," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2020.

[2] R. Gomezojeda, F. Moreno, D. Scaramuzza, and J. G. Jimenez, "PL-SLAM: a Stereo SLAM System Through the Combination of Points and Line Segments." *IEEE Trans. Robot.*, 2019.

[3] Y. Lu and D. Song, "Robust RGB-D Odometry Using Point and Line Features," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015.

[4] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Morenonoguer, "PL-SLAM: Real-time Monocular Visual SLAM with Points and Lines," in *in Proc. IEEE Int. Conf. Robot. Automat.*, 2017.

[5] P. F. Proença and Y. Gao, "Probabilistic RGB-D odometry based on points, lines and planes under depth uncertainty," *Robotics and Autonomous Syst.*, vol. 104, pp. 25–39, 2018.

[6] X. Zhang, W. Wang, X. Qi, Z. Liao, and R. Wei, "Point-plane SLAM using supposed planes for indoor environments," *Sensors*, vol. 19, no. 17, p. 3795, 2019.

[7] R. Murartal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, 2017.

[8] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time Dense Surface Mapping and Tracking," in *IEEE Int. Symposium on Mixed and Augmented reality*, 2011.

[9] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "ElasticFusion: dense SLAM without a pose graph," in *Robotics: Science and Syst.*, 2015.

[10] K. Wang, F. Gao, and S. Shen, "Real-time Scalable Dense Surfel Mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019.

[11] P. Kim, B. Coltin, and H. J. Kim, "Low-drift Visual Odometry in Structured Environments by Decoupling Rotational and Translational Motion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018.

[12] P. Kim, B. Coltin, and H. Jin Kim, "Linear RGB-D SLAM for Planar Environments," in *Proc. Springer Eur. Conf. Comput. Vision*, 2018.

[13] C. Raposo, M. Lourenço, M. Antunes, and J. P. Barreto, "Plane-based odometry using an RGB-D camera." in *Proc. Brit. Mach. Vision Conf.*, 2013.

[14] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane SLAM for hand-held 3D sensors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013.

[15] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, "Dense Planar SLAM," in *IEEE Int. Symposium on Mixed and Augmented reality*, 2014.

[16] P.-H. Le and J. Košecka, "Dense piecewise planar RGB-D SLAM for indoor environments," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2017.

[17] F. Nardi, B. Della Corte, and G. Grisetti, "Unified representation and registration of heterogeneous sets of geometric primitives," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 625–632, 2019.

[18] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2013.

[19] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "CPA-SLAM: Consistent Plane-model Alignment for Direct RGB-D SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016.

[20] T. Schops, T. Sattler, and M. Pollefeys, "Bad SLAM: bundle adjusted direct RGB-D SLAM," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019.

[21] T. Schöps, T. Sattler, and M. Pollefeys, "Surfelmeshing: online surfel-based mesh reconstruction," *IEEE Trans. on Pattern Analysis and Machine Intell.*, 2019.

[22] S. Schreiberhuber, J. Prankl, T. Patten, and M. Vincze, "ScalableFusion: High-resolution Mesh-based Real-time 3D Reconstruction," in *in Proc. IEEE Int. Conf. Robot. Automat.*, 2019.

[23] J. Straub, N. Bhandari, J. J. Leonard, and J. W. Fisher, "Real-time Manhattan World Rotation Estimation in 3D," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2015.

[24] Y. Zhou, L. Kneip, C. Rodriguez, and H. Li, "Divide and Conquer: Efficient Density-based Tracking of 3D Sensors in Manhattan Worlds," in *Proc. Asian Conf. Comput. Vision*, 2016.

[25] K. Joo, T.-H. Oh, J. Kim, and I. So Kweon, "Globally Optimal Manhattan Frame Estimation in Real-time," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016.

[26] P. Kim, B. Coltin, and H. J. Kim, "Visual Odometry with Drift-free Rotation Estimation Using Indoor Scene Regularities," in *Proc. Brit. Mach. Vision Conf.*, 2017.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vision*, 2011.

[28] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A Fast Line Segment Detector with a False Detection Control," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 32, no. 4, pp. 722–732, 2010.

[29] L. Zhang and R. Koch, "An Efficient and Robust Line Segment Matching Approach Based on LBD Descriptor and Pairwise Geometric Consistency," *Elsevier Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[30] A. J. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen, "Efficient Organized Point Cloud Segmentation with Connected Components," *Semantic Perception Mapping and Exploration*, 2013.

[31] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a Versatile and Accurate Monocular SLAM System," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[32] V. A. Prisacariu, O. Kähler, S. Golodetz, M. Sapienza, T. Cavallari, P. H. Torr, and D. W. Murray, "Infinitam v3: A Framework for Large-scale 3D Reconstruction with Loop Closure," *arXiv preprint arXiv:1708.00783*, 2017.

[33] Z. C. Marton, R. B. Rusu, and M. Beetz, "On Fast Surface Reconstruction Methods for Large and Noisy Datasets," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009.

[34] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014.

[35] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2012.

[36] Y. Lu and D. Song, "Robustness to Lighting Variations: An RGB-D Indoor Visual Odometry Using Line Segments," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2015.

[37] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An Evaluation of the RGB-D SLAM System," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012.

[38] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time Large-scale Dense 3D Reconstruction with Loop Closure," in *Proc. Springer Eur. Conf. Comput. Vision*, 2016.

6 Co-Planar Parametrization for Stereo-SLAM and Visual-Inertial Odometry

This work proposes a visual-inertial system for monocular-imu and stereo-imu setups. Motivated by those methods that leverage more constraints into the bundle optimization modules, we explore explicit methods to present those constraints to real relationships.

The relationship we focus on is co-planarity, meaning point or line landmarks are located in the same planar area. First, the paper builds an efficient and robust parametrization of co-planar points and lines, where the plane represents those points and lines to take the place of the original individual representations. The strategy leverages specific geometric constraints to improve camera pose optimization in terms of efficiency and accuracy.

In addition, to support the robustness of the co-planar parametrization method, a reliable plane detection module is proposed based on neural layer and RANSAC outlier removing approaches. The pipeline consists of extracting 2D points and lines, predicting planar regions, and filtering the outliers via RANSAC. Our parametrization scheme then represents co-planar points and lines as their 2D image coordinates and parameters of planes.

Contributions. Xin Li, Yan Li, and Federico Tombari proposed connecting neural networks with geometric methods for robust localization methods. I implemented the plane-instance segmentation network and evaluated the computation time in experiments. Federico Tombari and Jinlong Lin suggested to design the VIO system for monocular and stereo sensors. Evin Pınar Örnek helped to polish out the paper and videos. Xin Li evaluated our camera pose results in experiments.

Co-Planar Parametrization for Stereo-SLAM and Visual-Inertial Odometry

Xin Li[1*], Yanyan Li[2*] , Evin Pınar Örnek[2], Jinlong Lin[1], and Federico Tombari[2,3]

[1] Peking University
[2] Technical University of Munich
[3] Google
[*] Authors contribute equally

Co-Planar Parametrization for Stereo-SLAM and Visual-Inertial Odometry

Xin Li, Yanyan Li, Evin Pınar Örnek, Jinlong Lin, and Federico Tombari

Abstract—**This letter proposes a novel SLAM framework for stereo and visual inertial odometry estimation. It builds an efficient and robust parametrization of co-planar points and lines which leverages specific geometric constraints to improve camera pose optimization in terms of both efficiency and accuracy. The pipeline consists of extracting 2D points and lines, predicting planar regions and filtering the outliers via RANSAC. Our parametrization scheme then represents co-planar points and lines as their 2D image coordinates and parameters of planes. We demonstrate the effectiveness of the proposed method by comparing it to traditional parametrizations in a novel Monte-Carlo simulation set. Further, the whole stereo SLAM and VIO system is compared with state-of-the-art methods on the public real-world dataset EuRoC. Our method shows better results in terms of accuracy and efficiency than the state-of-the-art. The code is released at https://github.com/LiXin97/Co-Planar-Parametrization.**

Index Terms—**SLAM, Visual Learning.**

I. Introduction

SIMULTANEOUS Localization and Mapping (SLAM) and Visual Inertial Odometry (VIO) algorithms aim at camera pose estimation and scene reconstruction under unknown environments. They are ubiquitously employed in robotics for tasks such as planning, obstacle avoidance and navigation. When applied to indoor environments, these methods have to face important challenges due to the poor visual features available in the scene, which is often mostly characterized by low textured surfaces.

It has been shown that the structural regularities in the environment (e.g., lines and planes) bring valuable information to both SLAM and VIO systems [1], [2]. Such features can guide the SLAM optimization process by introducing additional constraints. However, how to organize such structural information and integrate it with the optimization in an efficient way is still an open question. Traditional representations focused on improving the trajectory accuracy, yet they ignored the high computational burden. In this work, we aim to tackle this problem by designing a better representation for planar structures, which simultaneously improves the accuracy and the efficiency of integrated stereo SLAM and VIO systems.

So far in the literature, several works leveraged points and lines detected from an RGB image to handle challenging environments [2]–[5]. Yet, the inner geometric relationship between those features is ignored in most of them. Different than using independent features of line segments and points, planar regions require fewer parameters to represent environments. Such planar regions and features can be found in almost all man-made environments, and they have been studied and leveraged in stereo SLAM and VIO systems [1], [6]–[9]. They introduce more constraints to the system that are helpful to improve overall accuracy. Nevertheless, they also rely on a high number of optimization parameters yielding limitations in real-world scenarios.

In this work, we propose a novel method to employ planarity constraints to improve the accuracy and efficiency of SLAM models based on VIO or stereo in indoor environments. Our method detects the co-planar point and line features through a deep learning based plane detection followed by RANSAC filtering. We then introduce a novel parametrization to represent these co-planar features in an unified manner instead of using them as independent features. The resulting parametrization decrease the size of Hessian matrix, as well as make it sparser as shown in Fig. 1(e). As a result, solving the bundle adjustment problem for estimating the correct camera parameters and 3D landmarks through second-order Newton optimization, which relies on calculating Schur complement on the Hessian matrix, becomes more efficient.

Furthermore, we show how our parametrization model can be integrated in a stereo SLAM or VIO pipeline as shown in Fig. 2 as we want to prove that our plane extraction and parametrization methods are general. By taking either a stereo image input, or an image with IMU sensor data, we solve the tracking and mapping problem through a graph based optimization. The non-planar 3D landmarks are integrated in the traditional way as 3D points, whereas the planar landmarks are introduced within the pipelines through proposed co-planarity parameters.

For evaluation, we used the public real-world EuRoC dataset and a newly created Monte-Carlo simulation set to perform further ablation studies. We compare our stereo-SLAM method against point-line SLAM approaches, as well as our VIO

Manuscript received May 12, 2020; accepted September 9, 2020. Date of publication September 28, 2020; date of current version October 8, 2020. This letter was recommended for publication by Associate Editor Todor Stoyanov and Editor Sven Behnke upon evaluation of the Reviewers' comments. *(Xin Li and Yanyan Li contributed equally to this work.) (Corresponding author : Yanyan Li.)*

Xin Li and Jinlong Lin are with Peking University, Beijing 100871, China (e-mail: lixin97@pku.edu.cn; linjl@ss.pku.edu.cn).

Yanyan Li and Evin Pınar Örnek are with the Department of Informatic, Technical University of Munich, 80333 Mnchen, Germany (e-mail: yanyan.li@tum.de; evin.oernek@tum.de).

Federico Tombari is with the Department of Informatic, Technical University of Munich, 80333 Mnchen, Germany, and also with Google Inc., Mountain View, CA 94043 USA (e-mail: tombari@in.tum.de).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3027230

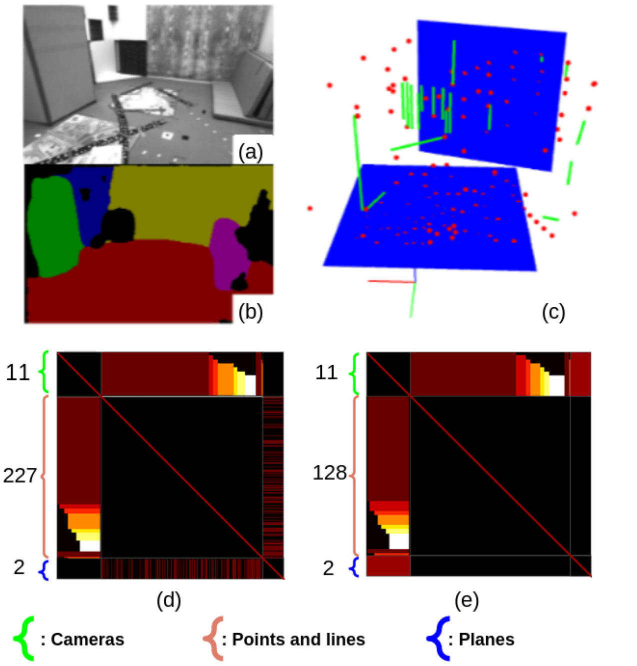


Fig. 1. System results: (a) input RGB frame; (b) plane instance segmentation; (c) reconstruction for points, lines and infinite planes; (d) and (e) Hessian matrices that show the spatial correlation of camera and 3D landmarks within the traditional [10], [11] and proposed parametrizations, respectively. Black areas represent zeros, non-zeros otherwise. (e) is sparser than (d). Number of camera parameters (green), points and lines features (orange) and plane parameters (blue) are shown.

method against the state-of-the-art plane-based VIO models. Our method shows improvement in accuracy on both pipelines while benefiting from lower runtime, demonstrating the effectiveness of co-planar constraints for SLAM. In summary, our paper proposes the following contributions:

- a novel two-stage plane detection strategy from RGB images, leveraging a neural network based plane segmentation and a robust outlier filtering
- a novel parametrization for co-planar points and lines that unifies the parameters, resulting in an efficient bundle adjustment optimization through the smaller and sparser Hessian matrix
- the deployment of these contributions within two different camera tracking frameworks, based respectively on VIO and stereo SLAM, both individually reporting state-of-the-art results.

II. Related Work

Feature-based SLAM is traditionally addressed by tracking keypoints along successive frames and then minimizing some error functions (typically based on re-projection errors) to estimate the camera poses [13]. For point/based only method, there are many successful proposals, such as PTAM [14], SVO [15] and ORB-SLAM [3]. However, using only point features has strong limitations within textureless environments as well as under illumination changes.

To deal with these problems, line-segment based methods were proposed [16], [17]. Moreover, planar regions and associated features have been leveraged by SLAM systems. In early works [1], planes in the scene were detected by RANSAC among estimated 3D points, which is time consuming and not stable. These plane-based mapping and tracking methods, however, are common within RGB-D sensors since it is easier to segment planes from depth maps. Salas-Moreno *et al.* [18] present a dense mapping approach by using bounded planes and surfels with RGB-D sensors. Point-Plane SLAM [19] computes orthogonal relationships between planes from depth maps, then uses constraints for pose estimation. CPA-SLAM [20] models the scene as a global plane model, which is helpful to remove drift by aligning current RGB-D frame with the plane model. By using IMU, VIO methods can deal with fast motion easily. MSCKF [21] and ROVIO [22] are popular filter-based methods, but the first one does not maintain estimates of 3D landmarks in the state vector. Different to those methods, an optimization strategy is used VINS-MONO [5] and Mesh-VIO [6] for pose estimation.

Instead of a set of features, planes are also used to construct co-planar regularities for points and lines. Instead of extracting planes from sparse point cloud, Mesh-VIO [6] builds 2D Delaunay triangulation based on 2D points first, and then project them into 3D from their correspondences. They find vertical and horizontal planes from the gravity vector given by the IMU, then merge the co-planar constraints in the optimization module. With the introduction of deep learning, methods were proposed to estimate planes from a single RGB image, hence opening up new possibilities for SLAM systems. PlaneReconstruction [23] and PlaneRCNN [24] are state-of-the-art plane instance segmentation methods for a single image. In addition to planes, they also estimate depth and normal maps from a single RGB image.

Inverse depth [10] and parallax angle [25] were proposed to represent point features in monocular systems. Inverse depth parametrization uses the inverse of the depth from its anchor camera, which works more accurately for distant features. Instead of using depth, the parallax angle is used in [25] which obtains good performance in both nearby and distant features. TextSLAM [26] suggests to extract text-based visual information and treats each detected text as a planar feature. In line parametrization methods, Plücker coordinate is a popular representation method for 3D line initialization and transformation. Each 3D line, however, has only 4 degrees of freedom (4DoFs), and the six parameters of Plücker coordinates lead to over-parameterization [27]. So, an orthogonal representation based on only four parameters is used in the optimization to solve this problem.

III. Proposed Method

In this section, we first explain our co-planar parametrization strategy, which includes plane instance detection and RANSAC based filtering steps. Then, we introduce the implementation details of our stereo and VIO versions that use the proposed parametrization in a sliding window optimization fashion.

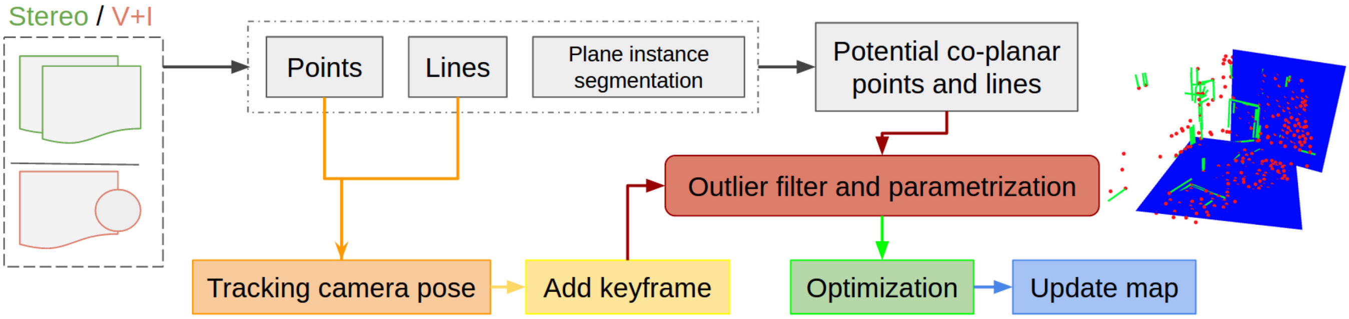

Stereo / V+I
Points
Lines
Plane instance segmentation
Potential co-planar points and lines
Outlier filter and parametrization
Tracking camera pose
Add keyframe
Optimization
Update map

Fig. 2. The pipeline of our plane-parametrized SLAM system. The overall pipeline follows the classical tracking and mapping approaches [3], along with the sliding-window based optimization. The pipeline can take as input either a stereo image pair or an image with IMU sensor data. 2D features and initial camera pose is estimated in a similar way as previous works [5], [12]. Then we detect planar regions via plane instance segmentation. After selecting the potential co-planar points and lines on the planar region, we remove the outliers with RANSAC. We present the remaining robust points and lines with the proposed parametrization, which can be directly integrated as an additional constraint in SLAM optimization.

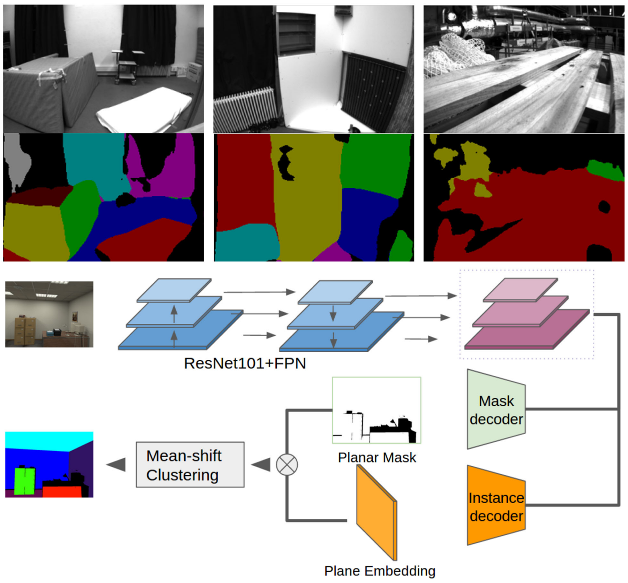


Fig. 3. Examples of plane instance segmentation on EuRoC dataset and architecture of the plane instance segmentation network.

A. Coplanarity-Based Parametrization

A plane is defined by equation $\mathbf{n}X_c^T + d = 0$, where $\mathbf{n} = (n_1, n_2, n_3) \in R^3$ is the normal of the plane, X_c is a 3D point in camera coordinates, and $d \in R$ is the distance from the plane to the origin of the camera c. However, this representation has an over-parametrization problem, and it cannot be solved with the Gauss-Newton approach due to singularity issue [28]. So we optimize the normal $\mathbf{n}$ on the tangent space S^2 with another optimization method, which is similar to Mesh-VIO [6]. In this section, we first describe how the co-planar points and features are detected. Then, we explain our parametrization for points and lines, respectively.

a) Plane Instance Segmentation: In order to detect planar regions in the scene in real-time, we use a plane instance segmentation network, which is a simplified version of PlaneReconstruction [23]. This network has two branches: planar mask decoder and a plane embedding decoder. The first branch decodes a binary mask for planar regions. The second one decodes the feature maps to an embedding space where mean-shift clustering is used to group each pixel into planar instances, iteratively. We train this plane detection network on ScanNet dataset [29] for 30 epochs.

b) Co-planar Feature Extraction: Since the plane instance segments extracted by the neural network might be at times inaccurate, we refine them by extracting 2D point and line features from images. Selecting the extracted features that align with the detected plane segments will lead us to robust features. We use ORB features [30] and LSD segment detection [31] to extract sets of co-planar points $[S_1^x, \ldots S_m^x]$ and co-planar lines $[S_1^l, \ldots S_m^l]$, where each distinct set consists of co-planar features $S_n^x = [x_i \ldots x_j], n \in [1, m]$ and x_i is a 2D pixel. For a stereo input, we obtain 3D points and lines by triangulating left-right image pairs. Whereas for VIO, the visual input is monocular and we triangulate sequential frames. During SLAM optimization, when a frame is detected as a new keyframe, we associate the features of this new frame with previous keyframes (i.e. check if they match and if they do not match, initiate new 3D landmarks with these features). After associating the landmarks, we build the potential co-planar points and lines, as shown in Fig. 2.

Due to the presence of outliers in the potential co-planar sets, we employ the following refinement strategy. First, for the current frame, we preserve the features that have been successfully triangulated. Then, we classify them according to detected 2D plane instance segments. If the number of features detected in a plane instance region is greater than a certain threshold, it will be considered as a potential planar region in 3D. If it is smaller than the threshold, plane will not be considered. After that, we use a RANSAC filter to find co-planar constraints in the potential planar region. We take out points C_x and lines C_l in the potential planar region and feed them to the filter. Specifically, corresponding rules in Eq. 1 are selected to fit parameters Γ of the plane according to the type of z ($\forall z \in \mathcal{Z}, \mathcal{Z} = [C_x, C_l]$),

$$f(c, \Gamma) = \begin{cases} \delta_\perp(P_x, \Gamma), z \in C_x \\ \max(\delta_\perp(c_{ls}, \Gamma), \delta_\perp(c_{le}), \Gamma), z \in C_l \end{cases} \qquad (1)$$

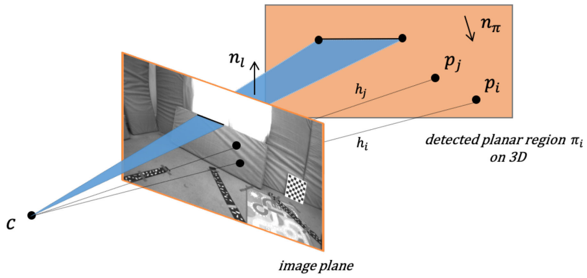


Fig. 4. Point and line features are shown on a detected planar region π_j with a normal n_π. h_i is the depth from camera frame origin to the 3D point p_i. n_l is the normal of a line on the plane n_π. Our parametrization rewrites the plane equation in terms of image pixel coordinates and combine line and point features.

where $\delta_\perp(\cdot,\cdot)$ denotes the perpendicular distance from a 3D point P_x to the plane in 3D, c_{ls} and c_{le} are the start and end points of the line respectively. Note that we only consider lines which have both endpoints lie on the same planar region. If the size of the largest consensus set exceeds a threshold θ_{cp} (80% in our experiments), we add the corresponding plane candidate to the system and establish point-plane and line-plane associations in the consensus set. We remove the outliers from the initial sets. When new 3D points and lines are generated in the system, we check if they belong to existing planes using the same metric defined above and store those correspondences. It is important to note that it would be also possible to detect planar regions by using only RANSAC (without the deep learning method). However, when there are unknown number of planes in a scene, RANSAC does not work optimal. It requires several iterations, where at each time a single planar region is detected and inlier points are removed. Yet, the false-detections accumulate over each time and results degenerate. We prevent this issue by detecting all planes through a neural network initially.

c) Parametrization of Points: After associating points and lines to co-planar regions as previously described, we obtain refined co-planar feature sets and parameters for each plane instance. As shown in Fig. 4, 3D points are the intersections of the detected plane and the camera-to-landmark rays. For each 3D point $P_x^c = (x^c, y^c, z^c)$ which lies on the plane π in camera frame c, we have the function $\mathbf{n}_\pi^T P_x^c + d_\pi = 0$. For example, the depth from origin of camera frame to the 3D point P_i is h_i. A normalized 3D point is presented as $(\hat{x}, \hat{y}, 1)$, where $(x^c, y^c, z^c) = (\hat{x}, \hat{y}, 1) \cdot h_i$. Also,

$$(\hat{x}, \hat{y}, 1)^T = K^{-1}(u, v, 1)^T \tag{2}$$

where K is the intrinsic matrix of camera c, and (u, v) is the 2D point corresponding to the landmark P_x^c. Then, the co-planar relationship for points can be represented as

$$h_i \cdot \mathbf{n}_\pi^T K^{-1}(u, v, 1)^T + d_\pi = 0, \tag{3}$$

where the relationship contains 2D pixel of the landmark and parameters of the plane. So in our parametrization, the point $\mathbf{p}^*$ lying on a planar region can be represented as

$$\mathbf{p}^* = [\mathbf{n}_\pi, d_\pi]. \tag{4}$$

d) Parametrization of Lines: For line features, the Plücker coordinates $\mathcal{L} = [\mathbf{n}_\text{l}^\top, \mathbf{d}^\top]^\top$ are used to initialize 3D lines, where $\mathbf{d} \in \mathbb{R}^3$ is the line's direction vector in camera frame c, and $\mathbf{n}_\text{l} \in \mathbb{R}^3$ is the normal vector of the plane determined by the line and the camera frame's origin point (Fig. 4). Furthermore, the line is the intersection of two known planes π_l and π_P, so the dual Plücker matrix $\mathbf{L}^*$ can be computed by:

$$\mathbf{L}^* = \begin{bmatrix} [\mathbf{d}]_\times & \mathbf{n}_\text{l} \\ -\mathbf{n}_\text{l}^\top & 0 \end{bmatrix} = \pi_l \pi_P^\top - \pi_P \pi_l^\top \in \mathbb{R}^{4\times4} \tag{5}$$

where $[\cdot]_\times$ is the skew-symmetric matrix of a three-dimensional vector, and $\pi = [\mathbf{n}, d]$ is a 4D vector. Then we can easily get Plücker coordinates $\mathcal{L} = [\mathbf{n}_\text{l}^\top, \mathbf{d}^\top]^\top$ from the dual Plücker matrix.

e) Resulting Hessian matrix: Compared with other proposed representations, which treat points and lines as independent features, our method uses one plane parameter to represent all co-planar features. Novel parametrization is then used in the bundle adjustment, which is solved by a second-order Newton optimization method, the Levenberg-Marquardt algorithm. This relies on taking the gradients of the residuals with respect to parameters (3D landmarks and camera poses) and solving the normal equations. Hence, when there are less number of parameters, Hessian matrix will be smaller. When there are less dependencies between the parameters, the sparse structure of Hessian can be employed more efficiently through Schur complement. The resulting Hessian matrix is illustrated in Fig. 1 and it's effects on efficiency are further shown in Experiments section, in Tab. III. The optimization equations are explained in next subsection. Further interested reader is referred to [32].

B. System Implementation

In this section, implementation details are introduced for both versions of our approach, i.e. the stereo SLAM and VIO, respectively.

a) Tracking: The goal of the tracking module is to extract 2D features and estimate the camera pose for each frame. In the stereo version, we estimate camera pose via point and line features, where stereo keypoints are defined by three coordinates $x_s = (u_L, v_L, u_R)$, here (u_L, v_L) are coordinates on the left image and u_R is the horizontal coordinate for the corresponding matches in the right image. Similar to points, lines between two images are matched by Line Band Descriptor (LBD) [33]. Furthermore, motion model is used to provide an initial pose that is refined by a frame-to-frame tracking strategy similar to ORB-SLAM [3]. Instead, for the VIO version, the initialization strategy of IMU is similar to VINS-Mono [5], which relis on a loose coupling strategy to align IMU pre-integration with the visual-only part. Different than visual-only (stereo) branch, the initial pose for optimization in VIO is obtained from IMU pre-integration [2], [5] so that the visual part can be regarded as a purely monocular version. Monocular keypoints are defined by two coordinates $x_m = (u_L, v_L)$ which are triangulated from multiple views.

In the system, we use different strategies for keyframe detection in stereo and VIO pipelines. For the former one, a new keyframe can be added only after at least 20 frames. Each keyframe tracks more than 40 points and 10% of keypoints should be new keypoints compared to the nearest keyframe. However, for the latter one, we consider the average parallax (with rotation compensation) of tracked features between two keyframes, which should be more than 10 degrees (similar to VINS-Mono [5]).

b) Mapping: When a keyframe is detected and inserted, we associate its 2D features to 3D corresponding landmarks in the sliding window (or local map) by 2D feature matching. For each non-associated 2D feature, we triangulate it with other keyframes in the VIO version, while for stereo, non-associated points and lines are usually triangulated by each stereo pair. Different from points, 3D lines are triangulated by two intersecting planes colored in blue in Fig. 4, which are observed in different views.

Based on the potential co-planar regions and the RANSAC filter, 3D landmarks are divided into two sets for optimization: planar features and non-planar features. Inverse depth algorithm is used to represent points; and Plücker coordinates and orthonormal representations are used to represent lines following He [2], which are then fed to window-based bundle adjustment for optimizing poses and landmarks.

c) Bundle Adjustment With Co-Planar Parametrization: In this part, we use re-projection error functions to optimize camera pose and landmark positions. Two different error functions are used for planar and non-planar features. Non-planar features are represented by traditional parametrization and optimized directly. However, co-planar features are refined by optimizing the parameters of the proposed parametrization. For point features, the re-projection error $\mathbf{r}^{\mathrm{p}}_{ik}$ strands for the the distance between the projected point of the jth map point and the observed point in the kth frame, which is noted as

$$\mathbf{r}^{\mathrm{p}}_{ik} = x_{ik} - \Pi(T_{kw}, P^{w}_{i}) \tag{6}$$

where $\Pi()$ re-projects the ith global 3D point P^{w}_{i} coordinates into the kth frame. For general points, P^{w}_{i} is represented as (x^{w}, y^{w}, z^{w}). Points lying on a plane are represented with Eq. (3).

For line features, the re-projection error $\mathbf{r}^{\mathrm{l}}_{jk}$ is defined as the distance between the re-projected line of the jth map line and two endpoints of its corresponding 2D line in the kth keyframe, which is given by,

$$\mathbf{r}^{\mathrm{l}}_{jk} = \begin{bmatrix} \frac{\mathbf{s}^{\top}\mathbf{n}_{\mathrm{l}}}{\sqrt{n_1^2+n_2^2}} & \frac{\mathbf{e}^{\top}\mathbf{n}_{\mathrm{l}}}{\sqrt{n_1^2+n_2^2}} \end{bmatrix}^{\top} \tag{7}$$

where $\mathbf{n}_{\mathrm{l}} = [n_1, n_2, n_3]^{\top}$ is the 2D line re-projected from the 3D line to the camera frame, $\mathbf{s} = [\hat{x}_s, \hat{y}_s, 1]^{\top}$ and $\mathbf{e} = [\hat{x}_e, \hat{y}_e, 1]^{\top}$ are two end-points of the observed line segment in the kth image plane. For general lines, $\mathbf{n}_{\mathrm{l}}$ can be represented as in an orthonormal way [2]. Lines lying on a plane are represented with the Eq. (5).

Given by the Eq. (6) and Eq. (7), We can therefore construct a unified target function which optimizes all terms simultaneously,

$$E = \sum_{k,i} \rho_p({\mathbf{r}^{\mathrm{p}}_{ik}}^{\top} \Lambda_{ik} \mathbf{r}^{\mathrm{p}}_{ik}) + \sum_{k,j} \rho_l(\mathbf{r}^{\top}_{jk^{\mathrm{l}}} \Lambda_{jk} \mathbf{r}^{\mathrm{l}}_{jk}) \tag{8}$$

here ρ_p and ρ_l present robust Cauchy cost functions. Respectively, Λ_{ik} and Λ_{jk} are the information matrices of points and lines, as calculated in [2], [5].

d) Tightly-Coupled Optimization for Inertial Constraints: For the VIO case, we fuse the data coming from the visual and inertial sensors via non-linear optimization in a tightly coupled form. Different from the stereo case, visual features are transferred to the IMU body coordinate system via extrinsic parameters $[\mathbf{R}_{bc} \quad \mathbf{t}_{bc}]$ between camera and IMU. So the unified target function for the VIO branch can be shown as,

$$\begin{aligned} E = & \sum_{k,i} \rho_p({\mathbf{r}^{\mathrm{p_{ik}}}}^{\top} \Lambda_{ik} \mathbf{r}^{\mathrm{p}}_{ik}) + \sum_{kj} \rho_l(\mathbf{r}^{\top}_{jk^{\mathrm{l}}} \Lambda_{jk} \mathbf{r}^{\mathrm{l}}_{jk}) \\ & + \sum_{b} \rho_l({\mathbf{r}^{\mathrm{b}}}^{\top} \Lambda_b \mathbf{r}^{\mathrm{b}}) + E_m \end{aligned} \tag{9}$$

where $\mathbf{r}^{\mathrm{b}}$ is the IMU residual, and E_m is the prior residual from marginalization operator in the sliding window. For more details, readers are referred to [5].

IV. Experiments

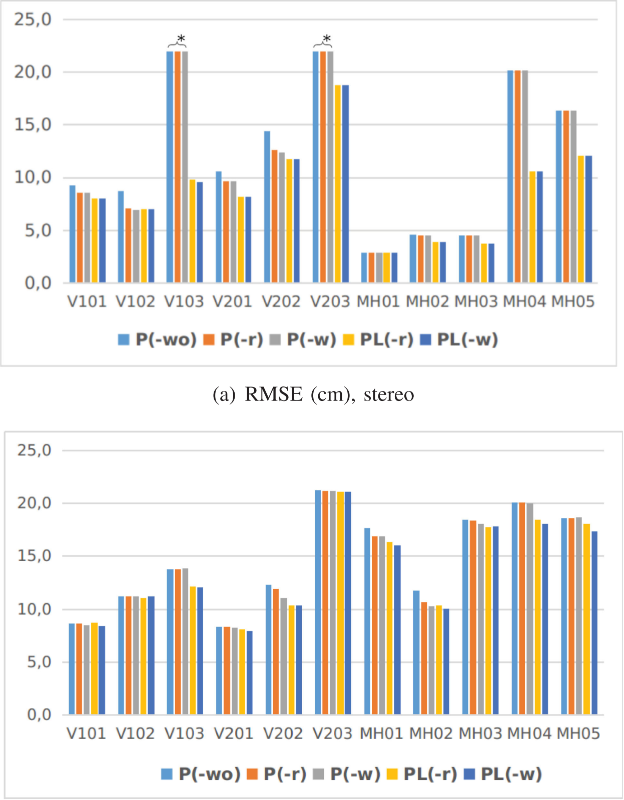
To evaluate the proposed method, we benchmark it against the state of the art on the EuRoC dataset [34]. In addition, we perform Monte-Carlo simulations to verify the robustness and efficiency of the novel parametrization. We evaluate both stereo and VIO pipelines with Absolute Trajectory Error (ATE) which measures absolute translational distances between the ground truth pose and the corresponding estimated pose. All the experiments run on an Intel Core i7-8550U @ 1.8 GHz and 16 GB RAM.

A. EuRoC Dataset

EuRoC is a popular public dataset for stereo SLAM and VIO systems, which collects stereo images and inertial data from an aerial vehicle in indoor environments [34]. There are two scenarios in this dataset: Vicon Room (V) and Machine Hall (MH), with eleven sequences in total. VH is an indoor environment and has several planar regions, whereas MH is the interior of an industrial facility where planar regions are unevenly distributed.

a) Ablation Studies: In order to evaluate the performance of the proposed parametrization in EuRoC, we fix the front-end and compare five formulations: $P(-wo)$, $P(-w)$, $PL(-w)$, $P(-r)$, and $PL(-r)$, where P denotes a point-based method, and PL denotes a point-line-based system. $(-wo)$ means the traditional parametrization (only inverse depth), and both $(-r)$ and $(-w)$ use co-planar constraints in the optimization module, but in different ways. $(-r)$ uses more equations between point-to-plane and line-to-plane, which are merged into optimization as in Mesh-VIO [6], [8]. Whereas $(-w)$ presents these residuals within the proposed co-planar parametrization.

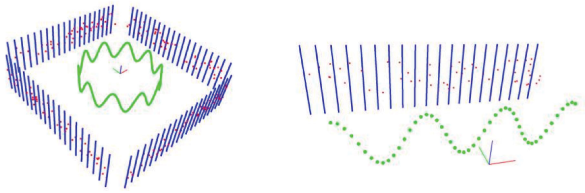
The results of the stereo and VIO versions on EuRoC dataset are presented in Fig. 5 and Fig. 5, respectively. In general, the proposed parametrization $PL(-w)$ results in lower RMSE compared to traditional parametrizations, $P(-wo)$ and $P(-w)$, in both cases, and especially in the MH sequences, where the



(a) RMSE (cm), stereo

(b) RMSE (cm), VIO

Fig. 5. Comparison in terms of ATE of different parametrization variants: $P(-wo)$, $P(-r)$, $P(-w)$, $PL(-r)$ and $PL(-w)$. The top part shows results for stereo, and the bottom one for VIO. The proposed parametrization $PL(-w)$ achieves the best results for all sequences where structural regularities are detected and enforced. * shows lost tracking on V103 and V203 sequences.



(a) Sequence a

(b) Sequence b

Fig. 6. Two simulation environments are illustrated, where points and lines are in red and blue, respectively. Camera follows green trajectories.

line features can provide more robust constraints with planar regions in the large industrial environment.

For stereo approaches, as shown in Fig. 5, line features make the system more robust especially in $V103$ and $V203$ sequences, where severe motion blur happened. In other Vicon sequences, $PL(-w)$ performs better than $PL(-r)$ because the proposed two-stage co-planar approach removes distances between those co-planar features and planes directly. In $MH01$, $MH02$ and $MH03$ which are textured sequences, all approaches obtain similar results. In Fig. 5, $P(-wo)$ and $P(-w)$ perform equally on $MH03$, $MH04$ and $MH05$ sequences, because there are not any structural regularities detected. When there are some planar regions detected, as in $V202$, $MH01$ and $MH02$, the proposed parametrization $P(-w)$ obtains better performance than traditional methods. If enough features can be obtained and few good co-planar sets, our system's performance will degenerate to that of traditional methods, as in sequences $V102$ and $V201$. The computation time of different operations in V101 is presented in Table II.

b) EuroC Evaluation: We compare our stereo branch against the stereo version of ORB-SLAM2 [12] and FMD-SLAM [35]. It is important to note that, for fairness of comparison, the tested ORB-SLAM2 does not have loop closure. Furthermore, we compare our VIO version against the recently proposed MSCKF [21], ROVIO [22], VINS-MONO [5], and Mesh-VIO [6]. Results are given in Table I. These VIO algorithms use all a monocular camera, except Mesh-VIO that uses a stereo camera. Results of previous works are taken from Rosinol *et al.* [6].

The left part of Table I shows that the $PL(-w)$ approach is an accurate and robust method compared with state-of-the-art VIO methods on sequences. Compared with Mesh-VIO [6], which also uses planar information to build co-planar regularities in the optimization process, our method performs better on most sequences, where Mesh-VIO obtained more vertical planes from 3D mesh due to using gravity during plane detection. When horizontal and vertical planes are difficult to detect as in $V103$ and some of the MH sequences, Mesh-VIO tends to degenerate easily so that it cannot build co-planar constraints. In sequence $MH05$, we observe a 26% improvement compared to the second best performing algorithm (Mesh-VIO), and in sequence $V103$, a 15% improvement and 35% improvement compared to VINS-MONO and Mesh-VIO, respectively. It can be seen that the optimization methods of VINS-MONO, Mesh-VIO and $PL(-w)$ are more robust than the filter-based MSCKF. Meanwhile, our method is more robust for indoor environments that have lots of co-planar regularities.

The stereo SLAM comparison is shown on the right side of Table I. Stereo ORB-SLAM2 obtains comparable results to ours on all sequences except V203 and MH04. In those textured sequences, this method tracks the features in a stable and accurate way. Instead, V203 is a difficult sequence because of the fast motion and the strong illumination changes, and tracking fails for both ORB-SLAM2 and FMD-SLAM. Benefiting from using point and line features, our method is instead more robust and can deal also with this sequence. The average RMSE values, for fairness computed without taking sequence V203 into account, show that our method obtains 25.7% and 38.7% improvements compared to ORB-SLAM2 and FMD-SLAM, respectively.

B. Simulation Dataset

We create two simulation sequences with ideal co-planar environments to evaluate the efficiency with respect to performance under different parametric formulations. As shown in Fig. 6(a), the first sequence has 100 lines and 200 points generated in 4 directions, which are observed by virtual cameras that follow a sinusoidal trajectory with 150 simulated poses. The second sequence consists of 20 lines and 50 points observed by 50 camera poses as shown in Fig. 6(b).

TABLE I
COMPARISON IN TERMS OF RMSE (CM) OF THE PROPOSED $PL(-w)$ PIPELINE AGAINST THE STATE OF THE ART ON THE EUROC DATASET. BEST RESULTS ARE BOLDED. × SHOWS LOST TRACKING. AVERAGED RESULTS WITH * DO NOT INCLUDE THE SEQUENCE V203

	VIO							**Stereo**				
	MSCKF [21]	ROVIO [22]	VINS MONO [5]	Mesh VIO [6]	PL (-wo)	PL (-r)	PL (-w)	ORB SLAM2 [12]	FMD SLAM [35]	PL (-wo)	PL (-r)	PL (-w)
V101	34	10	7	**6**	8.4	8.5	8.4	9	9	8.4	**8.0**	**8.0**
V102	20	10	10	**7**	11.0	10.9	11.0	8	20	7.5	**7.0**	**7.0**
V103	67	14	13	17	**11.9**	**11.9**	**11.9**	20	53	10.6	9.8	**9.6**
V201	10	12	**8**	**8**	8.1	8.1	**8.0**	**7**	9	9.0	8.2	8.2
V202	16	14	**8**	10	12.0	10.5	10.5	10	**8**	12.4	11.8	11.8
V203	113	**14**	21	27	20.9	20.9	20.9	×	×	19.8	**18.8**	**18.8**
MH01	42	21	27	**14**	17.1	16.3	16.2	4	4	**2.9**	**2.9**	**2.9**
MH02	45	25	12	13	11.0	10.2	**10.0**	5	4	4.1	**3.9**	**3.9**
MH03	23	25	**13**	21	17.6	17.6	17.6	4	5	4.0	**3.7**	**3.7**
MH04	37	49	23	22	18.5	18.4	**18.2**	16	**9**	10.6	10.6	10.6
MH05	48	52	35	23	18.2	18.0	**17.7**	20	**9**	12.1	12.1	12.1
Average	41.3	22.3	16.0	15.2	14.1	13.8	**13.7**	10.6*	12.9*	8.1*	**7.8***	**7.8***

TABLE II
COMPUTATION TIME (MEAN, MS) OF DIFFERENT OPERATIONS IN THE V101 SEQUENCE OF EUROC. * MEANS THAT THE OPERATION IS USED FOR EACH FRAME, OTHERWISE IT IS PERFORMED ON KEYFRAMES ONLY. D&M NOTES DETECTION AND MATCHING. - MEANS THAT THE OPERATION IS NOT USED

Operation	P(-wo)	P(-r)	P(-w)	PL(-wo)	PL(-r)	PL(-w)
Point D&M*	4	4	4	4	4	4
Line D&M	-	-	-	96	96	96
Plane Seg.	-	29	29	-	29	29
Plane fitting	-	10	10	-	10	10
Optimization	43	44	40	36	46	42
Total time	56	60	55	57	58	54

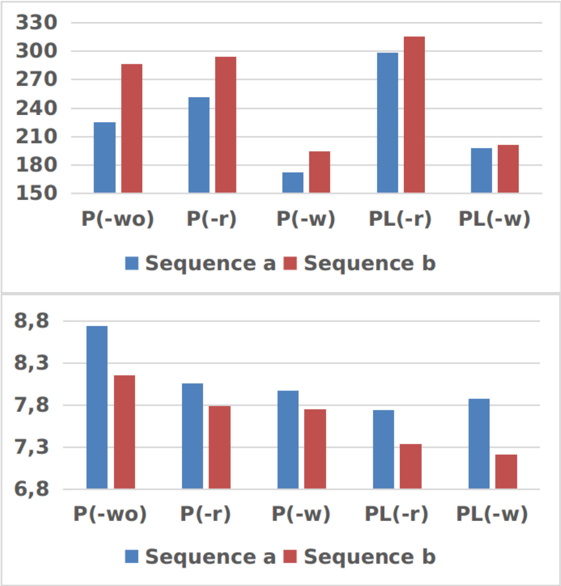


Fig. 7. Comparison of the optimization time (ms, top) and RMSE (cm, bottom) for pipelines $P(-wo)$, $P(-r)$, $P(-w)$, $PL(-r)$ and $PL(-w)$.

For line measurements, the virtual camera gets two endpoints from each measurements. Note that each measurement of a point, including endpoints of lines and point features, is corrupted by 1-pixel Gaussian random noise. In order to simplify the simulation, we simulate relative pose odometry measurements as pose estimation results from the tracking module, which have random noise as,

$$\bar{q}_m = \begin{bmatrix} \frac{1}{2}\mathbf{n}_\theta \\ 1 \end{bmatrix} \otimes \bar{q}, \quad \mathbf{p}_{Cm} = \mathbf{p}_C + \mathbf{n}_p \tag{10}$$

where $\mathbf{n}_\theta$ and $\mathbf{n}_p$ are the Gaussian white noises added to the relative pose, with $\sigma_\theta = 1$ deg and $\sigma_p = 10$ cm, respectively.

a) Performance: We pose the visual SLAM system as a non-linear least squares problem, solved via Gaussian-Newton. Maximum 10 iterations are allowed for each method in this simulation for a fair comparison. We run the simulation sequence 30 times and show median results for the accuracy of the estimated trajectory and optimization time. Fig. 7 shows similar performance across sequences, that is, $(-w)$ is more accurate and efficient than $(-r)$ and $(-wo)$. The second sequence (b) requires more optimization time and results in lower RMSE since more features are measured by each camera compared to the first. $P(-wo)$ requires less time than $P(-r)$ in two sequences because it does not use structural regularities and has small optimization computation as shown in Fig. 1(d). $P(-r)$ has a higher computational burden (Fig. 1(e)) and is more accurate than $P(-wo)$. While combining line features in the system, like $PL(-r)$, results are more accurate even if the method requires more time. Compared to $P(-r)$ and $PL(-r)$, our parametrizations for points and lines ($P(-w)$) are more efficient. In terms of optimization time, $P(-w)$ has a 31% improvement and $PL(-w)$ 33%, compared to $P(-r)$.

b) Number of Parameters: Furthermore, we analyze the reason of efficiency from the perspective of number of parameters that are to be updated. In traditional parametric methods (inverse depth for points and orthogonal approach for lines), each point, line and plane need 1 parameter, 4 parameters and 3 parameters, respectively. However, in our parametrization method, all points and lines in the plane are represented by only one plane parameter. Hence, there is only one parameter for each planar region during optimization. Table III shows the number of parameters that need to be updated in the global bundle adjustment on the second Monte Carlo sequence, where 20 lines

TABLE III
THE NUMBER OF LANDMARKS UPDATED IN OPTIMIZATION MODULE OF SEQUENCE 2

	P(-wo)	P(-r)	P(-w)	PL(-wo)	PL(-r)	PL(-w)
items	100	101	**51**	120	121	**51**
parameters	350	353	**303**	430	433	**303**

and 50 points are observed by 50 cameras. $P(-w)$ uses points only, so it has to update 100 items at each iteration. Similar to $P(-wo)$, we have to update 101 items and 121 items in $P(-r)$ and $PL(-r)$. Note that those two need to update one plane item because they use of co-planar constraints of point-to-plane and line-to-plane. In the proposed solutions, only 51 items (50 cameras and 1 plane) are updated in $P(-w)$ and $PL(-w)$ because they use the plane to represent co-planar points and lines.

V. Conclusion

We presented an efficient and robust co-planar parametrization method for points and lines by leveraging geometric and learning approaches together, which increases sparsity and reduces the size of Hessian matrix in each optimization module. Then, we illustrated how our co-planar parametrization can be implemented in stereo-SLAM and VIO pipelines. Our experiments show that our approach improves the efficiency and accuracy of both stereo and VIO optimization in indoor environments. As for future work, we plan to reconstruct dense maps from monocular data and merge together semantic segmentation and depth prediction to improve tracking and mapping simultaneously.

Acknowledgment

The authors would like to thank Dr. Yijia He for the helpful discussion.

References

[1] Y. Lu and D. Song, “Visual navigation using heterogeneous landmarks and unsupervised geometric constraints,” *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 736–749, Jun. 2015.

[2] Y. He, J. Zhao, Y. Guo, W. He, and K. Yuan, “Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features,” *Sensors*, vol. 18, no. 4, p. 1159, 2018.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: A versatile and accurate monocular slam system,” *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[4] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[5] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[6] A. Rosinol, T. Sattler, M. Pollefeys, and L. Carlone, “Incremental visual-inertial 3d mesh generation with structural regularities,” in *Proc. Conf. Robot. Autom.*, 2019, pp. 8220–8226.

[7] D. Zou, Y. Wu, L. Pei, H. Ling, and W. Yu, “Structvio: Visual-inertial odometry with structural regularity of man-made environments,” *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 999–1013, Aug. 2019.

[8] X. Li, Y. He, J. Lin, and X. Liu, “Leveraging planar regularities for point line visual-inertial odometry,” 2020, *arXiv:2004.11969*.

[9] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, “Kimera: An Open-Source Library for Real-Time Metric-Semantic Localization and Mapping,” in *Proc. Conf. Robot. Autom.*, 2020, pp. 1689–1696.

[10] J. Civera, A. J. Davison, and J. M. Montiel, “Inverse depth parametrization for monocular slam,” *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 932–945, Oct. 2008.

[11] A. Bartoli and P. Sturm, “The 3d line motion matrix and alignment of line reconstructions,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2001, vol. 1 pp. I–I.

[12] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[13] C. Cadena *et al.*, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[14] G. Klein and D. Murray, “Parallel tracking and mapping for small ar workspaces,” in *Proc. 6th IEEE Assoc. Comput. Machinery Int. Symp. Mixed Augmented Reality*, 2007, pp. 225–234.

[15] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “Svo: Semidirect visual odometry for monocular and multicamera systems,” *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.

[16] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, “Building a 3-d line-based map using stereo slam,” *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1364–1377, Dec. 2015.

[17] R. Gomez-Ojeda, F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, “Pl-slam: A stereo slam system through the combination of points and line segments,” *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 734–746, Jun. 2019.

[18] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, “Dense planar slam,” in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2014, pp. 157–164.

[19] X. Zhang, W. Wang, X. Qi, Z. Liao, and R. Wei, “Point-plane slam using supposed planes for indoor environments,” *Sensors*, vol. 19, no. 17, 2019, Art. no. 3795.

[20] L. Ma, C. Kerl, J. Stückler, and D. Cremers, “Cpa-slam: Consistent plane-model alignment for direct rgb-d slam,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1285–1291.

[21] A. I. Mourikis and S. I. Roumeliotis, “A multi-state constraint kalman filter for vision-aided inertial navigation,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 3565–3572.

[22] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, “Robust visual inertial odometry using a direct ekf-based approach,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 298–304.

[23] Z. Yu, J. Zheng, D. Lian, Z. Zhou, and S. Gao, “Single-image piece-wise planar 3D reconstruction via associative embedding,” in *Proc. Comput. Vis. Pattern Recognit.*, 2019, pp. 1029–1037.

[24] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz, “Planercnn: 3D plane detection and reconstruction from a single image,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, Jun. pp. 4445–4454.

[25] L. Zhao, S. Huang, L. Yan, and G. Dissanayake, “Parallax angle parametrization for monocular slam,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3117–3124.

[26] B. Li, D. Zou, D. Sartori, L. Pei, and W. Yu, “Textslam: Visual slam with planar text features,” in *Proc. Conf. Robot. Autom.*, 2020, pp. 2102–2108.

[27] A. Bartoli and P. Sturm, “Structure-from-motion using lines: Representation, triangulation, and bundle adjustment,” *Comput. Vis. Image Understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[28] M. Kaess, “Simultaneous localization and mapping with infinite planes,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 4605–4611.

[29] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “Scannet: Richly-annotated 3D reconstructions of indoor scenes,” in *Proc. IEEE Comput. Vis. Pattern Recognit*, 2017, pp. 2432–2443.

[30] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, “Orb: An efficient alternative to sift or surf,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[31] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, “Lsd: A line segment detector,” *Image Process. Line*, vol. 2, pp. 35–55, 2012.

[32] M. I. A. Lourakis and A. A. Argyros, “Sba: A software package for generic sparse bundle adjustment,” *ACM Trans. Math. Softw.*, vol. 36, no. 1, pp. 1–30, Mar. 2009.

[33] L. Zhang and R. Koch, “An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency,” *J. Vis. Commun. Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[34] M. Burri *et al.*, “The euroc micro aerial vehicle datasets,” *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016.

[35] F. Tang, H. Li, and Y. Wu, “Fmd stereo slam: Fusing mvg and direct formulation towards accurate and fast stereo slam,” in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 133–139.

7 E-Graph: Minimal Solution for Rigid Rotation with Extensibility Graphs

Camera pose estimation is a fundamental ability in intelligent agents, including robots, autonomous driving cars, and virtual/augmented reality devices, which provide tracking and re-localization services when those agents interact with unknown environments. Features, initial camera poses, and a front-end module generate measurement associations. While a back-end part acting as a critical formulation is responsible for refining and smoothing initials to accurate poses and landmarks by defining the problem as factor graphs. The typical visual factor graphs are constructed via covisibility relationships where each factor represents that an image frame measures a landmark. Benefiting from those relationships, relative transformations between different camera views can be estimated. Those vertices tend to be clustered into several local groups. When landmarks are distributed evenly in environments, those local clusters can be connected. Those co-visibility graphs achieve robust performance. However, extracting enough and even landmarks to maintain the connections between those local groups in low-textured scenes takes a lot of work, which brings problems in tracking and optimization modules.

To solve this problem, this paper proposes a new pose estimation approach based on a new type of pose graph structure. During the tracking process, points, lines, and planes will be extracted from each RGB-D frame. Instead of using those features to estimate 6-DoF camera poses, we first analyze vanishing directions and plane normals from line and plane features. When unparalleled directions are detected, a local coordinate $\mathbf{V}_i$ will be built. Then, we will check the association relationships between $\mathbf{V}_i$ and the set of other local coordinates $\mathbf{V}_m, m \in (0, \dots, n)$, where n is the number of local coordinates. When we detect matched coordinates $\mathbf{V}_i$ and $\mathbf{V}_j$, the orientation $\mathbf{R}_i$ of the current frame can be computed based on the former rotation $\mathbf{R}_j$ directly, but not affected by cumulative errors generated in the tracking process from frame F_{j+1} to F_i.

Contributions. Yan Li implemented and proposed the prototype idea of using vanishing directions of those 3D lines to build a new type of landmarks. Federico Tombari clarified some details about how to design experiments to evaluate the idea. All the experiments are carried out by Yan Li.

E-Graph: Minimal Solution for Rigid Rotation with Extensibility Graphs

Yanyan Li[1] and Federico Tombari[1,2]

[1] Technical University of Munich
[2] Google

E-Graph: Minimal Solution for Rigid Rotation with Extensibility Graphs

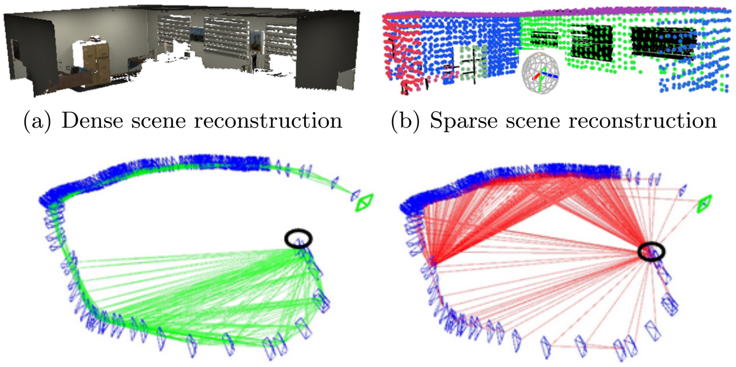
Yanyan Li[1,2][0000−0001−7292−9175] and Federico Tombari[1,3][0000−0001−5598−5212]

[1] Technical University of Munich, Munich, Germany
[2] Meta-Bounds Tech, Shenzhen, China
[3] Google, Zurich, Switzerland
yanyan.li@tum.de, tombari@in.tum.de

Abstract. Minimal solutions for relative rotation and translation estimation tasks have been explored in different scenarios, typically relying on the so-called co-visibility graphs. However, how to build direct rotation relationships between two frames without overlap is still an open topic, which, if solved, could greatly improve the accuracy of visual odometry. In this paper, a new minimal solution is proposed to solve relative rotation estimation between two images without overlapping areas by exploiting a new graph structure, which we call Extensibility Graph (E-Graph). Differently from a co-visibility graph, high-level landmarks, including vanishing directions and plane normals, are stored in our E-Graph, which are geometrically extensible. Based on E-Graph, the rotation estimation problem becomes simpler and more elegant, as it can deal with pure rotational motion and requires fewer assumptions, e.g. Manhattan/Atlanta World, planar/vertical motion. Finally, we embed our rotation estimation strategy into a complete camera tracking and mapping system which obtains 6-DoF camera poses and a dense 3D mesh model. Extensive experiments on public benchmarks demonstrate that the proposed method achieves state-of-the-art tracking performance.

1 Introduction

Camera pose estimation is a long-standing problem in computer vision as a key step in algorithms for visual odometry, Simultaneous Localization and Mapping (SLAM) and related applications in robotics, augmented reality, autonomous driving (to name a few). As part of the camera pose estimation problem, the minimal case [40] provides an estimate of whether the problem can be solved and how many elements are required to obtain a reliable estimate. According to the input data type and scenarios, different solutions [1, 28, 15, 9] were proposed, most of which became very popular in the computer vision and robotic community, such as the seven-point [1] and five-point [28] approaches. A typical limitation of traditional pose estimation solutions based on the minimal case [1, 28, 31, 9] is that both rotation and translation estimation rely on the co-visibility features between two frames, this having as a consequence that the length of an edge between two nodes is often relatively short. Therefore, tracking errors tend



(a) Dense scene reconstruction (b) Sparse scene reconstruction

(c) Covisibility graph (d) Extensibility graph

Fig. 1. Dense (a) and sparse (b) scene reconstruction of the office-room scene from the ICL dataset [10] obtained by the proposed method. (c) and (d): keyframes (in blue) and connected frames are linked with green and red lines, respectively, to build up the proposed covisibility and extensibility graphs. The black ellipses denote the start points of the camera trajectory.

to accumulate easily based on a frame-to-frame or frame-to-keyframe strategy. To solve this issue, more advanced tracking systems [26, 3] with optimization solutions, including local and global bundle adjustment approaches, were exploited to refine poses from minimal solutions. Loop Closure is a common algorithm used in feature-based [24] and direct [7] methods to remove drift. However, it also requires the camera to revisit the same place, which is a limiting assumption in many scenarios.

Compared with point features, lines and planes require more computation to be extracted and described. Early multi-feature SLAM systems [8] use them to increase the number of features to combat low-textured scenes. After that, coplanar, parallel and perpendicular relationships were explored [39, 18, 20] to add more constraints in the optimization module, still following a similar tracking strategy as ORBSLAM [25] or DSO [36] for the initial pose estimation.

Different to the tightly coupled estimation strategy, some works [43] proposed to decouple the 6-DoF pose estimation into rotation and translation estimation aiming to achieve a more accurate rotation estimation, based on the idea that pose drift is mainly caused by the rotation component [14]. At the same time, based on an estimated rotation matrix [31], only two points are required to compute the translation motion, leading to more robustness in low-textured regions.

The Manhattan World (MW) [43] and Atlanta World (AW) [13] assumptions introduce stronger constraints since they require a single orthogonal scene, or a scene with a unified vertical direction. Unlike loop closure that removes drift by detecting trajectory loops, the assumption of MW and AW is introduced for indoor tracking scenarios [19, 14] to improve the accuracy of camera pose estimation, since most indoor artificial environments follow this assumption. MW and AW improve accuracy when the main structure of the scene has orthogonal elements However, since this assumption requires the observation of vertical/orthogonal environmental features (such as straight lines or planes), the SLAM system using this method is also limited in the types of scenarios it can be successfully applied to.

In this paper we propose a rigid rotation estimation approach based on a novel graph structure, which we dub Extensibility Graph (E-Graph), for landmark association in RGB-D data. Our approach is designed to reduce drift and improve the overall trajectory accuracy in spite of loop closure or MW/AW assumptions. Benefiting of E-Graph, the drift-free rotation estimation problem is simplified to the alignment problem of rotating coordinate systems. Importantly, our rotation step does not need overlaps between two frames by making use of vanishing directions of lines and plane normals in the scene, hence can relate a higher number of keyframes with respect to standard co-visibility graphs, with benefits in terms of accuracy and robustness in presence of pure rotational motions.

In addition, we develop a complete tracking and dense mapping system base on the proposed E-Graph and rotation estimation strategies, which we demonstrate to outperform state-of-the-art SLAM approaches [20, 38, 26, 3]. To summarize, the main contributions of this paper are as follows: i) a new perspective for reducing drift is proposed based on our novel graph structure, E-Graph, which connects keyframes across long distances; ii) a novel drift-free rotation alignment solution between two frames without overlapping areas based on E-Graph; iii) a complete SLAM system based on the two previous contributions to improve robustness and accuracy in pose estimation and mapping. The proposed approach is evaluated on common benchmarks such as ICL [10] and TUM-RGBD [33], demonstrating an improved performance compared to the state of the art.

2 Related work

By making the assumption of planar motion [9], two-view relative pose estimation is implemented based on a single affine correspondence. Point features are common geometric features used in VO and SLAM [3] systems. To remove the drift from point-based front ends, different types of back ends are explored in tracking methods. Loop closing is an important module to remove drift, which happens when the system recognizes that a place [6, 23] has been visited before. After closing the loop, associated keyframes in the covisibility graph will be adjusted. Benefiting of loop closure and optimization modules, ORB-SLAM series [26, 3] organize the keyframes efficiently, which provides robust support for tracking tasks. Different from sparse point features used in ORB-SLAM, BAD-SLAM [32] implements a direct bundle adjustment formulation supported by GPU processing.

However, in indoor environments, to cover texture-less regions that have few point features, more geometric features are merged into the front end of systems. At the early stage, methods build re-projection error functions for lines and planes. CPA-SLAM [22] makes use of photometric and plane re-projection terms to estimate the camera pose. Based on estimated camera poses, detected planes are merged together with a global plane model. Similar to our method, CPA-SLAM and KDP-SLAM [11] can build constraints between non-overlapping frames. However those constraints are used to build heavy optimization targets instead of improving the efficiency. Furthermore, the relationship between parallel lines (vanishing points) and perpendicular planes is explored in [17, 41]. Based on the regularities of those structural features, they obtain a more accurate performance. Instead of exploring the parallel/perpendicular relationships between lines/planes, [30, 18] make use of constraints between co-planar points and lines in the optimization module.

Those regularities aim to build constraints between local features, [14, 20] introduce global constraints by modeling the environment as a special shape, like MW and AW. The MW assumption is suitable for a cuboid scenario, which is supposed to be built by orthogonal elements. Based on this assumption, those methods estimate each frame's rotation between the frame and the Manhattan world directly, which is useful to avoid drift between frames in those scenes. L-SLAM [14] groups normal vectors of each pixel into an orthogonal coordinate by projecting them into a Gaussian Sphere [43] and tracks the coordinate axes to compute the relative rotation motion. Similar to the main idea of L-SLAM, [15] provides a RGB-D compass by using a single line and plane. Since the line lies on the plane, the underlying assumption of the system is the MW-based rotation estimation method. However, the limitation of this strategy is also very obvious, that it works only in Manhattan environments. Based on ORB-SLAM2 [26], Structure-SLAM [19, 20] merges the MW assumption with keyframe-based tracking, to improve the robustness of the system in non-MW indoor scenes, which refine decoupled camera pose by using a frame-to-model strategy. Compared with MW-based tracking methods, our approach is less sensitive to the structure of environments.

3 Minimal case in orientation estimation

Commonly, the 6-DoF Euclidean Transform $T \in SE(3)$ defines motions as a set of rotation $R \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$. Based on point correspondences, camera pose estimation can be defined as,

$$\mathbf{P}^{'} = R\mathbf{P} + \mathbf{t} \tag{1}$$

where $\mathbf{P}^{'}$ and $\mathbf{P}$ are 3D correspondences, and $[R, \mathbf{t}]$ defines the relative motion between two cameras. For monocular sensors, their image normalized representations are $\mathbf{X}_c^{'}$ and $\mathbf{X}_c$,

$$\mathbf{X}_c^{'} = \alpha(R\mathbf{X}_c + \gamma\mathbf{t}) \tag{2}$$

where α and γ are depth-related parameters. After multiplying (2) by $\mathbf{X}_c^{'T}[\mathbf{t}]_x$, we can obtain the classic essential matrix equation,

$$\mathbf{X}_c^{'T} E\mathbf{X}_c = 0 \tag{3}$$

where $E = [\mathbf{t}]_x R$ and $[\mathbf{t}]_x$ is the skew symmetric matrix formed by $\mathbf{t}$.

For RGB-D sensors, the task is simplified since the absolute depth information is directly provided by sensors. Equation (1) can be solved by using 3 non-collinear correspondences only [29], although the distance between two frames is supposed to be kept small to extract enough correspondences.

3.1 Minimal solution for rotation

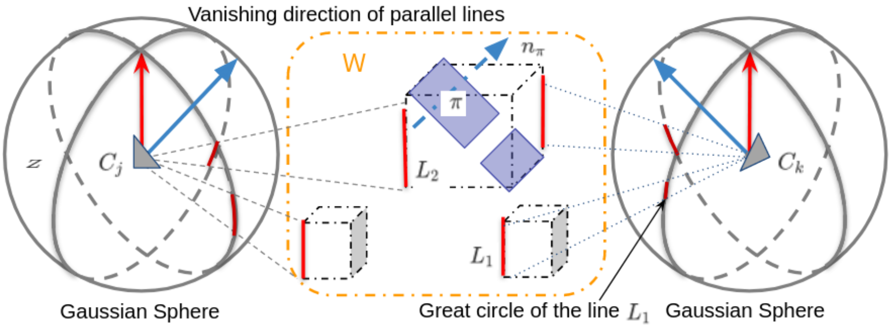

Vanishing direction of parallel lines
W
n_π
π
z
C_j
C_k
L_2
L_1
Gaussian Sphere
Great circle of the line L_1
Gaussian Sphere

Fig. 2. Minimal case of rotation estimation in EG

Different from traditional methods based on co-visibility graphs, the proposed method decouples rotation and translation estimation into two separate stages. Moreover, the rotation estimation task does not require feature correspondences. As shown in Figure 2, non-parallel direction vectors $\mathbf{v}_m, m \in [0, 1, \ldots, n]$ are detected in the camera coordinate C_j, where $\mathbf{v}_m^j = [_x v_m^j, _y v_m^j, _z v_m^j]^T$. In Euclidean 3D space, the size of a finite and linearly independent set of vectors is less then four. According to the Gram-Schmidt orthogonalization process, we can obtain an orthogonal set $S = [\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2]$,

$$\begin{aligned} \mathbf{u}_0 &= \mathbf{v}_0^j \\ \mathbf{u}_1 &= \mathbf{v}_1^j - proj_{[\mathbf{v}_0^j]}(\mathbf{v}_1^j) \\ \mathbf{u}_2 &= \mathbf{v}_2^j - proj_{[\mathbf{v}_0^j]}(\mathbf{v}_2^j) - proj_{[\mathbf{v}_1^j]}(\mathbf{v}_2^j) \end{aligned} \tag{4}$$

by using the projection operator $proj_{[\mathbf{u}]}(\mathbf{v}) = \frac{<\mathbf{u},\mathbf{v}>}{||\mathbf{u}||||\mathbf{v}||}\mathbf{v}$, where $< \mathbf{u}, \mathbf{v} >$ shows the inner product of the vectors $\mathbf{u}$ and $\mathbf{v}$. Furthermore, we obtain the normalized vectors $\mathbf{e}_0$, $\mathbf{e}_1$ and $\mathbf{e}_2$ via $\mathbf{e}_m = \frac{\mathbf{u}_m}{||\mathbf{u}_m||}$.

For the Euclidean space $\mathbb{R}^3$, the relevant orthonormal basis set based on the detected direction vectors is $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$. In the j^{th} camera coordinate, the orthonormal set is detected as $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$, while $(\mathbf{e}_0^*, \mathbf{e}_1^*, \mathbf{e}_2^*)$ in the k^{th} camera coordinate.

Therefore, from the perspective of the orthonormal set, those j^{th} and k^{th} coordinates are represented as $[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]^T$ and $[\mathbf{e}_0^*, \mathbf{e}_1^*, \mathbf{e}_2^*]^T$, respectively.

Given $\begin{bmatrix}\mathbf{e}_0^T\\ \mathbf{e}_1^T\\ \mathbf{e}_2^T\end{bmatrix}[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]$ is the identity matrix, the matrix $[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]$ is an orthogonal matrix and the columns of $[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]^T$ are orthonormal vectors as well, which can be used to build the orthonomal basis set of the j^{th} camera coordinate. Therefore, in $\mathbb{R}^3$ an arbitrary vector $\mathbf{x}$ can be represented by two orthonormal sets, $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)^T$ and $(\mathbf{e}_0^*, \mathbf{e}_1^*, \mathbf{e}_2^*)^T$, independently,

$$\begin{aligned}\mathbf{x} &= (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)^T(x_0, x_1, x_2)^T\\ &= (\mathbf{e}_0^*, \mathbf{e}_1^*, \mathbf{e}_2^*)^T(x_0^*, x_1^*, x_2^*)^T\end{aligned} \tag{5}$$

Finally, $(x_0, x_1, x_2)^T = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)(\mathbf{e}_0^*, \mathbf{e}_1^*, \mathbf{e}_2^*)^T(x_0^*, x_1^*, x_2^*)^T$ where the rotation motion $R_{c_jc_k}$ from camera k to camera j is $[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2][\mathbf{e}_0^*, \mathbf{e}_1^*, \mathbf{e}_2^*]^T$.

Two-Observation case. In the spatial case where two linearly independent direction vectors are detected, $\mathbf{u}_2$ can be achieved by the cross product process of $\mathbf{u}_0$ and $\mathbf{u}_1$. Obviously, the new set $[\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_0 \times \mathbf{u}_1]$ maintains the orthogonal property, which is the minimal solution for relative pose estimation problems.

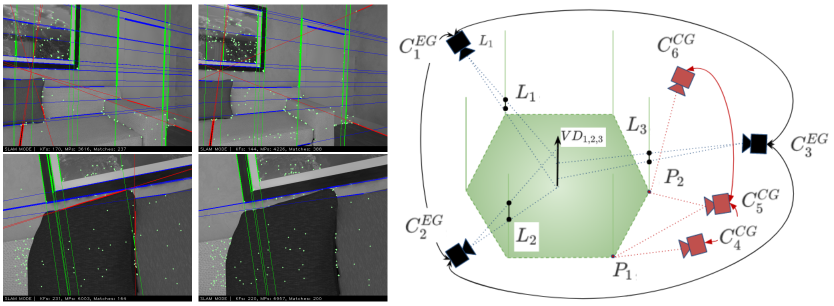
Orthogonal-Observation case. As discussed in Section 2, the MW assumption is enforced mostly by SLAM/VO methods designed to work indoor [15, 14, 19, 38], achieving particularly good results when the MW assumption holds. When the observation vectors $\mathbf{v}_m^j$ are orthogonal, the projection operation between different vectors is zero and the proposed method degenerates to a multi-MW case,

$$\begin{aligned}R_{c_jc_k} &= R_{c_jM_i}R_{c_kM_i}^T\\ &= [\frac{\mathbf{v}_0^j}{||\mathbf{v}_0^j||}, \frac{\mathbf{v}_1^j}{||\mathbf{v}_1^j||}, \frac{\mathbf{v}_2^j}{||\mathbf{v}_2^j||}][\frac{\mathbf{v}_0^k}{||\mathbf{v}_0^k||}, \frac{\mathbf{v}_1^k}{||\mathbf{v}_1^k||}, \frac{\mathbf{v}_2^k}{||\mathbf{v}_2^k||}]^T.\end{aligned} \tag{6}$$

For single-MW scenarios, a global orthogonal set can be obtained by every frame, therefore R_{c_jw}, from world to camera C_j, can be computed by $R_{c_jM}R_{c_0M}^T$, here R_{c_0w} is an identity matrix.

Compared with the visual compass [15] method making use of a combination of line and plane features from MW [14] to estimate camera rotation, our graph is more robust and flexible. Furthermore, compared to [31] that generates four rotation candidates after aligning two frames' vanishing points, our method not only leverages plane features, but also solves the ambiguity regarding the directions of the vanishing points [31].

After the relative rotation pose estimation step between two frames, in case of no overlap between them, we need to make use of their neighboring frames to compute translation vectors. Note that only two correspondences are required in translation estimation by making use of Equation 1, which is particularly suited to deal with scenes and environments characterized by different texture types compared to traditional approaches [26, 3].



(a) (b) (c)

Fig. 3. Vanishing point detection and Rotation connection examples. (a) Detection results of J-Linkage. (b) Refined results by our system. (c) E-Graph (black) and co-visibility graph (red).

4 Extensibility Graph (E-Graph)

As shown in Figure 3(c), the E-Graph method builds rotation connections (edges) between frames $[C_1^{EG}, C_2^{EG}, C_3^{EG}]$ that share global directions instead of any low-level correspondences (like points and lines). At the same time, no connection between C_4^{CG} and C_6^{CG} can be made since these frames have no co-visible features within the co-visibility graph. The proposed connection strategy will be detailed in the following subsections.

4.1 Landmarks from a RGB-D frame

Similar to the co-visibility graph, the proposed graph is also a topological representation of scenes. The difference is that the proposed graph is built based on the scene structure rather than on overlapping parts between frames. The distance between connected frames in a co-visibility graph tends to be small (see Figure 3) since two frames that are distant from each other rarely overlap, leading to the pose of the current frame being estimated based on the last frame or last keyframes only. The issue can be alleviated by using global bundle adjustment and loop closure modules, although they bring in intensive computation and trajectory constraints (e.g. need to re-visit a certain area).

In our graph $\mathscr{G} = [\mathscr{N}_c, \mathscr{N}_{lm}, \mathscr{E}]$ frames and landmarks are regarded as nodes $\mathscr{N}_c$ and $\mathscr{N}_{lm}$ respectively, while $\mathscr{E}$ represents the edges among connected frames. Note that landmarks are border-less planes and vanishing directions, e.g. $VD_{1,2,3}$, of lines detected in multiple views. In particular, an edge is established between two frames every time two or more structural elements are matched across them.

Features and landmarks. Vanishing directions are estimated from parallel lines detected by a joint 2D-3D process, where LSD [35] is used to extract 2D line features from RGB images. Meanwhile, AHP [5] is carried out to extract plane features from depth maps.

Firstly, as shown in Figure 3(a), we make use of the J-Linkage algorithm to classify detected 2D lines into different groups of parallel lines as described in [34]. However, there are still outliers left based on the 2D process. To solve this issue, we take advantage of depth maps to check the directions of lines in each group by using RANSAC to detect the best direction vector VD_n to represent the group S_n.

As for planar landmarks, we make use of the Hessian ($\boldsymbol{\pi} = (\mathbf{n}^{\pi}, d^{\pi})$) to represent a plane detected from the i^{th} frame, where $\mathbf{n}^{\pi}$ denotes the normal vector and d^{π} represents the distance between the camera center and this plane, which is transferred to world coordinates via the initial pose T_{wc_i}.

4.2 Data Association

After generating vanishing directions and planes, we now explain how to initialize and update them.

Initialization. Combined with the first keyframe Kf_0, detected planes and optimized vanishing directions are used to initialize the E-Graph. The camera pose T_0 of Kf_0 is set as the world coordinate for landmarks in the E-Graph. Planes $\boldsymbol{\pi}_i$ measured by Kf_0 are transferred to the graph directly as,

$$\mathscr{G}_0 = [\mathscr{N}_{c_0}, \mathscr{N}_{lm_0}, \mathscr{E}_0] \tag{7}$$

where $\mathscr{N}_{c_0}$ is Kf_0 and $\mathscr{E}_0$ has no edges yet. $\mathscr{N}_{lm_0}$ contains $[\boldsymbol{\pi}_i, VD_i, PD_j]$, where VD_i and PD_j refer to two different types of 3D lines detected in the RGB-D frame: the former refers to lines that are parallel to at least another 3D line, the latter to lines that are not parallel to any line. The first type of lines can generate vanishing directions VD_i in a single view, which are stored into the graph directly, similarly to planes. In addition, lines that do not have parallel lines detected in this RGB-D frame are marked as potential vanishing direction PD_j. In case parallel lines will be detected in successive frames, these lines will also be transferred to VD_j, otherwise, they are removed from the E-Graph.

Landmarks fusion. For each new input frame we need to extract vectors $\mathbf{n}^{\pi}$, VD and PD from the current frame. After rotating VD_i^c to the world coordinate frame as VD_i^w, if the direction between VD_i^c is parallel to $VD_k^w, k \in [0, \dots, m]$, where m is the number of vanishing directions saved in E-Graph, VD_i^c is then associated to the graph. To solve the unsure issues [31] of vanishing directions, we will unify the direction during the association process by using

$$\tilde{VD}_i^c = \begin{cases} VD_i^c \;\; (|norm(VD_i^w \cdot VD_k^w) - 1| < th_{vd}) \\ -VD_i^c \;\; (|norm(VD_i^w \cdot VD_k^w) + 1| < th_{vd}) \end{cases} \tag{8}$$

where $norm(\cdot)$ shows a dot product between two normalized vectors and $|\cdot|$ is the absolute difference. th_{vd} is a threshold to check the angle distance between two vectors. To include additional graph connections, we also try to associate

PD^c_j with VD^w_k and PD^w_k. If new pairs can be made at this stage, the associated PD vectors are transferred to the vanishing directions and fused into the graph.

Since the vanishing direction is independent from translation motion, VD^w_i, the vanishing direction in the world coordinate can be obtained as

$$VD^w_i = R_{wc} VD^c_i \tag{9}$$

where R_{wc} is the rotation motion from the camera coordinate frame to the world coordinate frame.

In certain indoor scenes, e.g. a corridor or hallway, when a robot moves along the wall, an extended planar region is detected across multiple views, with most of these views encompassing no overlap. To address this issue, we extract the normal vector $[n^c_x, n^c_y, n^c_z]$ of the plane in the camera coordinate, which can be fused into the world coordinate in the same way as the vanishing directions.

Edge connection. In E-Graph, all landmarks come from keyframes that follow the decision mechanisms of a feature-based SLAM system [24, 20], which we summarize in the following. A new keyframe is detected if it satisfies one of the following two conditions: 1) 20 frames have passed from the last keyframe; 2) the current frame tracks less than 85% points and lines correspondences with the last keyframe. Furthermore, when the current frame detects a new plane or a new vanishing direction, the frame is considered as a new keyframe. In addition, new landmarks connected to this keyframe are also merged into the graph at this stage.

By sequentially processing keyframes, if more than two pairs of matched landmarks are observed between two keyframes, an edge will be created to connect the respective two graph nodes. As shown in Figure 2, C_j and C_k detect the plane $\boldsymbol{\pi}$ and the same vanishing point generated by L_1 and L_2. Notably, even if these two frames do not have any correspondence, they can still be connected in our E-Graph.

5 Experiments

In this section, the proposed system is evaluated on different indoor benchmarks: ICL-NUIM [10] and TUM RGB-D [33]. ICL-NUIM [10] contains eight synthetic sequences recorded in two scenarios (living room and office room). TUM RGB-D [33] is recorded in real scenarios and includes varied sequences in terms of texture, scene size, presence of moving objects, etc.

Rotation estimation. The proposed rotation algorithm is compared with other state-of-the-art orientation estimation approaches. Compass [15] makes use of a single line and plane. OPRE [43] and GOME [12] estimate the distribution of surface normal vectors based on depth maps. OLRE [2] and ROVE [16] take advantage of vanishing directions for rotation estimation. Importantly, Compass, GOME, OLRE, OPRE, and P-SLAM [20] are all based on the MW assumption, while our method, ORB-SLAM2 [25] and ROVE are designed for general scenes.

Translation estimation. Since the rotation of the current frame is estimated from a keyframe that may not be overlapping with the current frame, we follow the 3D translation estimation model [26, 20] to estimate the translation $\mathbf{t}$ based on the predicted rotation. In this module, re-projection errors from point-line-plane feature correspondences are used to build a target optimization function, $\mathbf{t} = argmin(\sum_{j=0}^{n} e_{i,j}^{\pi} \Lambda^{\pi} e_{i,j}^{\pi} + e_{i,j}^{L} \Lambda^{L} e_{i,j}^{L} + e_{i,j}^{P} \Lambda^{L} e_{i,j}^{P})$, where e^{π}, e^{L} and e^{P} are re-projection error functions for planes, lines and points, respectively. The target function is optimized by using the Levenberg-Marquardt method. The translation is compared with the following state-of-the-art methods. ORB-SLAM2 [26] and ORB-SLAM3 [3] are popular keypoint-based SLAM systems. In our experiments, for fairness of comparison the loop closure is removed to reduce the effect of the back-ends. SP-SLAM [39] additionally uses points and planes in the tracking and optimization modules based on ORB-SLAM2. P-SLAM [19] assumes the indoor environments as MW, and includes a refinement module to make the tracking process more robust. Moreover, we also compare our system with GPU-based methods, including BadSLAM [32] and BundleFusion [4].

Dense mapping. In this paper, a mapping module is implemented to reconstruct unknown environments in sparse and dense types. The sparse map is reconstructed by the point-line-plane features extracted from keyframes, which supports a frame-to-map pose refinement step. Since sparse maps cannot provide enough information for robots, our system also generates a dense mesh map incrementally based on CPU. When a new keyframe is generated from the tracking thread, we make use of the estimated camera pose and the RGB-D pair to build a dense TSDF model based on [42, 27]. After that, the marching cubes method [21] is exploited to extract the surface from voxels.

Metrics. The metrics used in our experiments include absolute trajectory error (ATE), absolute rotation error (ARE), and relative pose error (RPE) that shows the difference in relative motion between two pairs of poses to evaluate the tracking process. Our results are reported in Table 2 and obtained on an Intel Core @i7-8700 CPU @3.20GHz and without any use of GPU resources.

5.1 ICL NUIM dataset

As shown in Table 1, the proposed method outperforms other MW-based and feature-based methods in terms of average rotation error. In *office room* sequences, OPRE and P-SLAM also perform well since orthogonal planar features can be found in the environment. However, in *office room 0*, parts of the camera movement only contain a single plane and some lines, leading to performance degradation, while our method achieves robust orientation tracking by taking advantage of a set of non-parallel planes and lines.

Furthermore, we compare the translation results against two feature-based methods as shown in Table 2. The first four sequences are related to a living room scenario, while the remaining sequences are from an office scenario. All methods obtain good results in *living room 0* where the camera moves back and

Table 1. Comparison of the average value of the absolute rotation error (degrees) on ICL-NUIM and TUM RGB-D structural benchmarks. The best result for each sequence is bolded. × shows that the method fails to track the orientation.

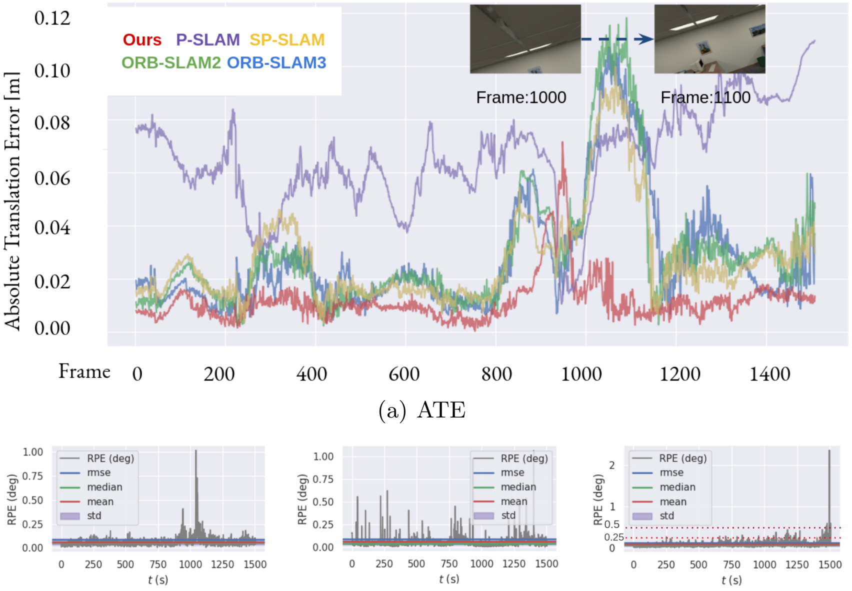
Sequence	Ours	Compass [15]	OPRE [43]	GOME [12]	ROVE [16]	OLRE [2]	ORB2 [26]	P-SLAM [20]
office room 0	**0.11**	0.37	0.18	5.12	29.11	6.71	0.40	0.57
office room 1	**0.22**	0.37	0.32	×	34.98	×	2.30	**0.22**
office room 2	0.39	0.38	0.33	6.67	60.54	10.91	0.51	**0.29**
office room 3	0.24	0.38	**0.21**	5.57	10.67	3.41	0.36	**0.21**
living room 0	0.44	**0.31**	×	×	×	×	0.97	0.36
living room 1	**0.24**	0.38	0.97	8.56	26.74	3.72	0.22	0.26
living room 2	0.36	**0.34**	0.49	8.15	39.71	4.21	0.83	0.44
living room 3	0.36	0.35	1.34	×	×	×	0.42	**0.27**
f3_stru_notex	4.46	1.96	3.01	4.07	×	11.22	×	4.71
f3_stru_tex	**0.60**	2.92	3.81	4.71	13.73	8.21	0.63	2.83
f3_l_cabinet	**1.45**	2.04	36.34	3.74	28.41	38.12	2.79	2.55
f3_cabinet	2.47	2.48	2.42	2.59	×	×	5.45	**1.18**

forth between the two parallel walls. P-SLAM detects a good MW model, and ORB-SLAM3 also observes enough features, benefiting from paintings hanging on the wall and small furniture. Compared with the living room, the office room has many low-textured regions. The performance of feature-based algorithms is not as good as in the living room scenes, especially in *office room 1* and *office room 3*.

Table 2. Comparison in terms of translation RMSE (m) for ICL-NUIM and TUM RGB-D sequences. × means that the system fails in the tracking process.

Sequence	Ours	P-SLAM[20]	ORB-SLAM3[3]
office room 0	**0.014**	0.068	0.035
office room 1	**0.013**	0.020	0.091
office room 2	0.020	0.011	**0.010**
office room 3	**0.011**	0.012	0.096
living room 0	0.008	**0.006**	**0.006**
living room 1	**0.006**	0.015	0.206
living room 2	**0.017**	0.020	0.018
living room 3	0.021	**0.012**	0.019
f1_360	0.114	×	**0.108**
f1_room	**0.095**	×	×
f2_rpy	**0.002**	0.154	0.003
f2_xyz	**0.003**	0.009	0.004
f3_l_o_house	0.012	0.122	**0.009**
f3_stru_notex	**0.017**	0.025	×
f3_l_cabinet	**0.058**	0.071	0.072

To analyze the relationship between rotation and translation results of different methods, absolute translation and rotation errors on the *office room 0* sequence are presented in Figure 4. When the camera moves to the ceiling, the number of detected features decreases, then an interesting phenomenon is witnessed (see also Figure 4(a)): the tracking error of feature-based systems quickly and drastically increases, then gradually fades as the number of features increases. At the same time, our method and P-SLAM exhibit a more robust performance when they face this challenge. An important difference is that, while P-SLAM underperforms due to the non-rigid MW scene, our method's performance is accurate thanks to the use of the E-Graph, which demonstrates to be more flexible than MW-based paradigms.



(a) ATE

(b) Ours (c) P-SLAM (d) ORB-SLAM3

Fig. 4. Comparison of the proposed system against state-of-the-art methods in the *office room 0* sequence of ICL NUIM in terms of mean/average absolute translation errors (top) and rotation errors (bottom).

5.2 TUM RGB-D

Different types of sequences are included from the TUM RGB-D benchmark, which aims to test general indoor scenes with low-textured scenes and sharp rotational motions. *f1_360*, *f1_room*, *f2_rpy* and *f2_xyz* are recorded in real office scenes, but the camera's rotation motion changes sharply especially in the first sequence. *f3_l_o_house*, *f3_sn_near* and *f3_l_cabinet* contain more structural information, where *f3_sn_near* is built on two white corners, and *f3_l_cabinet* records several movements around the white cabinet. Table 1 shows that ROVE, OLRE and ORB-SLAM2 have problems in low/non-textured regions. In *f3_l_cabinet*

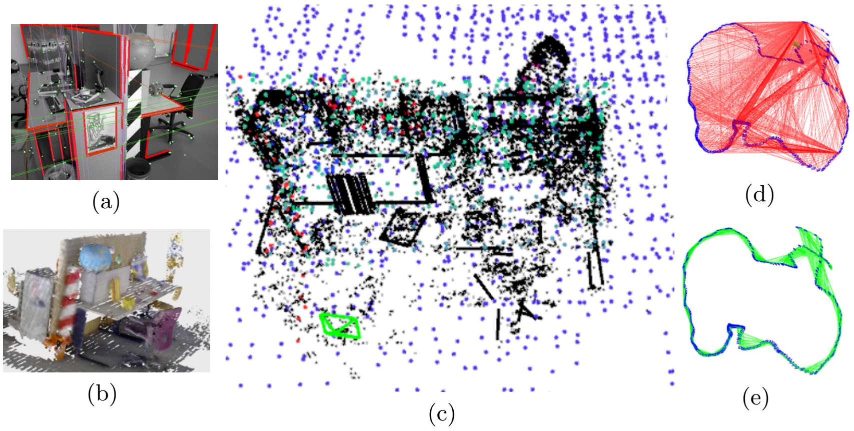


Fig. 5. Scene and graphs of *f3_l_o_house*. (a) 2D image, (b) dense mesh model, (c) sparse map, (d) E-Graph, (e) co-visibilitity graph.

that is not a rigid MW environment, the quality of depth maps is noisy, the surface normal maps extracted by OPRE have a negative effect on rotation estimation.

Table 3. ATE RMSE results (cm) on the TUM RGB-D dataset. Results for BundleFusion and BadSLAM are taken from [32]

Sequence	Ours	BundleFusion [4]	ElasticFusion [37]	BadSLAM [32]
	CPU	GPU	GPU	GPU
f1_desk	1.0	1.6	2.0	1.7
f2_xyz	0.7	1.1	1.1	1.1
f3_office	1.4	2.2	3.6	1.7

For structural sequences listed in Table 1, P-SLAM shows stable performance. In Table 2, general scenes are added as a comparison. As listed in Table 2, the keypoint-based method [3] cannot achieve robust results in *f3_sn_near*, i.e. , a textureless scenario, while the MW-based method [20] has problems when the scene structure breaks the MW assumption, by reporting a low performance in *f2_rpy* and *f3_l_o_house*, and even losing track in *f1_360* and *f1_room*. Therefore, the proposed method shows more robust performances in different types of scenarios, compared with MW-based systems [20, 15] and feature-based approaches [26, 3]. Furthermore, compared with GPU-based systems, our system only works on limited computation sources. As shown in Figure 5, *f3_l_o_house* is used to compare E-Graph and co-visibility graph. As clearly shown, E-Graph allows connecting more distant keyframes than a co-visibility graph. When two keyframes can be connected together, drifting phenomena can more easily be limited, in a similar way to the underlying idea behind loop closure. The cabinet scene is also a difficult sequence for point-based methods (see Figure 6(b)) since point features are concentrated in a few boundary regions. However, our method can deal with this type of scene where the same plane is observed in a number of frames.

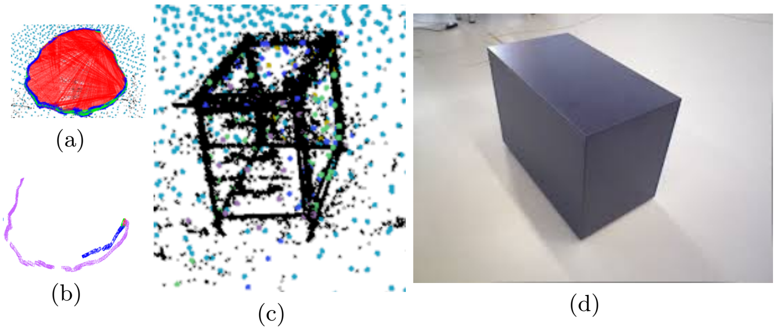


Fig. 6. Scene and graph of *f3_cabinet*. (a) E-Graph, (b) trajectory from ORB-SLAM3, (c) sparse map, (d) 2D image.

6 Conclusion

This paper proposed a new graph structure, E-Graph, to reduce tracking drift based on plane normals and vanishing directions in a scene, which can be used to build a rotation connection between two frames without visual overlap. The advantage of this idea is that rotation errors that occur between two frames have small or no effect on this relative rotation estimation step. Based on the proposed graph, a minimal solution is presented, that shows that two landmarks and two correspondences can be used to solve the relative camera pose. Therefore, the proposed method is better suited for texture-less scenes compared with traditional minimal solutions based on co-visible features. However, the proposed method also has limitations. Compared with point-based systems, our approach requires more types of features. Furthermore, since we need vanishing directions and plane vectors, the method is more suitable for man-made scenes.

Feature work. The E-Graph is a new tool to establish connections across frames and keyframes. An interesting topic for future exploration is considering a covisibility graph and our graph together to revisit pose estimation and obtain further improvements in drift removal.

Acknowledgements. We gratefully acknowledge Xin Li, Keisuke Tateno, Nicolas Brasch and Dr. Liang Zhao for the helpful discussion.

References

1. Andrew, A.M.: Multiple view geometry in computer vision. Kybernetes (2001)
2. Bazin, J.C., Pollefeys, M.: 3-line ransac for orthogonal vanishing point detection. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 4282–4287. IEEE (2012)
3. Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M., Tardós, J.D.: Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam. arXiv preprint arXiv:2007.11898 (2020)
4. Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., Theobalt, C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. ACM Transactions on Graphics (ToG) **36**(4), 1 (2017)
5. Feng, C., Taguchi, Y., Kamat, V.R.: Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 6218–6225. IEEE (2014)
6. Galvez-Lpez, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics **28**(5), 1188–1197 (2012). https://doi.org/10.1109/TRO.2012.2197158
7. Gao, X., Wang, R., Demmel, N., Cremers, D.: Ldso: Direct sparse odometry with loop closure. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2198–2204. IEEE (2018)
8. Gomez-Ojeda, R., Moreno, F.A., Zuniga-Noël, D., Scaramuzza, D., Gonzalez-Jimenez, J.: Pl-slam: A stereo slam system through the combination of points and line segments. IEEE Transactions on Robotics **35**(3), 734–746 (2019)
9. Guan, B., Zhao, J., Li, Z., Sun, F., Fraundorfer, F.: Minimal solutions for relative pose with a single affine correspondence. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1929–1938 (2020)
10. Handa, A., Whelan, T., McDonald, J., Davison, A.: A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM. In: IEEE International Conference on Robotics and Automation (2014)
11. Hsiao, M., Westman, E., Zhang, G., Kaess, M.: Keyframe-based dense planar slam. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). pp. 5110–5117 (2017). https://doi.org/10.1109/ICRA.2017.7989597
12. Joo, K., Oh, T.H., Kim, J., Kweon, I.S.: Globally optimal manhattan frame estimation in real-time. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1763–1771 (2016)
13. Joo, K., Oh, T.H., Rameau, F., Bazin, J.C., Kweon, I.S.: Linear rgb-d slam for atlanta world. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 1077–1083. IEEE (2020)
14. Kim, P., Coltin, B., Jin Kim, H.: Linear rgb-d slam for planar environments. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 333–348 (2018)
15. Kim, P., Coltin, B., Kim, H.J.: Indoor rgb-d compass from a single line and plane. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4673–4680 (2018)
16. Lee, J.K., Yoon, K.J.: Real-time joint estimation of camera orientation and vanishing points. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1866–1874 (2015)
17. Li, H., Xing, Y., Zhao, J., Bazin, J., Liu, Z., Liu, Y.: Leveraging structural regularity of atlanta world for monocular slam. In: 2019 International

Conference on Robotics and Automation (ICRA). pp. 2412–2418 (2019). https://doi.org/10.1109/ICRA.2019.8793716

18. Li, X., Li, Y., Pınar Örnek, E., Lin, J., Tombari, F.: Co-planar parametrization for stereo-slam and visual-inertial odometry. arXiv e-prints pp. arXiv–2009 (2020)
19. Li, Y., Brasch, N., Wang, Y., Navab, N., Tombari, F.: Structure-slam: Low-drift monocular slam in indoor environments. IEEE Robotics and Automation Letters **5**(4), 6583–6590 (2020)
20. Li, Y., Yunus, R., Brasch, N., Navab, N., Tombari, F.: Rgb-d slam with structural regularities. In: 2021 IEEE international conference on Robotics and automation (ICRA) (2021)
21. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987)
22. Ma, L., Kerl, C., Stckler, J., Cremers, D.: Cpa-slam: Consistent plane-model alignment for direct rgb-d slam. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 1285–1291 (2016). https://doi.org/10.1109/ICRA.2016.7487260
23. Mei, C., Sibley, G., Newman, P.: Closing loops without places. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3738–3744. IEEE (2010)
24. Mur-Artal, Raúl, M.J.M.M., Tardós, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Transactions on Robotics **31**(5), 1147–1163 (2015). https://doi.org/10.1109/TRO.2015.2463671
25. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE transactions on robotics **31**(5), 1147–1163 (2015)
26. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. IEEE Transactions on Robotics **33**(5), 1255–1262 (2017). https://doi.org/10.1109/TRO.2017.2705103
27. Nießner, M., Zollhöfer, M., Izadi, S., Stamminger, M.: Real-time 3d reconstruction at scale using voxel hashing. ACM Transactions on Graphics (ToG) **32**(6), 1–11 (2013)
28. Nistér, D.: An efficient solution to the five-point relative pose problem. IEEE transactions on pattern analysis and machine intelligence **26**(6), 756–770 (2004)
29. Pomerleau, F., Colas, F., Siegwart, R.: A review of point cloud registration algorithms for mobile robotics. Foundations and Trends in Robotics **4**(1), 1–104 (2015)
30. Rosinol, A., Sattler, T., Pollefeys, M., Carlone, L.: Incremental visual-inertial 3d mesh generation with structural regularities. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 8220–8226. IEEE (2019)
31. Salaün, Y., Marlet, R., Monasse, P.: Robust and accurate line-and/or point-based pose estimation without manhattan assumptions. In: European Conference on Computer Vision. pp. 801–818. Springer (2016)
32. Schps, T., Sattler, T., Pollefeys, M.: Bad slam: Bundle adjusted direct rgb-d slam. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 134–144 (2019). https://doi.org/10.1109/CVPR.2019.00022
33. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A Benchmark for the Evaluation of RGB-D SLAM Systems. In: IEEE International Conference on Intelligent Robots and Systems (2012)
34. Tardif, J.P.: Non-iterative approach for fast and accurate vanishing point detection. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 1250–1257. IEEE (2009)

35. Von Gioi, R.G., Jakubowicz, J., Morel, J.M., Randall, G.: LSD: A Fast Line Segment Detector with a False Detection Control. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(4), 722–732 (2010)
36. Wang, R., Schworer, M., Cremers, D.: Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3903–3911 (2017)
37. Whelan, T., Leutenegger, S., Salas-Moreno, R., Glocker, B., Davison, A.: Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems (2015)
38. Yunus, R., Li, Y., Tombari, F.: Manhattanslam: Robust planar tracking and mapping leveraging mixture of manhattan frames. arXiv preprint arXiv:2103.15068 (2021)
39. Zhang, X., Wang, W., Qi, X., Liao, Z., Wei, R.: Point-plane slam using supposed planes for indoor environments. Sensors **19**(17), 3795 (2019)
40. Zhao, J., Kneip, L., He, Y., Ma, J.: Minimal case relative pose computation using ray-point-ray features. IEEE transactions on pattern analysis and machine intelligence **42**(5), 1176–1190 (2019)
41. Zhou, H., Zou, D., Pei, L., Ying, R., Liu, P., Yu, W.: Structslam: Visual slam with building structure lines. IEEE Transactions on Vehicular Technology **64**(4), 1364–1375 (2015). https://doi.org/10.1109/TVT.2015.2388780
42. Zhou, Q.Y., Koltun, V.: Dense scene reconstruction with points of interest. ACM Transactions on Graphics (ToG) **32**(4), 1–8 (2013)
43. Zhou, Y., Kneip, L., Rodriguez, C., Li, H.: Divide and conquer: Efficient density-based tracking of 3d sensors in manhattan worlds. In: Asian Conference on Computer Vision. pp. 3–19. Springer (2016)

Part III

Conclusion and Future Work

Conclusion and Future Work 8

8.1 Contributions and Conclusions

Contributions

Primitive detection modules based on geometry and learning algorithms. Primitives detection is one of the most basic modules in SLAM systems. It supports tracking and mapping parts to pursue robust and accurate performances. To improve the pixel utilization rate of monocular images, we provide a semi-dense surface normal detection method (Chapter 4) for SLAM systems. First, dense surface normal maps of monocular images are predicted by convolutional neural networks. Furthermore, we use a sphere mean-shift algorithm to filter out noisy regions based on the dominant directions of indoor environments. For Stereo images, we provide a plane detection method (Chapter 6) that takes advantage of convolutional neural networks to predict potential planar regions first and then uses 3D points and lines in those regions to remove outlier regions based on the RANSAC algorithm.

Robust rotation estimation approach for monocular and RGB-D sensors. A 6-DoF rigid camera motion $\mathbf{T}$ contains 3-DoF rotation $\mathbf{R} \in SO(3)$ and 3-DoF translation $\mathbf{t} \in \mathbb{R}^3$. Since a translation estimation task is a linear problem given rotation results, the 3-DoF orientation estimation tasks are critical for pose estimation. In this thesis, we provide novel solutions for rotation estimation by exploring structural relationships. Based on the assumption of Manhattan World, we provide an architecture for monocular (Chapter 4) and RGB-D (Chapter 5) sensors. For monocular SLAM, we estimate rotation by tracking the perpendicular/orthogonal dominant directions from predicted surface normals. We can estimate relative rotation motions after tracking Manhattan world measurements between two frames. To deal with non-rigid Manhattan scenarios, our architecture provides a bundle adjustment module to refine the local map and camera poses at the same time.

Parametrizations for co-planar primitives In this thesis, we propose two types of primitives that are constructed by a set of basic point-line-plane landmarks with special structural regularities. Given co-planar relationships of points and lines, we propose a parametrization approach (Chapter 6) that represents those points and lines via parameters of the plane they are lying on. Compared with individual landmarks representation, the noise of landmarks' depth is limited mainly.

Joint co-visibility and extensibility factor graphs for optimization. Regarding factor graph optimization tasks, the graph architecture that optimizes those co-observed landmarks and camera poses is termed as **Co-visibility Graphs**, where co-observed landmarks can only be measured by those closing frames in space. Compared to co-visibility graph architecture, the graph architecture using connections between structural regularities and camera orientation is termed as **Extensibility Graphs** (Chapter 7).

Conclusion

SLAM and VO methods [13, 55] working as general trackers have presented impressive performances in many scenarios, but those methods still suffer from tracking lost issues in some man-made regions where not enough point features are being detected. Although those regions are only a part of the scene, this tracking loss problem still dramatically reduces the system's performance. To make tracking and mapping systems more accurate, robust, and reliable in man-made scenes, we propose solutions to improve primitives detection, pose estimation, primitives representation, and factor graph optimization modules in this dissertation.

Traditionally, monocular SLAM methods, especially feature-based systems, take advantage of very limited pixels of images since only small parts of input pixels can be worked as point features. We agree that adding line features will help increase tracking information, but the increase is limited. To solve the problem, our first work, Structure-SLAM, proposes a decoupled pose estimation architecture for a monocular RGB sensor, which uses points and lines and leverages dense surface normals estimated from a neural network into the tracking module. Furthermore, the proposed method estimates rotation motion based on surface normals under the assumption of Manhattan World. A translation vector is solved under a linear, close-formed relationship given rotation and features.

The architecture of Structure-SLAM is extended for RGB-D sensors, where real scales for scenes can be recovered benefitting from depth maps. In this completed architecture, plane primitives are added to estimate camera poses jointly with points and lines. After classifying those lines and planes into three orthogonal groups, we add parallel and perpendicular constraints in the bundle adjustment module. Since we have reconstructed dominant planar point clouds during the tracking process, we mesh them to dense representations for robot navigation applications. Those planar point clouds are also explored by building deformable surfaces based on superpixel representations.

The assumption of Manhattan World shows advantages in scenes that satisfy the assumption. However, it brings problems to our system in more general indoor environments since different parts of indoor areas cannot be modeled by a single Manhattan World. Therefore, we evolved the architecture by making the Manhattan World model work locally. During the tracking process, we record every Manhattan Frame constructed by at least two perpendicular planes into a structure graph. Therefore, in the rotation estimation module, we will revisit the graph when we detect that the new coming frames contain structure information. Based on the matching strategies, relative rotation between two frames may be estimated without visual overlaps.

Due to perpendicular planes are not very common in indoor scenes, we extend primitives to vanishing points that are built by parallel lines on image planes. And then, we break down the limitation of perpendicular relationships by using Schmidt quadrature operations on non-parallel primitives. In this work, those structure primitives, including vanishing directions and surface normals, will be maintained in a new factor graph structure, Extensibility Graph, which is proposed to distinguish those new constraints from ones in co-visibility graphs. The advantage of our new graph is that the edges connecting primitives and rotation vertices in extensibility graphs can be built without visual overlaps. Specifically, an edge in the co-visibility graph is generated because of the observation between landmarks and the camera. Taking the factor that different parallel line landmarks have the same direction vector. In contrast, a plane, no matter how big, only has a normal vector; the rotations of some cameras, even though they do not have any overlaps, are possibly connected directly in our extensibility graphs. However, it will never happen in co-visibility graphs. In other words, some long-distance rotation connections will be generated in our graphs, while corresponding connections in co-visibility graphs will be much shorter. Short connections have difficulties in limiting pose drift issues.

8.2 Future Work

While exploring the relationship between structural regularities and SLAM problems, we notice that some important and exciting topics still await exploration in parameterization, optimization, and dense reconstruction areas. This section categorizes future work into fundamental theories, SLAM system development, and deep neural networks for scene understanding.

Fundamental Theories

Representation and Optimization of 3D Lines. 3D lines are commonly detected in objects' surfaces and structures of man-made environments, some exhibiting a high degree of structural forms. When it comes to leveraging those line landmarks into co-visibility graphs of SLAM systems, the robustness of pose estimation will be improved compared with point-only factor graphs. The optimization processes of normalized direction and normal vectors are regarded as the same in the popular minimal parameterization method, Orthonomal Representation. However, the idea will be changed when we consider a set of parallel line landmarks simultaneously. Specifically, for a set of parallel lines, the normalized direction vectors of each line are the same in mathematics, but the normal vectors are different from one to another. Therefore, the Orthonormal approach can not be directly extended to represent a set of structural lines. To solve this issue, a future direction represents the normalized direction vector on S^2 and parameterizes the normalized normal vector on S^1 to take place of $SO(3)$ and $SO(2)$ used in $\mathcal{O}$.

Orientation Closure for SLAM systems. In the incremental localization process, the camera poses tend to drift during the accumulation of errors in estimated poses and maps. A loop closure module is widely used in SLAM systems to reduce the drift generated in the process, which uses co-visible features between two ends of a loop. Given a relative pose between two remote images at the end of the loop, a constraint between estimated camera poses and new relative poses provided by the loop closure module can be used to optimize pose graphs and factor graphs. Loop closure shows impressive pose drift-removing abilities, but the limitation is that the robot has to revisit the same place. Taking the fact that visual-based 6-DoF pose estimation can be transferred to a linear least-square problem with a known orientation part [103] and the relative rotation estimation between two frames does not require visual over-laps, a type of new closure, Orientation Closure, can be expected to be used in SLAM systems to decrease camera pose drift. Furthermore, another important thing is that the Orientation Closure does not require the same place revisiting.

SLAM system development

Structural Regularities and Self-supervised Networks for dense monocular SLAM. Normals predicted by supervised deep networks are leveraged in our monocular SLAM architecture, Structure-SLAM, but limited by the ground truth surface normal maps, domain shift issues have a negative effect on the performance of normal prediction in real-world scenes. Taking the fact that there are several dominant directions in man-made environments, most planar regions are possible to parallel to one of those dominant directions and perpendicular to some of the other directions. Furthermore, by tracking the dominant directions in different views, the normal estimation method provides consistent surface normal loss functions. The predicted surface normals are used in the rotation estimation of a monocular SLAM method.

Tracking, Understanding, and Mapping in Large Scenes. Some tracking and mapping applications are required in large indoor scenes, like office buildings, where traditional RGB-D sensors are inefficient for fast reconstructions since the valid depth perception distance of those sensors is very short. Therefore, future work is to design a new device based on monocular and solid-state LiDAR sensors to meet tracking, understanding, and mapping requirements for large-scale environments. First, a robust initialization algorithm is proposed to activate a co-visibility graph with a real scale, which uses the semi-direct strategy based on visual features and sparse LiDAR points. Furthermore, sequential inputs are fed to the faster tracking module based on structural regularities. Based on our tracking and sparse mapping algorithms, a keyframe-based depth completion module based on neural convolutional networks is implemented to reconstruct dense maps incrementally.

Deep neural Networks for scene understanding

3D reconstruction from a single RGB image. In man-made environments, especially for indoor scenes, it is common to observe orthogonal and parallel planes and lines that can be noted as Manhattan World, and those image frames with this information can be noted as Manhattan Frame. The Manhattan Frame has an important usage application of computer vision and robotics, which can be used to regress the rotational motion between the camera and the real world. Based on the explored constraints between structural cues, a future direction is to explore a network to extract and reconstruct parallel and perpendicular planes from a single RGB image, which predicts instance-wise normal and depth maps and recovers angles between different instances.

Part IV

Appendix

A. Jacobian Matrices in Optimization

1. Jacobian Matrix from Re-projection Factors of Lines

The Plücker representation of the mapline $\mathcal{L}^w$ in the world coordinate is denoted by

$$\mathcal{L}^i = \begin{bmatrix} \mathbf{n}^i \\ \mathbf{d}^i \end{bmatrix} = \mathcal{T}_{c_i,w} \begin{bmatrix} \mathbf{n}^w \\ \mathbf{d}^w \end{bmatrix} \tag{.1}$$

here $\mathcal{T}_{c_i,w} = \begin{bmatrix} \mathbf{R}_{c_i,w} & [\mathbf{t}_{c_i,w}]_\times \mathbf{R}_{c_i,w} \\ \mathbf{0} & \mathbf{R}_{c_i,w} \end{bmatrix}$, and $[\cdot]_\times$ is the skew-symmetric operation. Then, the $\begin{bmatrix} \mathbf{n}^i & \mathbf{d}^i \end{bmatrix}$ can be denoted by

$$\begin{aligned} \begin{bmatrix} \mathbf{n}^i & \mathbf{d}^i \end{bmatrix} &= \begin{bmatrix} \frac{\mathbf{n}}{||\mathbf{n}||} & \frac{\mathbf{d}}{||\mathbf{d}||} & \frac{\mathbf{n}}{||\mathbf{n}||} \times \frac{\mathbf{d}}{||\mathbf{d}||} \end{bmatrix} \begin{bmatrix} ||\mathbf{n}|| & 0 \\ 0 & ||\mathbf{b}|| \\ 0 & 0 \end{bmatrix} \\ &= \sqrt{||\mathbf{n}||^2 + ||\mathbf{b}||^2} \mathbf{R}(\boldsymbol{\varphi}) \begin{bmatrix} \cos\theta & 0 \\ 0 & \sin\theta \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{.2}$$

here $\boldsymbol{\varphi} = [\varphi_1 \ \varphi_2 \ \varphi_3]^T$ and $|| \cdot ||$ is the normalization operation. We make use of $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix}, \mathbf{U} \in SO(3)$ to represent $\mathbf{R}(\boldsymbol{\varphi})$. $\mathbf{W} = \begin{bmatrix} w_1 & -w_2 \\ w_2 & w_1 \end{bmatrix} = \frac{1}{\sqrt{||\mathbf{n}||^2 + ||\mathbf{d}||^2}} \begin{bmatrix} ||\mathbf{n}|| & -||\mathbf{d}|| \\ ||\mathbf{d}|| & ||\mathbf{n}|| \end{bmatrix}$.

On the image plane, the reprojected endpoints, $\mathbf{p}_s$ and $\mathbf{p}_e$, of a 3D line in the camera coordinate can be obtained by

$$\bar{\mathbf{p}}_k = K\bar{\mathbf{P}}_k, k \in (s, e) \tag{.3}$$

here $\bar{\mathbf{P}}_k$ is a normalized 3D endpoint in the camera coordinate. Therefore, the re-projected line $\mathbf{l}_j = \bar{\mathbf{p}}_s \times \bar{\mathbf{p}}_e = K\bar{\mathbf{P}}_s \times K\bar{\mathbf{P}}_e = \mathcal{K}(\bar{\mathbf{P}}_s \times \bar{\mathbf{P}}_e) = \mathcal{K}\mathbf{n}^j$, and $\mathcal{K} = \begin{bmatrix} f_y & 0 & 0 \\ 0 & f_x & 0 \\ -f_y c_x & -f_x c_y & f_x f_y \end{bmatrix}$.

The reprojected line $\mathbf{l}_j$ lying on the image plane can be represented as

$$\mathbf{l}_j = \mathcal{K}\mathbf{n}_j^c = w_1^j \begin{bmatrix} f_y & 0 & 0 \\ 0 & f_x & 0 \\ -f_y c_x & -f_x c_y & f_x f_y \end{bmatrix} \mathbf{u}_1^j. \tag{.4}$$

When the mapline $\mathcal{L}^w$ is detected by frame F_i and F_j, we can find the 2D correspondences lying on related images. The error between the re-projected line $\mathbf{l}^j$ and two endpoints, $\mathbf{p}_s$ and $\mathbf{p}_e$, of the extracted 2D line in the frame, can be written as

$$\mathbf{r}_l(\mathbf{p}_{k,s}^j, \mathbf{p}_{k,e}^j, \mathcal{L}^w, \mathcal{X}) = \begin{bmatrix} d(\mathbf{p}_{k,s}^j, \mathbf{l}^j) \\ d(\mathbf{p}_{k,e}^j, \mathbf{l}^j) \end{bmatrix} \tag{.5}$$

where $d(\mathbf{p}_{k,s}^j, \mathbf{l}) = \frac{\mathbf{p}_{k,s}^{jT} \cdot \mathbf{l}}{\sqrt{l_1^2 + l_2^2}}$, and $\mathbf{l}^j = \begin{bmatrix} l_0 & l_1 & l_2 \end{bmatrix}^T$ is the 2D line re-projected from the j^{th} 3D mapline. $\mathbf{p}_k = \begin{bmatrix} \bar{x}_k & \bar{y}_k & 1 \end{bmatrix}^T$, $k \in (s, e)$, are the normalized coordinates of endpoints.

Jacibian for Line landmarks

In the iterative optimization process, the line representation is represented as $\zeta_{|4\times 1} = \begin{bmatrix} \theta^T & \gamma \end{bmatrix}^T$, the pose is detected as $\xi_{|6\times 1} = \begin{bmatrix} \rho^T & \phi^T \end{bmatrix}^T$. Based on the chain rule, the Jacobian matrix $\mathbf{J} = \frac{\partial \mathbf{r}_l}{\partial \delta \zeta}$ can be calculated via $\frac{\partial \mathbf{r}_l}{\partial \delta \theta}$ and $\frac{\partial \mathbf{r}_l}{\partial \delta \gamma}$, which are denoted as

$$\begin{cases} \frac{\partial \mathbf{r}_l}{\partial \delta \theta} = \frac{\partial \mathbf{r}_l}{\partial \mathbf{l}} \frac{\partial \mathbf{l}}{\partial \mathcal{L}_c} \frac{\partial \mathcal{L}_c}{\partial \mathcal{L}_w} \frac{\partial \mathcal{L}_w}{\partial \mathcal{O}} \frac{\partial \mathcal{O}}{\partial \delta \theta} \\ \frac{\partial \mathbf{r}_l}{\partial \delta \gamma} = \frac{\partial \mathbf{r}_l}{\partial \mathbf{l}} \frac{\partial \mathbf{l}}{\partial \mathcal{L}_c} \frac{\partial \mathcal{L}_c}{\partial \mathcal{L}_w} \frac{\partial \mathcal{L}_w}{\partial \mathcal{O}} \frac{\partial \mathcal{O}}{\partial \delta \gamma} \end{cases} \tag{.6}$$

where $\frac{\partial \mathbf{l}}{\partial \mathcal{L}_c} = \begin{bmatrix} \mathcal{K} & \mathbf{0} \end{bmatrix}$, $\frac{\partial \mathcal{L}_c}{\partial \mathcal{L}_w} = \mathcal{T}_{wc}^{-1}$, and the representation of $\mathcal{O}$ is $\begin{bmatrix} \mathbf{n}^T & \mathbf{d}^T & \omega_n & \omega_d \end{bmatrix}^T$.

Therefore, $\frac{\partial \mathcal{L}_w}{\partial \mathcal{O}}$ can be obtained by $\frac{\partial \begin{bmatrix} \omega_1 \mathbf{n} \\ \omega_2 \mathbf{d} \end{bmatrix}}{\partial \mathcal{O}} = \begin{bmatrix} \frac{\mathcal{L}_w}{\partial \mathbf{n}} & \frac{\mathcal{L}_w}{\partial \mathbf{d}} & \frac{\mathcal{L}_w}{\partial \omega_1} & \frac{\mathcal{L}_w}{\partial \omega_2} \end{bmatrix} = \begin{bmatrix} \omega_1 \mathbf{I} & \mathbf{0} & \mathbf{n} & \mathbf{0} \\ \mathbf{0} & \omega_2 \mathbf{I} & \mathbf{0} & \mathbf{d} \end{bmatrix}$.

Futhermore, $\frac{\partial \mathcal{O}}{\partial \begin{bmatrix} \delta\theta \\ \delta\gamma \end{bmatrix}}$ can be denoted as

$$\frac{\partial \mathcal{O}}{\partial \begin{bmatrix} \delta\theta \\ \delta\gamma \end{bmatrix}} = \begin{bmatrix} \frac{\partial \mathbf{n}}{\partial \delta\theta} & \frac{\partial \mathbf{n}}{\partial \delta\gamma} \\ \frac{\partial \mathbf{d}}{\partial \delta\theta} & \frac{\partial \mathbf{d}}{\partial \delta\gamma} \\ \frac{\partial \omega_n}{\partial \delta\theta} & \frac{\partial \omega_n}{\partial \delta\gamma} \\ \frac{\partial \omega_d}{\partial \delta\theta} & \frac{\partial \omega_d}{\partial \delta\gamma} \end{bmatrix} \tag{.7}$$

Since $\mathbf{U} \in SO(3)$ and $\mathbf{W} \in SO(2)$, we take advantage of the perturbation model and Lie algebras to update them. Specifically, $\Delta\mathbf{U} = \exp([\delta\theta]_\times)$ and $\Delta\mathbf{W} = \exp([\delta\gamma]_\times)$ are operated on $\mathbf{U}$ and $\mathbf{W}$, respectively.

$$\begin{aligned} \mathbf{U}^{'} &= \mathbf{U}\exp([\delta\theta]_\times) \approx \mathbf{U}(\mathbf{I} + [\delta\theta]_\times) \\ \mathbf{W}^{'} &= \mathbf{W}\exp([\delta\gamma]_\times) \approx \mathbf{W}(\mathbf{I} + [\delta\gamma]_\times) \end{aligned} \tag{.8}$$

Therefore, the Jacobian matrix can be computed as

$$\begin{aligned} \frac{\partial \mathbf{u}_1}{\partial \delta\theta} &= \begin{bmatrix} \mathbf{0} & -\mathbf{u}_3 & \mathbf{u}_2 \end{bmatrix} \\ \frac{\partial \mathbf{u}_2}{\partial \delta\theta} &= \begin{bmatrix} \mathbf{u}_3 & \mathbf{0} & -\mathbf{u}_1 \end{bmatrix} \\ \frac{\partial \omega_1}{\partial \delta\gamma} &= -\omega_2 \\ \frac{\partial \omega_2}{\partial \delta\gamma} &= \omega_1 \end{aligned} \tag{.9}$$

Therefore, the Equation .7 can be represented as

$$\frac{\partial \mathcal{O}}{\partial \begin{bmatrix} \delta\theta \\ \delta\gamma \end{bmatrix}} = \begin{bmatrix} \frac{\partial \mathbf{n}}{\partial \delta\theta} & \frac{\partial \mathbf{n}}{\partial \delta\gamma} \\ \frac{\partial \mathbf{d}}{\partial \delta\theta} & \frac{\partial \mathbf{d}}{\partial \delta\gamma} \\ \frac{\partial \omega_n}{\partial \delta\theta} & \frac{\partial \omega_n}{\partial \delta\gamma} \\ \frac{\partial \omega_d}{\partial \delta\theta} & \frac{\partial \omega_d}{\partial \delta\gamma} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & -\mathbf{u}_3 & \mathbf{u}_2 & \mathbf{0} \\ -\mathbf{u}_3 & \mathbf{0} & -\mathbf{u}_1 & \mathbf{0} \\ 0 & 0 & 0 & -\omega_2 \\ 0 & 0 & 0 & \omega_1 \end{bmatrix} \tag{.10}$$

Therefore, all components on the chain are computed for the final Jacobian matrix.

B. Supplementaries for Methodology

1. Structure-SLAM for Unknown Scenes

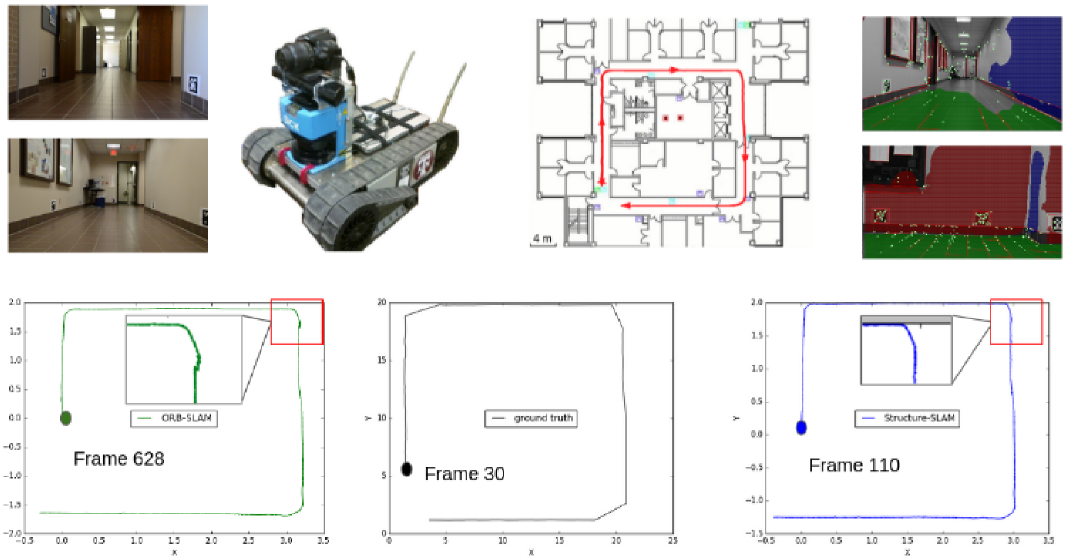


Figure .1 RBB4[104] has 12,000 frames recorded in a corridor. The dataset includes one camera ground truth pose every 50 frames

In the experiment, we use TUM RGB-D and ICL-NUIM images for network finetune. Therefore, in the Supplymentries, we test the Structure-SLAM system in a public sequence in which the scene was not used for training. As shown in Figure .1, the proposed method based on surface normals performs more robust in initialization and corner regions than ORB-SLAM.

2. Co-planar Primitives Detection

When points, lines, and planes are detected in the tracking process, re-projection errors are generally used to optimize landmarks and poses. In the Kimera-VIO method, the co-planar relationships, like point-plane and line-plane, are detected to add new distance constraints between those co-planar points/lines and the corresponding planes during the bundle adjustment model. When more and more features and constraints are added to the optimization modules, the computation burdens will be more intensive.

To solve the problem, the co-planar feature group is regarded as a new type of primitive, which works as a whole in the optimization module. The process of co-planar primitive detection is an efficient solution, which contains three steps as follows:

- (1) plane segmentation based on a single RGB image;
- (2) points and lines lie on the planar regions are regarded as potential co-planar features, which will be rechecked by feeding points, lines, and parameters of the plane into a RANSAC model;
- (3) The plane parameters in the optimization module represent those point and line inliers.

The detection process will be introduced in this section.

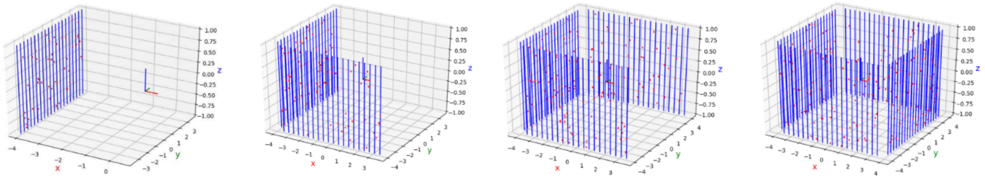
Plane Segmentation. For a single RGB image input, the results predicted by networks, as shown in Figure .2, have some outlier regions, which can not be used for co-planar feature association.

Figure .2 Planes segmentation based on neural networks. Black color shows non-planar regions.

However, the RANSAC method assumes the presence of a single plane instance in the input data. So, to extract more than one plane from the same point cloud, one has to apply RANSAC iteratively to extract planes one after another. A standard approach to carry out this is to remove its inliers every time a plane is detected and execute the algorithm again to find the next plane. This is suboptimal for the following reasons: We do not know the number of plane models in the point cloud. So, we do not know how many times to run the algorithm. Results will be affected if points belonging to other planes are detected as inliers for the current one and removed from the point cloud.

As shown in Figure .3, we generate four scenes with different numbers of planes. Each 3D plane consists of 90 inlier points (50 points and 40 endpoints from 20 lines), which are corrupted by 0.1 m Gaussian random noise, and 9 Outliers (10%).

We have tested RANSAC in these four scenes where the number of planes changes from one to four. For our method, the RANSAC can be guided by detecting a plane region first. The performance of the standard RANSAC deteriorates as the number of planes increases due to false positives. Results are shown in Table .1.



(a) Scene A (b) Scene B (c) Scene C (d) Scene D

Figure .3 From left to right, scenes contain different numbers of planes, which are built by points and endpoints of lines.

	Scene A	Scene B	Scene C	Scene D
max iterations	30	234	666	892
max inlier ratio	52.76%	27.29%	19.17%	17.37%
Fitting accuracy	100%	96.3%	64.4%	40%

Table .1 Analysis of our method and the RANSAC in finding one plane with different scenes. We report the mean from 1000 runs

In the experiments, RANSAC is used to extract the parameters of a plane. Then, by comparing parameters with the ground truth, the plane will be regarded as a correct one if the parameters between detection $\imath_{dt}$ and ground truth π_{gt} is less than a threshold t_r

$$\begin{cases} t_{angle} & = \frac{\mathbf{n}_{dt} \cdot \mathbf{n}_{gt}}{||\mathbf{n}_{dt}||||\mathbf{n}_{gt}||} \\ t_{dis} & = d_{dt} - d_{gt} \end{cases} \tag{.11}$$

In the experiments, we assume that the correct model can be obtained in 99.99% of the cases. Scene A has one plane, and RANSAC can fit the plane accurately in 30 iterations. When the number of planes increases in Scene B, Scene C, and Scene D, the fitting accuracy decreases from 96.3% to 40%, but the process takes more iterations, increasing from 234 to 892. If a scene has many planes, we will divide those planes first. Therefore, our method solves this issue in Scene B, Scene C, and Scene D.

C. Licenses and Publications Discussed in This Dissertation

Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments

Yanyan Li[1], Nikolas Brasch[1], Yida Wang[1], Nassir Navab[1,2] and Federico Tombari[1,3]

[1] Technical University of Munich
[2] Johns Hopkins University
[3] Google

Title	Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments
Version	Published Version
Paper document	Attached
License document	Attached

Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments

Yanyan Li, Nikolas Brasch, Yida Wang, Nassir Navab, and Federico Tombari

Abstract—**In this letter a low-drift monocular SLAM method is proposed targeting indoor scenarios, where monocular SLAM often fails due to the lack of textured surfaces. Our approach decouples rotation and translation estimation of the tracking process to reduce the long-term drift in indoor environments. In order to take full advantage of the available geometric information in the scene, surface normals are predicted by a convolutional neural network from each input RGB image in real-time. First, a drift-free rotation is estimated based on lines and surface normals using spherical mean-shift clustering, leveraging the weak Manhattan World assumption. Then translation is computed from point and line features. Finally, the estimated poses are refined with a map-to-frame optimization strategy. The proposed method outperforms the state of the art on common SLAM benchmarks such as ICL-NUIM and TUM RGB-D.**

Index Terms—**SLAM, visual learning.**

I. Introduction

Visual Simultaneous Localization and Mapping (V-SLAM) systems are important for autonomous robots and augmented reality, as they are used to estimate poses and reconstruct unknown environments. In numerous SLAM use cases and applications, monocular cameras are the most common sensors in indoor scenarios. Indoor environments are often characterized by a lack of textured surfaces, and by irregularly distributed feature points. In particular, low-textured walls, floor and ceiling are difficult to deal with by both state-of-the-art feature-based methods [1] as well as direct methods [2], [3]. For low-textured scenes, SLAM systems combining point and line features have been proposed to target low-textured scenes, e.g. Stereo-PLSLAM [4], PLVO [5], Mono-PLSLAM [6] and [7], extending the working scenarios to low-textured environments with visible structural edges. Since the map is built from a sequence of input frames, small errors accumulate over time, resulting in drift which affects dense reconstruction by leading to misaligned surfaces and artifacts.

There are two main strategies to overcome these errors. Loop closure detection [1], [8] combined with pose graph optimization detects previously seen landmarks and optimizes the pose graph based on the new constraints, thus correcting the accumulated drift. Loop closure, however, brings in an extra computational burden and removes the drift only when revisiting the same place. Another strategy consists of assuming an underlying (global) structure in the world frame, then each tracked frame can be directly aligned to this world structure instead of the last frame or keyframes. The most common formulation of a structured scene is the Manhattan World (MW) [9], [10] where the environment shown in Fig. 1(a) consists of geometric structures (planes and lines) oriented in one of three orthogonal orientations. It is particularly useful in indoor environments where structures such as walls, floor and ceilings often show consistent alignment over multiple rooms, enabling a global alignment.

The MW approach is an efficient method to keep the accumulated drift low by providing a drift-free strategy for rotation estimation, as the rotational component is the main source of overall drift [11], [12].

The state of the art of monocular approaches relying on a MW [9], [10] are based on parallel and orthogonal lines alone, as it is difficult to extract 3D information, except for vanishing points, from a monocular RGB image, which is a quite strong limitation for most scenarios. Furthermore, indoor environments often consist of large planar regions with few features for pose estimation. RGB-D methods [12], [13], directly measure the structure of the scene in the form of depth maps, this allows them to compute dense surface normals for each pixel.

Inspired by recent works based on convolutional neural networks (CNN) and scene geometry prediction approaches from a single view [14], [15], we propose a monocular SLAM framework which leverages the underlying scene structure to carry out low-drift SLAM even in presence of low-textured environments, in the form of densely predicted normal maps from a CNN, analogously to existing works based on dense RGB-D sensors.

Specifically, we propose the following contributions:

- A low drift real-time monocular SLAM framework for structured environments, with decoupled rotation and translation
- Dense monocular normal estimation for rotation estimation leveraging the MW assumption

Manuscript received February 24, 2020; accepted July 23, 2020. Date of publication August 11, 2020; date of current version August 21, 2020. This work was supported by China Scholarship Council. This letter was recommended for publication by Associate Editor J. Andrade-Cetto and Editor S. Behnke upon evaluation of the reviewers' comments. *(Yanyan Li and Nikolas Brasch contributed equally to this work.) (Corresponding author: Yanyan Li.)*

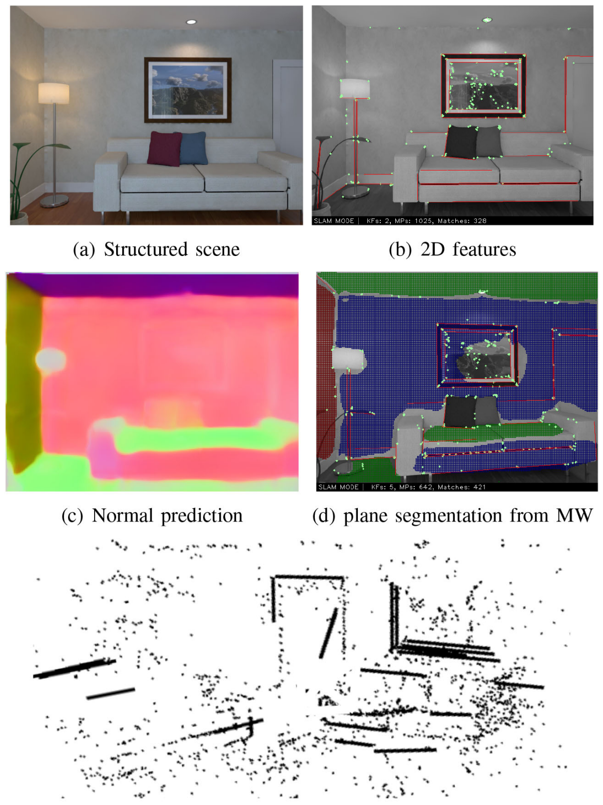
Yanyan Li, Nikolas Brasch, and Yida Wang are with the Technical University of Munich, 80333 München, Germany (e-mail: yanyan.li@tum.de; nikolas.brasch@tum.de; yida.wang@tum.de).

Nassir Navab is with the Technical University of Munich, 80333 München, Germany, and also with the Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: nassir.navab@tum.de).

Federico Tombari is with the Technical University of Munich, 80333 München, Germany, and also with the Google, 8002 Zürich, Switzerland (e-mail: tombari@in.tum.de).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3015456


SLAM MODE | KFs: 2, MPs: 1025, Matches: 328
SLAM MODE | KFs: 5, MPs: 642, Matches: 421

(a) Structured scene (b) 2D features

(c) Normal prediction (d) plane segmentation from MW

(e) Sparse output map

Fig. 1. The proposed approach targets low-textured indoor scenes to carry out low-drift monocular SLAM based on dense normal prediction and leveraging the Manhattan World assumption.

- A method for translation estimation relying on point and line features

We evaluate numerically on common SLAM benchmarks such as ICL-NUIM [16] and TUM RGB-D [17] showing that the proposed approach outperforms the state of the art in monocular SLAM.

II. Related Work

A. Monocular SLAM

PTAM [18] is a monocular, keyframe-based SLAM system which was the first work to introduce the idea of splitting camera tracking and mapping into parallel threads, and demonstrate to be successful for real time augmented reality applications in small-scale environments. Strasdat *et al.* [8] present a large scale monocular SLAM system in which the front-end bases on optical flow implemented on a GPU, followed by FAST feature matching and motion-only BA, and a back-end based on sliding-window BA. As a complete SLAM pipeline, ORB-SLAM [1] combines feature based tracking, sparse point mapping, descriptor based re-localization and loop closure altogether. In addition to point features several works propose the use of lines [4], [5] for low-textured environments, we propose to use additional dense structural information in the form of predicted normal maps.

Inspired by the recent success of deep learning based depth prediction, CNN-SLAM [19] incorporates a neural network which estimates depth information within the popular LSD-SLAM [2] framework to create dense scene reconstructions in metric scale, where depth predictions are used to initialize the SLAM system and merged continuously with the semi-dense depth maps optimized by the SLAM system. Instead of estimating depth maps only for key-frames in CNN-SLAM, our approach predicts surface normals from every RGB frame in real-time. In CodeSLAM [20], a neural network learns a compact latent representation for the structure of a scene conditioned on the RGB image, showing that the joint optimization of both structure and pose can improve monocular pose estimation. By predicting normal maps instead of depth maps we avoid the necessary differentiation operation which could introduce noise. Predicting normal maps also seems to generalize better between datasets as depth does.

B. RGB-D SLAM

Probabilistic-VO [7] combines points together with lines and planes for pose estimation while modeling their uncertainties. Due to the combination of 2D-3D point and line correspondences and 3D-3D plane matches, a weighting between reprojection and euclidean errors must be chosen empirically. CPA-SLAM [21] extended DVO-SLAM with global plane landmarks. Pose estimation and soft assignment of depth measurements to planes are computed in an Expectation-Maximization framework. KDP-SLAM [22] combines photometric and geometric loss based on plane segments instead of points for frame-to-frame pose estimation and additionally aligns plane segments with global planes in a Smoothing and Mapping (SAM) framework.

C. Manhattan World

Straub *et al.* [11] and Zhou *et al.* [23] show that the main source of drift in traditional feature-based systems is caused by the rotation estimation.

Even if the MW assumption is a good constraint for indoor SLAM, it is difficult to enforce it in monocular methods because only limited 3D information can be obtained. Zhou *et al.* [10] applies J-linkage [24] to classify parallel line segments into different groups and estimate the dominant direction from the vanishing points. If depth maps are available, surface normals can be computed directly. Joo *et al.* [25] provide a branch-and-bound framework for Manhattan Frame estimation. MVO [23] propose a unit sphere mean shift method to find the rotation matrix between the Manhattan World and the camera system. For the translational part, they compute and align density distributions of points in each orthogonal direction, avoiding the costly matching of points. OPVO [26] use planes to estimate the Manhattan Frame rotation, limiting its application to environments with at least 2 orthogonal planes. LPVO [12] adds vanishing points of lines for the rotation estimation. Both use point based methods for translation estimation. L-SLAM [13] replaces the graph based translation estimation from LPVO with a Kalman filter based SLAM update, using the LPVO translation estimation in the prediction step. Compared with [12], [13], we build an initialization module based on points, lines and predicted normals. Further more, a refinement module is added to optimize the pose after the decoupled initialization.

III. Scene Structure Analysis

The structural information used in the system is analyzed in this section. First, we describe the methods for extraction and triangulation of points and lines; Then, an architecture for surface normal prediction is introduced.

A. Points and Lines Analysis

Point features, due to their descriptiveness, compactness and robustness to illumination changes, are the most common features used in visual SLAM systems. In our method, ORB features [27] are adopted which are fast enough to extract and robust enough to get matched. Since it's hard to extract sufficient feature points for robust pose estimation in low-textured environments, we further supplement them with line segments extracted and encoded using the LSD [28] and LBD [29] accordingly.

Similar to ORB-SLAM [1], once the 2D point features $p_n = (u_n, v_n)$ and line segments $l_m = (p_{m,s}, p_{m,e})$ are extracted in the new keyframe F_i, new features are triangulated to 3D points P_n and lines L_m with correspondences located on other connected keyframes.

Due to the factorization of rotation and translation estimation, it is possible to estimate the pose even in cases with pure rotation and no translation or with small parallax, which would not be possible with pure monocular feature based approaches. The rotation can be estimated from the Manhattan World Frame, this means fewer landmarks are needed to obtain the remaining 3 degrees of freedom for the translation.

B. Surface Normal Prediction

We use learned knowledge to reason about the 3D environment, instead of measuring dense depth values directly. Therefore, a 2D convolutional architecture(CNN) is trained to segment planar regions and predict pixel-wise surface normals. The proposed CNN is composed of a ResNet101-FPN [14] encoder for feature extraction and a two-branch decoder for planar area segmentation and normal estimation. As the planar and non-planar regions are unbalanced in indoor scenarios, we use the balanced cross entropy loss for training

$$\mathscr{L}_p = -1(1-w)\sum_{i\in P}\log p_i - w\sum_{i\in P_{neg}}\log(1-p_i), \tag{1}$$

where P and P_{neg} represent planar and non-planar regions, respectively. p_i represents the probability of the ith pixel being located in a planar region. We use w to balance the contributions of planar and non-planar pixels. Then the loss function for the normal estimation is filtered by the planar mask.

$$\mathscr{L}_n = -\frac{1}{n}\sum_{i\in P} n_i \cdot n_i^*, \tag{2}$$

where n_i and n_i^* are the predicted normal and ground truth normal for the ith pixel.

IV. Initialization

In this section, we describe the strategy of computing the relative poses between two frames and reconstructing an initial map. In order to be robust to different motions, we decouple pose estimation into rotation and translation which is explained further in the following paragraphs.

Rotation. First, we assume that there is a Manhattan coordinate system M shown in Fig. 3, we compute the relative rotation R_{C_1M} from Manhattan coordinate frame M to the first frame C_1 by clustering the normal map v_i^s of C_1 on the unit Gaussian sphere [12], [23] centered on the M. Following [12], [23], we project the normals onto the tangent plane of each Manhattan world axis r_n, where $n \in [1,2,3]$, for the current estimation. Instead of testing several random matrices, we found that setting R_{C_0M} to identity and running multiple mean-shift iterations is enough to obtain a good estimate. In order to remove noise from normal maps, we only consider the vectors $v_{in}^{s'}$ which are close to the axis r_n.

Then, the refined surface normal vectors $v_{in}^{s'}$ are projected to two-dimensional vectors $m_{in}^{'}$ in the nth tangential plane. We compute the cluster mean $s_n^{'}$ for the nth tangential plane under a Gaussian kernel by

$$s_n^{'} = \frac{\sum_{in} e^{-c\|m_{in}^{'}\|^2} m_{in}^{'}}{\sum_{in} e^{-c\|m_{in}^{'}\|^2}} \tag{3}$$

where c is a hyper parameter that defines the width of the kernel, which is set to 2 in our experiments. Then, we transform the cluster centers back onto the Gaussian sphere as s_n, which are used to update the angle between the camera and the MW axis $\hat{r}_n$ combining with the current rotation Q_n,

$$\hat{r}_n = Q_n s_n, \tag{4}$$

here $Q_n = [r_{mod(n,3)}, r_{mod(n+1,3)}, r_{mod(n+2,3)}]$ and $mod()$ is a modulus operation. The tangent plane and the cluster centers are iteratively computed until the rotation estimate is converged. Then we obtain $R_{C_1M} = [\hat{r}_1, \hat{r}_2, \hat{r}_3]^T$.

Translation. As for the translation estimation, 2D correspondences of points $[p_i^1, p_i^2]$ between two frames and their relative rotation $R_{C_1C_2}$ are used

$$X_i^2 = \begin{bmatrix} x_i^2 \\ y_i^2 \\ z_i^2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} X_i^1 + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{5}$$

where X_i^j represents a 3D point in the jth camera. By eliminating the scale z_i^2, we obtain

$$\begin{bmatrix} \tilde{x}_i^2 \\ \tilde{y}_i^2 \\ 1 \end{bmatrix} = \begin{bmatrix} (r_1 \cdot X_i^1 + t_1)/(r_3 \cdot X_i^1 + t_3) \\ (r_2 \cdot X_i^1 + t_2)/(r_3 \cdot X_i^1 + t_3) \\ 1 \end{bmatrix} \tag{6}$$

where $[\tilde{x}_i^j\ \tilde{x}_i^j\ 1]^T$ represents the ith normalized 3D point in the jth camera frame. Since X_i^1 is also a 3D point, we need to eliminate z_i^1 and build

$$\begin{bmatrix} -\tilde{y}_i^1 t_3 + t_2 \\ \tilde{x}_i^1 t_3 - t_1 \\ -\tilde{x}_i^1 t_2 + \tilde{y}_i^1 t_1 \end{bmatrix}^T \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \begin{bmatrix} \tilde{x}_i^1 \\ \tilde{y}_i^1 \\ 1 \end{bmatrix} = 0 \tag{7}$$

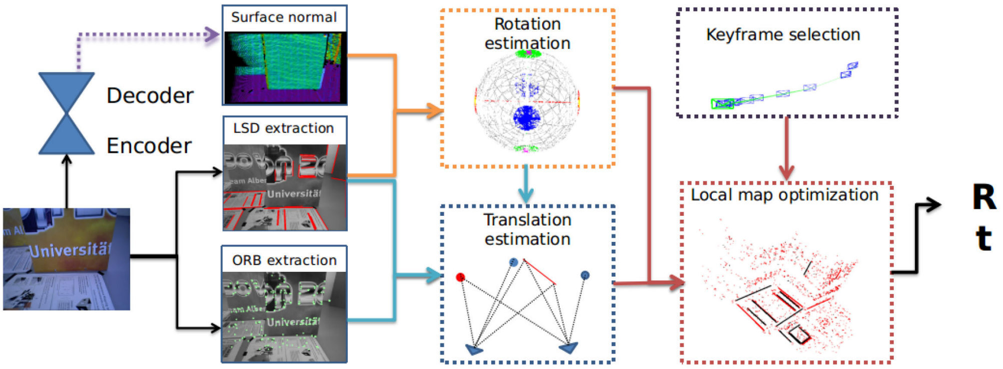


Fig. 2. Proposed SLAM framework (StructureSLAM). In the front-end, the encoder-decoder network predicts dense surface normals. In parallel, point and line features are extracted from the RGB image. In the back-end, first the scene structure in the form of normals and lines is used to estimate the global rotation of the camera. Then, the remaining 3-DoF for the translation are obtained using point and line features. The initial pose estimate is validated and refined using the local map. Keyframes are selected based on the availability of point and line features.

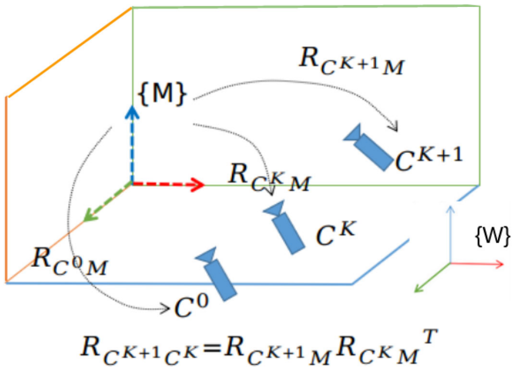


Fig. 3. Rotation estimation between multiple frames via the Manhattan world.

where $[\tilde{x}_i^j\ \tilde{x}_i^j\ 1]^T = K^T(u_i^j\ v_i^j\ 1)$ and K is the intrinsic matrix of the camera [12]. $(u_i^j\ v_i^j)$ is the ith pixel in the jth frame. Based on eq. (6) and eq.(7), we construct a translation relationship between those 2D correspondences. Then, we solve the system in eq. (7) using SVD to obtain the translation.

V. Tracking

Instead of estimating rotation and translation between two frames, we estimate the rotation between each frame and the underlying Manhattan World. The residual rotation errors are independent of the sequence length and cannot be propagated between frames. Point and line correspondences are used to estimate translation (3 DoFs) by a combination of frame-to-frame and frame-to-map methods.

A. Manhattan Rotation Estimation

This section describes the rotation estimation between camera and Manhattan system.

Given the surface normals and mask of planar regions from the network, we follow the mean-shift clustering approach, as desribed in IV, to find the dominant axes on the euclidean sphere and estimate the rotation $R_{C_K M}$. Since normal maps might contain errors due to the networks inference process, the clustering approach is used to remove outliers first. Furthermore, the initial rotation will be refined in following sections.

B. Translation Estimation

After obtaining the rotation matrix, we use the points and line segments to estimate the 3-DoF translational motion, which requires less features than the full 6-DoF estimation. We re-project the 3D points from the last frame to the current one and define the error function, based on the re-projection error, as follows,

$$e^p_{k,j} = p_k - \pi(R_{k,j}P_j + t_{k,j}) \tag{8}$$

here $\pi()$ is the projection function. Since the rotation matrix $R_{k,j}$ has already been estimated in the last step, we fix the rotation and only optimize the translation using the right half of the Jacobian matrix for eq. (8),

$$\frac{\partial e^p_{k,j}}{\partial \xi} = \begin{bmatrix} \frac{xyf_x}{z^2} & -\frac{z^2+x^2}{z^2}f_x & \frac{yf_x}{z} & -\frac{f_x}{z} & 0 & \frac{xf_x}{z^2} \\ \frac{z^2+y^2}{z^2}f_y & -\frac{xyf_y}{z^2} & -\frac{xf_y}{z} & 0 & -\frac{f_y}{z} & \frac{yf_y}{z^2} \end{bmatrix} \tag{9}$$

For the lines we obtain the normalized line function from the 2D endpoints p_{start} and p_{end} as follows,

$$l = \frac{p_{start} \times p_{end}}{\|p_{start}\|\|p_{end}\|} = (a, b, c) \tag{10}$$

We formulate the error function based on the point-to-line distance between l and the projected 3D endpoints P_{start} and P_{end} from the matched 3D line in the keyframe. For each endpoint P_x, the error function can be noted as,

$$e^l_{k,j} = l\pi(R_{k,j}P_x + t_{k,j}) \tag{11}$$

The Jacobian matrix for the line error eq. (11) is given by

$$\frac{\partial e^l_{k,j}}{\partial \xi} = \begin{bmatrix} -\frac{f_y l_y z^2+f_x l_x xy+f_y l_y y^2}{z^2}, & \frac{f_x l_x z^2+f_x l_x x^2+f_y l_y xy}{z^2}, \\ -\frac{f_x l_x y-f_y l_y x}{z}, & \frac{f_x l_x}{z}, \\ \frac{f_y l_y}{z}, & -\frac{f_x l_x x+f_y l_y y}{z^2} \end{bmatrix} \tag{12}$$

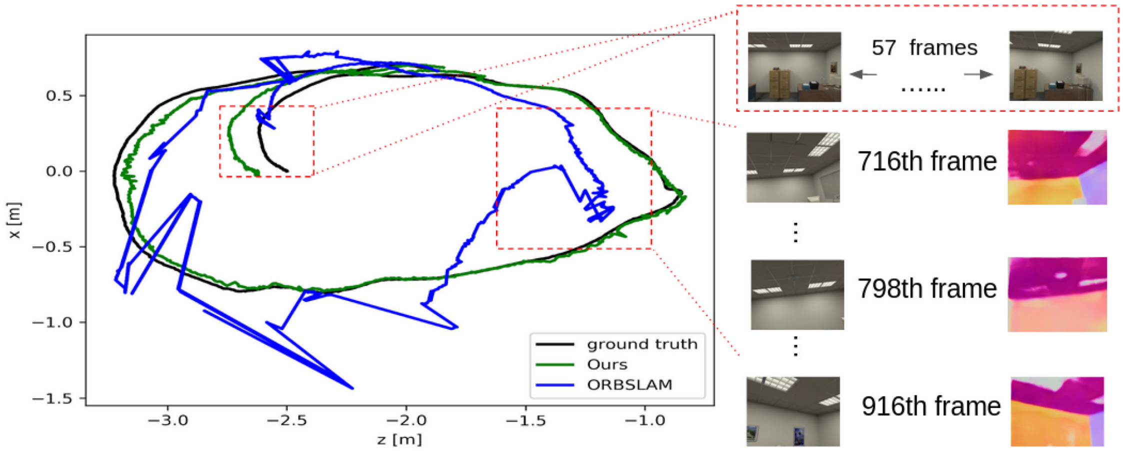


Fig. 4. Trajectory analysis, comparing the proposed method, ORB-SLAM and the ground-truth on the "of-k3" sequence in the ICL NUIM dataset.

The combined least squares cost for points and lines can be written as

$$t* = argmin \sum_{j\in(k-2,k-1)}^{M} ({e^{p}_{k,j}}^{T} e^{p}_{k,j} + {e^{l}_{k,P_x}}^{T} e^{l}_{k,P_x}) \quad (13)$$

The system is solved using the Levenberg-Marquardt algorithm.

C. Fallback and Pose Refinement

The pose estimate is based on the MW assumption. In cases of Non-Manhattan Worlds or where the Manhattan Frame is not visible in the current frame the estimated pose will be incorrect. To check whether the pose estimate obtained from the previous steps is correct we project all features from the last n keyframes onto the current frame and compute the re-projection error. By applying a threshold to filter the features we require a minimum number of inliers to accept the pose.

When not enough inliers are found we fall back to a frame-to-frame tracking method until we estimate a pose that agrees again with the Manhattan World. As a fallback we first track the new frame based on the last frame using an efficient re-projection search scheme [30] for points and lines, using the same least squares method as for the translation, this time using the full Jacobian matrix. In the case we do not get a good solution, measured based on the number of inliers, we try to estimate the pose based on the last keyframe using descriptor matching for the points [30] and re-projection based search for the lines. To reduce the drift, in the final step we optimize the pose of the new frame based on a local map constructed from the last n keyframes [30]. Here we do not use the MW assumption anymore, as we found that the initial rotation estimation is enough to reduce the drift and errors in the predicted normal maps can lead to inconsistent pose estimates.

In contrast to other work, based on Manhattan frames for rotation estimation this heuristic allows us to fall back to a purely feature based pose estimation in case the estimate from the MW pose estimation is wrong or not available.

VI. Experiments

Implementation details: We train the network implemented for normal estimation based on the ScanNet [31] dataset with a batch size of 32 for 8 epochs. The backbone is pretraind on ImageNet [32] for feature extraction and PlaneReconstruction [14] for understanding plane regions. We use the Adam optimizer with a learning rate of 10^{-4} and a weight decay of 10^{-5}. Our model is trained in an end-to-end manner and can predict normal maps in real-time. As a baseline we use the original GeoNet [15] model trained by the authors for 400 k iterations on NYU-DepthV2 [33]. Models used in the experiments are not fine-tuned on other datasets. All experiments were carried out with an Intel Core i7-8700 CPU (with @3.20 GHz) and a NVIDIA 2080 Ti GPU. We run each sequence 5 times and show median results for the accuracy of the estimated trajectory. We evaluate our proposed SLAM system on public datasets and compare its performances with other state-of-the-art methods. The evaluation metrics used in the experiments are the absolute trajectory error (ATE) and the relative pose error (RPE) [17], which measure the absolute and relative pose differences between the estimated and the ground truth motion.

Evaluation and datasets: In order to evaluate our method, on the one hand, we compare against several monocular SLAM frameworks, as CNN-SLAM [19] that connects SLAM with predicted depth maps based on keyframes, LSD-SLAM [2] that is popular direct method and ORB-SLAM [1]. We align the trajectories for ORB-SLAM, LSD and the proposed method to the ground truth trajectories using a similarity transformation [1] due to the unknown real scale. On the other hand, we run our SLAM architecture with different normal maps to evaluate the importance of accurate normals, by switching our normals with the ones from the state-of-the-art, but not real-time capable network, GeoNet [15] and normal maps computed from the depth maps provided by the dataset using [34].

- ICL-NUIM dataset [16] is a synthetic indoor datasets that provide RGB images, depth maps and ground-truth camera poses. There are two scenes, named "living room" and "office" which are noted as "lr" and "of" in our experiments.

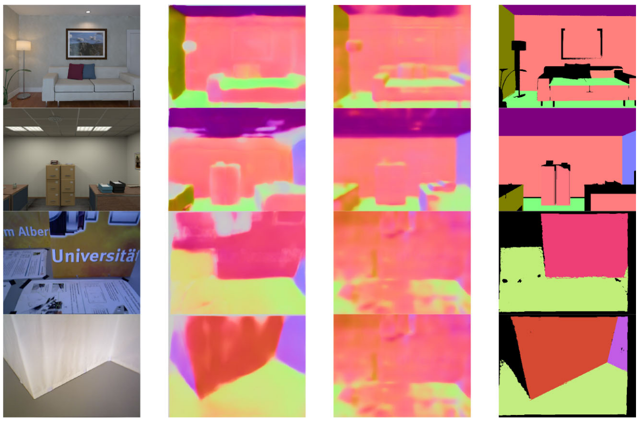

(a) Input (b) Ours (c) GeoNet [15] (d) AHC [34]

Fig. 5. Results of normal prediction model on ICL-NUIM (top) and TUM-RGBD (bottom) scenes for different approaches.

TABLE I
PERFORMANCE OF THE SURFACE NORMAL PREDICTION ON THE SCANNET [31] TEST SET

Error			Accuracy		
mean	median	rmse	<11.25°	<22.5°	<30°
15.2°	7.8°	24.4°	0.672	0.797	0.841

- TUM RGB-D dataset [17] was collected using a real RGB-D sensor in real scenes as well as specially designed scenes to challenge current SLAM algorithms, featuring challenging scenes with good structure, but without texture.
- HRBB4 dataset [35] which has 12,000 frames of 640 × 320 pixels recorded by a monocular camera in a corridor.

A. Normal Prediction

Fig. 5 presents qualitative results on unseen images of different normal estimation methods. In our method, we mask out the lampshade (first row) and small boxes (second row), as these regions are classified as non-planar. The first two rows, show common examples for indoor environments. Both of them show good results, GeoNet shows smaller inaccuracies. For the last two rows, which are very uncommon scenes, the planar region detection and normal estimation of our model are still generating reasonable results, while the quality of the normal predictions from GeoNet decreased severely.

The agglomerative hierarchical clustering (AHC) algorithm [34] is an efficient method to detect planes in a depth map. However it difficult to detect planes (like in the third an forth row) where the quality of the depth maps decrease due to a highly slanted surface. In the Table I, the performance of the network is evaluated on the ScanNet [31] dataset generated by [36] against the ground truth.

B. Pose Estimation

In order to evaluate our method in different environments, we select structured image sequences from the ICL-NUIM dataset [16] and the TUM RGB-D dataset [17]. Table II shows the RMSE for all methods on several sequences, 'lr' and 'of' stand for the living room and office room sequences in the ICL-NUIM dataset. 's-t-near' and 's-not-near' are the structure-texture-near and structure-notexture-near sequences in the TUM RGB-D dataset, respectively. 's-t-near' and 's-t-far' are showing the same environment consisting of multiple textured planes, 's-not-near' and 's-not-far' consist of a similar structure, but without texture.

From the six row to the eight row, different normal maps are given to the same backbone. It is obvious that using AHC-based normal maps (obtained from ground truth depth map) obtain the best results compared to other methods. It also shows the potential of our SLAM architecture, given precise normal maps. Performances from $-w$ Ours (combination of our normal prediction network and the backbone) is more robust than $-w$ GeoNet (combination of GeoNet and the backbone), especially in the 's-not-far' sequence. For those non-textured images, it is difficult for GeoNet to predict accurate normals. In the backbone, conic areas around each axis are used during the sphere mean-shift method to filter the normal maps, this allows the handling of normal outliers up to a certain point. In cases were the number of outliers is too high, it is difficult to obtain a good rotation from the back-end of the architecture. Different to the monocular methods, LPVO [12] works directly with RGB-D images, which prevents scale drift and allows tracking directly on the depthmap. In comparison our method achieves comparable performance without the use of a depth sensor.

Our method obtains good results and shows robust performance in all five sequences. In the first two sequences, the difference between the point based ORB-SLAM and our method, that connects structure and geometric information, is not significant. However ORB-SLAM is not able to find enough point matches over a sequence of frames and looses tracking in some of the sequences, these are marked with a cross (×). Our method, which additionally uses lines for the translation estimation achieves even better results.

When we compare -w Ours, -w GeoNet with ORB-SLAM in textured sequences, they obtain similar results because those sequences have a sufficient number of features distributed evenly on each frame. However for indoor environments, like Fig. 4, it is difficult to obtain enough point features because of large non-textured planar regions. In the 'of-kt3' sequence, there is little change in the first 57 frames, so ORB-SLAM cannot initialize successfully, because it needs enough points for homography/fundamental model selection. After initialization, it is also challenging for ORB-SLAM to track via the point-based motion model. For our case, the initial rotation matrix is estimated by the mean-shift method instead of estimating the essential or homography matrices. This means we can deal with pure rotational motion. Furthermore, points and line segments are used for 3 DoFs translation only, which is more robust even in large non-textured scene.

In order to present the robustness of our method, we compute the RPE for those sequences, which can be processed robustly by ORB-SLAM and our method. For 's-t-far' and 's-t-near' that are textured sequences, ORB-SLAM and the proposed method have similar performences. The relative translation errors for

TABLE II
COMPARISON OF TRANSLATION RMSE (M) FOR ICL-NUIM [16] AND TUM RGB-D [17] SEQUENCES USING MONOCULAR CAMERA. WE USE **BOLD** NUMBERS TO MARK THE BEST RESULT PER SEQUENCE. $-w$ MEANS THAT THE PROPOSED FRAMEWORK USES THE CORRESPONDING SURFACE NORMALS. × INDICATES THAT THE ALGORITHM FAILS DUE TO LOST TRACKING

Methods	lr-kt1	lr-kt2	lr-kt3	of-kt1	of-kt2	of-kt3	s-t-near	s-t-far	s-not-near	s-not-far
LSD-SLAM [2]	0.059	0.323	-	**0.157**	0.213	-	-	0.214	-	-
CNN-SLAM [19]	0.540	0.211	-	0.790	0.172	-	-	0.037	-	-
ORB-SLAM [1]	0.024	0.061	0.035	×	**0.031**	0.326	0.016	0.015	×	×
LPVO [12]	*0.04*	*0.03*	*0.10*	*0.05*	*0.04*	*0.03*	*0.11*	*0.17*	*0.08*	*0.07*
-*w* Ours	**0.016**	**0.045**	0.046	×	**0.031**	0.065	**0.014**	**0.014**	**0.065**	**0.281**
-*w* GeoNet [15]	×	0.047	**0.026**	×	0.048	**0.043**	0.107	**0.014**	0.068	×
-*w* AHC [34]	*0.016*	*0.028*	*0.020*	*0.763*	*0.021*	*0.020*	*0.015*	*0.013*	*0.015*	*0.220*

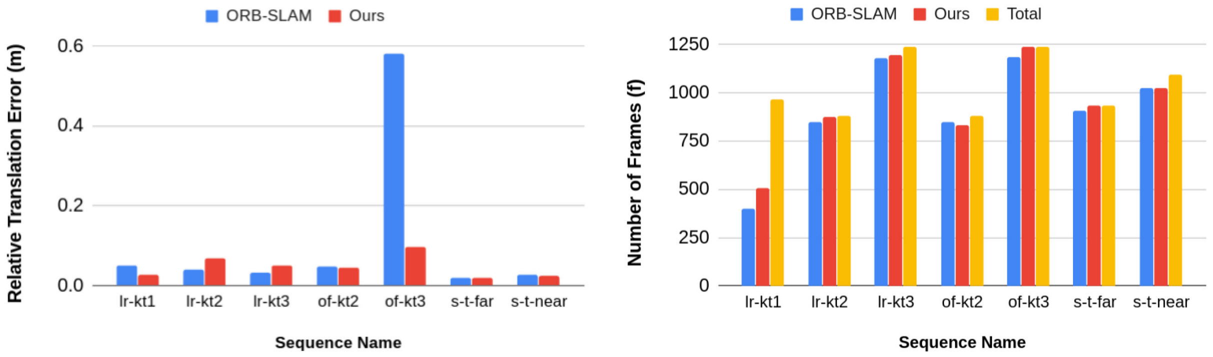


Fig. 6. Relative translational error comparison between ORB-SLAM and our method on different sequences (left) and a comparison of the average runtime length on each sequence before tracking is lost (right).

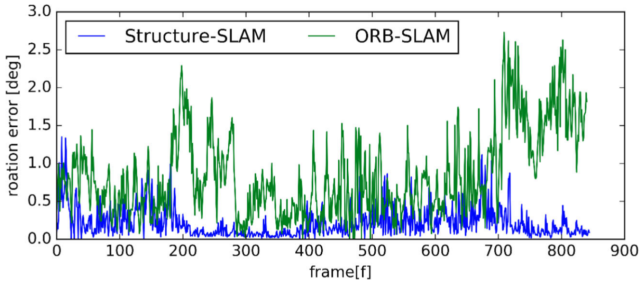


Fig. 7. rotation error comparison between ORB-SLAM and our method on sequence lr-k2.

the sequence ‘of-kt3’ in Fig. 6 (left) is significantly larger for ORB-SLAM, which corresponds to the result presented in Fig. 4. As shown in Fig. 7, the proposed method, Structure-SLAM, is more stable in rotation estimation compared with ORB-SLAM.

We also compare the number of frames tracked by different methods. Compared with ORB-SLAM, our method retrieves the camera pose more reliable. Especially in ‘lr-kt2,’ ‘of-kt3’ and ‘s-t-far,’ our method initializes fast and tracks all frames in the sequences, as can be seen in sequence ‘of-kt3’ in Fig. 6 on the right. Similar results can be found for HRBB4 in Fig. 8. Compared with ORB-SLAM which only initializes after the 628th frame, our method is able to initialization much earlier around frame 110. Furthermore, the proposed method shows a more stable behaviour in the upper right corner of the corridor where the environment changes drastically.

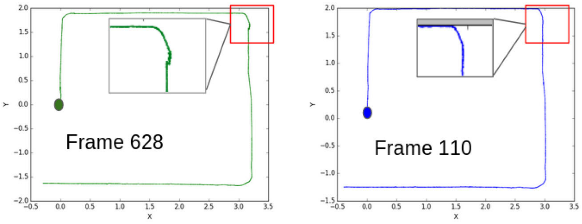


Fig. 8. The estimated trajectories of the camera on the HRBB4 [35] dataset. Left: ORB-SLAM, Right: Structure-SLAM.

VII. CONCLUSION

We have proposed a SLAM system for monocular cameras based on points, lines and surface normals. Using the Manhattan World assumption for rotation estimation and point and line features for windowed translation estimation we achieve state-of-the-art performance. We have shown that normals, learned from a single RGB image, can be used to estimate the rotation between frames leveraging the MW assumption. Compared to other state-of-the-art methods based on global rotation estimation, in our method there exists a fallback level using points and lines to estimate the full pose, in case no Manhattan frame can be found. This enables the tracking over short sequences to later re-localize within the Manhattan world. In the future, global bundle adjustment could be used to correct the frames during these sequences without global frames. Furthermore, we would like to leverage the learned structure information for the translation estimation as well.

References

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

[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Springer Eur. Conf. Comput. Vision*, 2014, pp. 834–849.

[3] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[4] R. Gomez-Ojeda, F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A stereo SLAM system through the combination of points and line segments," *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 734–746, Jun. 2019.

[5] Y. Lu and D. Song, "Robust RGB-D odometry using point and line features," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 3934–3942.

[6] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Morenonoguer, "PL-SLAM: Real-time monocular visual SLAM with points and lines," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4503–4508.

[7] P. F. Proença and Y. Gao, "Probabilistic RGB-D odometry based on points, lines and planes under depth uncertainty," *Robot. Auton. Syst.*, vol. 104, pp. 25–39, 2018.

[8] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," *Robot.: Sci. Syst. VI*, vol. 2, no. 3, pp. 73–80, 2010.

[9] H. Li, J. Yao, J.-C. Bazin, X. Lu, Y. Xing, and K. Liu, "A monocular SLAM system leveraging structural regularity in manhattan world," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2518–2525.

[10] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, "StructSLAM: Visual SLAM with building structure lines," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1364–1375, Apr. 2015.

[11] J. Straub, N. Bhandari, J. J. Leonard, and J. W. Fisher, "Real-time manhattan world rotation estimation in 3D," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2015, pp. 1913–1920.

[12] P. Kim, B. Coltin, and H. J. Kim, "Low-drift visual odometry in structured environments by decoupling rotational and translational motion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7247–7253.

[13] P. Kim, B. Coltin, and H. J. Kim, "Linear RGB-D SLAM for planar environments," in *Proc. Springer Eur. Conf. Comput. Vision*, 2018, pp. 333–348.

[14] Z. Yu, J. Zheng, D. Lian, Z. Zhou, and S. Gao, "Single-image piece-wise planar 3D reconstruction via associative embedding," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 1029–1037.

[15] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia, "GeoNet: Geometric neural network for joint depth and surface normal estimation," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 283–291.

[16] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 1524–1531.

[17] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.

[18] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2009, pp. 83–86.

[19] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 6565–6574.

[20] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "CodeSLAM–learning a compact, optimisable representation for dense visual SLAM," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 2560–2568.

[21] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1285–1291.

[22] M. Hsiao, E. Westman, G. Zhang, and M. Kaess, "Keyframe-based dense planar SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5110–5117.

[23] Y. Zhou, L. Kneip, C. Rodriguez, and H. Li, "Divide and conquer: Efficient density-based tracking of 3D sensors in Manhattan worlds," in *Proc. Asian Conf. Comput. Vision*, 2016, pp. 3–9.

[24] R. Toldo and A. Fusiello, "Robust multiple structures estimation with J-linkage," in *Proc. Springer Eur. Conf. Comput. Vision*, 2008, pp. 537–547.

[25] K. Joo, T.-H. Oh, J. Kim, and I. S. Kweon, "Globally optimal manhattan frame estimation in real-time," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 1763–1771.

[26] P. Kim, B. Coltin, and H. J. Kim, "Visual odometry with drift-free rotation estimation using indoor scene regularities," in *Proc. Brit. Mach. Vision Conf.*, 2017, pp. 7–19.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vision*, 2011, pp. 2564–2571.

[28] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, Apr. 2010.

[29] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *J. Vis. Commun. Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[30] R. Murartal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[31] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 2432–2443.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2009, pp. 248–255.

[33] P. K. N. Silberman, D. Hoiem, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Springer Eur. Conf. Comput. Vision*, 2012, pp. 746–760.

[34] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 6218–6225.

[35] Y. Lu, D. Song, and J. Yi, "High level landmark-based visual navigation using unsupervised geometric constraints in local bundle adjustment," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 1540–1545.

[36] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa, "PlaneNet: Piece-wise planar reconstruction from a single RGB image," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 2579–2588.

CCC RightsLink

Sign in/Register

IEEE Requesting permission to reuse content from an IEEE publication

Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments

Author: Yanyan Li
Publication: IEEE Robotics and Automation Letters
Publisher: IEEE
Date: October 2020

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

RGB-D SLAM with Structural Regularities

Yanyan Li[1], Raza Yunus[1], Nikolas Brasch[1], Nassir Navab[1,2] and Federico Tombari[1,3]

[1] Technical University of Munich
[2] Johns Hopkins University
[3] Google

Title	RGB-D SLAM with Structural Regularities
Version	Published Version
Paper document	Attached
License document	Attached

2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561560

RGB-D SLAM with Structural Regularities

Yanyan Li[1], Raza Yunus[1], Nikolas Brasch[1], Nassir Navab[1,2] and Federico Tombari[1,3]

***Abstract*— This work proposes a RGB-D SLAM system specifically designed for structured environments and aimed at improved tracking and mapping accuracy by relying on geometric features that are extracted from the surrounding. Structured environments offer, in addition to points, also an abundance of geometrical features such as lines and planes, which we exploit to design both the tracking and mapping components of our SLAM system. For the tracking part, we explore geometric relationships between these features based on the assumption of a Manhattan World (MW). We propose a decoupling-refinement method based on points, lines, and planes, as well as the use of Manhattan relationships in an additional pose refinement module. For the mapping part, different levels of maps from sparse to dense are reconstructed at a low computational cost. We propose an instance-wise meshing strategy to build a dense map by meshing plane instances independently. The overall performance in terms of pose estimation and reconstruction is evaluated on public benchmarks and shows improved performance compared to state-of-the-art methods. The code is released at `https://github.com/yanyan-li/PlanarSLAM`.**

I. Introduction

Visual Simultaneous Localization and Mapping (SLAM) algorithms are used to estimate the 6D camera pose while reconstructing the surrounding unknown environment. They have shown to be useful in a wide range of applications, such as autonomous robots, self-driving cars and augmented/virtual reality, where camera pose estimation enables cars, robots and mobile devices to localize themselves, while the dense map provides a representation of the environment, *e.g.* for robot-environment or human-environment interaction.

Many SLAM applications have to deal with structured scenes, *i.e.* man-made environments that are usually characterized by low-textured surfaces - a typical example is an indoor scene, or an outdoor parking place. This induces a lack of visual features, that visual SLAM systems typically leverage to improve camera pose estimation and/or 3D reconstruction, *e.g.* by carrying out loop closure and bundle adjustment to reduce drift. In order to deal with structured scenes, specific SLAM methods based on points and line segments, like S-SLAM [1], Stereo-PLSLAM [2], PLVO [3], Mono-PLSLAM [4] and Probabilistic-VO [5] have been proposed, extending the working environment to scenes where more lines than points can be detected. SP-SLAM [6] merges plane features into ORB-SLAM2 [7], achieving robust results in low-textured scenes.

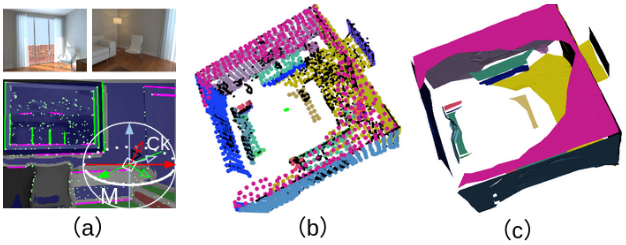


Fig. 1. RGB-D SLAM system. (a) Examples of a typical structured scene, and 2D features and orthogonal lines and planes segmentation. (b) Point cloud including points, lines and planes. (c) Real-time mesh on a CPU.

For the reconstruction, there are sparse, semi-dense and dense methods. Compared to the first two classes, which only provide incomplete maps, dense reconstruction is required to provide sufficient information for applications such as robot-environment interaction and 3D scene understanding. Many algorithms have been proposed to reconstruct indoor scenes via RGB-D sensors. KinectFusion [8] is a pioneering work relying on the truncated signed distance field (TSDF) representation of the map. In order to reconstruct large scale scenarios, surfel-based methods, like ElasticFusion [9], were proposed. Instead of reconstructing each pixel, Wang et al. [10] extracts superpixels from RGB images and depth maps, which is more efficient but still has redundant information especially in indoor scenarios where large planes can be commonly found.

In this paper, we build on our monocular Structure-SLAM [1] and propose a robust RGB-D SLAM system specifically designed to deal with structured environments, which improves tracking and mapping at the same time. Figure 1 illustrates the components of such structured scenes, which contains points, lines and plane segments. Following the decoupling strategy of Structure-SLAM, we estimate a drift-free rotation matrix first, and then the 3-DoF translation. The initial rotation and translation are refined via a map-to-frame strategy. Different to [1], [11], [12], plane features are merged into our Manhattan-based framework, which is used to estimation the initial translation vector and retain Manhattan relationships as constrains in the refinement module. Furthermore, an efficient meshing module is proposed that reconstructs the scene structure based on the obtained planar regions in the sparse map. In summary our contributions are:

- Based on the concept of MW-based decoupled pose estimation, we improve the translation estimation by combining point and line features with planes and an additional pose refinement step with Manhattan relationships.

[1]:Technical University of Munich, Germany; {`yanyan.li, raza.yunus, nikolas.brasch, nassir.navab, federico.tombari`}`@tum.de`; [2]:Johns Hopkins University, USA;[3]:Google Inc.

- We propose a planar instance-wise mesh based reconstruction method generating a compact representation of the environment from a sparse point cloud.
- A general framework for real-time RGB-D SLAM where these components are used to localize and map under structured environments with high accuracy.

We evaluate the performance of our approach in terms of both camera pose estimation and reconstruction on public benchmarks, showing improved performance compared to state-of-the-art methods.

II. Related work

In the following we review the literature related to RGB-D based SLAM systems as well as methods leveraging structural regularity as the MW assumption.

a) RGB-D SLAM.: In [13], [14] it was proposed to use planes over point features whenever possible, as the averaging over multiple depth measurements reduces the noise significantly. In Dense Planar SLAM [15] surfels belonging to the same planar areas are smoothed by fitting a plane to them and back-projecting the surfels onto the plane. Le *et al.* [16] rely on a scene layout consisting of a ground plane and several walls, and use dynamic programming to infer a sequentially consistent assignment of pixels to planes. In Probabilistic-VO [5], the uncertainties of points, lines and planes are modelled explicitly and used during pose estimation, where points, lines and planes are represented in a uniform framework in [17]. A direct SLAM system combining photo-metric and geometric terms is proposed in DVO-SLAM[18] and extended in CPA-SLAM [19] with global planes, where depth measurements are assigned to the global planes with weights.

b) Dense Reconstruction.: While the aforementioned methods have the goal to estimate precise poses and therefore only maintain a map with the most reliable information, several works have been proposed with the goal to create a complete dense reconstruction of the environment. KinetFusion [8] and ElasticFusion [9] explore dense reconstruction for RGB-D sensors. The first method fuses all depth data into a volumetric dense representation, which is used to track the camera pose using ICP. The size of the map is usually limited in volumetric methods due to memory constraints. Different from KinectFusion, ElasticFusion is a map-centric system that reconstructs surfel-based maps of the environment. In order to decrease the number of surfels in the map, superpixel-based surfels are proposed by [10], which reduces the number of surfels compared with ElasticFusion. Recently BAD-SLAM [20] proposed a direct bundle-adjustment approach for RGB-D SLAM. In [21] a textured mesh is extracted from a dense surfel cloud. A direct mesh based reconstruction approach for RGB-D sensors was proposed in [22].

c) Structural Regularity.: A line of works exploits additional constrains and regularities in the world, to improve the reconstruction performance. In [23] and [24] the authors showed that the rotation estimation error is the main reason for long-term drift.

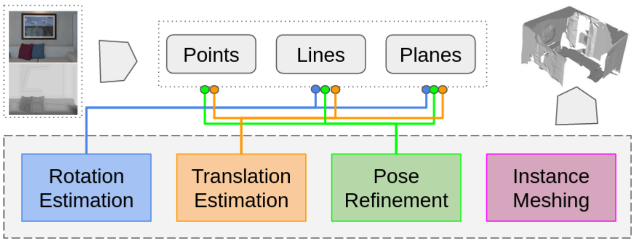


Fig. 2. Overview of the proposed framework. Point, line and plane features are extracted from the RGB-D frame. Rotation and translation are estimated in a decoupled fashion first and refined afterwards. The planar segments are used to create a mesh-based reconstruction of the environment.

A branch-and-bound framework for Manhattan Frame estimation is proposed in [25]. In MVO [24] a method using mean shift on the unit sphere is used to find the transformation between the MW and the current frame. When only planes are used for the rotation estimation as in OPVO [26] at least 2 orthogonal planes must be detected in each frame, the addition of vanishing points extracted from lines can be used alternatively, as done in LPVO [11]. The methods mentioned above use point features to estimate the translation. Structure-SLAM [1] is a monocular system that predicts normals via a convolutional neural network leverages normals with points and lines in a decoupling strategy. Since predicted normals are not as accurate as those computed from a depth map, the system provide a refinement/fallback module based on points and lines. Compared with Structure-SLAM, optimized vanishing points of lines and plane features are used for rotation and translation in this work. Then, the fallback part is removed and the refinement part incorporates geometric relationship of planes. Instead of the sparse point-line map, a dense mesh as output is more useful for robotics applications. L-SLAM [12] is also based on the MW assumption, which obtains translation, rotation and pixels of potential planar regions from LPVO. Then it refines 3D translational and 1-D plane offsets with a linear Kalman Filter. However, we use a more robust front-end for initial translation estimation. Furthermore, the 6D pose refinement step is used to optimize rotation and translation simultaneously and allows an offset to the initial rotation from MW, which is more robust to non-MW (curved surfaces and few planar regions) compared with L-SLAM and LPVO (see Figure 6).

III. Proposed framework

Given a sequence of RGB-D frames from a structured environment, the goal of our method is to reconstruct the 3D scene while simultaneously estimating the 6D camera pose at each frame. Section IV provides an overview of the proposed tracking pipeline, which decouples rotation and translation, while section V describes different types of mapping presentations generated by the system. We now describe the system's underlying features and structural components.

A. Extended feature set

In our method we use ORB features [27], which are fast to extract and match. In low-textured environments, it is hard to extract sufficient points for robust pose estimation, therefore we extend the feature set with lines, which are extracted using the LSD [28] approach, as described by LBD [29]. Furthermore, it is common to find non-textured planar regions in indoor environments, where plane instances extracted from the depth maps are valuable cues to extend points and lines. Planes are detected using the connected component analysis method [30]. They are represented by the Hessian normal form $\pi = (\hat{n}, d)$, where $\hat{n} = (n_x, n_y, n_z)$ is the normal of the plane, representing its orientation and d is the distance from the camera origin to the plane.

a) Points and lines: After the extraction of 2D point features $x_j = (u_j, v_j)$ and line segments $l_j = (x_{j,start}, x_{j,end})$ in frame F_i, we can back-project points and lines using the camera intrinsic parameters and the depth map to obtain 3D points X_j and 3D lines L_j. The depth map is not always correct, especially at depth discontinuities *e.g.* object boundaries. Therefore a robust fitting method for 3D lines is needed. First, we count the number of pixels with non-zero depth values intersected by the detected line segment. If the number exceeds a certain threshold, the 3D line L_j will be estimated via RANSAC to remove potential outliers.

b) Normals and planes.: Smooth normals are computed by averaging the tangential vectors from the depth image inside a patch of 10×10 pixels using integral images. After plane detection, we use the strategy of [6] to associate the observed planes with those present in the map. To match an observed plane with one from the map, we first check the angle between their normals. If it is below the threshold θ_n, we check the point-to-plane distance between them. The plane which has the minimum distance to the observed plane, and also lies below the distance threshold θ_P, is matched to the observed plane. In the experiments, θ_n and θ_P are set at 10 degrees and 0.1 m respectively. Furthermore, we also keep parallel and perpendicular relationships [6] between the map planes to leverage additional constraints during the tracking process. These are determined by the angle between the plane normals. Since they only provide constraints for orientation, we do not consider their distance.

B. Decoupling pose estimation and refinement.

To reduce error propagation between frames, we build on our monocular architecture [1] that computes rotational motion based on the MW assumption. Then the corresponding translational motion is estimated by features, with the fixed rotation computed from last step. In the font-end of this work, we use optimized lines for rotation estimation and planes for translation estimation.

Differently to Structure-SLAM [1] that uses a point-line local map to optimize translation and rotation together, we leverage planes in the local map and also make use of the geometric relationship (parallel and perpendicular) of those planes as constrains, which improves the accuracy of the system as it will be shown in Figure 4 and Table I.

IV. Tracking

Differently from traditional pose estimation methods, we decouple the 6D camera pose into rotation and translation. Based on the MW assumption, we obtain the rotational motion $R_{c_i m}$ between the MW and camera c_i. In this way, the rotation estimation will not be affected by the pose of the last frame or last keyframe, which reduces drift effectively. Afterwards, point, line and plane features as well as the initial rotation matrix are used for translation estimation, which consists of just 3 Degrees-of-Freedom (DoFs).

A. Rotation estimation

Instead of tracking the camera from frame-to-frame directly, the drift-free rotation estimation method estimates the rotation R_{cm} between each frame and the Manhattan coordinate frame, by modeling the indoor environments as a MW, thus reducing the drift generated from frame-to-frame tracking. As shown in Figure 1, Manhattan coordinate frames can be aligned to the starting frame of the camera via $R_{k+1,m}$. Generally, the coordinate of the first frame is regarded as the world frame, *i.e.* $R_{1,m} = R^T_{m,w}$. So we can obtain pose in the world coordinate by using,

$$R_{k+1,w} = R_{k+1,m} R_{m,w} \tag{1}$$

Here $R_{m,w}$ represents the relation from the world to MW, which is obtained by the MW initialization step and $R_{k+1,m}$ is the relation from MW to the $(k+1)^t h$ frame. These two matrices are computed via a sphere mean-shift method [24], where the normals and normalized vanishing directions are projected onto the tangent planes of the current rotation estimate. Then a mean shift step is performed on the tangent planes, which generates new centers and back-projects them to the sphere as new estimates. We refer the reader to [24] and [26] for more details on the sphere mean-shift method. To handle difficult scenes where only one or no plane at all is detected, we feed the unit sphere with both vanishing directions of the refined 3D lines and surface normals of planes, which is a more robust approach than [26], [1] under these challenging conditions.

B. Translation estimation

After estimating the rotation, points, lines and planes are used to estimate the translation. We re-project 3D points from the last frame into the current one and define the error function, based on the re-projection error, as follows,

$$e^p_{k,j} = p_k - \Pi(R_{k,j} P_j + t_{k,j}) \tag{2}$$

where $\Pi(\cdot)$ is the projection function. Since the rotation matrix $R_{k,j}$ has been obtained in the last step, we fix the rotation and only estimate the translation using the Jacobian matrix corresponding to (2).

As for lines, we obtain the normalized line function from the 2D endpoints p_{start} and p_{end} as follows

$$l = [p_{start} \times p_{end}] / [\|p_{start}\| \|p_{end}\|] = (a, b, c). \tag{3}$$

Then, we formulate the error function based on the point-to-line distance [4] between l and the projected 3D endpoints P_{start} and P_{end} from the matched 3D line in the keyframe. For each endpoint P_x, the error function can be noted as,

$$e^l_{k,P_x} = l\Pi(R_{k,j} P_x + t_{k,j}). \tag{4}$$

To get a minimal parameterization of a plane π for optimization, we represent it as $q(\pi) = (\phi, \psi, d)$ where ϕ and ψ are the azimuth and elevation angles of the normal and d is the distance from the Hessian form

$$q(\pi) = (\phi = arctan(\frac{n_y}{n_x}), \psi = arcsin(n_z), d). \tag{5}$$

So, the error function between the observed plane π_k in the frame and corresponding map plane π_x is

$$e^{\pi}_{k,\pi_x} = q(\pi_k) - q(T_{cw}^{-T}\pi_x) \tag{6}$$

where T_{cw}^{-T} is the transformation from world to camera coordinates.Assuming that the observations follow a Gaussian distribution, the final non-linear least squares cost function $t*$ can be written as in (7), where $\Lambda_{p_{k,j}}$, $\Lambda_{p_{k,P_x}}$ and Λ_{k,π_x} are the inverse covariance matrices of points, lines and planes, and ρ_p, ρ_l and ρ_π are robust Huber cost functions, respectively.

$$\begin{aligned} t* =& argmin \sum_{j}^{M} \rho_p \left({e^p_{k,j}}^T \Lambda_{p_{k,j}} e^p_{k,j} \right) \\ &+ \rho_l \left({e^l_{k,P_x}}^T \Lambda_{p_{k,P_x}} e^l_{k,P_x} \right) + \rho_\pi \left({e^\pi_{k,\pi_x}}^T \Lambda_{k,\pi_x} e^\pi_{k,\pi_x} \right) \end{aligned} \tag{7}$$

Here, a solution is determined using the Levenberg-Marquardt algorithm.

C. Pose refinement

The last two steps assume that the scene is a good Manhattan model, nevertheless several general indoor environments are not strictly adhering to the MW assumption, leading to degradation in accuracy. So, after obtaining the initial pose via the decoupled rotation and translation strategy, the refinement module [1] fine-tunes the pose to compensate for deviations from the MW or unstable initial estimates. In the refinement step, to reduce the drift from frame-to-frame pose estimation, the local map constructed by previous keyframes is used to optimize the pose based on a map-to-frame strategy [7].

Similar to [6], [7], [31], we also use keyframes to build a local map, although our map has point, line and plane landmarks, which are projected into the current frame to search for matches. Furthermore, we explore the relationship between planes in the local map and planes detected in the current frame. The parallel and perpendicular constraints between those planes are described as (8),

$$\begin{cases} e^{\pi_\parallel}_{k,n_x} = & ||q_n(n_k) - q_n(R_{cw}n_x)|| \\ e^{\pi_\perp}_{k,n_x} = & ||q_n(R_\perp n_k) - q_n(R_{cw}n_x)|| \end{cases} \tag{8}$$

where $q_n(\pi) = (\phi, \psi)$ and R_{cw} is the transformation from world to camera coordinates. For perpendicular planes, their plane normal is rotated by 90 degrees ($R_\perp$) to construct the error function. These two error functions are merged to (7) to build a joint optimization function in the refinement module.

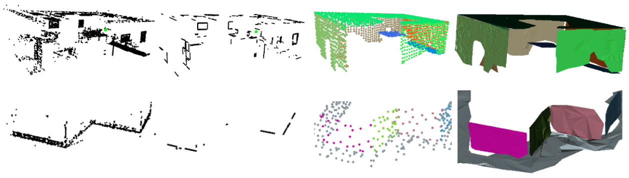


Fig. 3. Different levels of maps provided by the system.Top row: office room of the ICL-NUIM; bottom row: structure-nontexture-near of TUM RGB-D;

V. Mapping

This section describes the keyframe-based 3D mapping strategy used in our SLAM framework. Keyframes and 3D features build up a co-visibility graph, where nodes and edges are updated whenever a new keyframe and new features are available.

A. Sparse Mapping

As shown in Figure 3, the sparse map module is reconstructed by point-line-plane features extracted from keyframes. The first frame is set as the first keyframe and the global map is initialized by the landmarks thereby detected. When new points, lines and planes are detected in a new keyframe, which are not in the global map, they will be saved to a local map first. Then we check the quality of the landmarks in the local map, and then push reliable landmarks into a global map after culling bad ones. Different to the matching methods for points and lines, for each detected plane in a new keyframe, we first check whether it is associated with a plane in the map using the strategy described in section III. If we find an association, we add the 3D points of the new plane to the associated plane in the global map and filter out redundancies using a voxel grid to get a compact point cloud again. If the incoming plane is not associated to any plane in the global map, we add it to the map as a new plane.

B. Planar instance-wise meshing

The sparse map obtained in the previous section is still not adequate for applications involving robot-environment interactions, but it provides information about planar and non-planar instances. Therefore, we construct a denser map using an instance-wise meshing strategy. Indoor scenes can be divided into planar and non-planar regions. Planar areas like floors, walls and ceiling have often a large extent, however a dense pixel-wise information does not add to the quality and is highly redundant. So instead of using surfel or TSDF, we regard plane regions as instances that include a small and fixed number of elements independently of their size.

In particular, we input plane instances to the meshing module, which meshes them independently. First, the points belonging to a plane are organized as a kd-tree data-structure. Different to unstructured inputs, our method needs less time for searching several nearest neighbors. Then, we use Greedy Surface Triangulation (GST) [33] to build an instance-wise mesh, which is designed to deal with planar surfaces. Note

Sequence	Ours	Ours/-wo	ORB [7]	PS-SLAM [6]	LPVO [12]	L-SLAM [11]	DVO [18]	InfiniTAM [32]
lr-kt0	**0.006**	0.025	0.025	0.016	0.015	0.012	0.108	×
lr-kt1	0.015	0.036	0.008	0.018	0.039	0.027	0.059	**0.006**
lr-kt2	0.020	0.053	0.023	0.017	0.034	0.053	0.375	**0.013**
lr-kt3	**0.012**	0.059	0.021	0.025	0.102	0.143	0.433	×
of-kt0	0.041	0.068	0.037	0.032	0.061	**0.020**	0.244	0.042
of-kt1	0.020	0.028	0.029	0.019	0.052	**0.015**	0.178	0.025
of-kt2	**0.011**	0.060	0.039	0.026	0.039	0.026	0.099	×
of-kt3	0.014	0.012	0.065	0.012	0.030	0.011	0.079	**0.010**
snot-far	0.022	0.026	×	**0.020**	0.075	0.141	0.213	0.037
snot-near	0.025	×	×	**0.013**	0.080	0.066	0.076	0.022
cabinet	**0.035**	0.057	0.075	0.067	0.520	0.291	0.690	**0.035**
large-cabinet	**0.071**	0.813	0.124	0.079	0.279	0.140	0.979	0.512

TABLE I

Comparison of Translation RMSE (m) for ICL-NUIM and TUM-RGB-D sequences. × means the method fails in the tracking process. -wo means only using decoupled tracking without the refinement step.

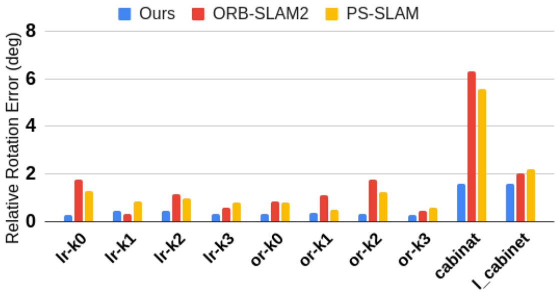


Fig. 4. Comparison of relative pose error (RPE) for rotation on the ICL-NUIM and TUM RGB-D sequences.

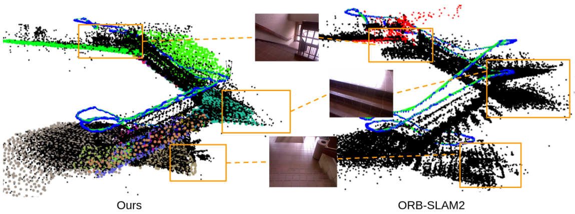


Fig. 5. Qualitative results of sparse reconstruction and trajectory between the proposed method and ORB-SLAM2 in the TAMU dataset.

that in our experiments, the initial search radius for selecting neighbors for triangulation is set to $5m$ and the multiplier is set as 5 to modify the final search radius to adapt to different point densities on the plane regions.

VI. Experiments

We evaluate the proposed SLAM system on two well known public datasets, the ICL-NUIM [34] and TUM RGB-D [35] benchmarks, comparing its performance with other state-of-the-art methods such as ORB-SLAM2 [7], PS-SLAM [6] that are feature-based methods, but removed the global bundle adjustment modules in the following experiments. Methods based on the MW assumption such as LPVO [11] and L-SLAM [12]. DVO-SLAM [18] is a direct method and InfiniTAM [32] uses a GPU for real-time tracking and mapping based on RGB and depth images. Additionally, we provide the reconstruction accuracy of our reconstructed model on the ICL-NUIM dataset and compare it with other popular methods for dense reconstruction. Lastly, to demonstrate that our system is robust over time, we also test on a sequence from the TAMU [3] dataset containing long sequences covering a large indoor area. All experiments are carried out with an Intel Core i7-8700 CPU (with @3.20GHz) and without any use of GPU. The ICL-NUIM dataset [34] provides synthetic scenes for two indoor environments, one living room and one office room scenario. These scenes contain large areas of low textured surfaces such as walls, ceilings, floors, etc. There are four sequences for each scene. We evaluate our method on all sequences.

time	Feat. extr.	Rotat.	Transla	Refinement	Total
Median	19.9	2.1	4.8	13.0	42.5
Mean	20.5	3.0	5.4	13.1	43.7
Std.	3.6	0.4	2.8	4.8	9.4

TABLE II

Measured tracking times (ms) on the TUM RGB-D sequences

Sequence	RGB-D	ElasticFu	InfiniTAM	SPFu	Ours
kt0	4.4	0.7	1.3	0.7	**0.4**
kt1	3.2	0.7	1.1	0.9	**0.6**
kt2	3.1	0.8	**0.1**	1.1	0.6
kt3	16.7	2.8	2.8	1.0	**0.8**

TABLE III

RMSE Reconstruction error (cm) on the ICL-NUIM dataset in centimeters.

A. ICL-NUIM RGB-D Dataset

Table I shows that our method obtains the best performance on three out of the eight sequences, based on the translation RMSE (ATE). InfiniTAM also performs well on lr-kt1, lr-kt2 and of-kt3 sequences, but the method also loses tracking in other sequences. As the dataset contains large structured areas, the Manhattan-based methods LPVO and L-SLAM are able to get a good estimate of the orientation and provide good results throughout. However, they usually

Sequences	Ours	ORB-SLAM2	length
Corridor-A	1.62	3.13	88
Stair-A	0.94	1.44	66
Entry-Hall	1.33	2.22	80

TABLE IV
COMPARISON OF THE ACCUMULATED DRIFT (M) IN DIFFERENT LARGE SCALE SEQUENCES.

need two planes, or alternatively, one plane and a vanishing direction to be visible at all times to estimate a good Manhattan frame. As shown in Figure 6, there are several challenging scenes in lr-kt3, where only a white wall and two leaves from a plant are captured when the camera is close to the wall. In this situation, OPVO and L-SLAM are unable to yield a good performance. When a bad initial pose is obtained in our system due to the scene not being a rigid MW, the refinement step based on point-line-plane features allows us to recover the pose nevertheless, while L-SLAM ignores optimizing rotation in the LKF module. Moreover, while DVO, being a dense method, may struggle because of the large areas of walls, floor etc. not containing enough gradient for the photometric error, ORB-SLAM2 and PS-SLAM perform well, as both environments contain sufficient ORB features extracted from furniture, objects etc. As our method takes advantage of all geometric elements, it is able to perform robustly in most sequences. In addition, Figure 4 shows the relative pose error for ORB-SLAM, PS-SLAM and our method. Our method obtains notably better results than the other two in relative translation and rotation. Especially the rotation error is much lower for our method, due to the use of the decoupled MW rotation estimation.

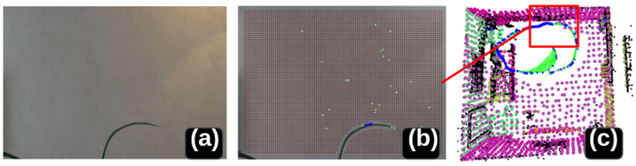


Fig. 6. Results in lr-k3. (a) input image; (b) point-line features and the segmented plane; (c) reconstructed 3D map and trajectory.

B. TUM RGB-D Dataset

The TUM RGB-D benchmark [35] is one of the most popular datasets for RGB-D SLAM systems, which provides indoor sequences under different texture and structure conditions. This allows us to separately test sequences which have structure, texture or both. In order to evaluate our method in challenging environments, we select four structured image sequences, the first three with low texture and the last one with a large scale environment. As all sequences listed in Table I have structure, but the large-cabinet sequence is not a rigid Manhattan scenario. Manhattan-based methods are able to provide good pose estimates on snot-far sequence, but the results degenerate in large-cabinet and cabinet sequences. The first two sequences include the same environment consisting of multiple non-textured planes. Here ORB-SLAM2 is not able to find enough point correspondences along the sequence and loses tracking. Our method, which additionally uses lines and planes for translation estimation, achieves better results. As shown in Figure 4, cabinet and large-cabinet are challenging sequences because of several low-texture frames. Our method's tracking strategy limits the relative rotation error to under 2 degrees, which is better than ORB-SLAM2 and PS-SLAM. The statistics of the time spent for each operation are shown in Table II, where we use different CPU threads to deal with points, lines and planes in the feature extraction and refinement modules.

C. Large scale sequence

The TAMU dataset [36] provides large-scale indoor sequences (constant lighting). While it does not provide ground-truth camera poses, the start and end point are the same, which can be used to evaluate the overall drift by computing the final position errors. As shown in Figure 5, the trajectory in the sequence Stair-C is a loop between two floors, where the improvement of our method over the whole trajectory length is 34.7% in drift compared to ORB-SLAM2. Similar situations can also be found in Corridor-A and Entry-Hall. More qualitative results are provided in the supplementary material.

D. Reconstruction Accuracy

We reconstruct models from ICL-NUIM and compare the results with state-of-the-art mapping methods, as shown in Table III. The accuracy of the reconstruction results is defined as the mean difference between the predicted model and the ground-truth model [34]. We compare the proposed mapping module against RGB-D SLAM [37], ElasticFusion [9], InfiniTAM [38], and SuperpixelFusion [10].

The SuperpixelFusion method is constrained by using ORB-SLAM for pose estimation, whereas our method also works well in low-textured environments. InfiniTAM obtains the best results in kt2, but shows worse performance on the kt0 and kt3 sequences, potential due to the large low-textured regions. ElasticFusion shows a similar behavior. Our method reconstructs more accurate maps than the others, but InfiniTAM and ElasticFusion provide more complete models than our map since we ignore small objects even though features based on points, lines and planes cover most of the pixels. Remarkably, all fusion methods, except for SuperpixelFusion and ours, rely on GPU based acceleration.

VII. CONCLUSIONS

We have proposed a RGB-D SLAM system based on points, lines and planes. Using the MW assumption for rotation estimation, and point, line and plane features for translation estimation, we achieve state-of-the-art performance. Also, a novel instance-wise meshing approach can reconstruct planar regions in the environment efficiently. The resulting dense map allows for interactions with the environment in robotic and AR/VR applications. In the future we would like to extend the planar reconstruction with a meshing of the non-planar parts in the environment to allow the complete reconstruction of more complex scenes.

References

[1] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari, "Structure-slam: Low-drift Monocular SLAM in Indoor Environments," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2020.

[2] R. Gomezojeda, F. Moreno, D. Scaramuzza, and J. G. Jimenez, "PL-SLAM: a Stereo SLAM System Through the Combination of Points and Line Segments." *IEEE Trans. Robot.*, 2019.

[3] Y. Lu and D. Song, "Robust RGB-D Odometry Using Point and Line Features," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015.

[4] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Morenonoguer, "PL-SLAM: Real-time Monocular Visual SLAM with Points and Lines," in *in Proc. IEEE Int. Conf. Robot. Automat.*, 2017.

[5] P. F. Proença and Y. Gao, "Probabilistic RGB-D odometry based on points, lines and planes under depth uncertainty," *Robotics and Autonomous Syst.*, vol. 104, pp. 25–39, 2018.

[6] X. Zhang, W. Wang, X. Qi, Z. Liao, and R. Wei, "Point-plane SLAM using supposed planes for indoor environments," *Sensors*, vol. 19, no. 17, p. 3795, 2019.

[7] R. Murartal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, 2017.

[8] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time Dense Surface Mapping and Tracking," in *IEEE Int. Symposium on Mixed and Augmented reality*, 2011.

[9] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "ElasticFusion: dense SLAM without a pose graph," in *Robotics: Science and Syst.*, 2015.

[10] K. Wang, F. Gao, and S. Shen, "Real-time Scalable Dense Surfel Mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019.

[11] P. Kim, B. Coltin, and H. J. Kim, "Low-drift Visual Odometry in Structured Environments by Decoupling Rotational and Translational Motion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018.

[12] P. Kim, B. Coltin, and H. Jin Kim, "Linear RGB-D SLAM for Planar Environments," in *Proc. Springer Eur. Conf. Comput. Vision*, 2018.

[13] C. Raposo, M. Lourenço, M. Antunes, and J. P. Barreto, "Plane-based odometry using an RGB-D camera." in *Proc. Brit. Mach. Vision Conf.*, 2013.

[14] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane SLAM for hand-held 3D sensors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013.

[15] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, "Dense Planar SLAM," in *IEEE Int. Symposium on Mixed and Augmented reality*, 2014.

[16] P.-H. Le and J. Košecka, "Dense piecewise planar RGB-D SLAM for indoor environments," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2017.

[17] F. Nardi, B. Della Corte, and G. Grisetti, "Unified representation and registration of heterogeneous sets of geometric primitives," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 625–632, 2019.

[18] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2013.

[19] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "CPA-SLAM: Consistent Plane-model Alignment for Direct RGB-D SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016.

[20] T. Schops, T. Sattler, and M. Pollefeys, "Bad SLAM: bundle adjusted direct RGB-D SLAM," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019.

[21] T. Schöps, T. Sattler, and M. Pollefeys, "Surfelmeshing: online surfel-based mesh reconstruction," *IEEE Trans. on Pattern Analysis and Machine Intell.*, 2019.

[22] S. Schreiberhuber, J. Prankl, T. Patten, and M. Vincze, "ScalableFusion: High-resolution Mesh-based Real-time 3D Reconstruction," in *in Proc. IEEE Int. Conf. Robot. Automat.*, 2019.

[23] J. Straub, N. Bhandari, J. J. Leonard, and J. W. Fisher, "Real-time Manhattan World Rotation Estimation in 3D," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2015.

[24] Y. Zhou, L. Kneip, C. Rodriguez, and H. Li, "Divide and Conquer: Efficient Density-based Tracking of 3D Sensors in Manhattan Worlds," in *Proc. Asian Conf. Comput. Vision*, 2016.

[25] K. Joo, T.-H. Oh, J. Kim, and I. So Kweon, "Globally Optimal Manhattan Frame Estimation in Real-time," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016.

[26] P. Kim, B. Coltin, and H. J. Kim, "Visual Odometry with Drift-free Rotation Estimation Using Indoor Scene Regularities," in *Proc. Brit. Mach. Vision Conf.*, 2017.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vision*, 2011.

[28] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A Fast Line Segment Detector with a False Detection Control," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 32, no. 4, pp. 722–732, 2010.

[29] L. Zhang and R. Koch, "An Efficient and Robust Line Segment Matching Approach Based on LBD Descriptor and Pairwise Geometric Consistency," *Elsevier Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[30] A. J. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen, "Efficient Organized Point Cloud Segmentation with Connected Components," *Semantic Perception Mapping and Exploration*, 2013.

[31] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a Versatile and Accurate Monocular SLAM System," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[32] V. A. Prisacariu, O. Kähler, S. Golodetz, M. Sapienza, T. Cavallari, P. H. Torr, and D. W. Murray, "Infinitam v3: A Framework for Large-scale 3D Reconstruction with Loop Closure," *arXiv preprint arXiv:1708.00783*, 2017.

[33] Z. C. Marton, R. B. Rusu, and M. Beetz, "On Fast Surface Reconstruction Methods for Large and Noisy Datasets," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009.

[34] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014.

[35] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2012.

[36] Y. Lu and D. Song, "Robustness to Lighting Variations: An RGB-D Indoor Visual Odometry Using Line Segments," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2015.

[37] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An Evaluation of the RGB-D SLAM System," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012.

[38] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time Large-scale Dense 3D Reconstruction with Loop Closure," in *Proc. Springer Eur. Conf. Comput. Vision*, 2016.

CCC RightsLink

Sign in/Register

RGB-D SLAM with Structural Regularities

Conference Proceedings:
2021 IEEE International Conference on Robotics and Automation (ICRA)

Author: Yanyan Li

Publisher: IEEE

Date: 30 May 2021

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Co-Planar Parametrization for Stereo-SLAM and Visual-Inertial Odometry

Xin Li[1*], Yanyan Li[2*] , Evin Pınar Örnek[2], Jinlong Lin[1], and Federico Tombari[2,3]

[1] Peking University
[2] Technical University of Munich
[3] Google
[*] Authors contribute equally

Title	Co-Planar Parametrization for Stereo-SLAM and Visual-Inertial Odometry
Version	Published Version
Paper document	Attached
License document	Attached

Co-Planar Parametrization for Stereo-SLAM and Visual-Inertial Odometry

Xin Li, Yanyan Li, Evin Pınar Örnek, Jinlong Lin, and Federico Tombari

***Abstract*—This letter proposes a novel SLAM framework for stereo and visual inertial odometry estimation. It builds an efficient and robust parametrization of co-planar points and lines which leverages specific geometric constraints to improve camera pose optimization in terms of both efficiency and accuracy. The pipeline consists of extracting 2D points and lines, predicting planar regions and filtering the outliers via RANSAC. Our parametrization scheme then represents co-planar points and lines as their 2D image coordinates and parameters of planes. We demonstrate the effectiveness of the proposed method by comparing it to traditional parametrizations in a novel Monte-Carlo simulation set. Further, the whole stereo SLAM and VIO system is compared with state-of-the-art methods on the public real-world dataset EuRoC. Our method shows better results in terms of accuracy and efficiency than the state-of-the-art. The code is released at https://github.com/LiXin97/Co-Planar-Parametrization.**

***Index Terms*—SLAM, Visual Learning.**

I. Introduction

SIMULTANEOUS Localization and Mapping (SLAM) and Visual Inertial Odometry (VIO) algorithms aim at camera pose estimation and scene reconstruction under unknown environments. They are ubiquitously employed in robotics for tasks such as planning, obstacle avoidance and navigation. When applied to indoor environments, these methods have to face important challenges due to the poor visual features available in the scene, which is often mostly characterized by low textured surfaces.

It has been shown that the structural regularities in the environment (e.g., lines and planes) bring valuable information to both SLAM and VIO systems [1], [2]. Such features can guide the SLAM optimization process by introducing additional constraints. However, how to organize such structural information and integrate it with the optimization in an efficient way is still an open question. Traditional representations focused on improving the trajectory accuracy, yet they ignored the high computational burden. In this work, we aim to tackle this problem by designing a better representation for planar structures, which simultaneously improves the accuracy and the efficiency of integrated stereo SLAM and VIO systems.

So far in the literature, several works leveraged points and lines detected from an RGB image to handle challenging environments [2]–[5]. Yet, the inner geometric relationship between those features is ignored in most of them. Different than using independent features of line segments and points, planar regions require fewer parameters to represent environments. Such planar regions and features can be found in almost all man-made environments, and they have been studied and leveraged in stereo SLAM and VIO systems [1], [6]–[9]. They introduce more constraints to the system that are helpful to improve overall accuracy. Nevertheless, they also rely on a high number of optimization parameters yielding limitations in real-world scenarios.

In this work, we propose a novel method to employ planarity constraints to improve the accuracy and efficiency of SLAM models based on VIO or stereo in indoor environments. Our method detects the co-planar point and line features through a deep learning based plane detection followed by RANSAC filtering. We then introduce a novel parametrization to represent these co-planar features in an unified manner instead of using them as independent features. The resulting parametrization decrease the size of Hessian matrix, as well as make it sparser as shown in Fig. 1(e). As a result, solving the bundle adjustment problem for estimating the correct camera parameters and 3D landmarks through second-order Newton optimization, which relies on calculating Schur complement on the Hessian matrix, becomes more efficient.

Furthermore, we show how our parametrization model can be integrated in a stereo SLAM or VIO pipeline as shown in Fig. 2 as we want to prove that our plane extraction and parametrization methods are general. By taking either a stereo image input, or an image with IMU sensor data, we solve the tracking and mapping problem through a graph based optimization. The non-planar 3D landmarks are integrated in the traditional way as 3D points, whereas the planar landmarks are introduced within the pipelines through proposed co-planarity parameters.

For evaluation, we used the public real-world EuRoC dataset and a newly created Monte-Carlo simulation set to perform further ablation studies. We compare our stereo-SLAM method against point-line SLAM approaches, as well as our VIO

Manuscript received May 12, 2020; accepted September 9, 2020. Date of publication September 28, 2020; date of current version October 8, 2020. This letter was recommended for publication by Associate Editor Todor Stoyanov and Editor Sven Behnke upon evaluation of the Reviewers' comments. *(Xin Li and Yanyan Li contributed equally to this work.) (Corresponding author : Yanyan Li.)*

Xin Li and Jinlong Lin are with Peking University, Beijing 100871, China (e-mail: lixin97@pku.edu.cn; linjl@ss.pku.edu.cn).

Yanyan Li and Evin Pınar Örnek are with the Department of Informatic, Technical University of Munich, 80333 Mnchen, Germany (e-mail: yanyan.li@tum.de; evin.oernek@tum.de).

Federico Tombari is with the Department of Informatic, Technical University of Munich, 80333 Mnchen, Germany, and also with Google Inc., Mountain View, CA 94043 USA (e-mail: tombari@in.tum.de).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3027230

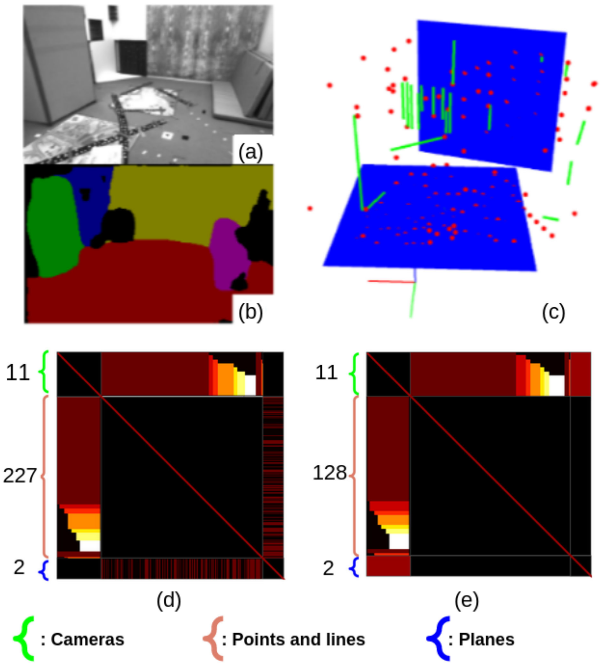


Fig. 1. System results: (a) input RGB frame; (b) plane instance segmentation; (c) reconstruction for points, lines and infinite planes; (d) and (e) Hessian matrices that show the spatial correlation of camera and 3D landmarks within the traditional [10], [11] and proposed parametrizations, respectively. Black areas represent zeros, non-zeros otherwise. (e) is sparser than (d). Number of camera parameters (green), points and lines features (orange) and plane parameters (blue) are shown.

method against the state-of-the-art plane-based VIO models. Our method shows improvement in accuracy on both pipelines while benefiting from lower runtime, demonstrating the effectiveness of co-planar constraints for SLAM. In summary, our paper proposes the following contributions:

- a novel two-stage plane detection strategy from RGB images, leveraging a neural network based plane segmentation and a robust outlier filtering
- a novel parametrization for co-planar points and lines that unifies the parameters, resulting in an efficient bundle adjustment optimization through the smaller and sparser Hessian matrix
- the deployment of these contributions within two different camera tracking frameworks, based respectively on VIO and stereo SLAM, both individually reporting state-of-the-art results.

II. Related Work

Feature-based SLAM is traditionally addressed by tracking keypoints along successive frames and then minimizing some error functions (typically based on re-projection errors) to estimate the camera poses [13]. For point/based only method, there are many successful proposals, such as PTAM [14], SVO [15] and ORB-SLAM [3]. However, using only point features has strong limitations within textureless environments as well as under illumination changes.

To deal with these problems, line-segment based methods were proposed [16], [17]. Moreover, planar regions and associated features have been leveraged by SLAM systems. In early works [1], planes in the scene were detected by RANSAC among estimated 3D points, which is time consuming and not stable. These plane-based mapping and tracking methods, however, are common within RGB-D sensors since it is easier to segment planes from depth maps. Salas-Moreno *et al.* [18] present a dense mapping approach by using bounded planes and surfels with RGB-D sensors. Point-Plane SLAM [19] computes orthogonal relationships between planes from depth maps, then uses constraints for pose estimation. CPA-SLAM [20] models the scene as a global plane model, which is helpful to remove drift by aligning current RGB-D frame with the plane model. By using IMU, VIO methods can deal with fast motion easily. MSCKF [21] and ROVIO [22] are popular filter-based methods, but the first one does not maintain estimates of 3D landmarks in the state vector. Different to those methods, an optimization strategy is used VINS-MONO [5] and Mesh-VIO [6] for pose estimation.

Instead of a set of features, planes are also used to construct co-planar regularities for points and lines. Instead of extracting planes from sparse point cloud, Mesh-VIO [6] builds 2D Delaunay triangulation based on 2D points first, and then project them into 3D from their correspondences. They find vertical and horizontal planes from the gravity vector given by the IMU, then merge the co-planar constraints in the optimization module. With the introduction of deep learning, methods were proposed to estimate planes from a single RGB image, hence opening up new possibilities for SLAM systems. PlaneReconstruction [23] and PlaneRCNN [24] are state-of-the-art plane instance segmentation methods for a single image. In addition to planes, they also estimate depth and normal maps from a single RGB image.

Inverse depth [10] and parallax angle [25] were proposed to represent point features in monocular systems. Inverse depth parametrization uses the inverse of the depth from its anchor camera, which works more accurately for distant features. Instead of using depth, the parallax angle is used in [25] which obtains good performance in both nearby and distant features. TextSLAM [26] suggests to extract text-based visual information and treats each detected text as a planar feature. In line parametrization methods, Plücker coordinate is a popular representation method for 3D line initialization and transformation. Each 3D line, however, has only 4 degrees of freedom (4DoFs), and the six parameters of Plücker coordinates lead to over-parameterization [27]. So, an orthogonal representation based on only four parameters is used in the optimization to solve this problem.

III. Proposed Method

In this section, we first explain our co-planar parametrization strategy, which includes plane instance detection and RANSAC based filtering steps. Then, we introduce the implementation details of our stereo and VIO versions that use the proposed parametrization in a sliding window optimization fashion.

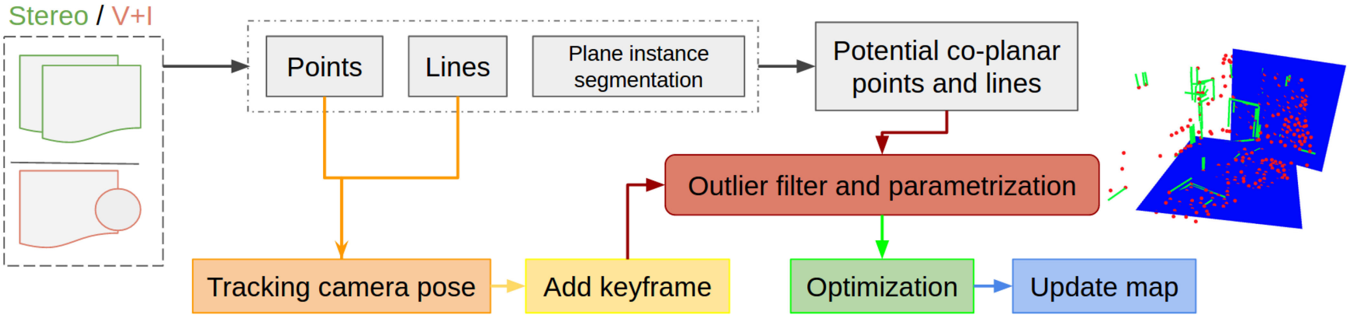

Stereo / V+I
Points
Lines
Plane instance segmentation
Potential co-planar points and lines
Outlier filter and parametrization
Tracking camera pose
Add keyframe
Optimization
Update map

Fig. 2. The pipeline of our plane-parametrized SLAM system. The overall pipeline follows the classical tracking and mapping approaches [3], along with the sliding-window based optimization. The pipeline can take as input either a stereo image pair or an image with IMU sensor data. 2D features and initial camera pose is estimated in a similar way as previous works [5], [12]. Then we detect planar regions via plane instance segmentation. After selecting the potential co-planar points and lines on the planar region, we remove the outliers with RANSAC. We present the remaining robust points and lines with the proposed parametrization, which can be directly integrated as an additional constraint in SLAM optimization.

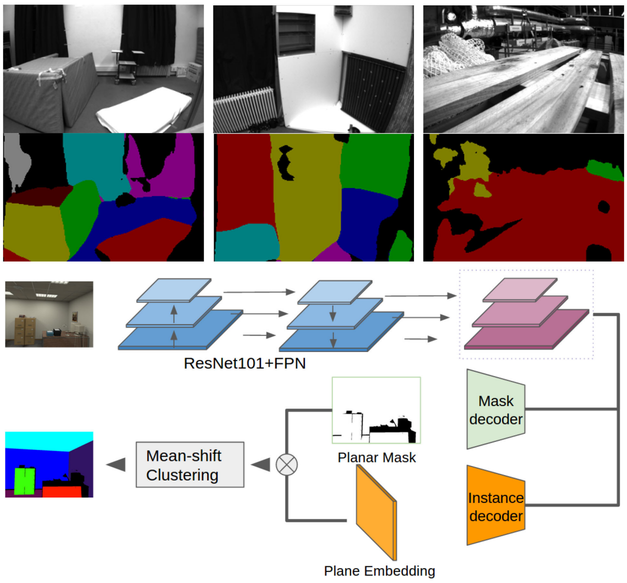


Fig. 3. Examples of plane instance segmentation on EuRoC dataset and architecture of the plane instance segmentation network.

A. *Coplanarity-Based Parametrization*

A plane is defined by equation $\mathbf{n}X_c^T + d = 0$, where $\mathbf{n} = (n_1, n_2, n_3) \in R^3$ is the normal of the plane, X_c is a 3D point in camera coordinates, and $d \in R$ is the distance from the plane to the origin of the camera c. However, this representation has an over-parametrization problem, and it cannot be solved with the Gauss-Newton approach due to singularity issue [28]. So we optimize the normal $\mathbf{n}$ on the tangent space S^2 with another optimization method, which is similar to Mesh-VIO [6]. In this section, we first describe how the co-planar points and features are detected. Then, we explain our parametrization for points and lines, respectively.

a) Plane Instance Segmentation: In order to detect planar regions in the scene in real-time, we use a plane instance segmentation network, which is a simplified version of PlaneReconstruction [23]. This network has two branches: planar mask decoder and a plane embedding decoder. The first branch decodes a binary mask for planar regions. The second one decodes the feature maps to an embedding space where mean-shift clustering is used to group each pixel into planar instances, iteratively. We train this plane detection network on ScanNet dataset [29] for 30 epochs.

b) Co-planar Feature Extraction: Since the plane instance segments extracted by the neural network might be at times inaccurate, we refine them by extracting 2D point and line features from images. Selecting the extracted features that align with the detected plane segments will lead us to robust features. We use ORB features [30] and LSD segment detection [31] to extract sets of co-planar points $[S_1^x, \ldots S_m^x]$ and co-planar lines $[S_1^l, \ldots S_m^l]$, where each distinct set consists of co-planar features $S_n^x = [x_i \ldots x_j], n \in [1, m]$ and x_i is a 2D pixel. For a stereo input, we obtain 3D points and lines by triangulating left-right image pairs. Whereas for VIO, the visual input is monocular and we triangulate sequential frames. During SLAM optimization, when a frame is detected as a new keyframe, we associate the features of this new frame with previous keyframes (i.e. check if they match and if they do not match, initiate new 3D landmarks with these features). After associating the landmarks, we build the potential co-planar points and lines, as shown in Fig. 2.

Due to the presence of outliers in the potential co-planar sets, we employ the following refinement strategy. First, for the current frame, we preserve the features that have been successfully triangulated. Then, we classify them according to detected 2D plane instance segments. If the number of features detected in a plane instance region is greater than a certain threshold, it will be considered as a potential planar region in 3D. If it is smaller than the threshold, plane will not be considered. After that, we use a RANSAC filter to find co-planar constraints in the potential planar region. We take out points C_x and lines C_l in the potential planar region and feed them to the filter. Specifically, corresponding rules in Eq. 1 are selected to fit parameters Γ of the plane according to the type of z ($\forall z \in \mathcal{Z}, \mathcal{Z} = [C_x, C_l]$),

$$f(c, \Gamma) = \begin{cases} \delta_\perp(P_x, \Gamma), z \in C_x \\ \max(\delta_\perp(c_{ls}, \Gamma), \delta_\perp(c_{le}), \Gamma), z \in C_l \end{cases} \qquad (1)$$

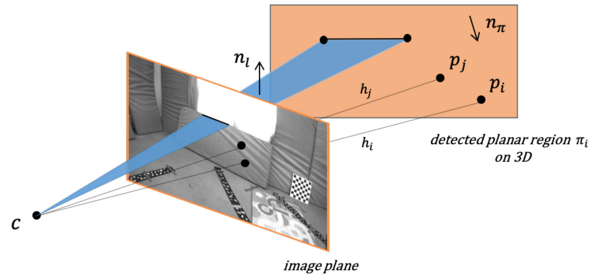


Fig. 4. Point and line features are shown on a detected planar region π_j with a normal n_π. h_i is the depth from camera frame origin to the 3D point p_i. n_l is the normal of a line on the plane n_π. Our parametrization rewrites the plane equation in terms of image pixel coordinates and combine line and point features.

where $\delta_\perp(\cdot,\cdot)$ denotes the perpendicular distance from a 3D point P_x to the plane in 3D, c_{ls} and c_{le} are the start and end points of the line respectively. Note that we only consider lines which have both endpoints lie on the same planar region. If the size of the largest consensus set exceeds a threshold θ_{cp} (80% in our experiments), we add the corresponding plane candidate to the system and establish point-plane and line-plane associations in the consensus set. We remove the outliers from the initial sets. When new 3D points and lines are generated in the system, we check if they belong to existing planes using the same metric defined above and store those correspondences. It is important to note that it would be also possible to detect planar regions by using only RANSAC (without the deep learning method). However, when there are unknown number of planes in a scene, RANSAC does not work optimal. It requires several iterations, where at each time a single planar region is detected and inlier points are removed. Yet, the false-detections accumulate over each time and results degenerate. We prevent this issue by detecting all planes through a neural network initially.

c) Parametrization of Points: After associating points and lines to co-planar regions as previously described, we obtain refined co-planar feature sets and parameters for each plane instance. As shown in Fig. 4, 3D points are the intersections of the detected plane and the camera-to-landmark rays. For each 3D point $P_x^c = (x^c, y^c, z^c)$ which lies on the plane π in camera frame c, we have the function $\mathbf{n}_\pi^T P_x^c + d_\pi = 0$. For example, the depth from origin of camera frame to the 3D point P_i is h_i. A normalized 3D point is presented as $(\hat{x}, \hat{y}, 1)$, where $(x^c, y^c, z^c) = (\hat{x}, \hat{y}, 1) \cdot h_i$. Also,

$$(\hat{x}, \hat{y}, 1)^T = K^{-1}(u, v, 1)^T \tag{2}$$

where K is the intrinsic matrix of camera c, and (u, v) is the 2D point corresponding to the landmark P_x^c. Then, the co-planar relationship for points can be represented as

$$h_i \cdot \mathbf{n}_\pi^T K^{-1}(u, v, 1)^T + d_\pi = 0, \tag{3}$$

where the relationship contains 2D pixel of the landmark and parameters of the plane. So in our parametrization, the point $\mathbf{p}^*$ lying on a planar region can be represented as

$$\mathbf{p}^* = [\mathbf{n}_\pi, d_\pi]. \tag{4}$$

d) Parametrization of Lines: For line features, the Plücker coordinates $\mathcal{L} = [\mathbf{n}_\mathrm{l}^\top, \mathbf{d}^\top]^\top$ are used to initialize 3D lines, where $\mathbf{d} \in \mathbb{R}^3$ is the line's direction vector in camera frame c, and $\mathbf{n}_\mathrm{l} \in \mathbb{R}^3$ is the normal vector of the plane determined by the line and the camera frame's origin point (Fig. 4). Furthermore, the line is the intersection of two known planes π_l and π_P, so the dual Plücker matrix $\mathbf{L}^*$ can be computed by:

$$\mathbf{L}^* = \begin{bmatrix} [\mathbf{d}]_\times & \mathbf{n}_\mathrm{l} \\ -\mathbf{n}_\mathrm{l}^\top & 0 \end{bmatrix} = \pi_l \pi_P^\top - \pi_P \pi_l^\top \in \mathbb{R}^{4\times4} \tag{5}$$

where $[\cdot]_\times$ is the skew-symmetric matrix of a three-dimensional vector, and $\pi = [\mathbf{n}, d]$ is a 4D vector. Then we can easily get Plücker coordinates $\mathcal{L} = [\mathbf{n}_\mathrm{l}^\top, \mathbf{d}^\top]^\top$ from the dual Plücker matrix.

e) Resulting Hessian matrix: Compared with other proposed representations, which treat points and lines as independent features, our method uses one plane parameter to represent all co-planar features. Novel parametrization is then used in the bundle adjustment, which is solved by a second-order Newton optimization method, the Levenberg-Marquardt algorithm. This relies on taking the gradients of the residuals with respect to parameters (3D landmarks and camera poses) and solving the normal equations. Hence, when there are less number of parameters, Hessian matrix will be smaller. When there are less dependencies between the parameters, the sparse structure of Hessian can be employed more efficiently through Schur complement. The resulting Hessian matrix is illustrated in Fig. 1 and it's effects on efficiency are further shown in Experiments section, in Tab. III. The optimization equations are explained in next subsection. Further interested reader is referred to [32].

B. System Implementation

In this section, implementation details are introduced for both versions of our approach, i.e. the stereo SLAM and VIO, respectively.

a) Tracking: The goal of the tracking module is to extract 2D features and estimate the camera pose for each frame. In the stereo version, we estimate camera pose via point and line features, where stereo keypoints are defined by three coordinates $x_s = (u_L, v_L, u_R)$, here (u_L, v_L) are coordinates on the left image and u_R is the horizontal coordinate for the corresponding matches in the right image. Similar to points, lines between two images are matched by Line Band Descriptor (LBD) [33]. Furthermore, motion model is used to provide an initial pose that is refined by a frame-to-frame tracking strategy similar to ORB-SLAM [3]. Instead, for the VIO version, the initialization strategy of IMU is similar to VINS-Mono [5], which relis on a loose coupling strategy to align IMU pre-integration with the visual-only part. Different than visual-only (stereo) branch, the initial pose for optimization in VIO is obtained from IMU pre-integration [2], [5] so that the visual part can be regarded as a purely monocular version. Monocular keypoints are defined by two coordinates $x_m = (u_L, v_L)$ which are triangulated from multiple views.

In the system, we use different strategies for keyframe detection in stereo and VIO pipelines. For the former one, a new keyframe can be added only after at least 20 frames. Each keyframe tracks more than 40 points and 10% of keypoints should be new keypoints compared to the nearest keyframe. However, for the latter one, we consider the average parallax (with rotation compensation) of tracked features between two keyframes, which should be more than 10 degrees (similar to VINS-Mono [5]).

b) Mapping: When a keyframe is detected and inserted, we associate its 2D features to 3D corresponding landmarks in the sliding window (or local map) by 2D feature matching. For each non-associated 2D feature, we triangulate it with other keyframes in the VIO version, while for stereo, non-associated points and lines are usually triangulated by each stereo pair. Different from points, 3D lines are triangulated by two intersecting planes colored in blue in Fig. 4, which are observed in different views.

Based on the potential co-planar regions and the RANSAC filter, 3D landmarks are divided into two sets for optimization: planar features and non-planar features. Inverse depth algorithm is used to represent points; and Plücker coordinates and orthonormal representations are used to represent lines following He [2], which are then fed to window-based bundle adjustment for optimizing poses and landmarks.

c) Bundle Adjustment With Co-Planar Parametrization: In this part, we use re-projection error functions to optimize camera pose and landmark positions. Two different error functions are used for planar and non-planar features. Non-planar features are represented by traditional parametrization and optimized directly. However, co-planar features are refined by optimizing the parameters of the proposed parametrization. For point features, the re-projection error $\mathbf{r}_{ik}^{\mathrm{p}}$ strands for the the distance between the projected point of the jth map point and the observed point in the kth frame, which is noted as

$$\mathbf{r}_{ik}^{\mathrm{p}} = x_{ik} - \Pi(T_{kw}, P_i^w) \tag{6}$$

where $\Pi()$ re-projects the ith global 3D point P_i^w coordinates into the kth frame. For general points, P_i^w is represented as (x^w, y^w, z^w). Points lying on a plane are represented with Eq. (3).

For line features, the re-projection error $\mathbf{r}_{jk}^{\mathrm{l}}$ is defined as the distance between the re-projected line of the jth map line and two endpoints of its corresponding 2D line in the kth keyframe, which is given by,

$$\mathbf{r}_{jk}^{\mathrm{l}} = \left[\frac{\mathbf{s}^\top \mathbf{n}_{\mathrm{l}}}{\sqrt{n_1^2+n_2^2}} \quad \frac{\mathbf{e}^\top \mathbf{n}_{\mathrm{l}}}{\sqrt{n_1^2+n_2^2}} \right]^\top \tag{7}$$

where $\mathbf{n}_{\mathrm{l}} = [n_1, n_2, n_3]^\top$ is the 2D line re-projected from the 3D line to the camera frame, $\mathbf{s} = [\hat{x}_s, \hat{y}_s, 1]^\top$ and $\mathbf{e} = [\hat{x}_e, \hat{y}_e, 1]^\top$ are two end-points of the observed line segment in the kth image plane. For general lines, $\mathbf{n}_{\mathrm{l}}$ can be represented as in an orthonormal way [2]. Lines lying on a plane are represented with the Eq. (5).

Given by the Eq. (6) and Eq. (7), We can therefore construct a unified target function which optimizes all terms simultaneously,

$$E = \sum_{k,i} \rho_p({\mathbf{r}_{ik}^{\mathrm{p}}}^\top \Lambda_{ik} \mathbf{r}_{ik}^{\mathrm{p}}) + \sum_{k,j} \rho_l(\mathbf{r}_{jk^{\mathrm{l}}}^\top \Lambda_{jk} \mathbf{r}_{jk}^{\mathrm{l}}) \tag{8}$$

here ρ_p and ρ_l present robust Cauchy cost functions. Respectively, Λ_{ik} and Λ_{jk} are the information matrices of points and lines, as calculated in [2], [5].

d) Tightly-Coupled Optimization for Inertial Constraints: For the VIO case, we fuse the data coming from the visual and inertial sensors via non-linear optimization in a tightly coupled form. Different from the stereo case, visual features are transferred to the IMU body coordinate system via extrinsic parameters $[\mathbf{R}_{bc} \quad \mathbf{t}_{bc}]$ between camera and IMU. So the unified target function for the VIO branch can be shown as,

$$\begin{aligned} E = & \sum_{k,i} \rho_p({\mathbf{r}^{\mathrm{p_{ik}}}}^\top \Lambda_{ik} \mathbf{r}_{ik}^{\mathrm{p}}) + \sum_{kj} \rho_l(\mathbf{r}_{jk^{\mathrm{l}}}^\top \Lambda_{jk} \mathbf{r}_{jk}^{\mathrm{l}}) \\ & + \sum_{b} \rho_l({\mathbf{r}^{\mathrm{b}}}^\top \Lambda_b \mathbf{r}^{\mathrm{b}}) + E_m \end{aligned} \tag{9}$$

where $\mathbf{r}^{\mathrm{b}}$ is the IMU residual, and E_m is the prior residual from marginalization operator in the sliding window. For more details, readers are referred to [5].

IV. Experiments

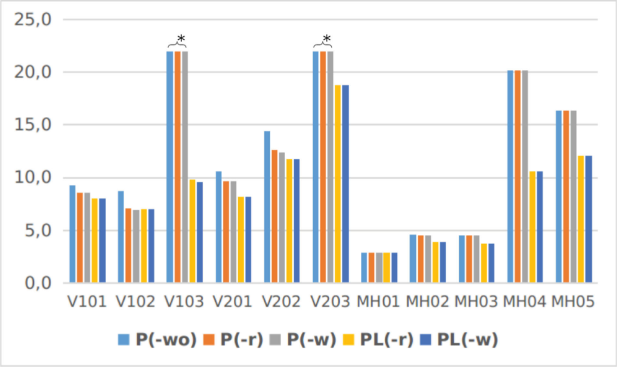
To evaluate the proposed method, we benchmark it against the state of the art on the EuRoC dataset [34]. In addition, we perform Monte-Carlo simulations to verify the robustness and efficiency of the novel parametrization. We evaluate both stereo and VIO pipelines with Absolute Trajectory Error (ATE) which measures absolute translational distances between the ground truth pose and the corresponding estimated pose. All the experiments run on an Intel Core i7-8550U @ 1.8 GHz and 16 GB RAM.

A. EuRoC Dataset

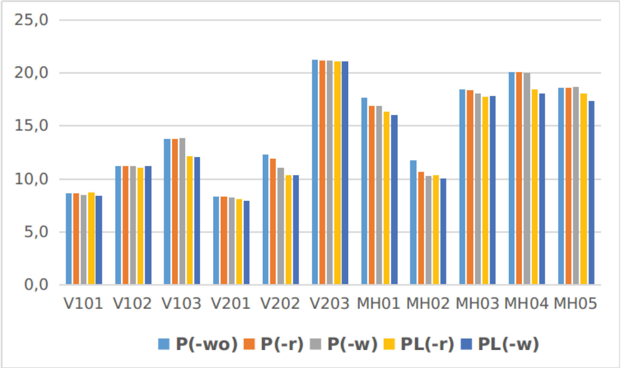
EuRoC is a popular public dataset for stereo SLAM and VIO systems, which collects stereo images and inertial data from an aerial vehicle in indoor environments [34]. There are two scenarios in this dataset: Vicon Room (V) and Machine Hall (MH), with eleven sequences in total. VH is an indoor environment and has several planar regions, whereas MH is the interior of an industrial facility where planar regions are unevenly distributed.

a) Ablation Studies: In order to evaluate the performance of the proposed parametrization in EuRoC, we fix the front-end and compare five formulations: $P(-wo)$, $P(-w)$, $PL(-w)$, $P(-r)$, and $PL(-r)$, where P denotes a point-based method, and PL denotes a point-line-based system. $(-wo)$ means the traditional parametrization (only inverse depth), and both $(-r)$ and $(-w)$ use co-planar constraints in the optimization module, but in different ways. $(-r)$ uses more equations between point-to-plane and line-to-plane, which are merged into optimization as in Mesh-VIO [6], [8]. Whereas $(-w)$ presents these residuals within the proposed co-planar parametrization.

The results of the stereo and VIO versions on EuRoC dataset are presented in Fig. 5 and Fig. 5, respectively. In general, the proposed parametrization $PL(-w)$ results in lower RMSE compared to traditional parametrizations, $P(-wo)$ and $P(-w)$, in both cases, and especially in the MH sequences, where the



(a) RMSE (cm), stereo



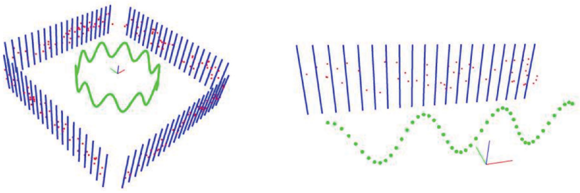
(b) RMSE (cm), VIO

Fig. 5. Comparison in terms of ATE of different parametrization variants: $P(-wo)$, $P(-r)$, $P(-w)$, $PL(-r)$ and $PL(-w)$. The top part shows results for stereo, and the bottom one for VIO. The proposed parametrization $PL(-w)$ achieves the best results for all sequences where structural regularities are detected and enforced. * shows lost tracking on V103 and V203 sequences.

line features can provide more robust constraints with planar regions in the large industrial environment.

For stereo approaches, as shown in Fig. 5, line features make the system more robust especially in $V103$ and $V203$ sequences, where severe motion blur happened. In other Vicon sequences, $PL(-w)$ performs better than $PL(-r)$ because the proposed two-stage co-planar approach removes distances between those co-planar features and planes directly. In $MH01$, $MH02$ and $MH03$ which are textured sequences, all approaches obtain similar results. In Fig. 5, $P(-wo)$ and $P(-w)$ perform equally on $MH03$, $MH04$ and $MH05$ sequences, because there are not any structural regularities detected. When there are some planar regions detected, as in $V202$, $MH01$ and $MH02$, the proposed parametrization $P(-w)$ obtains better performance than traditional methods. If enough features can be obtained and few good co-planar sets, our system's performance will degenerate to that of traditional methods, as in sequences $V102$ and $V201$. The computation time of different operations in V101 is presented in Table II.

b) EuroC Evaluation: We compare our stereo branch against the stereo version of ORB-SLAM2 [12] and FMD-SLAM [35]. It is important to note that, for fairness of comparison, the tested ORB-SLAM2 does not have loop closure. Furthermore, we compare our VIO version against the recently proposed MSCKF [21], ROVIO [22], VINS-MONO [5], and Mesh-VIO [6]. Results are given in Table I. These VIO algorithms use all a monocular camera, except Mesh-VIO that uses a stereo camera. Results of previous works are taken from Rosinol *et al.* [6].



(a) Sequence a (b) Sequence b

Fig. 6. Two simulation environments are illustrated, where points and lines are in red and blue, respectively. Camera follows green trajectories.

The left part of Table I shows that the $PL(-w)$ approach is an accurate and robust method compared with state-of-the-art VIO methods on sequences. Compared with Mesh-VIO [6], which also uses planar information to build co-planar regularities in the optimization process, our method performs better on most sequences, where Mesh-VIO obtained more vertical planes from 3D mesh due to using gravity during plane detection. When horizontal and vertical planes are difficult to detect as in $V103$ and some of the MH sequences, Mesh-VIO tends to degenerate easily so that it cannot build co-planar constraints. In sequence $MH05$, we observe a 26% improvement compared to the second best performing algorithm (Mesh-VIO), and in sequence $V103$, a 15% improvement and 35% improvement compared to VINS-MONO and Mesh-VIO, respectively. It can be seen that the optimization methods of VINS-MONO, Mesh-VIO and $PL(-w)$ are more robust than the filter-based MSCKF. Meanwhile, our method is more robust for indoor environments that have lots of co-planar regularities.

The stereo SLAM comparison is shown on the right side of Table I. Stereo ORB-SLAM2 obtains comparable results to ours on all sequences except V203 and MH04. In those textured sequences, this method tracks the features in a stable and accurate way. Instead, V203 is a difficult sequence because of the fast motion and the strong illumination changes, and tracking fails for both ORB-SLAM2 and FMD-SLAM. Benefiting from using point and line features, our method is instead more robust and can deal also with this sequence. The average RMSE values, for fairness computed without taking sequence V203 into account, show that our method obtains 25.7% and 38.7% improvements compared to ORB-SLAM2 and FMD-SLAM, respectively.

B. Simulation Dataset

We create two simulation sequences with ideal co-planar environments to evaluate the efficiency with respect to performance under different parametric formulations. As shown in Fig. 6(a), the first sequence has 100 lines and 200 points generated in 4 directions, which are observed by virtual cameras that follow a sinusoidal trajectory with 150 simulated poses. The second sequence consists of 20 lines and 50 points observed by 50 camera poses as shown in Fig. 6(b).

TABLE I
COMPARISON IN TERMS OF RMSE (CM) OF THE PROPOSED $PL(-w)$ PIPELINE AGAINST THE STATE OF THE ART ON THE EUROC DATASET. BEST RESULTS ARE BOLDED. × SHOWS LOST TRACKING. AVERAGED RESULTS WITH * DO NOT INCLUDE THE SEQUENCE V203

	VIO							**Stereo**				
	MSCKF [21]	ROVIO [22]	VINS MONO [5]	Mesh VIO [6]	PL (-wo)	PL (-r)	PL (-w)	ORB SLAM2 [12]	FMD SLAM [35]	PL (-wo)	PL (-r)	PL (-w)
V101	34	10	7	**6**	8.4	8.5	8.4	9	9	8.4	**8.0**	**8.0**
V102	20	10	10	**7**	11.0	10.9	11.0	8	20	7.5	**7.0**	**7.0**
V103	67	14	13	17	**11.9**	**11.9**	**11.9**	20	53	10.6	9.8	**9.6**
V201	10	12	**8**	**8**	8.1	8.1	**8.0**	**7**	9	9.0	8.2	8.2
V202	16	14	**8**	10	12.0	10.5	10.5	10	**8**	12.4	11.8	11.8
V203	113	**14**	21	27	20.9	20.9	20.9	×	×	19.8	**18.8**	**18.8**
MH01	42	21	27	**14**	17.1	16.3	16.2	4	4	**2.9**	**2.9**	**2.9**
MH02	45	25	12	13	11.0	10.2	**10.0**	5	4	4.1	**3.9**	**3.9**
MH03	23	25	**13**	21	17.6	17.6	17.6	4	5	4.0	**3.7**	**3.7**
MH04	37	49	23	22	18.5	18.4	**18.2**	16	**9**	10.6	10.6	10.6
MH05	48	52	35	23	18.2	18.0	**17.7**	20	**9**	12.1	12.1	12.1
Average	41.3	22.3	16.0	15.2	14.1	13.8	**13.7**	10.6*	12.9*	8.1*	**7.8***	**7.8***

TABLE II
COMPUTATION TIME (MEAN, MS) OF DIFFERENT OPERATIONS IN THE V101 SEQUENCE OF EUROC. * MEANS THAT THE OPERATION IS USED FOR EACH FRAME, OTHERWISE IT IS PERFORMED ON KEYFRAMES ONLY. D&M NOTES DETECTION AND MATCHING. - MEANS THAT THE OPERATION IS NOT USED

Operation	P(-wo)	P(-r)	P(-w)	PL(-wo)	PL(-r)	PL(-w)
Point D&M*	4	4	4	4	4	4
Line D&M	-	-	-	96	96	96
Plane Seg.	-	29	29	-	29	29
Plane fitting	-	10	10	-	10	10
Optimization	43	44	40	36	46	42
Total time	56	60	55	57	58	54

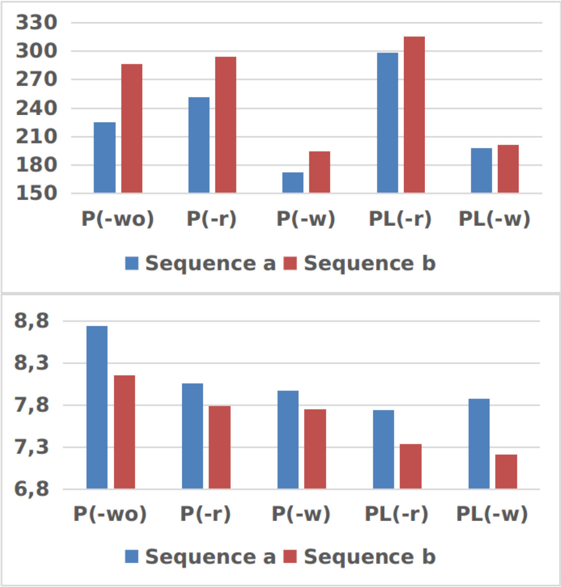


Fig. 7. Comparison of the optimization time (ms, top) and RMSE (cm, bottom) for pipelines $P(-wo)$, $P(-r)$, $P(-w)$, $PL(-r)$ and $PL(-w)$.

For line measurements, the virtual camera gets two endpoints from each measurements. Note that each measurement of a point, including endpoints of lines and point features, is corrupted by 1-pixel Gaussian random noise. In order to simplify the simulation, we simulate relative pose odometry measurements as pose estimation results from the tracking module, which have random noise as,

$$\bar{q}_m = \begin{bmatrix} \frac{1}{2}\mathbf{n}_\theta \\ 1 \end{bmatrix} \otimes \bar{q}, \quad \mathbf{p}_{Cm} = \mathbf{p}_C + \mathbf{n}_p \tag{10}$$

where $\mathbf{n}_\theta$ and $\mathbf{n}_p$ are the Gaussian white noises added to the relative pose, with $\sigma_\theta = 1$ deg and $\sigma_p = 10$ cm, respectively.

a) Performance: We pose the visual SLAM system as a non-linear least squares problem, solved via Gaussian-Newton. Maximum 10 iterations are allowed for each method in this simulation for a fair comparison. We run the simulation sequence 30 times and show median results for the accuracy of the estimated trajectory and optimization time. Fig. 7 shows similar performance across sequences, that is, $(-w)$ is more accurate and efficient than $(-r)$ and $(-wo)$. The second sequence (b) requires more optimization time and results in lower RMSE since more features are measured by each camera compared to the first. $P(-wo)$ requires less time than $P(-r)$ in two sequences because it does not use structural regularities and has small optimization computation as shown in Fig. 1(d). $P(-r)$ has a higher computational burden (Fig. 1(e)) and is more accurate than $P(-wo)$. While combining line features in the system, like $PL(-r)$, results are more accurate even if the method requires more time. Compared to $P(-r)$ and $PL(-r)$, our parametrizations for points and lines ($P(-w)$) are more efficient. In terms of optimization time, $P(-w)$ has a 31% improvement and $PL(-w)$ 33%, compared to $P(-r)$.

b) Number of Parameters: Furthermore, we analyze the reason of efficiency from the perspective of number of parameters that are to be updated. In traditional parametric methods (inverse depth for points and orthogonal approach for lines), each point, line and plane need 1 parameter, 4 parameters and 3 parameters, respectively. However, in our parametrization method, all points and lines in the plane are represented by only one plane parameter. Hence, there is only one parameter for each planar region during optimization. Table III shows the number of parameters that need to be updated in the global bundle adjustment on the second Monte Carlo sequence, where 20 lines

TABLE III
THE NUMBER OF LANDMARKS UPDATED IN OPTIMIZATION MODULE OF SEQUENCE 2

	P(-wo)	P(-r)	P(-w)	PL(-wo)	PL(-r)	PL(-w)
items	100	101	**51**	120	121	**51**
parameters	350	353	**303**	430	433	**303**

and 50 points are observed by 50 cameras. $P(-w)$ uses points only, so it has to update 100 items at each iteration. Similar to $P(-wo)$, we have to update 101 items and 121 items in $P(-r)$ and $PL(-r)$. Note that those two need to update one plane item because they use of co-planar constraints of point-to-plane and line-to-plane. In the proposed solutions, only 51 items (50 cameras and 1 plane) are updated in $P(-w)$ and $PL(-w)$ because they use the plane to represent co-planar points and lines.

V. CONCLUSION

We presented an efficient and robust co-planar parametrization method for points and lines by leveraging geometric and learning approaches together, which increases sparsity and reduces the size of Hessian matrix in each optimization module. Then, we illustrated how our co-planar parametrization can be implemented in stereo-SLAM and VIO pipelines. Our experiments show that our approach improves the efficiency and accuracy of both stereo and VIO optimization in indoor environments. As for future work, we plan to reconstruct dense maps from monocular data and merge together semantic segmentation and depth prediction to improve tracking and mapping simultaneously.

ACKNOWLEDGMENT

The authors would like to thank Dr. Yijia He for the helpful discussion.

REFERENCES

[1] Y. Lu and D. Song, “Visual navigation using heterogeneous landmarks and unsupervised geometric constraints,” *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 736–749, Jun. 2015.

[2] Y. He, J. Zhao, Y. Guo, W. He, and K. Yuan, “Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features,” *Sensors*, vol. 18, no. 4, p. 1159, 2018.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: A versatile and accurate monocular slam system,” *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[4] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[5] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[6] A. Rosinol, T. Sattler, M. Pollefeys, and L. Carlone, “Incremental visual-inertial 3d mesh generation with structural regularities,” in *Proc. Conf. Robot. Autom.*, 2019, pp. 8220–8226.

[7] D. Zou, Y. Wu, L. Pei, H. Ling, and W. Yu, “Structvio: Visual-inertial odometry with structural regularity of man-made environments,” *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 999–1013, Aug. 2019.

[8] X. Li, Y. He, J. Lin, and X. Liu, “Leveraging planar regularities for point line visual-inertial odometry,” 2020, *arXiv:2004.11969*.

[9] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, “Kimera: An Open-Source Library for Real-Time Metric-Semantic Localization and Mapping,” in *Proc. Conf. Robot. Autom.*, 2020, pp. 1689–1696.

[10] J. Civera, A. J. Davison, and J. M. Montiel, “Inverse depth parametrization for monocular slam,” *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 932–945, Oct. 2008.

[11] A. Bartoli and P. Sturm, “The 3d line motion matrix and alignment of line reconstructions,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2001, vol. 1 pp. I–I.

[12] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[13] C. Cadena *et al.*, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[14] G. Klein and D. Murray, “Parallel tracking and mapping for small ar workspaces,” in *Proc. 6th IEEE Assoc. Comput. Machinery Int. Symp. Mixed Augmented Reality*, 2007, pp. 225–234.

[15] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “Svo: Semidirect visual odometry for monocular and multicamera systems,” *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.

[16] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, “Building a 3-d line-based map using stereo slam,” *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1364–1377, Dec. 2015.

[17] R. Gomez-Ojeda, F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, “Pl-slam: A stereo slam system through the combination of points and line segments,” *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 734–746, Jun. 2019.

[18] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, “Dense planar slam,” in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2014, pp. 157–164.

[19] X. Zhang, W. Wang, X. Qi, Z. Liao, and R. Wei, “Point-plane slam using supposed planes for indoor environments,” *Sensors*, vol. 19, no. 17, 2019, Art. no. 3795.

[20] L. Ma, C. Kerl, J. Stückler, and D. Cremers, “Cpa-slam: Consistent plane-model alignment for direct rgb-d slam,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1285–1291.

[21] A. I. Mourikis and S. I. Roumeliotis, “A multi-state constraint kalman filter for vision-aided inertial navigation,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 3565–3572.

[22] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, “Robust visual inertial odometry using a direct ekf-based approach,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 298–304.

[23] Z. Yu, J. Zheng, D. Lian, Z. Zhou, and S. Gao, “Single-image piece-wise planar 3D reconstruction via associative embedding,” in *Proc. Comput. Vis. Pattern Recognit.*, 2019, pp. 1029–1037.

[24] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz, “Planercnn: 3D plane detection and reconstruction from a single image,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, Jun. pp. 4445–4454.

[25] L. Zhao, S. Huang, L. Yan, and G. Dissanayake, “Parallax angle parametrization for monocular slam,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3117–3124.

[26] B. Li, D. Zou, D. Sartori, L. Pei, and W. Yu, “Textslam: Visual slam with planar text features,” in *Proc. Conf. Robot. Autom.*, 2020, pp. 2102–2108.

[27] A. Bartoli and P. Sturm, “Structure-from-motion using lines: Representation, triangulation, and bundle adjustment,” *Comput. Vis. Image Understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[28] M. Kaess, “Simultaneous localization and mapping with infinite planes,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 4605–4611.

[29] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “Scannet: Richly-annotated 3D reconstructions of indoor scenes,” in *Proc. IEEE Comput. Vis. Pattern Recognit*, 2017, pp. 2432–2443.

[30] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, “Orb: An efficient alternative to sift or surf,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[31] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, “Lsd: A line segment detector,” *Image Process. Line*, vol. 2, pp. 35–55, 2012.

[32] M. I. A. Lourakis and A. A. Argyros, “Sba: A software package for generic sparse bundle adjustment,” *ACM Trans. Math. Softw.*, vol. 36, no. 1, pp. 1–30, Mar. 2009.

[33] L. Zhang and R. Koch, “An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency,” *J. Vis. Commun. Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[34] M. Burri *et al.*, “The euroc micro aerial vehicle datasets,” *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016.

[35] F. Tang, H. Li, and Y. Wu, “Fmd stereo slam: Fusing mvg and direct formulation towards accurate and fast stereo slam,” in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 133–139.

CCC RightsLink

Sign in/Register

IEEE Requesting permission to reuse content from an IEEE publication

Co-Planar Parametrization for Stereo-SLAM and Visual-Inertial Odometry

Author: Xin Li

Publication: IEEE Robotics and Automation Letters

Publisher: IEEE

Date: October 2020

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

E-Graph: Minimal Solution for Rigid Rotation with Extensibility Graphs

Yanyan Li[1] and Federico Tombari[1,2]

[1] Technical University of Munich
[2] Google

Title	E-Graph: Minimal Solution for Rigid Rotation with Extensibility Graphs
Version	Accepted Version
Paper document	Attached
License document	Attached

E-Graph: Minimal Solution for Rigid Rotation with Extensibility Graphs

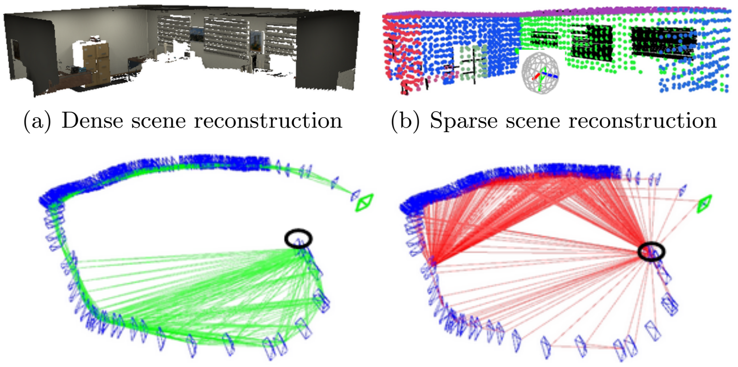
Yanyan Li[1,2][0000−0001−7292−9175] and Federico Tombari[1,3][0000−0001−5598−5212]

[1] Technical University of Munich, Munich, Germany
[2] Meta-Bounds Tech, Shenzhen, China
[3] Google, Zurich, Switzerland
yanyan.li@tum.de, tombari@in.tum.de

Abstract. Minimal solutions for relative rotation and translation estimation tasks have been explored in different scenarios, typically relying on the so-called co-visibility graphs. However, how to build direct rotation relationships between two frames without overlap is still an open topic, which, if solved, could greatly improve the accuracy of visual odometry. In this paper, a new minimal solution is proposed to solve relative rotation estimation between two images without overlapping areas by exploiting a new graph structure, which we call Extensibility Graph (E-Graph). Differently from a co-visibility graph, high-level landmarks, including vanishing directions and plane normals, are stored in our E-Graph, which are geometrically extensible. Based on E-Graph, the rotation estimation problem becomes simpler and more elegant, as it can deal with pure rotational motion and requires fewer assumptions, e.g. Manhattan/Atlanta World, planar/vertical motion. Finally, we embed our rotation estimation strategy into a complete camera tracking and mapping system which obtains 6-DoF camera poses and a dense 3D mesh model. Extensive experiments on public benchmarks demonstrate that the proposed method achieves state-of-the-art tracking performance.

1 Introduction

Camera pose estimation is a long-standing problem in computer vision as a key step in algorithms for visual odometry, Simultaneous Localization and Mapping (SLAM) and related applications in robotics, augmented reality, autonomous driving (to name a few). As part of the camera pose estimation problem, the minimal case [40] provides an estimate of whether the problem can be solved and how many elements are required to obtain a reliable estimate. According to the input data type and scenarios, different solutions [1, 28, 15, 9] were proposed, most of which became very popular in the computer vision and robotic community, such as the seven-point [1] and five-point [28] approaches. A typical limitation of traditional pose estimation solutions based on the minimal case [1, 28, 31, 9] is that both rotation and translation estimation rely on the co-visibility features between two frames, this having as a consequence that the length of an edge between two nodes is often relatively short. Therefore, tracking errors tend



(a) Dense scene reconstruction (b) Sparse scene reconstruction

(c) Covisibility graph (d) Extensibility graph

Fig. 1. Dense (a) and sparse (b) scene reconstruction of the office-room scene from the ICL dataset [10] obtained by the proposed method. (c) and (d): keyframes (in blue) and connected frames are linked with green and red lines, respectively, to build up the proposed covisibility and extensibility graphs. The black ellipses denote the start points of the camera trajectory.

to accumulate easily based on a frame-to-frame or frame-to-keyframe strategy. To solve this issue, more advanced tracking systems [26, 3] with optimization solutions, including local and global bundle adjustment approaches, were exploited to refine poses from minimal solutions. Loop Closure is a common algorithm used in feature-based [24] and direct [7] methods to remove drift. However, it also requires the camera to revisit the same place, which is a limiting assumption in many scenarios.

Compared with point features, lines and planes require more computation to be extracted and described. Early multi-feature SLAM systems [8] use them to increase the number of features to combat low-textured scenes. After that, coplanar, parallel and perpendicular relationships were explored [39, 18, 20] to add more constraints in the optimization module, still following a similar tracking strategy as ORBSLAM [25] or DSO [36] for the initial pose estimation.

Different to the tightly coupled estimation strategy, some works [43] proposed to decouple the 6-DoF pose estimation into rotation and translation estimation aiming to achieve a more accurate rotation estimation, based on the idea that pose drift is mainly caused by the rotation component [14]. At the same time, based on an estimated rotation matrix [31], only two points are required to compute the translation motion, leading to more robustness in low-textured regions.

The Manhattan World (MW) [43] and Atlanta World (AW) [13] assumptions introduce stronger constraints since they require a single orthogonal scene, or a scene with a unified vertical direction. Unlike loop closure that removes drift by detecting trajectory loops, the assumption of MW and AW is introduced for indoor tracking scenarios [19, 14] to improve the accuracy of camera pose estimation, since most indoor artificial environments follow this assumption. MW and AW improve accuracy when the main structure of the scene has orthogonal elements However, since this assumption requires the observation of vertical/orthogonal environmental features (such as straight lines or planes), the SLAM system using this method is also limited in the types of scenarios it can be successfully applied to.

In this paper we propose a rigid rotation estimation approach based on a novel graph structure, which we dub Extensibility Graph (E-Graph), for landmark association in RGB-D data. Our approach is designed to reduce drift and improve the overall trajectory accuracy in spite of loop closure or MW/AW assumptions. Benefiting of E-Graph, the drift-free rotation estimation problem is simplified to the alignment problem of rotating coordinate systems. Importantly, our rotation step does not need overlaps between two frames by making use of vanishing directions of lines and plane normals in the scene, hence can relate a higher number of keyframes with respect to standard co-visibility graphs, with benefits in terms of accuracy and robustness in presence of pure rotational motions.

In addition, we develop a complete tracking and dense mapping system base on the proposed E-Graph and rotation estimation strategies, which we demonstrate to outperform state-of-the-art SLAM approaches [20, 38, 26, 3]. To summarize, the main contributions of this paper are as follows: i) a new perspective for reducing drift is proposed based on our novel graph structure, E-Graph, which connects keyframes across long distances; ii) a novel drift-free rotation alignment solution between two frames without overlapping areas based on E-Graph; iii) a complete SLAM system based on the two previous contributions to improve robustness and accuracy in pose estimation and mapping. The proposed approach is evaluated on common benchmarks such as ICL [10] and TUM-RGBD [33], demonstrating an improved performance compared to the state of the art.

2 Related work

By making the assumption of planar motion [9], two-view relative pose estimation is implemented based on a single affine correspondence. Point features are common geometric features used in VO and SLAM [3] systems. To remove the drift from point-based front ends, different types of back ends are explored in tracking methods. Loop closing is an important module to remove drift, which happens when the system recognizes that a place [6, 23] has been visited before. After closing the loop, associated keyframes in the covisibility graph will be adjusted. Benefiting of loop closure and optimization modules, ORB-SLAM series [26, 3] organize the keyframes efficiently, which provides robust support for tracking tasks. Different from sparse point features used in ORB-SLAM, BAD-SLAM [32] implements a direct bundle adjustment formulation supported by GPU processing.

However, in indoor environments, to cover texture-less regions that have few point features, more geometric features are merged into the front end of systems. At the early stage, methods build re-projection error functions for lines and planes. CPA-SLAM [22] makes use of photometric and plane re-projection terms to estimate the camera pose. Based on estimated camera poses, detected planes are merged together with a global plane model. Similar to our method, CPA-SLAM and KDP-SLAM [11] can build constraints between non-overlapping frames. However those constraints are used to build heavy optimization targets instead of improving the efficiency. Furthermore, the relationship between parallel lines (vanishing points) and perpendicular planes is explored in [17, 41]. Based on the regularities of those structural features, they obtain a more accurate performance. Instead of exploring the parallel/perpendicular relationships between lines/planes, [30, 18] make use of constraints between co-planar points and lines in the optimization module.

Those regularities aim to build constraints between local features, [14, 20] introduce global constraints by modeling the environment as a special shape, like MW and AW. The MW assumption is suitable for a cuboid scenario, which is supposed to be built by orthogonal elements. Based on this assumption, those methods estimate each frame's rotation between the frame and the Manhattan world directly, which is useful to avoid drift between frames in those scenes. L-SLAM [14] groups normal vectors of each pixel into an orthogonal coordinate by projecting them into a Gaussian Sphere [43] and tracks the coordinate axes to compute the relative rotation motion. Similar to the main idea of L-SLAM, [15] provides a RGB-D compass by using a single line and plane. Since the line lies on the plane, the underlying assumption of the system is the MW-based rotation estimation method. However, the limitation of this strategy is also very obvious, that it works only in Manhattan environments. Based on ORB-SLAM2 [26], Structure-SLAM [19, 20] merges the MW assumption with keyframe-based tracking, to improve the robustness of the system in non-MW indoor scenes, which refine decoupled camera pose by using a frame-to-model strategy. Compared with MW-based tracking methods, our approach is less sensitive to the structure of environments.

3 Minimal case in orientation estimation

Commonly, the 6-DoF Euclidean Transform $T \in SE(3)$ defines motions as a set of rotation $R \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$. Based on point correspondences, camera pose estimation can be defined as,

$$\mathbf{P}^{'} = R\mathbf{P} + \mathbf{t} \tag{1}$$

where $\mathbf{P}^{'}$ and $\mathbf{P}$ are 3D correspondences, and $[R, \mathbf{t}]$ defines the relative motion between two cameras. For monocular sensors, their image normalized representations are $\mathbf{X}_c^{'}$ and $\mathbf{X}_c$,

$$\mathbf{X}_c^{'} = \alpha(R\mathbf{X}_c + \gamma\mathbf{t}) \tag{2}$$

where α and γ are depth-related parameters. After multiplying (2) by $\mathbf{X}_c^{'T}[\mathbf{t}]_x$, we can obtain the classic essential matrix equation,

$$\mathbf{X}_c^{'T} E \mathbf{X}_c = 0 \tag{3}$$

where $E = [\mathbf{t}]_x R$ and $[\mathbf{t}]_x$ is the skew symmetric matrix formed by $\mathbf{t}$.

For RGB-D sensors, the task is simplified since the absolute depth information is directly provided by sensors. Equation (1) can be solved by using 3 non-collinear correspondences only [29], although the distance between two frames is supposed to be kept small to extract enough correspondences.

3.1 Minimal solution for rotation

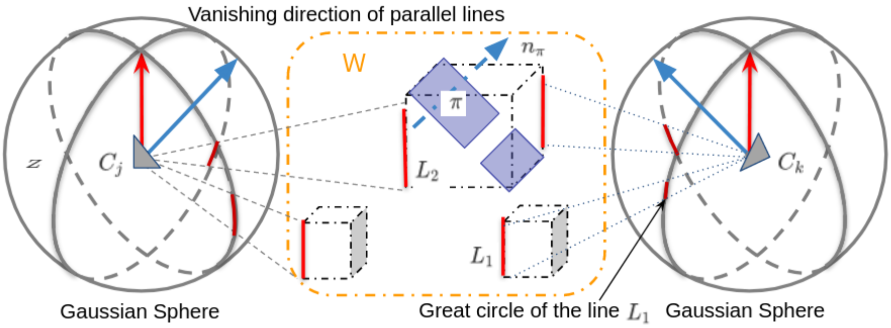

Vanishing direction of parallel lines
W
n_π
π
z
C_j
C_k
L_2
L_1
Gaussian Sphere
Great circle of the line L_1
Gaussian Sphere

Fig. 2. Minimal case of rotation estimation in EG

Different from traditional methods based on co-visibility graphs, the proposed method decouples rotation and translation estimation into two separate stages. Moreover, the rotation estimation task does not require feature correspondences. As shown in Figure 2, non-parallel direction vectors $\mathbf{v}_m, m \in [0, 1, \ldots, n]$ are detected in the camera coordinate C_j, where $\mathbf{v}^j_m = [{}_x v^j_m, {}_y v^j_m, {}_z v^j_m]^T$. In Euclidean 3D space, the size of a finite and linearly independent set of vectors is less then four. According to the Gram-Schmidt orthogonalization process, we can obtain an orthogonal set $S = [\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2]$,

$$\begin{aligned} \mathbf{u}_0 &= \mathbf{v}^j_0 \\ \mathbf{u}_1 &= \mathbf{v}^j_1 - proj_{[\mathbf{v}^j_0]}(\mathbf{v}^j_1) \\ \mathbf{u}_2 &= \mathbf{v}^j_2 - proj_{[\mathbf{v}^j_0]}(\mathbf{v}^j_2) - proj_{[\mathbf{v}^j_1]}(\mathbf{v}^j_2) \end{aligned} \tag{4}$$

by using the projection operator $proj_{[\mathbf{u}]}(\mathbf{v}) = \frac{<\mathbf{u},\mathbf{v}>}{||\mathbf{u}||||\mathbf{v}||}\mathbf{v}$, where $< \mathbf{u}, \mathbf{v} >$ shows the inner product of the vectors $\mathbf{u}$ and $\mathbf{v}$. Furthermore, we obtain the normalized vectors $\mathbf{e}_0$, $\mathbf{e}_1$ and $\mathbf{e}_2$ via $\mathbf{e}_m = \frac{\mathbf{u}_m}{||\mathbf{u}_m||}$.

For the Euclidean space $\mathbb{R}^3$, the relevant orthonormal basis set based on the detected direction vectors is $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$. In the j^{th} camera coordinate, the orthonormal set is detected as $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$, while $(\mathbf{e}^*_0, \mathbf{e}^*_1, \mathbf{e}^*_2)$ in the k^{th} camera coordinate.

Therefore, from the perspective of the orthonormal set, those j^{th} and k^{th} coordinates are represented as $[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]^T$ and $[\mathbf{e}^*_0, \mathbf{e}^*_1, \mathbf{e}^*_2]^T$, respectively.

Given $\begin{bmatrix} \mathbf{e}_0^T \\ \mathbf{e}_1^T \\ \mathbf{e}_2^T \end{bmatrix}$ $[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]$ is the identity matrix, the matrix $[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]$ is an orthogonal matrix and the columns of $[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]^T$ are orthonormal vectors as well, which can be used to build the orthonomal basis set of the j^{th} camera coordinate. Therefore, in $\mathbb{R}^3$ an arbitrary vector $\mathbf{x}$ can be represented by two orthonormal sets, $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)^T$ and $(\mathbf{e}_0^*, \mathbf{e}_1^*, \mathbf{e}_2^*)^T$, independently,

$$\begin{aligned} \mathbf{x} &= (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)^T (x_0, x_1, x_2)^T \\ &= (\mathbf{e}_0^*, \mathbf{e}_1^*, \mathbf{e}_2^*)^T (x_0^*, x_1^*, x_2^*)^T \end{aligned} \tag{5}$$

Finally, $(x_0, x_1, x_2)^T = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)(\mathbf{e}_0^*, \mathbf{e}_1^*, \mathbf{e}_2^*)^T (x_0^*, x_1^*, x_2^*)^T$ where the rotation motion $R_{c_j c_k}$ from camera k to camera j is $[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2][\mathbf{e}_0^*, \mathbf{e}_1^*, \mathbf{e}_2^*]^T$.

Two-Observation case. In the spatial case where two linearly independent direction vectors are detected, $\mathbf{u}_2$ can be achieved by the cross product process of $\mathbf{u}_0$ and $\mathbf{u}_1$. Obviously, the new set $[\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_0 \times \mathbf{u}_1]$ maintains the orthogonal property, which is the minimal solution for relative pose estimation problems.

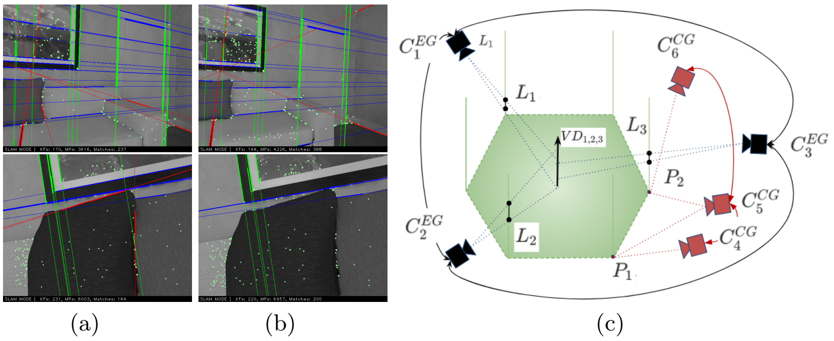
Orthogonal-Observation case. As discussed in Section 2, the MW assumption is enforced mostly by SLAM/VO methods designed to work indoor [15, 14, 19, 38], achieving particularly good results when the MW assumption holds. When the observation vectors $\mathbf{v}_m^j$ are orthogonal, the projection operation between different vectors is zero and the proposed method degenerates to a multi-MW case,

$$\begin{aligned} R_{c_j c_k} &= R_{c_j M_i} R_{c_k M_i}^T \\ &= [\frac{\mathbf{v}_0^j}{||\mathbf{v}_0^j||}, \frac{\mathbf{v}_1^j}{||\mathbf{v}_1^j||}, \frac{\mathbf{v}_2^j}{||\mathbf{v}_2^j||}][\frac{\mathbf{v}_0^k}{||\mathbf{v}_0^k||}, \frac{\mathbf{v}_1^k}{||\mathbf{v}_1^k||}, \frac{\mathbf{v}_2^k}{||\mathbf{v}_2^k||}]^T. \end{aligned} \tag{6}$$

For single-MW scenarios, a global orthogonal set can be obtained by every frame, therefore $R_{c_j w}$, from world to camera C_j, can be computed by $R_{c_j M} R_{c_0 M}^T$, here $R_{c_0 w}$ is an identity matrix.

Compared with the visual compass [15] method making use of a combination of line and plane features from MW [14] to estimate camera rotation, our graph is more robust and flexible. Furthermore, compared to [31] that generates four rotation candidates after aligning two frames' vanishing points, our method not only leverages plane features, but also solves the ambiguity regarding the directions of the vanishing points [31].

After the relative rotation pose estimation step between two frames, in case of no overlap between them, we need to make use of their neighboring frames to compute translation vectors. Note that only two correspondences are required in translation estimation by making use of Equation 1, which is particularly suited to deal with scenes and environments characterized by different texture types compared to traditional approaches [26, 3].



(a) (b) (c)

Fig. 3. Vanishing point detection and Rotation connection examples. (a) Detection results of J-Linkage. (b) Refined results by our system. (c) E-Graph (black) and co-visibility graph (red).

4 Extensibility Graph (E-Graph)

As shown in Figure 3(c), the E-Graph method builds rotation connections (edges) between frames $[C_1^{EG}, C_2^{EG}, C_3^{EG}]$ that share global directions instead of any low-level correspondences (like points and lines). At the same time, no connection between C_4^{CG} and C_6^{CG} can be made since these frames have no co-visible features within the co-visibility graph. The proposed connection strategy will be detailed in the following subsections.

4.1 Landmarks from a RGB-D frame

Similar to the co-visibility graph, the proposed graph is also a topological representation of scenes. The difference is that the proposed graph is built based on the scene structure rather than on overlapping parts between frames. The distance between connected frames in a co-visibility graph tends to be small (see Figure 3) since two frames that are distant from each other rarely overlap, leading to the pose of the current frame being estimated based on the last frame or last keyframes only. The issue can be alleviated by using global bundle adjustment and loop closure modules, although they bring in intensive computation and trajectory constraints (e.g. need to re-visit a certain area).

In our graph $\mathscr{G} = [\mathscr{N}_c, \mathscr{N}_{lm}, \mathscr{E}]$ frames and landmarks are regarded as nodes $\mathscr{N}_c$ and $\mathscr{N}_{lm}$ respectively, while $\mathscr{E}$ represents the edges among connected frames. Note that landmarks are border-less planes and vanishing directions, e.g. $VD_{1,2,3}$, of lines detected in multiple views. In particular, an edge is established between two frames every time two or more structural elements are matched across them.

Features and landmarks. Vanishing directions are estimated from parallel lines detected by a joint 2D-3D process, where LSD [35] is used to extract 2D line features from RGB images. Meanwhile, AHP [5] is carried out to extract plane features from depth maps.

Firstly, as shown in Figure 3(a), we make use of the J-Linkage algorithm to classify detected 2D lines into different groups of parallel lines as described in [34]. However, there are still outliers left based on the 2D process. To solve this issue, we take advantage of depth maps to check the directions of lines in each group by using RANSAC to detect the best direction vector VD_n to represent the group S_n.

As for planar landmarks, we make use of the Hessian ($\boldsymbol{\pi} = (\mathbf{n}^\pi, d^\pi)$) to represent a plane detected from the i^{th} frame, where $\mathbf{n}^\pi$ denotes the normal vector and d^π represents the distance between the camera center and this plane, which is transferred to world coordinates via the initial pose T_{wc_i}.

4.2 Data Association

After generating vanishing directions and planes, we now explain how to initialize and update them.

Initialization. Combined with the first keyframe Kf_0, detected planes and optimized vanishing directions are used to initialize the E-Graph. The camera pose T_0 of Kf_0 is set as the world coordinate for landmarks in the E-Graph. Planes $\boldsymbol{\pi}_i$ measured by Kf_0 are transferred to the graph directly as,

$$\mathscr{G}_0 = [\mathscr{N}_{c_0}, \mathscr{N}_{lm_0}, \mathscr{E}_0] \tag{7}$$

where $\mathscr{N}_{c_0}$ is Kf_0 and $\mathscr{E}_0$ has no edges yet. $\mathscr{N}_{lm_0}$ contains $[\boldsymbol{\pi}_i, VD_i, PD_j]$, where VD_i and PD_j refer to two different types of 3D lines detected in the RGB-D frame: the former refers to lines that are parallel to at least another 3D line, the latter to lines that are not parallel to any line. The first type of lines can generate vanishing directions VD_i in a single view, which are stored into the graph directly, similarly to planes. In addition, lines that do not have parallel lines detected in this RGB-D frame are marked as potential vanishing direction PD_j. In case parallel lines will be detected in successive frames, these lines will also be transferred to VD_j, otherwise, they are removed from the E-Graph.

Landmarks fusion. For each new input frame we need to extract vectors $\mathbf{n}^\pi$, VD and PD from the current frame. After rotating VD_i^c to the world coordinate frame as VD_i^w, if the direction between VD_i^c is parallel to $VD_k^w, k \in [0, \ldots, m]$, where m is the number of vanishing directions saved in E-Graph, VD_i^c is then associated to the graph. To solve the unsure issues [31] of vanishing directions, we will unify the direction during the association process by using

$$\tilde{VD}_i^c = \begin{cases} VD_i^c \ \ (|norm(VD_i^w \cdot VD_k^w) - 1| < th_{vd}) \\ -VD_i^c \ (|norm(VD_i^w \cdot VD_k^w) + 1| < th_{vd}) \end{cases} \tag{8}$$

where $norm(\cdot)$ shows a dot product between two normalized vectors and $|\cdot|$ is the absolute difference. th_{vd} is a threshold to check the angle distance between two vectors. To include additional graph connections, we also try to associate

PD_j^c with VD_k^w and PD_k^w. If new pairs can be made at this stage, the associated PD vectors are transferred to the vanishing directions and fused into the graph.

Since the vanishing direction is independent from translation motion, VD_i^w, the vanishing direction in the world coordinate can be obtained as

$$VD_i^w = R_{wc} VD_i^c \tag{9}$$

where R_{wc} is the rotation motion from the camera coordinate frame to the world coordinate frame.

In certain indoor scenes, e.g. a corridor or hallway, when a robot moves along the wall, an extended planar region is detected across multiple views, with most of these views encompassing no overlap. To address this issue, we extract the normal vector $[n_x^c, n_y^c, n_z^c]$ of the plane in the camera coordinate, which can be fused into the world coordinate in the same way as the vanishing directions.

Edge connection. In E-Graph, all landmarks come from keyframes that follow the decision mechanisms of a feature-based SLAM system [24, 20], which we summarize in the following. A new keyframe is detected if it satisfies one of the following two conditions: 1) 20 frames have passed from the last keyframe; 2) the current frame tracks less than 85% points and lines correspondences with the last keyframe. Furthermore, when the current frame detects a new plane or a new vanishing direction, the frame is considered as a new keyframe. In addition, new landmarks connected to this keyframe are also merged into the graph at this stage.

By sequentially processing keyframes, if more than two pairs of matched landmarks are observed between two keyframes, an edge will be created to connect the respective two graph nodes. As shown in Figure 2, C_j and C_k detect the plane $\boldsymbol{\pi}$ and the same vanishing point generated by L_1 and L_2. Notably, even if these two frames do not have any correspondence, they can still be connected in our E-Graph.

5 Experiments

In this section, the proposed system is evaluated on different indoor benchmarks: ICL-NUIM [10] and TUM RGB-D [33]. ICL-NUIM [10] contains eight synthetic sequences recorded in two scenarios (living room and office room). TUM RGB-D [33] is recorded in real scenarios and includes varied sequences in terms of texture, scene size, presence of moving objects, etc.

Rotation estimation. The proposed rotation algorithm is compared with other state-of-the-art orientation estimation approaches. Compass [15] makes use of a single line and plane. OPRE [43] and GOME [12] estimate the distribution of surface normal vectors based on depth maps. OLRE [2] and ROVE [16] take advantage of vanishing directions for rotation estimation. Importantly, Compass, GOME, OLRE, OPRE, and P-SLAM [20] are all based on the MW assumption, while our method, ORB-SLAM2 [25] and ROVE are designed for general scenes.

Translation estimation. Since the rotation of the current frame is estimated from a keyframe that may not be overlapping with the current frame, we follow the 3D translation estimation model [26, 20] to estimate the translation $\mathbf{t}$ based on the predicted rotation. In this module, re-projection errors from point-line-plane feature correspondences are used to build a target optimization function, $\mathbf{t} = argmin(\sum_{j=0}^{n} e^{\pi}_{i,j} \Lambda^{\pi} e^{\pi}_{i,j} + e^{L}_{i,j} \Lambda^{L} e^{L}_{i,j} + e^{P}_{i,j} \Lambda^{L} e^{P}_{i,j})$, where e^{π}, e^{L} and e^{P} are re-projection error functions for planes, lines and points, respectively. The target function is optimized by using the Levenberg-Marquardt method. The translation is compared with the following state-of-the-art methods. ORB-SLAM2 [26] and ORB-SLAM3 [3] are popular keypoint-based SLAM systems. In our experiments, for fairness of comparison the loop closure is removed to reduce the effect of the back-ends. SP-SLAM [39] additionally uses points and planes in the tracking and optimization modules based on ORB-SLAM2. P-SLAM [19] assumes the indoor environments as MW, and includes a refinement module to make the tracking process more robust. Moreover, we also compare our system with GPU-based methods, including BadSLAM [32] and BundleFusion [4].

Dense mapping. In this paper, a mapping module is implemented to reconstruct unknown environments in sparse and dense types. The sparse map is reconstructed by the point-line-plane features extracted from keyframes, which supports a frame-to-map pose refinement step. Since sparse maps cannot provide enough information for robots, our system also generates a dense mesh map incrementally based on CPU. When a new keyframe is generated from the tracking thread, we make use of the estimated camera pose and the RGB-D pair to build a dense TSDF model based on [42, 27]. After that, the marching cubes method [21] is exploited to extract the surface from voxels.

Metrics. The metrics used in our experiments include absolute trajectory error (ATE), absolute rotation error (ARE), and relative pose error (RPE) that shows the difference in relative motion between two pairs of poses to evaluate the tracking process. Our results are reported in Table 2 and obtained on an Intel Core @i7-8700 CPU @3.20GHz and without any use of GPU resources.

5.1 ICL NUIM dataset

As shown in Table 1, the proposed method outperforms other MW-based and feature-based methods in terms of average rotation error. In *office room* sequences, OPRE and P-SLAM also perform well since orthogonal planar features can be found in the environment. However, in *office room 0*, parts of the camera movement only contain a single plane and some lines, leading to performance degradation, while our method achieves robust orientation tracking by taking advantage of a set of non-parallel planes and lines.

Furthermore, we compare the translation results against two feature-based methods as shown in Table 2. The first four sequences are related to a living room scenario, while the remaining sequences are from an office scenario. All methods obtain good results in *living room 0* where the camera moves back and

Table 1. Comparison of the average value of the absolute rotation error (degrees) on ICL-NUIM and TUM RGB-D structural benchmarks. The best result for each sequence is bolded. × shows that the method fails to track the orientation.

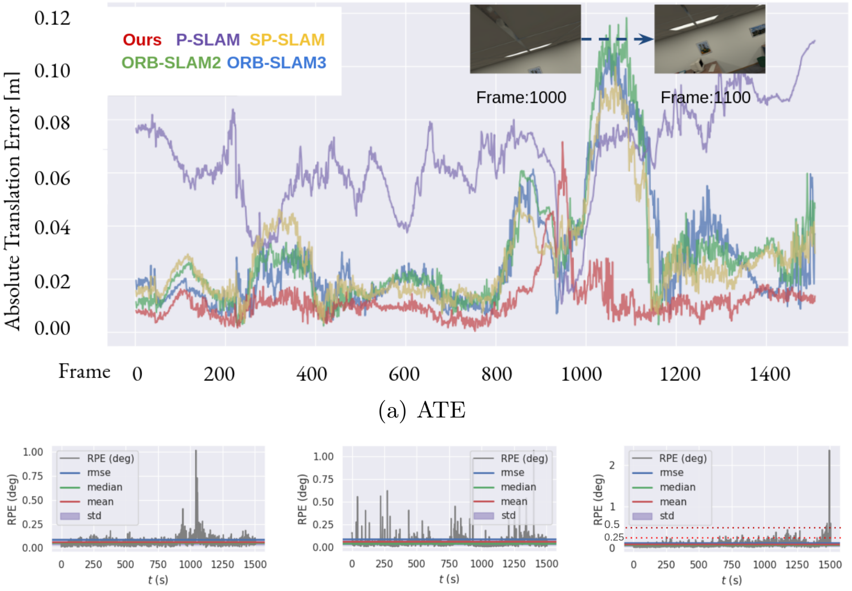
Sequence	Ours	Compass [15]	OPRE [43]	GOME [12]	ROVE [16]	OLRE [2]	ORB2 [26]	P-SLAM [20]
office room 0	**0.11**	0.37	0.18	5.12	29.11	6.71	0.40	0.57
office room 1	**0.22**	0.37	0.32	×	34.98	×	2.30	**0.22**
office room 2	0.39	0.38	0.33	6.67	60.54	10.91	0.51	**0.29**
office room 3	0.24	0.38	**0.21**	5.57	10.67	3.41	0.36	**0.21**
living room 0	0.44	**0.31**	×	×	×	×	0.97	0.36
living room 1	**0.24**	0.38	0.97	8.56	26.74	3.72	0.22	0.26
living room 2	0.36	**0.34**	0.49	8.15	39.71	4.21	0.83	0.44
living room 3	0.36	0.35	1.34	×	×	×	0.42	**0.27**
f3_stru_notex	4.46	1.96	3.01	4.07	×	11.22	×	4.71
f3_stru_tex	**0.60**	2.92	3.81	4.71	13.73	8.21	0.63	2.83
f3_l_cabinet	**1.45**	2.04	36.34	3.74	28.41	38.12	2.79	2.55
f3_cabinet	2.47	2.48	2.42	2.59	×	×	5.45	**1.18**

forth between the two parallel walls. P-SLAM detects a good MW model, and ORB-SLAM3 also observes enough features, benefiting from paintings hanging on the wall and small furniture. Compared with the living room, the office room has many low-textured regions. The performance of feature-based algorithms is not as good as in the living room scenes, especially in *office room 1* and *office room 3*.

Table 2. Comparison in terms of translation RMSE (m) for ICL-NUIM and TUM RGB-D sequences. × means that the system fails in the tracking process.

Sequence	Ours	P-SLAM[20]	ORB-SLAM3[3]
office room 0	**0.014**	0.068	0.035
office room 1	**0.013**	0.020	0.091
office room 2	0.020	0.011	**0.010**
office room 3	**0.011**	0.012	0.096
living room 0	0.008	**0.006**	**0.006**
living room 1	**0.006**	0.015	0.206
living room 2	**0.017**	0.020	0.018
living room 3	0.021	**0.012**	0.019
f1_360	0.114	×	**0.108**
f1_room	**0.095**	×	×
f2_rpy	**0.002**	0.154	0.003
f2_xyz	**0.003**	0.009	0.004
f3_l_o_house	0.012	0.122	**0.009**
f3_stru_notex	**0.017**	0.025	×
f3_l_cabinet	**0.058**	0.071	0.072

To analyze the relationship between rotation and translation results of different methods, absolute translation and rotation errors on the *office room 0* sequence are presented in Figure 4. When the camera moves to the ceiling, the number of detected features decreases, then an interesting phenomenon is witnessed (see also Figure 4(a)): the tracking error of feature-based systems quickly and drastically increases, then gradually fades as the number of features increases. At the same time, our method and P-SLAM exhibit a more robust performance when they face this challenge. An important difference is that, while P-SLAM underperforms due to the non-rigid MW scene, our method's performance is accurate thanks to the use of the E-Graph, which demonstrates to be more flexible than MW-based paradigms.



(a) ATE

(b) Ours

(c) P-SLAM

(d) ORB-SLAM3

Fig. 4. Comparison of the proposed system against state-of-the-art methods in the *office room 0* sequence of ICL NUIM in terms of mean/average absolute translation errors (top) and rotation errors (bottom).

5.2 TUM RGB-D

Different types of sequences are included from the TUM RGB-D benchmark, which aims to test general indoor scenes with low-textured scenes and sharp rotational motions. *f1_360*, *f1_room*, *f2_rpy* and *f2_xyz* are recorded in real office scenes, but the camera's rotation motion changes sharply especially in the first sequence. *f3_l_o_house*, *f3_sn_near* and *f3_l_cabinet* contain more structural information, where *f3_sn_near* is built on two white corners, and *f3_l_cabinet* records several movements around the white cabinet. Table 1 shows that ROVE, OLRE and ORB-SLAM2 have problems in low/non-textured regions. In *f3_l_cabinet*

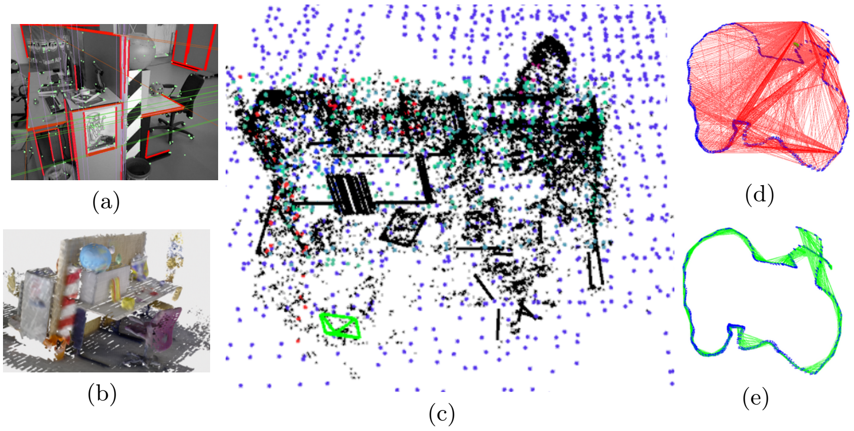


Fig. 5. Scene and graphs of *f3_l_o_house*. (a) 2D image, (b) dense mesh model, (c) sparse map, (d) E-Graph, (e) co-visibilitity graph.

that is not a rigid MW environment, the quality of depth maps is noisy, the surface normal maps extracted by OPRE have a negative effect on rotation estimation.

Table 3. ATE RMSE results (cm) on the TUM RGB-D dataset. Results for BundleFusion and BadSLAM are taken from [32]

Sequence	Ours	BundleFusion [4]	ElasticFusion [37]	BadSLAM [32]
	CPU	GPU	GPU	GPU
f1_desk	1.0	1.6	2.0	1.7
f2_xyz	0.7	1.1	1.1	1.1
f3_office	1.4	2.2	3.6	1.7

For structural sequences listed in Table 1, P-SLAM shows stable performance. In Table 2, general scenes are added as a comparison. As listed in Table 2, the keypoint-based method [3] cannot achieve robust results in *f3_sn_near*, i.e. , a textureless scenario, while the MW-based method [20] has problems when the scene structure breaks the MW assumption, by reporting a low performance in *f2_rpy* and *f3_l_o_house*, and even losing track in *f1_360* and *f1_room*. Therefore, the proposed method shows more robust performances in different types of scenarios, compared with MW-based systems [20, 15] and feature-based approaches [26, 3]. Furthermore, compared with GPU-based systems, our system only works on limited computation sources. As shown in Figure 5, *f3_l_o_house* is used to compare E-Graph and co-visibility graph. As clearly shown, E-Graph allows connecting more distant keyframes than a co-visibility graph. When two keyframes can be connected together, drifting phenomena can more easily be limited, in a similar way to the underlying idea behind loop closure. The cabinet scene is also a difficult sequence for point-based methods (see Figure 6(b)) since point features are concentrated in a few boundary regions. However, our method can deal with this type of scene where the same plane is observed in a number of frames.

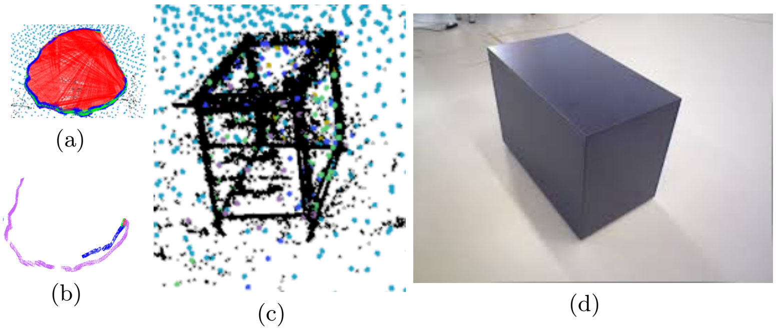


Fig. 6. Scene and graph of *f3_cabinet*. (a) E-Graph, (b) trajectory from ORB-SLAM3, (c) sparse map, (d) 2D image.

6 Conclusion

This paper proposed a new graph structure, E-Graph, to reduce tracking drift based on plane normals and vanishing directions in a scene, which can be used to build a rotation connection between two frames without visual overlap. The advantage of this idea is that rotation errors that occur between two frames have small or no effect on this relative rotation estimation step. Based on the proposed graph, a minimal solution is presented, that shows that two landmarks and two correspondences can be used to solve the relative camera pose. Therefore, the proposed method is better suited for texture-less scenes compared with traditional minimal solutions based on co-visible features. However, the proposed method also has limitations. Compared with point-based systems, our approach requires more types of features. Furthermore, since we need vanishing directions and plane vectors, the method is more suitable for man-made scenes.

Feature work. The E-Graph is a new tool to establish connections across frames and keyframes. An interesting topic for future exploration is considering a covisibility graph and our graph together to revisit pose estimation and obtain further improvements in drift removal.

Acknowledgements. We gratefully acknowledge Xin Li, Keisuke Tateno, Nicolas Brasch and Dr. Liang Zhao for the helpful discussion.

References

1. Andrew, A.M.: Multiple view geometry in computer vision. Kybernetes (2001)
2. Bazin, J.C., Pollefeys, M.: 3-line ransac for orthogonal vanishing point detection. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 4282–4287. IEEE (2012)
3. Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M., Tardós, J.D.: Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam. arXiv preprint arXiv:2007.11898 (2020)
4. Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., Theobalt, C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. ACM Transactions on Graphics (ToG) **36**(4), 1 (2017)
5. Feng, C., Taguchi, Y., Kamat, V.R.: Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 6218–6225. IEEE (2014)
6. Galvez-Lpez, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics **28**(5), 1188–1197 (2012). https://doi.org/10.1109/TRO.2012.2197158
7. Gao, X., Wang, R., Demmel, N., Cremers, D.: Ldso: Direct sparse odometry with loop closure. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2198–2204. IEEE (2018)
8. Gomez-Ojeda, R., Moreno, F.A., Zuniga-Noël, D., Scaramuzza, D., Gonzalez-Jimenez, J.: Pl-slam: A stereo slam system through the combination of points and line segments. IEEE Transactions on Robotics **35**(3), 734–746 (2019)
9. Guan, B., Zhao, J., Li, Z., Sun, F., Fraundorfer, F.: Minimal solutions for relative pose with a single affine correspondence. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1929–1938 (2020)
10. Handa, A., Whelan, T., McDonald, J., Davison, A.: A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM. In: IEEE International Conference on Robotics and Automation (2014)
11. Hsiao, M., Westman, E., Zhang, G., Kaess, M.: Keyframe-based dense planar slam. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). pp. 5110–5117 (2017). https://doi.org/10.1109/ICRA.2017.7989597
12. Joo, K., Oh, T.H., Kim, J., Kweon, I.S.: Globally optimal manhattan frame estimation in real-time. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1763–1771 (2016)
13. Joo, K., Oh, T.H., Rameau, F., Bazin, J.C., Kweon, I.S.: Linear rgb-d slam for atlanta world. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 1077–1083. IEEE (2020)
14. Kim, P., Coltin, B., Jin Kim, H.: Linear rgb-d slam for planar environments. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 333–348 (2018)
15. Kim, P., Coltin, B., Kim, H.J.: Indoor rgb-d compass from a single line and plane. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4673–4680 (2018)
16. Lee, J.K., Yoon, K.J.: Real-time joint estimation of camera orientation and vanishing points. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1866–1874 (2015)
17. Li, H., Xing, Y., Zhao, J., Bazin, J., Liu, Z., Liu, Y.: Leveraging structural regularity of atlanta world for monocular slam. In: 2019 International

Conference on Robotics and Automation (ICRA). pp. 2412–2418 (2019). https://doi.org/10.1109/ICRA.2019.8793716

18. Li, X., Li, Y., Pınar Örnek, E., Lin, J., Tombari, F.: Co-planar parametrization for stereo-slam and visual-inertial odometry. arXiv e-prints pp. arXiv–2009 (2020)
19. Li, Y., Brasch, N., Wang, Y., Navab, N., Tombari, F.: Structure-slam: Low-drift monocular slam in indoor environments. IEEE Robotics and Automation Letters **5**(4), 6583–6590 (2020)
20. Li, Y., Yunus, R., Brasch, N., Navab, N., Tombari, F.: Rgb-d slam with structural regularities. In: 2021 IEEE international conference on Robotics and automation (ICRA) (2021)
21. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987)
22. Ma, L., Kerl, C., Stckler, J., Cremers, D.: Cpa-slam: Consistent plane-model alignment for direct rgb-d slam. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 1285–1291 (2016). https://doi.org/10.1109/ICRA.2016.7487260
23. Mei, C., Sibley, G., Newman, P.: Closing loops without places. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3738–3744. IEEE (2010)
24. Mur-Artal, Raúl, M.J.M.M., Tardós, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Transactions on Robotics **31**(5), 1147–1163 (2015). https://doi.org/10.1109/TRO.2015.2463671
25. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE transactions on robotics **31**(5), 1147–1163 (2015)
26. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. IEEE Transactions on Robotics **33**(5), 1255–1262 (2017). https://doi.org/10.1109/TRO.2017.2705103
27. Nießner, M., Zollhöfer, M., Izadi, S., Stamminger, M.: Real-time 3d reconstruction at scale using voxel hashing. ACM Transactions on Graphics (ToG) **32**(6), 1–11 (2013)
28. Nistér, D.: An efficient solution to the five-point relative pose problem. IEEE transactions on pattern analysis and machine intelligence **26**(6), 756–770 (2004)
29. Pomerleau, F., Colas, F., Siegwart, R.: A review of point cloud registration algorithms for mobile robotics. Foundations and Trends in Robotics **4**(1), 1–104 (2015)
30. Rosinol, A., Sattler, T., Pollefeys, M., Carlone, L.: Incremental visual-inertial 3d mesh generation with structural regularities. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 8220–8226. IEEE (2019)
31. Salaün, Y., Marlet, R., Monasse, P.: Robust and accurate line-and/or point-based pose estimation without manhattan assumptions. In: European Conference on Computer Vision. pp. 801–818. Springer (2016)
32. Schps, T., Sattler, T., Pollefeys, M.: Bad slam: Bundle adjusted direct rgb-d slam. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 134–144 (2019). https://doi.org/10.1109/CVPR.2019.00022
33. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A Benchmark for the Evaluation of RGB-D SLAM Systems. In: IEEE International Conference on Intelligent Robots and Systems (2012)
34. Tardif, J.P.: Non-iterative approach for fast and accurate vanishing point detection. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 1250–1257. IEEE (2009)

35. Von Gioi, R.G., Jakubowicz, J., Morel, J.M., Randall, G.: LSD: A Fast Line Segment Detector with a False Detection Control. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(4), 722–732 (2010)
36. Wang, R., Schworer, M., Cremers, D.: Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3903–3911 (2017)
37. Whelan, T., Leutenegger, S., Salas-Moreno, R., Glocker, B., Davison, A.: Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems (2015)
38. Yunus, R., Li, Y., Tombari, F.: Manhattanslam: Robust planar tracking and mapping leveraging mixture of manhattan frames. arXiv preprint arXiv:2103.15068 (2021)
39. Zhang, X., Wang, W., Qi, X., Liao, Z., Wei, R.: Point-plane slam using supposed planes for indoor environments. Sensors **19**(17), 3795 (2019)
40. Zhao, J., Kneip, L., He, Y., Ma, J.: Minimal case relative pose computation using ray-point-ray features. IEEE transactions on pattern analysis and machine intelligence **42**(5), 1176–1190 (2019)
41. Zhou, H., Zou, D., Pei, L., Ying, R., Liu, P., Yu, W.: Structslam: Visual slam with building structure lines. IEEE Transactions on Vehicular Technology **64**(4), 1364–1375 (2015). https://doi.org/10.1109/TVT.2015.2388780
42. Zhou, Q.Y., Koltun, V.: Dense scene reconstruction with points of interest. ACM Transactions on Graphics (ToG) **32**(4), 1–8 (2013)
43. Zhou, Y., Kneip, L., Rodriguez, C., Li, H.: Divide and conquer: Efficient density-based tracking of 3d sensors in manhattan worlds. In: Asian Conference on Computer Vision. pp. 3–19. Springer (2016)

SPRINGER NATURE ORDER DETAILS

Dec 14, 2023

This Agreement between Yan Li ("You") and Springer Nature ("Springer Nature") consists of your order details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

Order Number	501868236
Order date	Dec 12, 2023
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Springer eBook
Licensed Content Title	E-Graph: Minimal Solution for Rigid Rotation with Extensibility Graphs
Licensed Content Author	Yanyan Li, Federico Tombari
Licensed Content Date	Jan 1, 2022
Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	1 - 29
Author of this Springer Nature content	yes
Title of new work	Tracking and Mapping with Structural Regularities
Institution name	Technical University of Munich
Expected presentation date	Dec 2023
Requestor Location	TUM Boltzmannstr. 3 Garching b. München, 85748 Germany Attn: TUM
Billing Type	Invoice
Billing Address	TUM Boltzmannstr. 3 Garching b. München, Germany 85748 Attn: TUM
Total	0.00 USD

Terms and Conditions

Springer Nature Customer Service Centre GmbH Terms and Conditions

The following terms and conditions ("Terms and Conditions") together with the terms specified in your [RightsLink] constitute the License ("License") between you as Licensee and Springer Nature Customer Service Centre GmbH as Licensor. By clicking 'accept' and completing the transaction for your use of the material ("Licensed

Material"), you confirm your acceptance of and obligation to be bound by these Terms and Conditions.

1. Grant and Scope of License

1. 1. The Licensor grants you a personal, non-exclusive, non-transferable, non-sublicensable, revocable, world-wide License to reproduce, distribute, communicate to the public, make available, broadcast, electronically transmit or create derivative works using the Licensed Material for the purpose(s) specified in your RightsLink Licence Details only. Licenses are granted for the specific use requested in the order and for no other use, subject to these Terms and Conditions. You acknowledge and agree that the rights granted to you under this License do not include the right to modify, edit, translate, include in collective works, or create derivative works of the Licensed Material in whole or in part unless expressly stated in your RightsLink Licence Details. You may use the Licensed Material only as permitted under this Agreement and will not reproduce, distribute, display, perform, or otherwise use or exploit any Licensed Material in any way, in whole or in part, except as expressly permitted by this License.

1. 2. You may only use the Licensed Content in the manner and to the extent permitted by these Terms and Conditions, by your RightsLink Licence Details and by any applicable laws.

1. 3. A separate license may be required for any additional use of the Licensed Material, e.g. where a license has been purchased for print use only, separate permission must be obtained for electronic re-use. Similarly, a License is only valid in the language selected and does not apply for editions in other languages unless additional translation rights have been granted separately in the License.

1. 4. Any content within the Licensed Material that is owned by third parties is expressly excluded from the License.

1. 5. Rights for additional reuses such as custom editions, computer/mobile applications, film or TV reuses and/or any other derivative rights requests require additional permission and may be subject to an additional fee. Please apply to journalpermissions@springernature.com or bookpermissions@springernature.com for these rights.

2. Reservation of Rights

Licensor reserves all rights not expressly granted to you under this License. You acknowledge and agree that nothing in this License limits or restricts Licensor's rights in or use of the Licensed Material in any way. Neither this License, nor any act, omission, or statement by Licensor or you, conveys any ownership right to you in any Licensed Material, or to any element or portion thereof. As between Licensor and you, Licensor owns and retains all right, title, and interest in and to the Licensed Material subject to the license granted in Section 1.1. Your permission to use the Licensed Material is expressly conditioned on you not impairing Licensor's or the applicable copyright owner's rights in the Licensed Material in any way.

3. Restrictions on use

3. 1. Minor editing privileges are allowed for adaptations for stylistic purposes or formatting purposes provided such alterations do not alter the original meaning or

intention of the Licensed Material and the new figure(s) are still accurate and representative of the Licensed Material. Any other changes including but not limited to, cropping, adapting, and/or omitting material that affect the meaning, intention or moral rights of the author(s) are strictly prohibited.

3. 2. You must not use any Licensed Material as part of any design or trademark.

3. 3. Licensed Material may be used in Open Access Publications (OAP), but any such reuse must include a clear acknowledgment of this permission visible at the same time as the figures/tables/illustration or abstract and which must indicate that the Licensed Material is not part of the governing OA license but has been reproduced with permission. This may be indicated according to any standard referencing system but must include at a minimum 'Book/Journal title, Author, Journal Name (if applicable), Volume (if applicable), Publisher, Year, reproduced with permission from SNCSC'.

4. STM Permission Guidelines

4. 1. An alternative scope of license may apply to signatories of the STM Permissions Guidelines ("STM PG") as amended from time to time and made available at https://www.stm-assoc.org/intellectual-property/permissions/permissions-guidelines/.

4. 2. For content reuse requests that qualify for permission under the STM PG, and which may be updated from time to time, the STM PG supersede the terms and conditions contained in this License.

4. 3. If a License has been granted under the STM PG, but the STM PG no longer apply at the time of publication, further permission must be sought from the Rightsholder. Contact journalpermissions@springernature.com or bookpermissions@springernature.com for these rights.

5. Duration of License

5. 1. Unless otherwise indicated on your License, a License is valid from the date of purchase ("License Date") until the end of the relevant period in the below table:

Reuse in a medical communications project	Reuse up to distribution or time period indicated in License
Reuse in a dissertation/thesis	Lifetime of thesis
Reuse in a journal/magazine	Lifetime of journal/magazine
Reuse in a book/textbook	Lifetime of edition
Reuse on a website	1 year unless otherwise specified in the License
Reuse in a presentation/slide kit/poster	Lifetime of presentation/slide kit/poster. Note: publication whether electronic or in print of presentation/slide kit/poster may require further permission.
Reuse in conference proceedings	Lifetime of conference proceedings
Reuse in an annual report	Lifetime of annual report

Reuse in training/CME materials	Reuse up to distribution or time period indicated in License
Reuse in newsmedia	Lifetime of newsmedia
Reuse in coursepack/classroom materials	Reuse up to distribution and/or time period indicated in license

6. Acknowledgement

6. 1. The Licensor's permission must be acknowledged next to the Licensed Material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract and must be hyperlinked to the journal/book's homepage.

6. 2. Acknowledgement may be provided according to any standard referencing system and at a minimum should include "Author, Article/Book Title, Journal name/Book imprint, volume, page number, year, Springer Nature".

7. Reuse in a dissertation or thesis

7. 1. Where 'reuse in a dissertation/thesis' has been selected, the following terms apply: Print rights of the Version of Record are provided for; electronic rights for use only on institutional repository as defined by the Sherpa guideline (www.sherpa.ac.uk/romeo/) and only up to what is required by the awarding institution.

7. 2. For theses published under an ISBN or ISSN, separate permission is required. Please contact journalpermissions@springernature.com or bookpermissions@springernature.com for these rights.

7. 3. Authors must properly cite the published manuscript in their thesis according to current citation standards and include the following acknowledgement: '*Reproduced with permission from Springer Nature*'.

8. License Fee

You must pay the fee set forth in the License Agreement (the "License Fees"). All amounts payable by you under this License are exclusive of any sales, use, withholding, value added or similar taxes, government fees or levies or other assessments. Collection and/or remittance of such taxes to the relevant tax authority shall be the responsibility of the party who has the legal obligation to do so.

9. Warranty

9. 1. The Licensor warrants that it has, to the best of its knowledge, the rights to license reuse of the Licensed Material. **You are solely responsible for ensuring that the material you wish to license is original to the Licensor and does not carry the copyright of another entity or third party (as credited in the published version).** If the credit line on any part of the Licensed Material indicates that it was reprinted or adapted with permission from another source, then you should seek additional permission from that source to reuse the material.

9. 2. EXCEPT FOR THE EXPRESS WARRANTY STATED HEREIN AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, LICENSOR PROVIDES THE LICENSED MATERIAL "AS IS" AND MAKES NO OTHER REPRESENTATION OR WARRANTY. LICENSOR EXPRESSLY DISCLAIMS ANY LIABILITY FOR ANY CLAIM ARISING FROM OR OUT OF THE CONTENT, INCLUDING BUT NOT LIMITED TO ANY ERRORS, INACCURACIES, OMISSIONS, OR DEFECTS CONTAINED THEREIN, AND ANY IMPLIED OR EXPRESS WARRANTY AS TO MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL LICENSOR BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, INDIRECT, PUNITIVE, OR EXEMPLARY DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, VIEWING OR USE OF THE LICENSED MATERIAL REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION APPLIES NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

10. Termination and Cancellation

10. 1. The License and all rights granted hereunder will continue until the end of the applicable period shown in Clause 5.1 above. Thereafter, this license will be terminated and all rights granted hereunder will cease.

10. 2. Licensor reserves the right to terminate the License in the event that payment is not received in full or if you breach the terms of this License.

11. General

11. 1. The License and the rights and obligations of the parties hereto shall be construed, interpreted and determined in accordance with the laws of the Federal Republic of Germany without reference to the stipulations of the CISG (United Nations Convention on Contracts for the International Sale of Goods) or to Germany´s choice-of-law principle.

11. 2. The parties acknowledge and agree that any controversies and disputes arising out of this License shall be decided exclusively by the courts of or having jurisdiction for Heidelberg, Germany, as far as legally permissible.

11. 3. This License is solely for Licensor's and Licensee's benefit. It is not for the benefit of any other person or entity.

Questions? For questions on Copyright Clearance Center accounts or website issues please contact springernaturesupport@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777. For questions on Springer Nature licensing please visit https://www.springernature.com/gp/partners/rights-permissions-third-party-distribution

Other Conditions:

Version 1.4 - Dec 2022

D. Abstracts of Publications not Discussed in This Dissertation

Open-Structure: a Structural Benchmark Dataset for SLAM Algorithms

Yanyan Li[1], Zhao Guo[2], Ze Yang[2], Yanbiao Sun[2], Liang Zhao[3] and Federico Tombari[1,4]

[1] Technical University of Munich
[2] Tianjin University
[3] University of Edinburgh
[4] Google

Abstract. This paper introduces a new benchmark dataset, Open-Structure, for evaluating visual odometry and SLAM methods, which directly equips point and line measurements, correspondences, structural associations, and co-visibility factor graphs instead of providing raw images. Based on the proposed benchmark dataset, these 2D or 3D data can be directly input to different stages of SLAM pipelines to avoid the impact of the data preprocessing modules in ablation experiments. First, we propose a dataset generator for real-world and simulated scenarios. In real-world scenes, it maintains the same observations and occlusions as actual feature extraction results. Those generated simulation sequences enhance the dataset's diversity by introducing various carefully designed trajectories and observations. Second, a SLAM baseline is proposed using our dataset to evaluate widely used modules in camera pose tracking, parametrization, and optimization modules. By evaluating these state-of-the-art algorithms across different scenarios, we discern each module's strengths and weaknesses within the camera tracking and optimization process. Our dataset and baseline are available at https://github.com/yanyan-li/Open-Structure.

wMPS-SLAM: An Online and Accurate Monocular Visual-wMPS SLAM System

Ze Yang[1], Yanyan Li[2], Yanbiao Sun[1], Jiarui Lin[1], Jigui Zhu[1]

[1] Tianjin University
[2] Technical University of Munich

Abstract. Tracking drift issues resulting from the lack of global positioning constraints prevent visual simultaneous localization and mapping (SLAM) systems from providing accurate pose estimation services, especially in indoor environments. Similar to the function of the Global Navigation Satellite System (GNSS), a new indoor equipment called workshop Measurement Positioning System (wMPS) has been developed to provide millimeter-level global positioning information in the large-volume metrology industrial community but is rarely applied to SLAM solutions. In this paper, we introduce a novel wMPS-SLAM framework that combines 3 degree-of-freedom (DoF) wMPS signals into our monocular pose estimation algorithm, enabling drift-free global 6DoF pose estimation in indoor scenarios. Firstly, a system calibration method is proposed to establish the relationship between the camera and wMPS and fuse the sensor information. Secondly, a variant similarity transformation scheme is employed to jointly estimate the scale factor and extrinsic parameters of the camera frame w.r.t the world frame for estimating the global camera poses. Lastly, a pose graph optimization method that minimizes global position and relative pose errors is utilized to fuse Visual Odometry (VO) estimation and wMPS measurements. Evaluation experiments conducted on two datasets demonstrate that wMPS-SLAM outperforms the state-of-the-art monocular visual solution (ORB-SLAM3) in terms of accuracy. Specifically, the absolute translation error (ATE) of the proposed system is reduced from 28.58mm to 7.018mm in real-world experiments.

Tightly-coupled fusion of iGPS measurements in optimization-based visual SLAM

Ze Yang[1], Yanyan Li[2], Jiarui Lin[1], Yanbiao Sun[1], Jigui Zhu[1]

[1] Tianjin University
[2] Technical University of Munich

Abstract. The monocular visual Simultaneous Localization and Mapping (SLAM) can achieve accurate and robust pose estimation with excellent perceptual ability. However, accumulated image error over time brings out excessive trajectory drift in a GPS-denied indoor environment lacking global positioning constraints. In this paper, we propose a novel optimization-based SLAM fusing rich visual features and indoor GPS (iGPS) measurements, obtained by workshop Measurement Position System, (wMPS), to tackle the problem of trajectory drift associated with visual SLAM. Here, we first calibrate the spatial shift and temporal offset of two types of sensors using multi-view alignment and pose optimization bundle adjustment (BA) algorithms, respectively. Then, we initialize camera poses and map points in a unified world frame by iGPS-aided monocular initialization and PnP algorithms. Finally, we employ a tightly-coupled fusion of iGPS measurements and visual observations using a pose optimization strategy for high-accuracy global localization and mapping. In experiments, public datasets and self-collected sequences are used to evaluate the performance of our approach. The proposed system improves the result of absolute trajectory error from the current state-of-the-art 19.16mm (ORB-SLAM3) to 5.87mm in the public dataset and from 31.20mm to 5.85mm in the real-world experiment. Furthermore, the proposed system also shows good robustness in the evaluations.

SRH-Net: Stacked Recurrent Hourglass Network for Stereo Matching.

Hongzhi Du[1], Yanyan Li[2], Yanbiao Sun[1], Jigui Zhu[1], Federico Tombari[2,3]

[1] Tianjin University
[2] Technical University of Munich
[3] Google

Abstract. The cost aggregation strategy shows a crucial role in learning-based stereo matching tasks, where 3D convolutional filters obtain state of the art but require intensive computation resources, while 2D operations need less GPU memory but are sensitive to domain shift. In this paper, we decouple the 4D cubic cost volume used by 3D convolutional filters into sequential cost maps along the direction of disparity instead of dealing with it at once by exploiting a recurrent cost aggregation strategy. Furthermore, a novel recurrent module, Stacked Recurrent Hourglass (SRH), is proposed to process each cost map. Our hourglass network is constructed based on Gated Recurrent Units (GRUs) and down/upsampling layers, which provides GRUs larger receptive fields. Then two hourglass networks are stacked together, while multi-scale information is processed by skip connections to enhance the performance of the pipeline in textureless areas. The proposed architecture is implemented in an end-to-end pipeline and evaluated on public datasets, which reduces GPU memory consumption by up to 56.1% compared with PSMNet using stacked hourglass 3D CNNs without the degradation of accuracy. Then, we further demonstrate the scalability of the proposed method on several high-resolution pairs, while previously learned approaches often fail due to the memory constraint. The code is released at https://github.com/hongzhidu/SRHNet.

ManhattanSLAM: Robust Planar Tracking and Mapping Leveraging Mixture of Manhattan Frames

Raza Yunus[1], Yanyan Li[1], Federico Tombari[1,2]

[1] Technical University of Munich
[2] Google

Abstract. In this paper, a robust RGB-D SLAM system is proposed to utilize the structural information in indoor scenes, allowing for accurate tracking and efficient dense mapping on a CPU. Prior works have used the Manhattan World (MW) assumption to estimate low-drift camera pose, in turn limiting the applications of such systems. This paper, in contrast, proposes a novel approach delivering robust tracking in MW and non-MW environments. We check orthogonal relations between planes to directly detect Manhattan Frames, modeling the scene as a Mixture of Manhattan Frames. For MW scenes, we decouple pose estimation and provide a novel drift-free rotation estimation based on Manhattan Frame observations. For translation estimation in MW scenes and full camera pose estimation in non-MW scenes, we make use of point, line and plane features for robust tracking in challenging scenes. Additionally, by exploiting plane features detected in each frame, we also propose an efficient surfel-based dense mapping strategy, which divides each image into planar and non-planar regions. Planar surfels are initialized directly from sparse planes in our map while non-planar surfels are built by extracting superpixels. We evaluate our method on public benchmarks for pose estimation, drift and reconstruction accuracy, achieving superior performance compared to other state-of-the-art methods. We will open-source our code in the future.

Bibliography

[1] Y. Li, Z. Guo, Z. Yang, Y. Sun, L. Zhao, and F. Tombari. “Open-Structure: a Structural Benchmark Dataset for SLAM Algorithms”. In: *arXiv preprint arXiv:2310.10931* (2023) (see pp. 1, 46).

[2] Z. Yang, Y. Li, J. Lin, Y. Sun, and J. Zhu. “wMPS-SLAM: An Online and Accurate Monocular Visual-wMPS SLAM System”. In: *IEEE Transactions on Instrumentation and Measurement* 72 (2023), pp. 1–11 (see p. 1).

[3] Z. Yang, Y. Li, J. Lin, Y. Sun, and J. Zhu. “Tightly-coupled fusion of iGPS measurements in optimization-based visual SLAM”. In: *Optics Express* 31.4 (2023), pp. 5910–5926 (see p. 1).

[4] Y. Li and F. Tombari. “E-Graph: Minimal Solution for Rigid Rotation with Extensibility Graphs”. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*. Springer. 2022, pp. 306–322 (see pp. 1, 12).

[5] Y. Li, R. Yunus, N. Brasch, N. Navab, and F. Tombari. “RGB-D SLAM with structural regularities”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 11581–11587 (see pp. 1, 12, 27, 61).

[6] R. Yunus, Y. Li, and F. Tombari. “Manhattanslam: Robust planar tracking and mapping leveraging mixture of manhattan frames”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 6687–6693 (see p. 1).

[7] H. Du, Y. Li, Y. Sun, J. Zhu, and F. Tombari. “SRH-Net: Stacked recurrent hourglass network for stereo matching”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 8005–8012 (see p. 1).

[8] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari. “Structure-slam: Low-drift monocular slam in indoor environments”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6583–6590 (see pp. 2, 12, 40, 46, 49).

[9] X. Li, Y. Li, E. P. Örnek, J. Lin, and F. Tombari. “Co-planar parametrization for stereo-SLAM and visual-inertial odometry”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6972–6979 (see pp. 2, 12, 40).

[10] J. J. Leonard and H. F. Durrant-Whyte. “Simultaneous map building and localization for an autonomous mobile robot.” In: *IROS*. Vol. 3. 1991, pp. 1442–1447 (see p. 8).

[11] R. C. Smith and P. Cheeseman. “On the representation and estimation of spatial uncertainty”. In: *The international journal of Robotics Research* 5.4 (1986), pp. 56–68 (see p. 8).

[12] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. “A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM”. In: *2014 IEEE international conference on Robotics and automation (ICRA)*. IEEE. 2014, pp. 1524–1531 (see pp. 9, 47).

[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163 (see pp. 9, 18, 19, 21, 26–29, 31, 45, 49, 126).

[14] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. "Kimera: an open-source library for real-time metric-semantic localization and mapping". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1689–1696 (see pp. 9, 28, 40, 49).

[15] T. Qin, P. Li, and S. Shen. "Vins-mono: A robust and versatile monocular visual-inertial state estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020 (see pp. 9, 18, 21, 27–29, 33, 49).

[16] L. von Stumberg and D. Cremers. "Dm-vio: Delayed marginalization visual-inertial odometry". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1408–1415 (see p. 9).

[17] J. Engel, V. Koltun, and D. Cremers. "Direct sparse odometry". In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 611–625 (see pp. 9, 52).

[18] M. Labbé and F. Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation". In: *Journal of Field Robotics* 36.2 (2019), pp. 416–446 (see pp. 9, 28).

[19] W. Förstner and E. Gülch. "A fast operator for detection and precise location of distinct points, corners and centres of circular features". In: *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*. Vol. 6. Interlaken. 1987, pp. 281–305 (see p. 15).

[20] C. Harris, M. Stephens, et al. "A combined corner and edge detector". In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244 (see p. 15).

[21] C. Tomasi and T. Kanade. "Detection and tracking of point". In: *Int J Comput Vis* 9 (1991), pp. 137–154 (see p. 15).

[22] E. Rosten, R. Porter, and T. Drummond. "Faster and better: A machine learning approach to corner detection". In: *IEEE transactions on pattern analysis and machine intelligence* 32.1 (2008), pp. 105–119 (see pp. 15, 16).

[23] "Object recognition from local scale-invariant features". In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee. 1999, pp. 1150–1157 (see pp. 15, 16, 21).

[24] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. "Speeded-up robust features (SURF)". In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359 (see p. 15).

[25] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571 (see p. 15).

[26] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "BRIEF: Binary Robust Independent Elementary Features". In: *Computer Vision – ECCV 2010*. Ed. by K. Daniilidis, P. Maragos, and N. Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792 (see p. 16).

[27] J. B. Burns, A. R. Hanson, and E. M. Riseman. "Extracting Straight Lines". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.4 (1986), pp. 425–455 (see p. 16).

[28] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. "LSD: A fast line segment detector with a false detection control". In: *IEEE transactions on pattern analysis and machine intelligence* 32.4 (2008), pp. 722–732 (see p. 16).

[29] C. Akinlar and C. Topal. "EDLines: A real-time line segment detector with a false detection control". In: *Pattern Recognition Letters* 32.13 (2011), pp. 1633–1642 (see p. 16).

[30] L. Zhang and R. Koch. "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency". In: *Journal of Visual Communication and Image Representation* 24.7 (2013), pp. 794–805 (see p. 16).

[31] R. Schnabel, R. Wahl, and R. Klein. "Efficient RANSAC for point-cloud shape detection". In: *Computer graphics forum*. Vol. 26. 2. Wiley Online Library. 2007, pp. 214–226 (see p. 16).

[32] B. Oehler, J. Stueckler, J. Welle, D. Schulz, and S. Behnke. “Efficient multi-resolution plane segmentation of 3D point clouds”. In: *Intelligent Robotics and Applications: 4th International Conference, ICIRA 2011, Aachen, Germany, December 6-8, 2011, Proceedings, Part II 4*. Springer. 2011, pp. 145–156 (see p. 16).

[33] C. Feng, Y. Taguchi, and V. R. Kamat. “Fast plane extraction in organized point clouds using agglomerative hierarchical clustering”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 6218–6225 (see p. 16).

[34] C. Feng, Y. Taguchi, and V. R. Kamat. “Fast plane extraction in organized point clouds using agglomerative hierarchical clustering”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 6218–6225 (see p. 16).

[35] J. Civera, A. J. Davison, and J. M. Montiel. “Inverse depth parametrization for monocular SLAM”. In: *IEEE transactions on robotics* 24.5 (2008), pp. 932–945 (see pp. 17, 29, 49).

[36] A. Bartoli and P. Sturm. “Structure-from-motion using lines: Representation, triangulation, and bundle adjustment”. In: *Computer vision and image understanding* 100.3 (2005), pp. 416–441 (see pp. 17, 50).

[37] L. Ma, C. Kerl, J. Stückler, and D. Cremers. “CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1285–1291 (see p. 17).

[38] M. Pizzoli, C. Forster, and D. Scaramuzza. “REMODE: Probabilistic, monocular dense reconstruction in real time”. In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 2609–2616 (see p. 17).

[39] C. Forster, M. Pizzoli, and D. Scaramuzza. “SVO: Fast semi-direct monocular visual odometry”. In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 15–22 (see pp. 17, 59).

[40] G. Vogiatzis and C. Hernández. “Video-based, real-time multi-view stereo”. In: *Image and Vision Computing* 29.7 (2011), pp. 434–441 (see p. 17).

[41] A. Elfes and L. Matthies. “Sensor integration for robot navigation: Combining sonar and stereo range data in a grid-based representataion”. In: *26th IEEE conference on decision and control*. Vol. 26. IEEE. 1987, pp. 1802–1807 (see p. 17).

[42] B. Curless and M. Levoy. “A volumetric method for building complex models from range images”. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 1996, pp. 303–312 (see p. 17).

[43] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. “Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera”. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 2011, pp. 559–568 (see p. 17).

[44] W. E. Lorensen and H. E. Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *ACM siggraph computer graphics* 21.4 (1987), pp. 163–169 (see p. 17).

[45] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. “KinectFusion: Real-time dense surface mapping and tracking”. In: *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE. 2011, pp. 127–136 (see pp. 17, 46, 59).

[46] M. Irani, B. Rousso, and S. Peleg. “Computing occluding and transparent motions”. In: *International Journal of Computer Vision* 12 (1994), pp. 5–16 (see p. 18).

[47] M. J. Black and P. Anandan. “The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields”. In: *Computer vision and image understanding* 63.1 (1996), pp. 75–104 (see p. 18).

[48] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges”. In: *IJCAI*. Vol. 3. 2003. 2003, pp. 1151–1156 (see p. 18).

[49] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. “MonoSLAM: Real-time single camera SLAM”. In: *IEEE transactions on pattern analysis and machine intelligence* 29.6 (2007), pp. 1052–1067 (see p. 18).

[50] G. Klein and D. Murray. “Parallel Tracking and Mapping for Small AR Workspaces”. In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. 2007, pp. 225–234 (see pp. 18, 19).

[51] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. “Consistency of the EKF-SLAM algorithm”. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 3562–3568 (see p. 18).

[52] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis. “Observability-based rules for designing consistent EKF SLAM estimators”. In: *The International Journal of Robotics Research* 29.5 (2010), pp. 502–528 (see p. 18).

[53] F. Janabi-Sharifi and M. Marey. “A kalman-filter-based method for pose estimation in visual servoing”. In: *IEEE transactions on Robotics* 26.5 (2010), pp. 939–947 (see p. 18).

[54] R. Mur-Artal and J. D. Tardós. “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras”. In: *IEEE transactions on robotics* 33.5 (2017), pp. 1255–1262 (see pp. 18, 19, 29, 31, 45, 48, 50, 53, 59).

[55] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós. “Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890 (see pp. 20, 31, 49, 126).

[56] L. Juan and O. Gwun. “A comparison of sift, pca-sift and surf”. In: *International Journal of Image Processing (IJIP)* 3.4 (2009), pp. 143–152 (see p. 20).

[57] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571 (see p. 20).

[58] A. Bruhn, J. Weickert, and C. Schnörr. “Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods”. In: *International journal of computer vision* 61 (2005), pp. 211–231 (see p. 20).

[59] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. “BRIEF: Computing a local binary descriptor very fast”. In: *IEEE transactions on pattern analysis and machine intelligence* 34.7 (2011), pp. 1281–1298 (see p. 21).

[60] B. D. Lucas and T. Kanade. “An iterative image registration technique with an application to stereo vision”. In: *IJCAI’81: 7th international joint conference on Artificial intelligence*. Vol. 2. 1981, pp. 674–679 (see p. 22).

[61] D. Nistér. “An efficient solution to the five-point relative pose problem”. In: *IEEE transactions on pattern analysis and machine intelligence* 26.6 (2004), pp. 756–770 (see p. 23).

[62] R. I. Hartley. “In defense of the eight-point algorithm”. In: *IEEE Transactions on pattern analysis and machine intelligence* 19.6 (1997), pp. 580–593 (see p. 23).

[63] H. C. Longuet-Higgins. “A computer algorithm for reconstructing a scene from two projections”. In: *Nature* 293.5828 (1981), pp. 133–135 (see p. 23).

[64] V. Lepetit, F. Moreno-Noguer, and P. Fua. “EP n P: An accurate O (n) solution to the P n P problem”. In: *International journal of computer vision* 81 (2009), pp. 155–166 (see pp. 24, 33).

[65] L. Kneip, H. Li, and Y. Seo. “Upnp: An optimal o (n) solution to the absolute pose problem with universal applicability”. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer. 2014, pp. 127–142 (see pp. 24, 48).

[66] P. Kim, B. Coltin, and H. J. Kim. “Low-drift visual odometry in structured environments by decoupling rotational and translational motion”. In: *2018 IEEE international conference on Robotics and automation (ICRA)*. IEEE. 2018, pp. 7247–7253 (see pp. 24, 46, 56).

[67] Y. Zhou, L. Kneip, C. Rodriguez, and H. Li. “Divide and conquer: Efficient density-based tracking of 3D sensors in Manhattan worlds”. In: *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part V 13*. Springer. 2017, pp. 3–19 (see pp. 24, 46, 57).

[68] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige. “g2o: A general framework for (hyper) graph optimization”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 9–13 (see p. 29).

[69] D. Wisth, M. Camurri, and M. Fallon. “VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots”. In: *IEEE Transactions on Robotics* (2022) (see p. 31).

[70] D. Gálvez-López and J. D. Tardos. “Bags of binary words for fast place recognition in image sequences”. In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197 (see p. 31).

[71] L. Zhou, S. Wang, and M. Kaess. “DPLVO: Direct point-line monocular visual odometry”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7113–7120 (see pp. 40, 52, 55).

[72] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cireşan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella. “Max-pooling convolutional neural networks for vision-based hand gesture recognition”. In: *2011 IEEE international conference on signal and image processing applications (ICSIPA)*. IEEE. 2011, pp. 342–347 (see p. 41).

[73] C. Papageorgiou and T. Poggio. “A trainable system for object detection”. In: *International journal of computer vision* 38 (2000), pp. 15–33 (see p. 41).

[74] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. “Understanding convolution for semantic segmentation”. In: *2018 IEEE winter conference on applications of computer vision (WACV)*. Ieee. 2018, pp. 1451–1460 (see p. 41).

[75] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov. “Deep learning classification of land cover and crop types using remote sensing data”. In: *IEEE Geoscience and Remote Sensing Letters* 14.5 (2017), pp. 778–782 (see p. 41).

[76] M. D. Zeiler and R. Fergus. “Stochastic pooling for regularization of deep convolutional neural networks”. In: *arXiv preprint arXiv:1301.3557* (2013) (see p. 41).

[77] A. F. Agarap. “Deep learning using rectified linear units (relu)”. In: *arXiv preprint arXiv:1803.08375* (2018) (see p. 42).

[78] J. Han and C. Moraga. “The influence of the sigmoid function parameters on the speed of backpropagation learning”. In: *International workshop on artificial neural networks*. Springer. 1995, pp. 195–201 (see p. 42).

[79] R. A. Dunne and N. A. Campbell. “On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function”. In: *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*. Vol. 181. Citeseer. 1997, p. 185 (see p. 42).

[80] Y. LeCun et al. “LeNet-5, convolutional neural networks”. In: *URL: http://yann. lecun. com/exdb/lenet* 20.5 (2015), p. 14 (see p. 43).

[81] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer. “PL-SLAM: Real-time monocular visual SLAM with points and lines”. In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 4503–4508 (see p. 45).

[82] S. J. Lee and S. S. Hwang. “Elaborate monocular point and line slam with robust initialization”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1121–1129 (see p. 45).

[83] D. M. Rosen, K. J. Doherty, A. Terán Espinoza, and J. J. Leonard. “Advances in inference and representation for simultaneous localization and mapping”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 4 (2021), pp. 215–242 (see p. 46).

[84] P. Kim, B. Coltin, and H. J. Kim. “Visual Odometry with Drift-Free Rotation Estimation Using Indoor Scene Regularities.” In: *BMVC*. Vol. 2. 6. 2017, p. 7 (see p. 46).

[85] Y. Li, R. Yunus, N. Brasch, N. Navab, and F. Tombari. “RGB-D SLAM with Structural Regularities”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 11581–11587 (see p. 46).

[86] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. J. Kelly, and S. Leutenegger. “Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping”. In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1144–1151 (see p. 46).

[87] B. Curless and M. Levoy. “A Volumetric Method for Building Complex Models from Range Images”. In: *ACM* (1996) (see p. 46).

[88] W. E. Lorensen and H. E. Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *ACM SIGGRAPH Computer Graphics* (1987), pp. 163–169 (see p. 46).

[89] W. E Lorense. “A High Resolution 3D Surface Construction Algorithm”. In: *Proc Siggraph* (1987) (see p. 46).

[90] R. Gomez-Ojeda, F.-A. Moreno, D. Zuniga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez. “PL-SLAM: A stereo SLAM system through the combination of points and line segments”. In: *IEEE Transactions on Robotics* 35.3 (2019), pp. 734–746 (see pp. 49, 50).

[91] X. Zhang, W. Wang, X. Qi, Z. Liao, and R. Wei. “Point-Plane SLAM Using Supposed Planes for Indoor Environments”. In: *Sensors* 19.17 (2019) (see pp. 52, 53).

[92] A. Rosinol, T. Sattler, M. Pollefeys, and L. Carlone. “Incremental visual-inertial 3d mesh generation with structural regularities”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8220–8226 (see p. 53).

[93] A. Rosinol. “Densifying sparse vio: a mesh-based approach using structural regularities”. MA thesis. ETH Zurich; Massachusetts Institute of Technology (MIT), 2018 (see p. 54).

[94] X. Li, Y. He, J. Lin, and X. Liu. “Leveraging planar regularities for point line visual-inertial odometry”. In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2020, pp. 5120–5127 (see p. 54).

[95] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu. “StructSLAM: Visual SLAM With Building Structure Lines”. In: *IEEE Transactions on Vehicular Technology* 64.4 (2015), pp. 1364–1375 (see p. 54).

[96] L. Zhou, G. Huang, Y. Mao, S. Wang, and M. Kaess. “EDPLVO: Efficient Direct Point-Line Visual Odometry”. In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 7559–7565 (see p. 55).

[97] C.-H. Chang and N. Kehtarnavaz. “Computationally efficient vanishing point detection for camera applications”. In: *2014 IEEE International Conference on Consumer Electronics (ICCE)*. 2014, pp. 39–40 (see p. 55).

[98] P. Kim, B. Coltin, and H. J. Kim. “Visual Odometry with Drift-Free Rotation Estimation Using Indoor Scene Regularities.” In: *BMVC*. Vol. 2. 6. 2017, p. 7 (see p. 57).

[99] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. “Real-time 3D reconstruction at scale using voxel hashing”. In: *ACM Transactions on Graphics (ToG)* 32.6 (2013), pp. 1–11 (see p. 59).

[100] J. Huang, A. Dai, L. J. Guibas, and M. Nießner. “3Dlite: towards commodity 3D scanning for content creation.” In: *ACM Trans. Graph.* 36.6 (2017), pp. 203–1 (see p. 61).

[101] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Nießner. “Scan2cad: Learning cad model alignment in rgb-d scans”. In: *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 2019, pp. 2614–2623 (see p. 61).

[102] Z. Jiang, C.-C. Hsu, and Y. Zhu. “Ditto: Building digital twins of articulated objects from interaction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5616–5626 (see p. 61).

[103] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert. “Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization”. In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, pp. 4597–4604 (see p. 128).

[104] Y. Lu, D. Song, and J. Yi. “High level landmark-based visual navigation using unsupervised geometric constraints in local bundle adjustment”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 1540–1545 (see p. 137).

List of Figures