IMPROVING THE APPLICABILITY OF VISUAL SLAM WITH SUBMODULAR SUBMATRIX SELECTION

A Dissertation
Presented to
The Academic Faculty

By

Yipu Zhao

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

Georgia Institute of Technology

December 2019

IMPROVING THE APPLICABILITY OF VISUAL SLAM WITH SUBMODULAR SUBMATRIX SELECTION

Approved by:

Dr. Patricio A. Vela, Advisor
School of Electrical Engineering
Georgia Institute of Technology

Dr. Anthony J. Yezzi
School of Electrical Engineering
Georgia Institute of Technology

Dr. Ghassan AlRegib
School of Electrical Engineering
Georgia Institute of Technology

Dr. Frank Dellaert
School of Interactive Computing
Georgia Institute of Technology

Dr. Yongkwon Cho
School of Civil and Environmental Engineering
Georgia Institute of Technology

Date Approved: July 29, 2019

A time will come to ride the wind and cleave the waves;

I will set my cloud-like sail to cross the sea which raves.

Bai Li, 744 A.D.

To my family, and the people who have faith in me.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Patricio A. Vela, for his support and guidance over the past four years. The nervous young man has learned a lot from you, since the afternoon he walked into your office for the first time. Throughout the years, you are not just offering me suggestions and insights on my research, but also leading by example on solving challenging problems with strategy. This thesis would not be possible without your support and guidance.

I would like to thank Dr. Anthony J. Yezzi for serving as the chair of my reading committee, for teaching me about the elegant mathematics in PDE, and for supporting me multiple times when I need it the most. I would also like to thank Dr. Ghassan AlRegib, who served as my reading committee and taught me beautiful formulations in digital image processing; Dr. Frank Dellaert, who inspired me to choose this research topic, and took time out from his insanely busy schedule to be on my committee; Dr. Yongkwon Cho, who agreed to be my committee on such short notice.

I am grateful to the collaborators and colleagues I have met in Georgie Tech. To Miguel, I will always remember the warm welcome and mentoring since day one. You taught me to keep optimistic no matter what happened. To Wenkai, it's always a pleasure working with you. I miss those days when we spent solid hours working on some hard problems. To Sam, for the generous day-to-day support over the years. To Justin, for countless skills (and fun facts) you shared with me. To Alex, Luisa, Fu-jen, Rui-nian and Shi-yu, for all the time we spent together inside and outside of IVALab.

More than anyone else, I am deeply grateful to my parents. Six years, two continents, eight thousand miles, they have always been supporting me, encouraging me, loving me. I cannot survive the grind without them. Finally, I would like to thank my angel, Yolanda. You support me, believe in me, and put your love in me, even on those dark days, especially on those dark days. You are the moon of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

BA	Bundle Adjustment
EIF	Extended Information Filter
EKF	Extended Kalman Filter
KF	Key Frame
KLT	Kanade-Lucas-Tomasi
HPL	Homogeneous-Points Line Parameterization
IDL	Inverse-Depth-Points Line Parameterization
LSQ	Least Squares
MSCKF	Multi-State Constraint Kalman Filter
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus
RMSE	Root Mean Square Error
RPE	Relative Position Error
ROE	Relative Orientation Error
SfM	Structure from Motion
SLAM	Simultaneous Localization and Mapping
VINS	Visual-Inertial State Estimation
VO	Visual Odometry
VSLAM	Visual Simultaneous Localization and Mapping
VI-SLAM	Visual-Inertial Simultaneous Localization and Mapping

SUMMARY

The objective of the thesis is to improve the applicability of Visual Simultaneous Localization and Mapping (VSLAM) on diverse platforms and scenarios, which has broad impact on practical applications in Robotics and Argumented Reality (AR).

Traditionally, a large fraction of effort on VSLAM has focused on performance, for instance accurate pose tracking, dense mapping, etc. The computation cost of VSLAM, on the other hand, is commonly overlooked: many VSLAM systems have to run on desktop CPUs or even GPUs to meet real-time requirements. Until very recently, the applicability of VSLAM draws the attention of community, with target applications on diverse platforms (e.g. micro flying vehicles, AR headset) and scenarios (e.g. low-texture, fast motion). However, state-of-the-art applicable VSLAM involves design choices that trade efficiency with significant sacrifice of performance, therefore with low tracking accuracy and high sensitivity to working environment.

In this thesis, we study feature-based BA SLAM, which has high performance in general but also high computation cost. A series of improvements are proposed to improve both the efficiency and performance of feature-based BA SLAM for diverse platforms and scenarios. As recognized in the SfM and SLAM community, the structure of the SLAM problem can be represented with two equivalent representations: factor graph and Jacobian matrix. From the perspective of information preservation, the full factor graph that contains many inter-connections between nodes should be used. However, for a real-time applicable SLAM, a small and sparse graph (Jacobian) is preferred. A rich body of work has explored offline or posterior graph sparsification. Instead, the scope of this thesis is on online graph selection and sparsification.

The thesis is based upon theorems developed in the submatrix selection literature. Originating from computational theory and machine learning, submatrix selection aims at identifying a subset of columns/rows from the original matrix, while maximizing matrix re-

vealing metrics such as the *Frobenius Norm* and *logDet*. An optimally selected submatrix not only preserves the most information from original matrix but is also much smaller and sparser than the original one. Small-size and sparsity are preferred for efficient numerical optimization; the performance-efficiency trade-off of optimization-related process such as VSLAM is improved thereafter. In Chapter 3, submatrix selection is introduced to guide the feature matching effort in the feature matching module of the VSLAM front-end. Chapter 4 extends the concept of submatrix selection to submatrix tuning, for improving the conditioning of the line-assisted VSLAM. In Chapter 5, the local map data structure and data selection process are explored, as it plays a critical role in VSLAM front-end robustness. Submatrix selection enables efficient construction and querying of a compact local map. In Chapter 6, submatrix selection is introduced to the BA-based VSLAM back-end, which results in a cost-effective back-end solution with superior performance when running on compute limited devices. Finally Chapter 7 explores VSLAM in mobile robotic systems for closed-loop and online usage. VSLAM pose estimation is integrated with high-rate inertial data to generate feedback control signal, which is crucial for close-loop navigation. The benefit of low-latency VSLAM is revealed in the closed-loop navigation, which is a major application of VSLAM.

CHAPTER 1
INTRODUCTION

Visual Odometry (VO) and Visual Simultaneous Localization and Mapping (VSLAM) are essential in a wide range of robotics and Augmented Reality (AR) applications. VO aims at local-consistent pose tracking with or without mapping; while VSLAM covers both local-consistent and global-consistent pose tracking, with explicit mapping and loop closing modules [1]. In the absence of absolute positioning signal, e.g. GPS (outdoor) or local electromagnetic beacons (indoor), VO/VSLAM complements traditional odometry methods such as wheel-based or inertial odometry.

Research over the past two decades has revealed a few key strategies for VO/VSLAM. A large fraction of effort has focused on the accuracy and robustness of pose tracking [2, 3, 4, 5, 6, 7] (and mapping [8, 9]), while meeting the real-time requirement (e.g. 30 fps) on desktops and laptops. However, the computational resources on practical robotics and AR platforms are more diverse and can be more limited. For instance, a micro flying vehicle (MAV) can only support lightweight computing kits [10], while AR headsets typically have ARM SoC with low power consumption. Although many state-of-the-art VO/VSLAM systems achieved good real-time performance on a PC or laptop with a powerful CPU, reaching the same level of performance on a less powerful device remains an open problem. Some VO/VSLAM systems fail to meet real-time processing under computational limits [11]. Other systems gain efficiency with significant performance loss [6, 10]. To improve the applicability of VO/VSLAM on practical robotics and AR applications, cost-efficiency of VO/VSLAM is essential, and has to be further improved.

In this chapter, we review the state-of-the-art in VSLAM and relevant topics of VO and VINS. After discussing the gap of applicability in the existing VO/VSLAM works, the contributions of this thesis are listed as an outline.

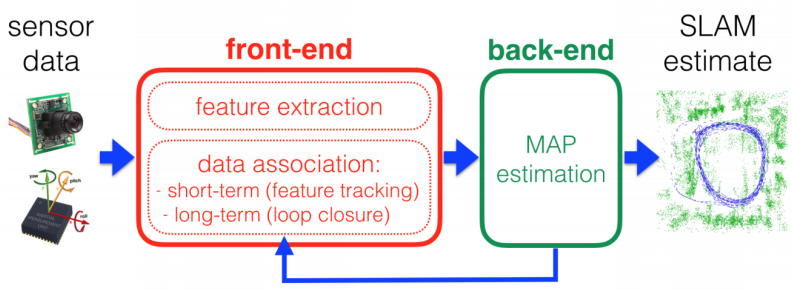


Figure 1.1: Typical pipeline of VSLAM/VINS [12]. Front-end consists of feature extraction and data association (feature tracking and loop closure detection). Back-end performs Maximum A Posteriori (MAP) estimation using front-end data associations.

1.1 Gap of Applicability in State-of-the-Art VSLAM/VINS Systems

This section covers the literature of VSLAM. We focus the discussion to VSLAM with frame-based, color cameras. VSLAM methods using alternative visual sensor such as event camera are excluded.

Two closely-related topics, VO and VINS are covered as well. Conventionally, VO can be considered as a component of VSLAM: VO aims at local-consistent pose tracking with or without mapping; while VSLAM covers both local-consistent and global-consistent pose tracking, with explicit mapping and loop closing modules [1]. VINS, meanwhile, can be considered as an extension of VSLAM with additional inertial input. VINS gains extra robustness by fusing visual input with inertial. For simplicity we use the term VSLAM to represent both VO and VSLAM.

The typical pipeline of VSLAM/VINS is illustrated in Fig 1.1. Two major component of VSLAM/VINS are an image processing font-end and a state estimation back-end. The role of front-end is to process input visual (and inertial) data, and to associate data captured at different time. It typically consists of feature extraction and data association (feature tracking and loop closure detection). The role of the back-end, meanwhile, is to estimate the state of the visual sensor (e.g. camera) and surrounding environment (e.g. map) using front-end measurements. In VSLAM back-end, the state estimation is typically posed as a Maximum A Posteriori (MAP) problem solved with filters or non-linear joint optimization.

1.1.1 Front-end of VSLAM/VINS

The front-end of VSLAM can be categorized into feature-based and direct methods, as illustrated in Fig 1.3:

Feature-based methods work with feature descriptors, which are computational efficient to extract and insensitive to image noise and view point change. Typically, a feature-based front-end consists of three modules: keypoint detection, feature(descriptor) extraction and feature matching. Most are based on point-feature binary descriptors, e.g. BRISK [13], ORB [14] and FREAK [15]. Due to the robustness and repeatability of point-feature binary descriptors, feature-based front-ends provide long-baseline data associations (illustrated in Fig 1.2), therefore are widely used in modern VSLAM (e.g. ORB-SLAM [4]) and VINS systems (e.g. OKVIS [16]). To increase the amount of long-baseline data associations, feature-based front-ends usually match current feature to a collection of historical features, i.e., local map. Systems that couple feature-based front-end and local map are highly accurate, and are robust in most scenarios where some textures exist. On low-textured scenarios where point feature may be lacking, e.g. corridors and hallways, line and edge can be utilized as alternative features. In the early days of VSLAM, lines were explored to cope with large view change of monocular camera tracking [17, 18]. More recently, with progress in line detection (e.g. LSD[19]) and descriptor (e.g. LBD[20]), line features have been demonstrated as reliable alternatives for VSLAM [21, 22, 23] and VINS [24].

The robustness of feature-based front-ends come with the price of high computational cost. With dedicated hardwares such as FPGA, certain computations that are highly parallelizable can be accelerated, e.g. feature extraction [26, 27, 28]. Nevertheless, matching the features between current image and the map is computationally expensive, since the cost of doing so scales linearly with the map size. To reduce the load in feature matching, direct front-ends omit the explicit feature extraction and matching module (partially or completely). One type of direct front-ends is based on optical flow methods such as KLT [29]. Compared with feature-based front-ends, front-ends using optical flow track pixels

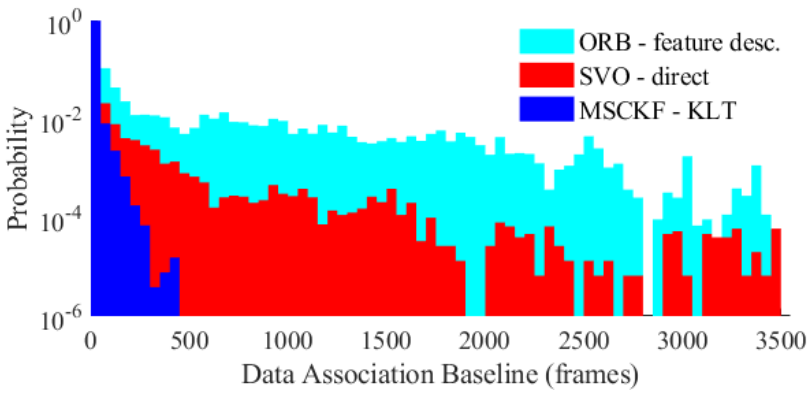


Figure 1.2: Distributions of data association baseline for 3 representative VSLAM front-ends when averaged on EuRoC MAV benchmark [25]: feature descriptor in ORB-SLAM (ORB) [4], KLT in MSCKF [10], and direct SVO [6]. For each association, the baseline is assessed with the length of life: from the first-measured frame to the last-measured frame. The feature-based front-end (ORB) extracts more long-baseline feature matchings than the KLT and direct methods.

in a short duration. Though computationally less costly, the tracking quality is worse than feature-based ones [30]. As a consequence, state-of-the-art VINS systems [31, 32] with KLT-based front-ends have worse tracking performance than feature-based ones [16]. Another direct method uses photometric error as an objective function to optimize for state estimation. These front-ends omit the explicit feature extraction and matching modules completely. The load of matching is postponed to the back-end optimization, which minimizes an direct objective with photometric residuals. In general, VSLAM [7, 6] and VINS [33] systems with photometric-based front-ends are more efficient than feature-based ones. To further improve the efficiency, state-of-the-art photometric-based front-ends only work with a sparse set of image patches, as opposed to the dense set used at the early stage [5].

The biggest issue of direct measurements, including KLT and photometric error, is the small region of attraction (due to the non-smooth nature of image) [34]. As a consequence, direct front-ends are sensitive to many factors: image noise, initial pose estimation, lighting condition changes, etc. Furthermore, immediate recovery from track failure (i.e., relocalization) is a known issue for direct systems. Therefore, direct systems require certain conditions [7, 35, 36] to work properly, e.g. global shutter camera with precise

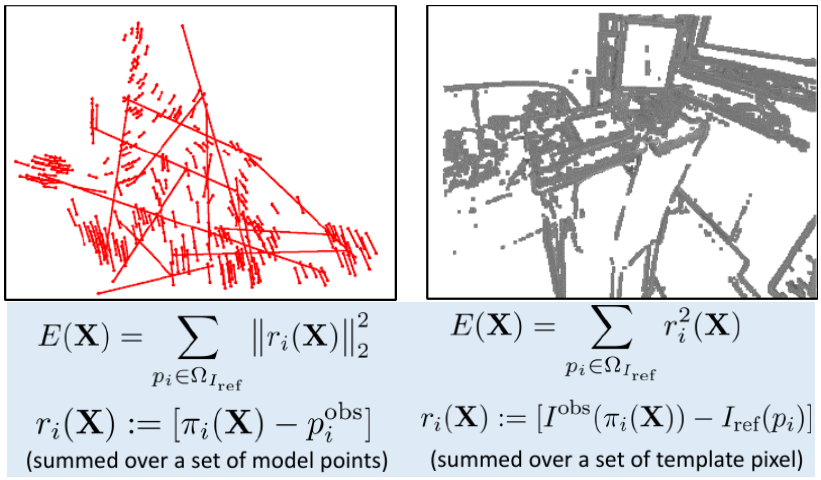


Figure 1.3: Front-end options: feature-based [4] vs. direct [7]. **Left**: feature-based front-end extracts a small set of correspondences between frames using feature matching. The correspondences contribute to state estimation as back-projection residual terms. **Right**: direct front-end works on a large set of pixels between frames. Data association and state estimation are typically conducted jointly, via minimizing the photometric residual terms.

calibration, consistent lighting condition, accurate motion prediction or smooth and slow camera motion. These conditions limit the applicability of direct systems for many robotics and AR applications, where VSLAM should work with noisy sensory input under changing environment for long duration. In addition, direct measurements rarely endure over long-baseline travel and can sometimes exhibit intermittent observation, both of which undermine strong localization and triangulation conditioning. For applications with frequent revisits, the percentage of long-baseline associations utilized by direct systems is less than that of feature-based ones, therefore limiting the performance of direct VSLAM systems. Semidirect systems [6] also leverage direct measurements in the pose tracking, therefore have poorer tracking performance than feature-based methods. Direct VINS are less sensitive since the motion priors from inertial sensors are relatively accurate. Still, performance of direct VINS are worse than feature-based VINS, as demonstrated in [11].

To summarize, the applicability of all image processing front-ends mentioned above are limited. In Fig 1.4, we compare these front-ends with regards to three applicability metrics: efficiency, accuracy and robustness:

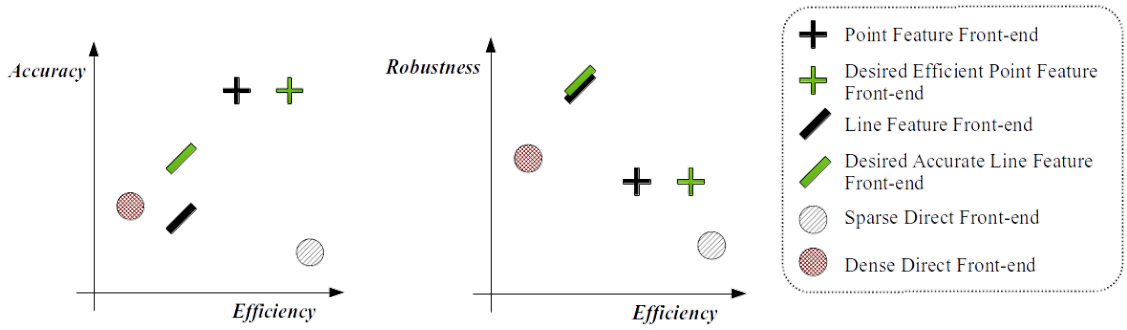


Figure 1.4: Illustration of applicability limits for image processing front-ends in VSLAM/VINS. **Left**: Accuracy vs. Efficiency. **Right**: Robustness vs. Efficiency.

1) When some level of textures exist, point-feature-based front-ends have high accuracy but also relatively high computation costs, due to the explicit feature extraction and matching. Direct front-ends are efficient, however the accuracy of these methods are poorer, therefore limiting the usage in practical applications. *It is desirable for the VSLAM front-end to have the same level as accuracy as feature-based front-ends, while being as efficient as direct ones.*

2) In low texture situations, line-feature-based front-ends provide extra robustness. However they are typically inaccurate due to the weak-constraint of line features. *It is desirable for the VSLAM front-end to have the same level of robustness as line-feature-based front-ends, while being more accurate.*

1.1.2 Back-end of VSLAM/VINS

To exploit the multi-core and multi-thread capabilities of modern compute hardware, state estimation in VSLAM/VINS is separated into multiple threads, each tackles a semi-independent problem. Modern VSLAM/VINS systems typically have 3 threads:

1) Pose tracking thread: conducts pose-only optimization on the current frame. It should meet strict frame-rate imposed constraints on processing time, including the overhead of the front-end. For efficiency, it is common to assume fixed map in pose tracking thread, and only optimize the state of current frame. Common optimization methods include PnP and pose-only BA.

2) Local optimization thread: conducts joint optimization of poses and mapped features at local scale (e.g. within a short history). Depending on the task requirement, the rate of local optimization ranges between frame-rate and sub-frame-rate. Compared with the pose tracking thread, the computation of local optimization is much higher due to the high dimensionality of states to be optimized.

3) Global optimization thread: conducts optimization of poses (and mapped features) at global scale (e.g. the entire history). Global optimization could be executed at the lowest frequency, e.g. only being triggered when a loop closure is detected. Furthermore, it is common for global optimization to work on the camera-only system (i.e., pose graph optimization), therefore further reducing the computational load.

The local optimization is the major bottleneck for applicable VSLAM/VINS, since the computational costs are high while the demanded processing rate is likewise high. In what follows, we first review existing optimization techniques in local optimization. Then, a gap of applicability in existing local optimization methods is revealed, which motivates the cost-efficiency local BA work (i.e., Good Graph) in this thesis.

There is a long history in SLAM community to apply filters, such as EKF [2] and EIF [37], to local optimization. The complexity of EKF and EIF are extremely high: $\mathcal{O}((n+m)^3)$, with n-D camera states and m-D map states. This is due to the dense structure (covariance matrix or information matrix) utilized to model joint distribution in EKF and EIF. Not surprisingly, the performance-efficiency trade-off of EKF and EIF failed to meet the large-scale, long-term requirement of VSLAM [38]. The efficiency of EKF can be significantly improved by optimizing the camera states only, while modeling map states as constraints between camera states. One representative design of efficient EKF is MSCKF [39], which has been well studied and applied to VINS [40, 10]. The complexity of MSCKF is $\mathcal{O}((n)^3+m)$, which is linear in the cardinality of the map. However, MSCKF has the downside of degraded mapping (e.g. map points are poorly distributed and inaccurate). Furthermore, all EKF variants introduced above are known to be inconsistent: they have

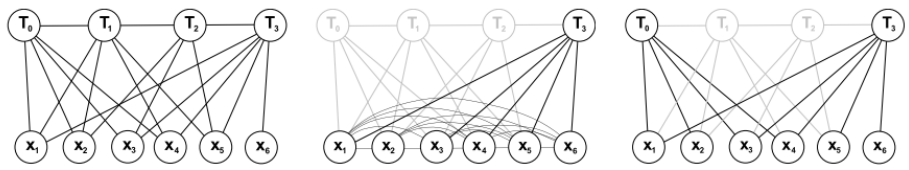


Figure 1.5: Illustration of filter and BA, as variants of general Markov Random Field (MRF) [38]. Notice the dense structure in filter (the edges between map nodes) and the sparse structure in keyframe BA.

to linearize and marginalize old states at a early stage, which may be conflicting with later states as more measurements arrive.

Bundle adjustment, on the other hand, has proven to be more accurate, especially in large-scale SfM and SLAM problem [38], when computational resources are sufficient. BA methods are more consistent than filters since there is no early linearization or marginalization. The downside of BA is the computation complexity: it could be cubic at the worst case! Therefore, BA requires extra effort in bounding the scale of the states to be optimized. One tactic for bounding the computational cost of BA in local optimization is exploiting the sparsity property of SLAM problem [41, 42, 43, 44]. As illustrated in Fig 1.5, SLAM is sparse: the map can be modeled as a set of independent states, and the time-varying camera poses can be modeled as Markov. Exploiting the sparse structure with specialized data structures and data organization methods leads to a sparse optimization problem that is efficiently solved with Schur marginalization and back substitution. In this way, the time complexity of local BA can be reduced to $\mathcal{O}(n^3 + mn)$: 1) Marginalize out all the map states, with cost $\mathcal{O}(mn)$; 2) Solve the reduced camera system, with $\mathcal{O}(n^3)$; and 3) Back substitute to collect the map states, with $\mathcal{O}(mn)$.

Sparse local BA is implemented in the back-end solvers of modern VSLAM [3, 5, 6, 7, 4] and VINS [16, 31, 45] systems. State-of-the-art non-linear solvers, such as iSAM2[44], g2o[46], Ceres[47], SLAM++[48] and ICE-BA[45], support sparse least squares optimization using state-of-the-art solvers, e.g. Levenberg-Marquardt and Conjugate Gradient Descent.

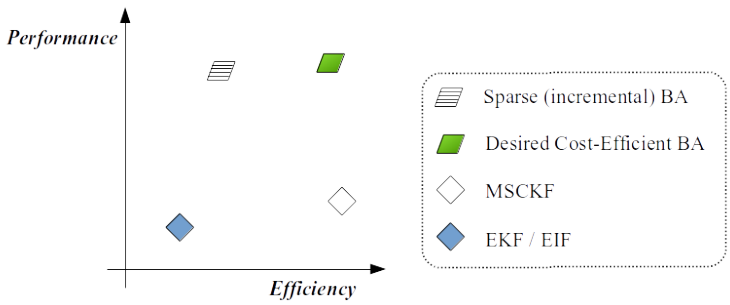


Figure 1.6: Illustration of applicability limits for optimization back-ends in VSLAM/VINS. The performance, which include both accuracy and robustness, are compared against efficiency, for back-end options.

Apart from sparsity, another key characteristics of SLAM problem to exploit is the incremental nature: measurements arrive sequentially, rendering the SLAM problem equivalent to an incremental estimation problem. Incremental solvers reuse the previously calculated factorization, and only perform calculations for states affected by the currently arrived measurements. Pioneering works of iSAM [43, 44] uses Givens rotation to update the QR factorization incrementally. Incremental Cholesky factorization updates have been incorporated into the BA-based back-end solvers [49]. Incremental algorithms for another compute intensive module, Schur elimination, have also been implemented [50]. More recently, the combination of sliding window and incremental algorithms has been explored [45].

Though a rich body of works exist in reducing the computational cost of the local BA, it is still more expensive than carefully designed filter back-ends such as MSCKF. As illustrated in Fig 1.6, a gap of applicability exists in the back-ends of VSLAM/VINS. MSCKF-based back-ends are computationally inexpensive, but with unsatisfactory quality in pose tracking and mapping. Sparse (and incremental) BA back-ends have the best performance, yet they are quite expensive to compute. Therefore, VSLAM/VINS systems with BA-based back-end haven't achieved the same level of success as filter-based ones when there are strict computational constraints. Still, BA-based back-ends have great potential on account of having better accuracy and robustness. Furthermore, BA-based back-ends

provide a more accurate and richer 3D map, which is crucial for downstream modules such as 3D scene understanding, interaction, and closed-loop navigation [51]. *Therefore, speeding up the sparse BA back-end without performance loss will have a huge impact on the applicability of BA-based VSLAM.*

1.2 General Problem of Submatrix Selection

The works to be described in this thesis are based upon theorems developed in computational theory and applied mathematics. Specifically, the general problem of *submatrix selection* is closely related. Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ (usually in full-rank), it is of interest to compress A via selecting a submatrix A_s, such that the selected submatrix behaves spectrally similarly to the original matrix, i.e., the singular values of the two matrices are comparable. In most cases, the selection is further limited to one dimension of full matrix A only, which is referred as *column (row) subset selection*.

Submatrix selection, especially column (row) subset selection, has been extensively studied for large-scale problems that $\mathbf{A}$ has tens of thousands of rows and columns, or more. A variety of algorithms are developed to solve submatrix selection for large-scale matrix, including random sampling [52, 53], greedy forward [54] and backward stepwise selection [55], forward stagewise regression [56, 57]), branch and bound [58, 59], and convex optimization such as ridge regression [60] and the lasso [61]. Random sampling methods [52, 53] have good performance when the scale of matrices is vastly huge; the performance degrades when the matrix gets smaller. Optimization based methods, such as branch and bound [58, 59] and convex optimization [60, 61], have good performance guarantee in general. However these methods are compute-expensive. Stepwise and stagewise methods are in the middle ground: they are less expensive to compute, but with degraded performance guarantee.

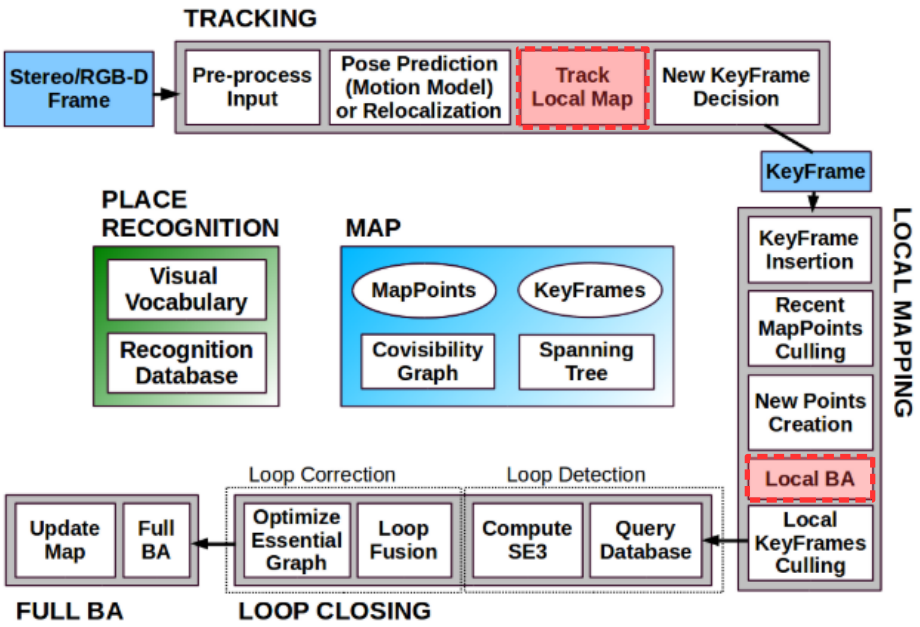
Compared with the existing studies on general submatrix selection, the problem dealt with in this thesis is similar in-spirit, but has several differences arising from the target

application. A large portion of the literature aims to solve in tractable ways large-scale linear problems requiring high-performance computing (HPC) that exceed the capabilities of HPC machines. These large-scale problems require reduction to be solveable, or require acceleration to be solveable on reasonable time-scales (possibly due to an iterative outer loop associated to the actual problem solution). Many of the same approaches to arriving at computationally tractable methods for large-scale problems apply to moderately-scaled problems on compute limited devices. However, the real-time constraints and trade-offs associated to the accelerated solutions need to be managed. The problems arising in SLAM have three key differences: 1) the scale of matrix in VSLAM is much smaller than those matrices in machine learning and data mining; 2) the compute budget (processing speed, memory, time cost) in VSLAM is highly restrictive, and 3) the performance requirement of submatrix selection is strict due to the sequential nature of VSLAM. The first difference is important because some of the accelerated solutions rely on theorems that hold in the asymptotic sense (as the matrix grows and the subset sought shrinks percentage-wise). Additionally, the desired submatrix selection algorithm in this thesis should have strict performance guarantees, while being highly efficient when working on small to medium scale matrices (e.g. with hundreds of rows and columns). In the meantime, the scalability to large-scale submatrix selection is less of a concern. Per such requirements, a specific family of submatrix selection algorithm, namely greedy stepwise selection, is extensively studied and verified.

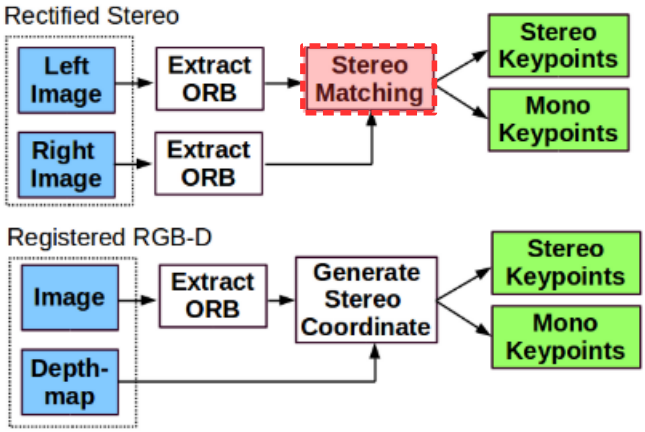
1.3 Pipeline of Feature-based BA VSLAM

In this thesis, we revisit the feature-based BA VSLAM, which has the best performance but also high computational costs. Two state-of-the-art feature-based BA VSLAM, ORB-SLAM [4] and PL-SLAM [23], are chosen as the base VSLAM system to improve upon.

The pipeline of ORB-SLAM [4] is illustrated in Fig 1.7. The computation load is separated into three parallel threads, i.e., tracking, local mapping and loop closing. As



(a) System Threads and Modules.



(b) Input pre-processing

Figure 1.7: Detailed pipeline of state-of-the-art feature-based BA VSLAM, ORB-SLAM [4]. The modules improved in this thesis are highlighted in red. **Top**: three threads are running in-parallel: 1) tracking, 2) local mapping and 3) loop closing. **Bottom**: when working with stereo input, pre-rectification of input image pair is conducted. Although it is not shown in this figure, ORB-SLAM also works with monocular input.

discussed early in VSLAM front-end options, the main bottlenecks of pose tracking are feature extraction and matching. In the example pipeline, ORB (feature) extraction is conducted as a part of the pre-processing module, while feature matching is conducted in the

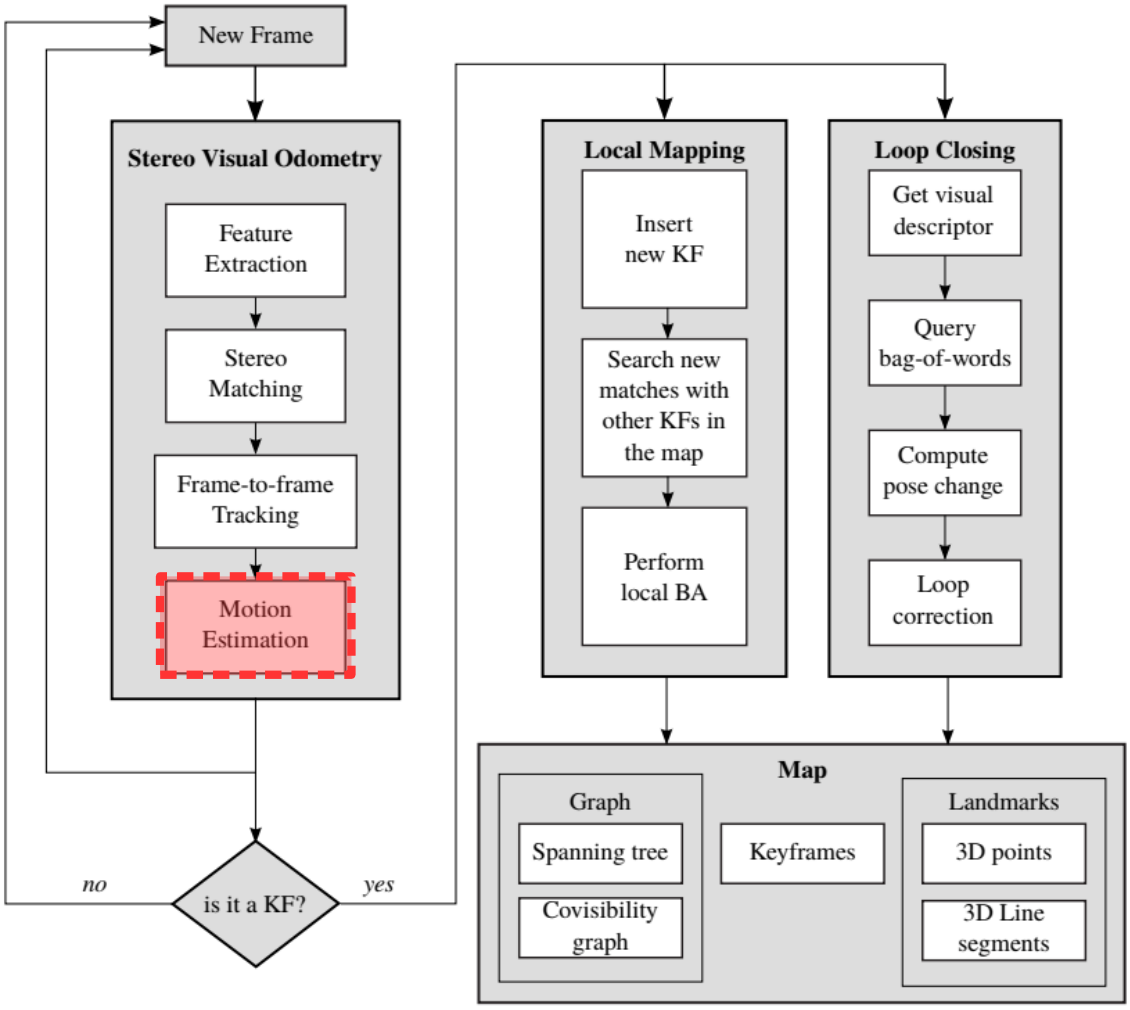


Figure 1.8: Detailed pipeline of state-of-the-art line-assisted BA VSLAM, PL-SLAM [23]. The module improved in this thesis is highlighted in red. Similar to ORB-SLAM [4], three major threads run in-parallel: 1) stereo VO (pose tracking), 2) local mapping and 3) loop closing.

stereo matching and the track local map modules. Naturally, the efficiency of these modules (ORB extraction, stereo matching, and track local map) are worth looking into if we want to improve the applicability of feature-based BA VSLAM. Apart from image processing front-end, the local mapping also affects the applicability of VSLAM heavily. As mentioned in VSLAM back-end options, both the computational costs and the demanded processing rate is high for local mapping. The cost-efficiency of the most compute-heavy module in local mapping, namely local BA, is worth looking into as well.

While point-feature-based ORB-SLAM has top-of-the-line performance in many scenarios, it does suffer when point features are lacking in the environment. Concrete examples include low-textured environment and motion-blurring. To run VSLAM robustly on these scenarios, it is desired to incorporate line features into a point-feature VSLAM pipeline. A state-of-the-art VSLAM system that tracks both point and line features is PL-SLAM [23]. The pipeline of PL-SLAM is illustrated in Fig 1.8. Similar to ORB-SLAM [4], the computation load is separated into three parallel threads. One primary limitation of line-assisted PL-SLAM is that, the real-time pose tracking with line features (frame-to-frame tracking) is not as accurate as the point counterpart. Due to the weak nature of line features and the frequent self-occlusion, lines are hard to triangulate and update with accumulated measurements. Here, hard means *with high uncertainty*, therefore the resulting 3D lines in the map are typically erroneous. The accuracy of line-assisted pose tracking needs further improvement for practical applications. In addition, the computational cost of modern line detectors such as LSD [19] is higher than point detectors. Through it is out of the scope of this thesis, accelerating feature extraction with dedicated hardware, such as FPGAs, is worth investigating as well.

1.4 Outline of the Thesis

The rest of this thesis is organized as follows:

- Chapter 2 describes the general forms of least squares optimization objectives used in multiple modules of VSLAM. The connection between least squares optimization and submatrix selection is revealed for the described optimization problems. A set of submatrix selection metrics have the property of submodularity, which enables efficient and near-optimal selection algorithms. Recognizing the universality of these efficient solutions leads to several improvements to VSLAM, as described in the remaining Chapters of the thesis.

- Chapter 3 applies submodular submatrix selection to a computation-intense module in the VSLAM front-end, i.e., map-to-frame feature matching. It is then combined with active feature matching, leading to a low-latency, performance guaranteed feature matching algorithm, dubbed Good Feature Matching [62, 63].

- Chapter 4 extends the idea of point feature selection to line features. A specific property of lines, i.e., extending along specific direction, is exploited to enable line feature refinement. The underlying optimization objective of line feature refinement is a convex optimization problem. An efficient, multi-start algorithm for generating sub-optimal solutions, dubbed Good Line Cutting [64], is described and evaluated.

- Chapter 5 details an appearance-based enhancement (Map Hashing [65]) for constructing, populating, and querying the local map. Local map is a critical accuracy-improving sub-component of the VSLAM front-end. An efficient hashing technique is applied to store and query appearance prior. Furthermore, submodular submatrix selection provides a means to reduce the quantity of hash queries through active, online table selection, thereby reducing the overhead of local map construction.

- Chapter 6 tackles the computational costs of the general BA problem, which is frequently solved in BA-based VSLAM back-end. A novel, rigorous method to determine the state subset in BA with strong performance guarantee is proposed, dubbed Good Graph [66]. Furthermore, we explore the potential of budget-awareness to determine the size of desired Good Graph on-the-fly.

- Chapter 7 explores the application of previously described efficiency improvements to closed-loop robot navigation, when integrated into a loosely-coupled visual-inertial state estimation system. The accurate and low-latency of described visual SLAM method is revealed in the closed-loop navigation scenario investigated. A reproducible benchmarking simulation [67] for closed-loop VSLAM evaluation is pre-

sented, which supports comprehensive evaluation of VSLAM in closed-loop navigation tasks.

- Chapter 8 concludes the thesis with a summary of findings and observations. Directions for future research are discussed as well.

CHAPTER 2

PRELIMINARY

2.1 Background

A key perspective of this thesis is speeding up compute-intensive modules in VSLAM with submodular submatrix selection. The majority of VSLAM modules, as studied in this thesis, can be formulated as some sort of least squares problem:

$$\arg\min_{\mathbf{x}} \sum_{i,j} \|\rho(\mathbf{x}(i), \mathbf{x}(j))\|^2_{\mathbf{\Sigma_{ij}}}, \tag{2.1}$$

where $\mathbf{x}$ is the vector of states to be optimized, ρ is the residual function, and $\mathbf{\Sigma_{ij}}$ is the covariance of each residual term.

With first-order approximation, these least squares problems can be further simplified into linear systems:

$$\arg\min_{\boldsymbol{\delta}} \|\mathbf{J}\boldsymbol{\delta} - \mathbf{b}\|^2, \tag{2.2}$$

where the Jacobian $\mathbf{J}$ is of interest. The Jacobian $\mathbf{J}$ can be equivalently represented as a factor graph [68]. A toy example of factor graph, as well as equivalent Jacobian and system (Hessian) matrix, are illustrated in Fig 2.1.

A toy example of SLAM (in factor graph representation), as well as corresponding Jacobian, is illustrated in Fig 2.1. Each factor (measurement) has corresponding non-zero filling in the Jacobian. Each factor has corresponding row, and each state (pose/landmark) has corresponding column.

Both the size and the sparsity of Jacobian $\mathbf{J}$ are closely related to cost-efficiency of corresponding VSLAM modules:

1) In the feature matching module of the VSLAM front-end, the number of rows in $\mathbf{J}$ is de-

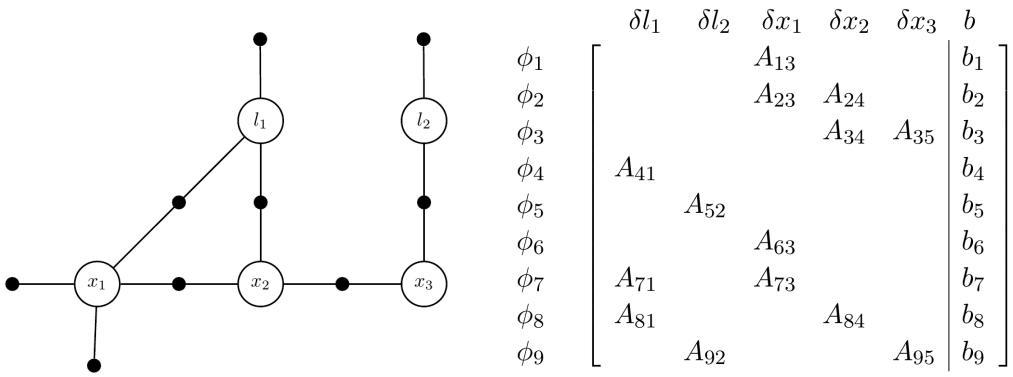


Figure 2.1: A toy example of factor graph and matrix representations [68]. **Left**: a factor graph with 3 poses x_i and 2 landmarks l_i. **Right**: corresponding Jacobian, where each factor (measurement) has corresponding row, and each state (pose/landmark) has corresponding column.

termined by the total number of feature matchings, while the number of columns is fixed (to the state of current frame). Each measurement contributes to a set of fillings accordingly. Collecting a **J** with a small row number is clearly cheaper in terms of feature matching effort. Since feature matching could be expensive in the presence of many matching candidates, it is desired to bound the effort of feature matching by building a small Jacobian.

2) For BA optimization in VSLAM back-end, both the row number and column number of **J** are correlated to computation cost. Similar to the front-end case, each row of **J** stands for a measurement, e.g. a feature matching or an odometry reading. Each column of **J**, on the other hand, stands for a state to be optimized. Collecting a **J** with a limit size on row and column is much cheaper: less computation effort is required in data association. Furthermore, the cost of numeral optimization is reduced when working on a **J** with less rows/columns, as the computation costs of most matrix manipulations involved in numeral optimization have cubic growth.

Naturally, it is attractive to build a principled solution that selects a small-size Jacobian from the full Jacobian **J** with little overhead. Apart from size limitation, it is also desired to preserve the spectral property of the full matrix **J** as much as possible. The general problem, i.e., submatrix selection, has been studied in the fields of computational theory and machine learning. The submatrix selection problem is defined as: given a matrix **J**,

Max-Trace	Trace $Tr(\mathbf{Q}) = \sum_1^m \mathbf{Q}_{ii}$ is max.
Min-Cond	Condition $\kappa(\mathbf{Q}) = \lambda_1(\mathbf{Q})/\lambda_m(\mathbf{Q})$ is min.
Max-MinEigenValue	Min. eigenvalue $\lambda_m(\mathbf{Q})$ is max.
Max-logDet	Log. of determinant $\log\det(\mathbf{Q})$ is max.

Table 2.1: Commonly used matrix-revealing metrics for square matrix $\mathbf{Q}$ of rank m.

select a subset of rows and columns so that the overall spectral properties of the selected submatrix are preserved as much as possible.

2.2 Matrix-Revealing Metrics

As extensively studied in the numerical methods and machine learning fields [69, 70], a number of matrix-revealing metrics exist to score the subset selection process. They are listed in Table 2.1. Subset selection with any of the listed matrix-revealing metrics is equivalent to a finite combinatorial optimization problem under the cardinality constraint:

$$\max_{S_1\subseteq\{1,2,\ldots,m\},S_2\subseteq\{1,2,\ldots,n\},|S_1|=k_1,|S_2|=k_2} f([\mathbf{J}(S_1,S_2)]^T[\mathbf{J}(S_1,S_2)]) \tag{2.3}$$

where S_1 is the index subset of selected rows from the full matrix $\mathbf{J}$, S_2 is the index subset of selected columns, $[\mathbf{J}(S_1,S_2)]$ is the corresponding submatrix indexed by S_1 and S_2, k_1 and k_2 are the cardinalities of row and column subset, and f the matrix-revealing metric.

2.3 Submatrix Selection Algorithms

While the combinatorial optimization can be solved by brute force, the exponentially-growing problem space quickly becomes impractical to search over for real-time VSLAM applications. To employ efficient subset selection strategies while limiting the loss in optimality, the submodularity property of subset selection is exploited [71, 72, 73, 74].

Definition [75] *A set function* $\mathbf{f}: 2^F \to \mathbb{R}$ *is submodular if, for any subsets* $A \subseteq B \subseteq F$*, and for any element* $e \in F \setminus B$*, it holds that:*

$$\mathbf{f}(A \cup e) - \mathbf{f}(A) \geq \mathbf{f}(B \cup e) - \mathbf{f}(B)$$

Submodularity formalizes the notion of diminishing returns in discrete domain: adding a row(block) to a small submatrix is more advantageous than adding it to a large submatrix. More importantly, a submodular function can be solved with greedy heuristic:

Proposition 1 *(Suboptimal submodular maximization [75])* Given a normalized, monotone, submodular set function $\mathbf{f} : 2^F \rightarrow \mathbb{R}$, and calling S^* the optimal solution of the maximization problem 2.3, then the set $S^{\#}$, computed by the greedy heuristic, is such that:

$$\mathbf{f}(S^{\#}) \geq (1 - 1/e)\mathbf{f}(S^*) \approx 0.63\mathbf{f}(S^*)$$

This bound ensures that the worst-case performance of a simple greedy algorithm cannot be far from the optimum. Except for *Min-Cond*, the metrics listed in Table 2.1 are either submodular or approximately submodular, and monotone increasing. The *Max-logDet* metric is submodular [73], while the *Max-Trace* is modular (a stronger property) [71]. Lastly, *Max-MinEigenValue* is approximately submodular [72]. Therefore, selecting rows with these metrics can be approximately solved with greedy methods. Using these known properties, the aim here is to arrive at an efficient submatrix selection algorithm without significant loss in optimality.

2.3.1 Greedy Selection

Subset selection with submodular metric has been studied for sensor selection [73] and feature selection [74], with reliance on a simple greedy algorithm commonly used to approximate the original NP-hard combinational optimization problem. The approximation ratio of the greedy approach is $1-1/e$ [71]. This approximation ratio is the best achievable by any polynomial time algorithm under the assumption that $P \neq NP$.

The computation complexity of the greedy selection is $\mathcal{O}(kn)$, when selecting k-size submatrix from n-size full matrix. When working with a large-size matrix, the cost of

greedy selection could be too expensive for real-time VSLAM applications. Several accelerating techniques exist for the greedy selection, which could be crucial for applying submatrix selection to large-size matrix with real-time requirement.

2.3.2 Lazy Greedy

The classical greedy algorithm can be enhanced into an accelerated version, lazy greedy [76]. The key idea of lazy greedy is utilizing a compute-cheap upper bound to reject unwanted candidates, therefore reducing the computation of actual margin gain at each iteration. Obviously, the speed up of lazy greedy hinges on the tightness of the upper bound. Consider an idealized case, where the computing upper bound takes zero-cost and a constant rejection ratio ρ is achieved with the upper bound. Hence the total complexity of selecting a k-size submatrix out of the n-size full matrix using lazy greedy algorithm is $\mathcal{O}(k(1-\rho)n)$: the lazy greedy algorithm has to run k rounds, in each round it will go through $(1-\rho)n$ candidates to identify the current best row/column.

For certain metric (e.g. *Max-MinEigenValue*), the tight upper bound exists. For the *logDet* metric, the upper bound derived from Hadamard's inequality [77] is quite loose (i.e., $\rho \approx 0$):

$$\log\det(\mathbf{Q}) \leq \sum_{i=1}^{m} \log(\mathbf{Q_{ii}}),\ \mathrm{rank}(\mathbf{Q}) = m. \tag{2.4}$$

Therefore the lazy greedy algorithm is not a general solution to efficient submatrix selection. Even when working with metric that has tight upper-bound, the amount of computation saved by lazy greedy is limited. As reported in [74] and further confirmed in our simulation in Chapter 3, the time cost of lazy greedy selection in VSLAM easily exceeds the real-time requirement (e.g. 30ms per frame). Hence the lazy greedy is still limited in applicability and efficiency.

Algorithm 1 STOCHASTIC-GREEDY

Input: $f : 2^V \to \mathbb{R}_+, k \in \{1, \ldots, n\}$.
Output: A set $A \subseteq V$ satisfying $|A| \leq k$.
1: $A \leftarrow \emptyset$.
2: **for** $(i \leftarrow 1;\ i \leq k;\ i \leftarrow i + 1)$ **do**
3: $R \leftarrow$ a random subset obtained by sampling s random elements from $V \setminus A$.
4: $a_i \leftarrow \text{argmax}_{a \in R} \Delta(a|A)$.
5: $A \leftarrow A \cup \{a_i\}$
6: **return** A.

Figure 2.2: Lazier greedy algorithm, originally proposed in [81].

2.3.3 Lazier-than-Lazy Greedy

Compared to the aforementioned deterministic methods (e.g. classic and lazy greedy), randomized submatrix selection has been studied as a faster alternative with probabilistic performance guarantee [78, 79]. Combining randomized selection with deterministic method yields fast yet near-optimal submatrix selection for specific matrix norms [70, 80] and general submodular functions [81, 82].

The combined algorithm is dubbed as lazier-than-lazy greedy, or lazier greedy for further simplicity. The general procedure of lazier greedy is presented in Fig 2.2. The idea of lazier greedy is simple: at each round of greedy selection, instead of going through all n candidates, only a random subset of candidates are evaluated to identify the current best candidate. Furthermore, the size of random subset s can be controlled with a decay factor ϵ: $s = \frac{n}{k}\log(\frac{1}{\epsilon})$. In this way, the total complexity is reduced from $\mathcal{O}(kn)$ (greedy) or $\mathcal{O}(k(1-\rho)n)$ (lazy greedy) to $\mathcal{O}(\log(\frac{1}{\epsilon})n)$. The majority of this thesis is based upon the following two theorems:

Theorem 2 *[81] Let f be a non-negative monotone submoduar function. Let us also set $s = \frac{n}{k}\log(\frac{1}{\epsilon})$. Then lazier greedy achieves a $(1 - 1/e - \epsilon)$ approximation guarantee in expectation to the optimum solution of problem Eq 3.8.*

Theorem 3 *[82] The expectation of approximation guarantee of $(1 - 1/e - \epsilon)$ is reached*

with a minimum probability of $1 - e(-0.5k(\sqrt{\mu} + ln(\epsilon + e^{-1})/\sqrt{\mu})^2)$*, when maximizing a monotone submodular function under cardinality constraint* k *with lazier-greedy.* $\mu \in (0, 1]$ *is the average of approximation ratio when maximizing margin gain at each iteration of lazier greedy.*

Notice that the symbols and formulations in Theorem 3 are adjusted from the original proof at [82] to be consistent with Theorem 2. According to these two theorems: 1) lazier greedy introduce a linear loss ϵ to the approximation ratio *in expectation*; and 2) the expectation of linear-loss approximation ratio can be guaranteed with high probability. Compared to the theoretical upper bound of approximation ratio, $1 - 1/e$, which no polynomial time algorithm can exceed [71], lazier greedy only loses a small portion from it (in expectation and probability).

CHAPTER 3
GOOD FEATURE MATCHING: LOW-LATENCY FRONT-END OF FEATURE-BASED VSLAM

3.1 Introduction

This chapter describes the Good Feature Matching algorithm that reduces the latency of feature matching in VSLAM/VINS. The key observation that motivates the research: the camera pose estimation in VSLAM is an over-constraint optimization problem with massive over-measurement. Taking less feature matchings reduces the computation cost in VSLAM, yet increases the effect of outliers and noise. Obviously a trade-off of performance-efficiency exists between aggressive feature subset selection and keeping the full feature set. When properly selected, the feature subset can actually balance the performance drop and efficiency improvement. The goal of this work is to identify a small subset of features (a.k.a. good features) that are most valuable towards pose estimation with minimum compute overhead. By only utilizing the good features in both data association and state optimization, the latency of pose tracking is improved, while the accuracy and robustness are preserved.

The primary outcome of this work is illustrated in Fig 3.1. At the left column, we present the latency-accuracy trade-off of 4 monocular VSLAM systems on a public benchmark (EuRoC MAV [25]). Each marker on the plot represents a successfully tracked sequence (zero track loss in 10 repeated trials) for the denoted VSLAM system. To better understand the latency-accuracy trade-off in each VSLAM system, we adjust the maximum number of features/patches extracted per frame (for GF-ORB, we also adjust the maximum number of good feature being matched per frame), to obtain the workable region of each system in the latency-accuracy plot (in dashed contour). According to the left

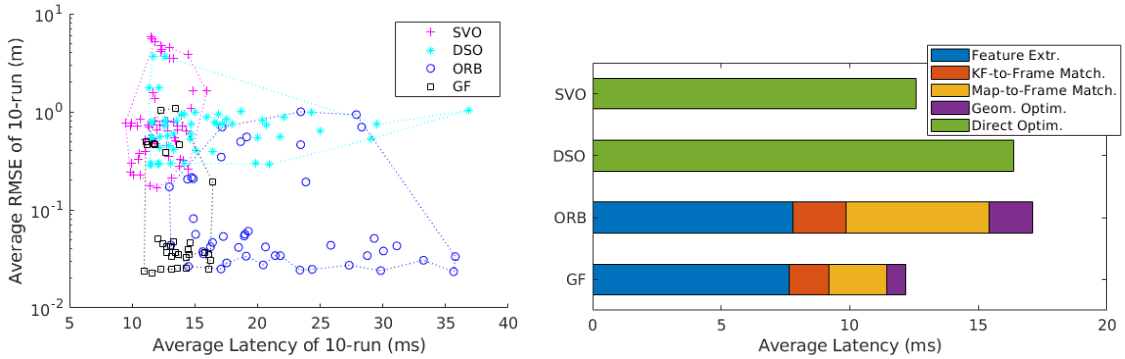


Figure 3.1: Latency reduction and accuracy preservation of proposed approach on EuRoC MAV benchmark. Four monocular VSLAM systems are assessed: semidirect SVO [6], direct DSO [7], feature-based ORB [4], and proposed GF-ORB. **Left**: latency vs. accuracy of four systems. The workable region (in dashed contour) of each system is obtained by adjusting the maximum number of features/patches per frame. **Right**: latency break down of each module in pose tracking pipelines, average on EuRoC benchmark. An example configuration that yields good trade-off of latency and accuracy is set: 800 features/patches extracted per frame; for GF-ORB we further limit the number of good features matched per frame to 100.

column of Fig 3.1, feature-based ORB-SLAM occupies the lower-right portion, as it is accurate yet with high-latency; direct DSO can reach lower latency than ORB-SLAM under some configurations, but it has an order of magnitude higher *absolute root-mean-square error* (RMSE) than ORB-SLAM; the tight-bounded working region of semidirect SVO is at the upper-left, meaning it is efficient yet inaccurate. The objectives of low-latency and high-accuracy are achieved with the proposed approach, GF-ORB-SLAM, whose markers are located in the lower-left region of the plot. We further present the break down of latency introduced by each module in pose tracking pipelines, under example configurations for all 4 VSLAM systems. When GF-ORB-SLAM is compared with the baseline ORB-SLAM, the time cost of feature extraction is identical, but the feature matching and subsequent modules have a significantly reduced time cost. The overall latency of GF-ORB is the lowest among all four systems, including the semidirect SVO.

Contributions of this work include:

1) Studying the **error model** of least squares pose optimization, which connects the performance of pose optimization to the spectral property of a weighted Jacobian matrix;

2) **Exploring metrics** connected to the least squares conditioning of pose optimization, with quantification of *Max-logDet* as the optimal metric;

3) An **efficient Good Feature Selection algorithm** that works with *Max-logDet* metric is introduced, which is an order of magnitude faster than state-of-the-art feature selection approaches;

4) Fusing Good Feature Selection and active matching into a **generic Good Feature Matching algorithm**, which is efficient and applicable to feature-based VSLAM; and

5) **Comprehensive evaluation** of Good Feature Matching on a state-of-the-art feature-based VSLAM system, with multiple benchmarks, sensor setups and computation platforms. Evaluation results demonstrate both latency reduction and accuracy and robustness preservation with the propose method. We open source our implementations for both monocular [1] and stereo SLAM [2].

3.2 Background

This work is closely connected to following two research topics in VSLAM:

3.2.1 Feature Selection

Feature selection has been widely applied in VSLAM for performance and efficiency purposes. Conventionally, fully data-driven and randomized methods such as RANSAC are used to reject outlier features [2]. Extensions to RANSAC improve its computational efficiency [83, 84]. These RANSAC-like approaches are utilized in many VSLAM systems [2, 3, 4] to improve the robustness of state estimation.

Apart from outlier rejection, feature selection methods are also utilized for inlier selection, which aims to identify valuable inlier matches from useless ones. One major benefit of inlier selection is the reduction of computation (and latency thereafter), since only a small set of selected inliers are processed by VSLAM. In addition, it is possible to improve

[1] https://github.com/ivalab/GF_ORB_SLAM

[2] https://github.com/ivalab/gf_orb_slam2

accuracy with inlier selection, as demonstrated in [85, 86, 87, 62]. The scope of this work is on inlier selection, which reduces the latency of VSLAM while preserving the accuracy and robustness.

Image appearance has been used to guide inlier selection: feature points with distinct color/texture patterns are more likely to get matched correctly [88, 89, 90]. However, these works solely rely on quantifying distinct appearance, while the structural information of the 3D world and the camera motion are ignored. While we agree that appearance cues are important in feature selection, the focus of this work is on the latter properties: identifying valuable features based on structural and motion information. The proposed structural-driven method can be combined with the appearance-based complementary approach [62].

To exploit the structural and motion information, covariance-based inlier selection methods are studied [2, 91, 92, 93, 94, 74]. Most of these works are based on pose covariance matrix, which has two key characteristics: 1) it contains both structural and motion information implicitly, and 2) it approximately represents the uncertainty ellipsoid of pose estimation. Based on the pose covariance matrix, different metrics were introduced to guide the inlier selection, such as information gain [2], entropy [92], trace [93], covariance ratio [94], minimum eigenvalue and log determinant [74]. Covariance-based inlier selection methods are studied for both filtering-based VSLAM [2, 91, 92, 93, 94] and BA-based VSLAM [50, 48, 74].

Observability matrix has been studied as an alternative of covariance matrix to guide feature selection [86, 87]. In these works, the connection between pose tracking accuracy and observability conditioning of SLAM as a dynamic system is studied. The insight of their work being: the better conditioned the SLAM system is, the more tolerant the pose estimator will be towards feature measurement error. To that end, the minimum singular value of observability matrix is used as the metric to guide feature selection. However, the efficient construction of observability matrix relies on the piecewise linear assumption, which limits the applicability of observability-based feature selection. Furthermore, we

argue that covariance matrix is better suited for static or *instantaneous* bundle adjustment (BA) problem as been formulated in pose tracking, and it can be constructed efficiently for non-linear optimizers.

The study in [74] is mostly related to our work. In [74], feature selection is performed by maximizing the information gain towards pose estimation within a prediction horizon. Two feature selection metrics were evaluated, minimal eigenvalue and log determinant (*Max-logDet*). Though the log determinant metric is utilized in our work, the algorithm for approximately selecting the feature subset maximizing *logDet* differs, as well as the matrix whose conditioning is optimized. Compared with [74], our work is more applicable for low-latency pose tracking thanks to two key advantages. First, the lazier-greedy algorithm presented in our work is efficient. It takes an order of magnitude less time than the lazy-greedy algorithm of [74], yet preserves the optimality bound. Second, we present the combination of efficient feature selection and active feature matching, which reduces the latency of both data association and state optimization. Meanwhile, [74] selects features after data association, therefore leaving the latency of data association unchanged. The experimental results in [74] supports our claim on applicability: there are occasions that feature selection actually increases the latency of full pipeline, compared with the original all-feature approach.

3.2.2 Active Matching

Another key perspective of this work is combining feature selection algorithm with active feature matching, which leads to latency-reduction in both data association and state optimization. Active matching refers to the guided feature matching methods that prioritize processing resource (e.g. CPU percentage, latency budget) on a subset of features. Compared with the brute force approach that treats all features equally, active matching is potentially more efficient, especially under resource constraints.

Active matching has been intensively studied for filter-based VSLAM, with represen-

tative works [95, 96, 97]. Traditional active matching methods are designed upon dense covariance matrix (i.e., majority of off-diagonal components are filled), therefore are obsolete in modern VSLAM driven by non-linear sparse optimizers. Furthermore, the algorithms used by these active matching methods were extremely compute-heavy, therefore impossible to integrate into the real-time pose tracking thread of modern VSLAM system. Therefore, the idea of active matching gets less attractive, as quoted from [51]: "the problem with this idea (active searching) was that ... too much computation is required to decide *where* to look." In this work, we demonstrate the worthy of revisiting the classic idea of active matching: the Good Feature Matching algorithm is extremely efficient and applicable, based upon specific matrices and selection algorithm tailored for non-linear optimization. To the best of our knowledge, this is the first work that demonstrates the applicability of latency-reduction and accuracy preservation in *real-time pose tracking* with active feature selection.

3.3 Conditioning of Pose Tracking Objective

Without loss of generality, this work is based on the least squares objective of pose tracking below:

$$\hat{x} = \arg\min_{x} \|\mathbf{h}(x, p) - z\|^2 , \tag{3.1}$$

where x is the pose of the camera, p are the 3D feature points and z are the corresponding 2D image measurements. The measurement function, $\mathbf{h}(x, p)$, is a combination of the $SE(3)$ transformation (world-to-camera) and pin-hole projection. For simplification, we omit the distortion of camera lens in $\mathbf{h}(x, p)$. In practice, it is typical to replace the quadratic loss in Eq 3.1 with some robust loss function, e.g. Huber Kernel. Nevertheless, least squares with robust kernel can be approximated with iteratively reweighted least squares [98]. Therefore, the formulations in the following can be extended to robust least squares by simply including a weight matrix.

Solving the least squares objective often involves the first-order approximation of all non-linear functions in Eq 3.1:

$$\begin{aligned}\mathbf{h}(x,p) &\approx \mathbf{h}(x^{(s)},p) + \mathbf{H_x}(x - x^{(s)}) \\ \mathbf{h}(x,p) &\approx \mathbf{h}(x,p^{(s)}) + \mathbf{H_p}(p - p^{(s)})\end{aligned} \tag{3.2}$$

where $\mathbf{H_x}$ is the measurement Jacobian linearized about the initial guess $x^{(s)}$. In Gauss-Newton style optimizer, minimization of Eq 3.1 is done iteratively via

$$x^{(s+1)} = x^{(s)} - \mathbf{H_x}^{+}(z - \mathbf{h}(x^{(s)},p)). \tag{3.3}$$

The accuracy of Gauss-Newton depends on the residual error ϵ_r, which can be decomposed into two terms: measurement error ϵ_z and map error ϵ_p. Assuming the input error are under independent Gaussian: $\epsilon_z(i) \sim N(0, \boldsymbol{\Sigma_z}(i))$ and $\epsilon_p(i) \sim N(0, \boldsymbol{\Sigma_p}(i))$. Then we can derive the covariance matrix of pose estimation:

$$\begin{aligned}\boldsymbol{\Sigma_r} &= \boldsymbol{\Sigma_z} + \mathbf{H_p}\boldsymbol{\Sigma_p}\mathbf{H_p}^T \\ \boldsymbol{\Sigma_x} &= \mathbf{H_x}^{+}\boldsymbol{\Sigma_r}(\mathbf{H_x}^{+})^T = \mathbf{H_x}^{+}\mathbf{W_r}(\mathbf{H_x}^{+}\mathbf{W_r})^T\end{aligned} \tag{3.4}$$

where the simplification holds since both $\mathbf{J}_\rho$ and $\boldsymbol{\Sigma_r}$ are block-diagonal matrices, and $\mathbf{W_r}$ is the diagonal weight matrix with Cholesky decomposed diagonal blocks $\boldsymbol{\Sigma_r}$: $\boldsymbol{\Sigma_r}(i) = \mathbf{W_r}(i)\mathbf{W_r}(i)^T$.

Finally, we can move everything to the left hand side of Eq 3.4:

$$\mathbf{W_r}^{-1}\mathbf{H_x}\boldsymbol{\Sigma_x}(\mathbf{W_r}^{-1}\mathbf{H_x})^T = \mathbf{I}, \tag{3.5}$$

where $\mathbf{W_r}^{-1}$ is still a block diagonal matrix, consisting of 2×2 blocks denoted by $\mathbf{W_r}^{-1}(i)$. Meanwhile, each row block of measurement Jacobian $\mathbf{H_x}$ can be written as $\mathbf{H_x}(i)$. Follow-

ing through on the block-wise multiplication results in the matrix $\mathbf{H_c}$:

$$\mathbf{H_c} = \begin{bmatrix} \mathbf{W_r}^{-1}(0)\mathbf{H_x}(0) \\ ... \\ \mathbf{W_r}^{-1}(n-1)\mathbf{H_x}(n-1) \end{bmatrix}, \tag{3.6}$$

from which the simplified pose covariance matrix follows:

$$\mathbf{\Sigma_x} = \mathbf{H_c}^{+}(\mathbf{H_c}^{+})^T = (\mathbf{H_c}^T\mathbf{H_c})^{-1}, \tag{3.7}$$

assuming that $\mathbf{H_c}$ is full rank (i.e., sufficient tracked map points exist). The conditioning of $\mathbf{H_c}$ determines the error propagation properties of the iteratively solved least-squares solution for the camera pose x.

The pose covariance matrix $\mathbf{\Sigma_x}$ represents the uncertainty ellipsoid in pose configuration space. According to Eq (3.7), one should use all the features/measurements available to minimize the uncertainty (i.e., variance) of pose estimation: with more measurements, the singular values of $\mathbf{H_c}$ should increase in magnitude. The worst case uncertainty would be proportional to the inverse of minimal singular value $\sigma_{min}(\mathbf{H_c})$, whereas in the best case it would be proportional to the inverse of maximal singular value $\sigma_{max}(\mathbf{H_c})$.

However, for the purpose of low-latency pose tracking, one should only utilize **sufficient** features. There is a tension between latency and error rejection. From the analysis, the uncertainty of least squares pose optimization problem is bounded by the extremal spectral properties of the matrix $\mathbf{H_c}$. Hence, one possible metric that measures the sufficiency of a feature subset would be, the factor of worst case scenario $\sigma_{min}(\mathbf{H_c})$. Meanwhile, one may argue that the extremal spectral properties only decides the upper and lower bounds of pose optimization uncertainty. The true values would depend on what the overall spectral properties of the system are. It follows then, that another possible measurement of sufficiency would be the overall spectral properties of $\mathbf{H_c}$.

3.4 Good Feature Selection using Max-LogDet

Define the *Good Feature Selection* problem to be: Given a set of 2D-3D feature matchings, find a constant-cardinality subset from them, such that the error of least squares pose optimization is minimized when using the subset only. Based on the previous discussion, the Good Feature Selection problem is equivalent to submatrix selection: Given a matrix H_c, select a subset of row blocks so that the overall spectral properties of the selected submatrix are preserved as much as possible.

3.4.1 Objective Formulation

Recall the discussion of submatrix selection in Chapter 2, a number of matrix-revealing metrics exist to score the subset selection process, as listed in Table 2.1. Subset selection with any of the listed matrix-revealing metrics is equivalent to a finite combinatorial optimization problem under cardinality constraint:

$$\max_{S\subseteq\{1,2,\ldots,n\},|S|=k} f([\mathbf{H_c}(\mathbf{S})]^T[\mathbf{H_c}(\mathbf{S})]) \tag{3.8}$$

where $\mathbf{S}$ contains the index subsets of selected row blocks from the full matrix $\mathbf{H_c}$, $[\mathbf{H_c}(\mathbf{S})]$ is the corresponding row-wise concatenated submatrix, k is the cardinality of subset, and f the matrix-revealing metric.

To explore which matrix-revealing metrics might best guide good feature/row block selection for least squares pose optimization, a simulation is conducted based on the Matlab simulation environment [99]. To simulate the residual error, both the 3D mapped features and the 2D measurements are perturbed with zero-mean Gaussian noise. A zero-mean Gaussian with the standard deviation of 0.02m are added to the 3D features stored as map. Three levels of measurement error are added to 2D measurements: zero-mean Gaussian with standard deviation of 0.5, 1.5 and 2.5 pixel. Subset size ranging from 80 to 200 are tested. To be statistically sound, 300 runs are repeated for each configuration. The sim-

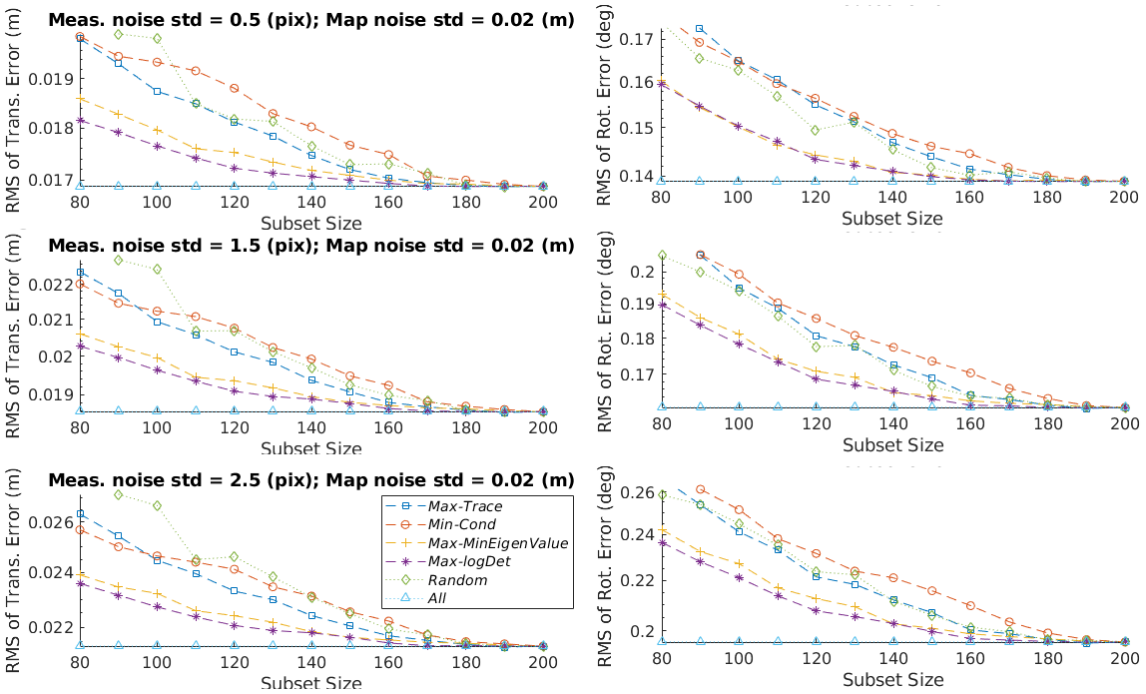


Figure 3.2: Simulation results of least squares pose optimization. **First column**: RMS of translational error under 3 levels of residual error. **Second column**: RMS of rotational error under 3 levels of residual error.

ulation results are presented in Fig 3.2, with the root-mean-square (RMS) of translational error (m) and rotational error (deg). Each of the matrix-revealing metrics in Table 2.1 is tested. For reference, the plots include simulation results with randomized subset selection (*Random*) and with all features available (*All*).

According to the simulation, two metrics stand out: *Max-MinEigenValue* and *Max-logDet*. Under all residual noise levels, their curves more quickly approach the baseline error (*All*) as a function of the subset size. Based on the outcomes, *Max-logDet* is chosen as the metric to guide Good Feature Selection. The reasons are two-fold:

First, according to Fig 3.2, the error curves of *Max-logDet* are always lower, if not at the same level, than those of *Max-MinEigenValue*. Similar trends are observed under other configurations as well. The subset selected with *Max-logDet* approximates the original full feature set better than the subset with *Max-MinEigenValue*. As discussed previously, greedy selection with *Max-logDet* has guaranteed approximation ratio due to submodular-

ity.

Second, the computational cost of *logDet* is lower than that of *MinEigenValue*. The main *logDet* computation is Cholesky factorization, with a complexity of $\mathcal{O}(0.33n^3)$ on a n square matrix, whereas for *MinEigenValue* the complexity is $\mathcal{O}(22n^3)$ [100].

3.5 Efficient Good Feature Matching

Assuming the combined matrix $\mathbf{H_c}$ is known, the key question of Good Feature Matching becomes: how to identify a row subset $[\mathbf{H_c}(\mathbf{S})]$ in $\mathbf{H_c}$, so that the overall spectral property (e.g. measured by *logDet*) is maximized? As introduced in Chapter 2, efficient algorithm exists for such submodular submatrix selection: lazier greedy.

3.5.1 Choice of Decay Factor

The approximation ratio and computational speed up of lazier greedy hinge on the decay factor ϵ. As illustrated in Fig 3.3 (middle), the approximation ratio decays linearly with ϵ, while the computational cost (FLOP) decays logarithmically. When the decay factor $\epsilon = 0$, the lazier greedy algorithm converges to the classical greedy, which has the best optimal guarantee and the highest computational cost.

As ϵ increases the resulting computational gain outpaces the loss in optimality, until hitting an inflection point after which the benefit reduces. When the decay factor is set to the maximal (i.e., $e^{-\frac{k}{n}}$), lazier greedy becomes randomized sampling (i.e., $s = 1$), which has an almost-zero expected approximation ratio (i.e., $1 - 1/e - e^{-\frac{k}{n}}$). Though the computational cost is the lowest in randomized sampling, the consistency of randomized sampling is limited as indicated from the closer to zero approximation ratio. As a consequence, a poorly-conditioned state optimization could occasionally be formulated when selecting good features with randomized sampling, especially when the size of feature pool is small. For sequential estimation problems such as VSLAM, inconsistent randomized sampling should be avoided: degraded estimation of the current state negatively impact the estima-

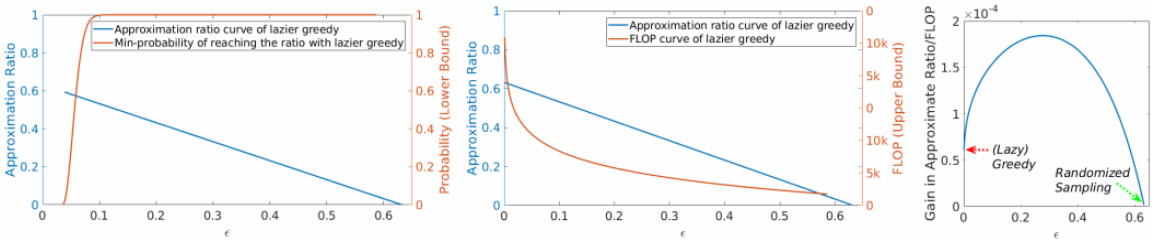


Figure 3.3: Illustration of performance and efficiency of lazier greedy, when selecting a subset of 450 rows from 1500 rows with average approximation ratio $\mu = 0.8$ in maximizing margin gain (*logDet*). **Left**: Approximation ratio and probabilistic guarantee of lazier greedy, versus the decay factor ϵ. **Middle**: Approximation ratio and computation cost (FLOP) of lazier greedy, versus the decay factor ϵ. **Right**: Efficiency of lazier greedy, versus the decay factor ϵ.

tion of follow-up states and leads to pose tracking degradation, or even failure. Therefore the choice of ϵ should lie somewhere between the two extremes (0 and $e^{-\frac{k}{n}}$). By setting ϵ to a small positive value, e.g. 0.1-0.3 as indicated in Fig 3.3 (right), lazier greedy will have a slightly degraded optimal bound but with a 3-4x higher efficiency than lazy greedy. Alg 1 describes an efficient algorithm for Good Feature Selection based on the near-optimal lazier-greedy.

Algorithm 1: Lazier-greedy Good Feature Selection algorithm.

Data: $H_c = \{\mathbf{H_c}(1),\ \mathbf{H_c}(2),\ \ldots,\ \mathbf{H_c}(n)\},\ k$
Result: $H_c^{sub} \subseteq H_c,\ |H_c^{sub}| = k$

$H_c^{sub} \leftarrow \emptyset$;
while $|H_c^{sub}| < k$ **do**
  $H_c^R \leftarrow$ a random subset obtained by sampling $s = \frac{n}{k}\log(\frac{1}{\epsilon})$ random elements from H_c;
  $\mathbf{H_c}(i) \leftarrow \arg\max_{\mathbf{H_c}(i) \in H_c^R} \log\det(\mathbf{H_c(i)}^T \mathbf{H_c(i)} + [\mathbf{H_c^{sub}}]^T[\mathbf{H_c^{sub}}])$;
  $H_c^{sub} \leftarrow H_c^{sub} \cup \mathbf{H_c}(i)$;
  $H_c \leftarrow H_c \setminus \mathbf{H_c}(i)$;
return H_c^{sub}.

3.5.2 Simulation of Lazier Greedy Feature Selection

To validate the benefits of lazier greedy, and to identify the proper value of decay factor ϵ, a simulation of Good Feature Selection is conducted. A testing process similar to the

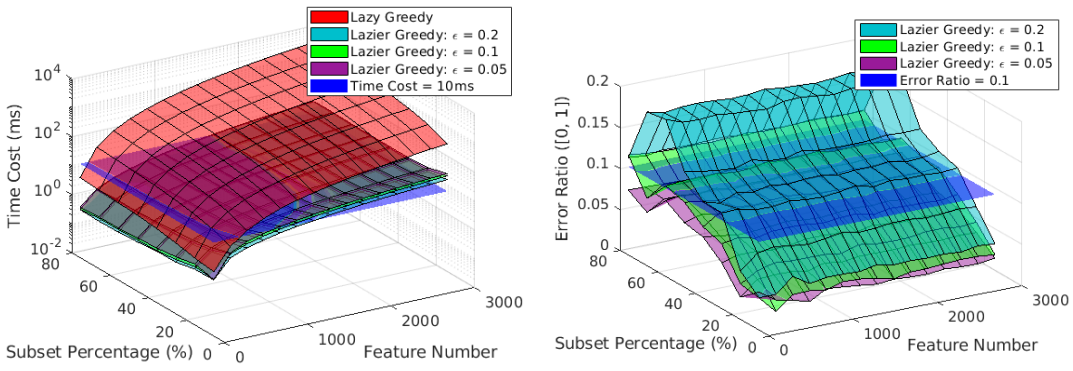


Figure 3.4: Lazy greedy vs. lazier greedy in feature selection simulation. **Left**: average time cost of lazy greedy vs. lazier greedy under different decay factor ϵ. **Right**: average error ratio of lazier greedy (compared with lazy greedy baseline; the smaller the better) under different ϵ.

Matlab one from the previous pose optimization simulation was implemented in C++ for speed assessment. The two feature selection algorithms tested are: lazy greedy [74] and lazier greedy (Alg 1). Like the simulation of pose optimization, a set of randomly-spawned 3D feature points, as well as the corresponding 2D measurements, are provided as input. Gaussian noise is added to both the 3D mapped features and the 2D measurements. The perturbed inputs are fed into a matrix building module, which estimates the combined matrix H_c for submatrix/feature selection.

To assess the performance and efficiency of Good Feature Selection comprehensively, we sweep through the three parameters: the number of 3D features from 100 to 3000, the percentage to select as subset from 10% to 80%, and the decay factor from 0.5 to 0.005. For each parameter combination, we randomly spawn 100 different worlds and evaluate each feature selection algorithm on each world. Due to the randomness of lazier greedy, we repeat it 20 times under each configuration.

Fig 3.4 plots the simulation results for computational time and error ratio as a function of the subset percentage and the number of features. The error ratio uses the lazy-greedy outcome as the baseline, then computes the normalized RMS of the difference versus lazier greedy. The multiple surfaces for lazier-greedy correspond to different decay factors ϵ. Referring to the time cost graph, lazier greedy is 1-3 orders of magnitude lower than that of

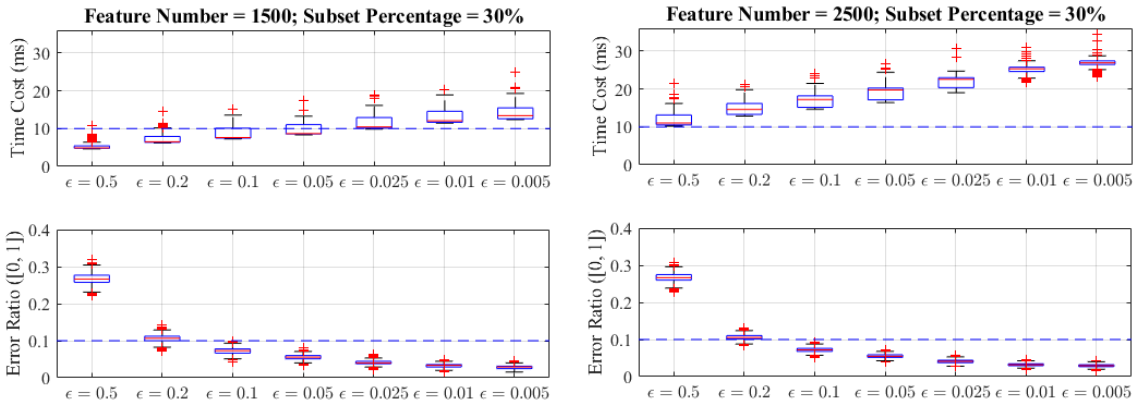


Figure 3.5: Lazier greedy with different decay factor ϵ under 2 example configurations: selecting 30% subset from 1500 and 2500 feature matchings. **First row**: time cost of lazier greedy. **Second row**: error ratio of lazier greedy.

lazy greedy, depending on ϵ. The plot includes a constant reference plane of $10ms$ time cost (in blue). The preference is to lie near to-or below-this reference plane, which lazier greedy can achieve over large regions of its parameter space while lazy greedy cannot. Moving to the error ratio graph, an error ratio of 0.1 indicates that the subset selected with lazier greedy is less than 10% different from the lazy greedy baseline. Though slow, this baseline has good performance for Good Feature Selection. According to Fig 3.4, the average error ratio of lazier greedy is below 0.1 for the majority of configuration surfaces when $\epsilon \leq 0.1$.

To further identify an acceptable decay factor ϵ, box-plots of time cost and error ratio under three configurations are presented in Fig 3.5, which vary by the number of matched features. We consider $\epsilon = 0.1$ a good option for Good Feature Selection: the time cost of corresponding lazier greedy is minimum under the requirement of less-than-0.1 error ratio. In what follows, all experiments run lazier greedy with $\epsilon = 0.1$.

3.5.3 Good Feature Matching Algorithm

Now we can tailor the lazier greedy to Good Feature Matching problem. Essentially, the assumption that all feature matchings are available shall be removed. At the beginning stage of Good Feature Matching, few feature matches is known. Under such a condition, the complete algorithm of Good Feature Matching is described in Alg 2.

Algorithm 2: Good Feature Matching in mono VSLAM.

Data: $P = \{p(1),\ p(2),\ \ldots,\ p(n)\}, Z = \{z(1),\ z(2),\ \ldots,\ z(m)\}, k$
Result: $M = < p(i), z(j) >,\ |M| = k$

foreach *3D feature* $p(i)$ **do**
    build Jacobians $\mathbf{H_x}(i),\ \mathbf{H_p}(i)$;
    $\mathbf{W}(i) = chol(\mathcal{I}_2 + \mathbf{H_p}(i)\boldsymbol{\Sigma_p}(i)\mathbf{H_p}(i)^T)$;
    $\mathbf{H_c}(i) = \mathbf{W}(i)^{-1}\mathbf{H_x}(i)$;
$M \leftarrow \emptyset, H_c^{sub} \leftarrow \emptyset$;
while $|M| < k$ **do**
    $H_c^R \leftarrow$ a random subset obtained by sampling $s = \frac{n}{k}\log(\frac{1}{\epsilon})$ random elements from H_c;
    while *1* **do**
        $\mathbf{H_c}(i) \leftarrow \arg\max_{\mathbf{H_c}(i) \in H_c^R} \log\det(\mathbf{H_c}(i)^T\mathbf{H_c}(i) + [\mathbf{H_c^{sub}}]^T[\mathbf{H_c^{sub}}])$;
        if *found matched measurement* $z(j)$ *for* $p(i)$ **then**
            $\mathbf{W}(i) = chol(\boldsymbol{\Sigma_z}(j) + \mathbf{H_p}(i)\boldsymbol{\Sigma_p}(i)\mathbf{H_p}(i)^T)$;
            $\mathbf{H_c}(i) = \mathbf{W}(i)^{-1}\mathbf{H_x}(i)$;
            $M \leftarrow M \cup < p(i), z(j) >$;
            break;
        else
            $H_c^R \leftarrow H_c^R \setminus \mathbf{H_c}(i)$;
            $H_c^R \leftarrow H_c^R \cup$ a random sample from H_c;
    $H_c^{sub} \leftarrow H_c^{sub} \cup \mathbf{H_c}(i)$;
    $H_c \leftarrow H_c \setminus \mathbf{H_c}(i)$;
    $Z \leftarrow Z \setminus z(j)$;
return M.

Good Feature Matching applies to stereo cameras as well as to monocular cameras. Compared to monocular VSLAM pipeline, stereo VSLAM has an additional module in data association: stereo matching, which associates measurements between left and right frames. Since the stereo algorithm associates existing 3D mapped features to 2D measurements from both frames, each paired measurement provides twice the number of rows to the least squares objective (in pose-only and joint BA). Stereo methods also provide for instant initialization of new map points through triangulated 2D measurements from the left and right frames. However, optimization for the current pose (as pursued in pose tracking) only benefits from the stereo matchings associated with existing 3D mapped features! By exploiting this property, we can design a lazy-stereo VSLAM pipeline that has lower

latency than the original stereo pipeline. Stereo matching is postponed after map-to-frame matching. Instead of searching for stereo matchings between all measurements, only those measurements associated with 3D map points are matched. After pose optimization, the remaining measurements are stereo-matched and triangulated as new 3D mapped features.

The lazy-stereo VSLAM pipeline should have the same level of accuracy and robustness as the original pipeline, with reduced pose tracking latency. Implementing the stereo good feature matching algorithm with the lazy-stereo pipeline will further reduce latency while preserving accuracy and robustness. Compared with monocular Alg 2, the stereo Alg 3 has additional steps of stereo matching at each successful iteration of map-to-frame feature matching (line 13 of Alg 3). Depending on the matching outcome, the block $H_c(i)$ contains map-to-frame information only (no stereo matching found; line 11-12 of Alg 3), or both map-to-frame and left-to-right information (stereo matching found; line 14-15 of Alg 3).

Algorithm 3: Good Feature Matching in stereo VSLAM.

Data: $P = \{p(1), \ldots, p(n)\}$, $Z = \{z(1), \ldots, z(m)\}$, $Z^r = \{z^r(1), \ldots, z^r(s)\}$, k
Result: $M = < p(i), z(j), z^r(r) >$, $|M| = k$
// line 1-9 identical with monocular version
10 **if** *found matched left measurement* $z(j)$ *for* $p(i)$ **then**
11 $\quad \mathbf{W}(i) = chol(\mathbf{\Sigma_z}(j) + \mathbf{H_p}(i)\mathbf{\Sigma_p}(i)\mathbf{H_p}(i)^T)$;
12 $\quad \mathbf{H_c}(i) = \mathbf{W}(i)^{-1}\mathbf{H_x}(i)$;
13 $\quad$ **if** *found matched right measurement* $z^r(d)$ *for* $p(i)$ **then**
14 $\quad\quad \mathbf{W^r}(i) = chol(\mathbf{\Sigma_z}^r(d) + \mathbf{H_p^r}(i)\mathbf{\Sigma_p}(i)\mathbf{H_p^r}(i)^T)$;
15 $\quad\quad \mathbf{H_c}(i) = [\mathbf{H_c}(i); \mathbf{W^r}(i)^{-1}\mathbf{H_x^r}(i)]$;
16 $\quad\quad M \leftarrow M \cup < p(i), z(j), z^r(d) >$;
17 $\quad$ **else**
18 $\quad\quad M \leftarrow M \cup < p(i), z(j), \emptyset >$;
19 $\quad$ break;
// rest of lines identical with line 15-21 of monocular version

3.6 Experiments

The Good Feature Matching algorithm is integrated into the map-to-frame matching function of monocular ORB-SLAM [4] and stereo ORB-SLAM [101]. For each input frame, the map-to-frame feature matchings are combined with keyframe-to-frame feature matchings, which are compute-economic to find. The combined matchings are feed into pose optimization for real-time camera pose estimation. In the following, the good-feature-matching enhanced ORB-SLAM is referred as *GF*, while the baseline ORB-SLAM is *ORB*. Two reference methods that prioritize feature matching with simple heuristics are also integrated into ORB-SLAM: 1) purely-randomized matching, i.e., *Rnd*; and 2) prioritizing map points with long tracking history, i.e., *Long*.

The EuRoC MAV benchmark [25] is chosen for the evaluation, since it covers a variety of challenging cases for pose tracking in VSLAM/VINS: fast motion, blurring, low light and low texture. For fair comparison between VSLAM and VO, the loop closing modules are disabled in all SLAM systems. Accuracy of real-time pose tracking is evaluated with the absolute root-mean-square difference (RMSE) between ground truth track and SLAM estimated track, as commonly used in SLAM evaluation [102]. The latency of real-time pose tracking per frame, i.e., time cost from receiving an image till publishing the state estimation, is also reported. The full evaluation results that include both RMSE and RPE/ROE are provided externally [3].

3.6.1 Monocular VSLAM with Latency Reduction

Apart from ORB-SLAM variants, we also report results of two state-of-the-art monocular VSLAM (*SVO* [6], *DSO* [7]), and two monocular VINS (*ROVIO* [33], *VIMono* [31]). Apart from the feature-based *ORB*, the other four baselines are with direct front-ends.

All results in the following are collected from desktops with the same spec: Intel i7-7700k quadcore 4.20GHz CPU (passmark score of 2583 per thread), Ubuntu 14.04 and

[3] `https://github.com/ivalab/FullResults_GoodFeature`

ROS Indigo. To ensure the results are reliable, a 10-run repeat is performed for each configuration, i.e., the benchmark sequence, the VSLAM approach and the parameter (number of features tracked per frame). Any results with tracking failure are discarded in the following, as it indicates some issue in robustness. In addition to the monocular ORB-SLAM baseline (*ORB*), two state-of-the-art monocular direct VO methods serve as baselines: *SVO*[4] [6] and *DSO*[5] [7]. *SVO* is a light-weight direct VO system targeting low-latency pose tracking while sacrificing tracking accuracy. The multi-threaded option in *SVO* is enabled, so that the depth update/mapping runs on a separate thread from pose tracking.

The latency and accuracy of VSLAM systems can be adjusted through a few internal parameters. One key parameter that significantly impacts both latency and accuracy is the *max feature number*, i.e., the maximum number of features/patches tracked per frame. Running VSLAM with high *max feature number* is beneficial for accuracy and robustness. Meanwhile, lowering the *max feature number* is preferred for latency reduction. To evaluate the trade-off between latency and accuracy for baseline systems (*ORB*, *SVO*, and *DSO*), all of them are configured to run 10-repeats for *max feature number* parameters ranging from 150 to 2000.

For a given *max feature number*, ORB-SLAM latency can be reduced via the Good Feature Matching algorithm. Adjusting the *good feature number*, i.e., the number of good features being matched in pose tracking, varies the observed latency. Tests with the three ORB-SLAM variants (*GF*, *Rnd* and *Long*) are configured to run 10-repeat under *good feature number* values ranging from 60 to 240. Meanwhile, the *max feature number* is fixed to 800, which yields a good balance of latency and accuracy for baseline *ORB*.

Fig. 3.6 present the latency-accuracy trade-off curves for monocular VSLAM implementations on three example EuRoC sequences Amongst the baseline methods, *ORB* has the best accuracy while *SVO* has the lowest latency. Lowering the *max feature number* reduces the latency of *ORB* baseline to some extent, however, it comes with clear loss of

[4] `http://rpg.ifi.uzh.ch/svo2.html`
[5] `https://github.com/JakobEngel/dso`

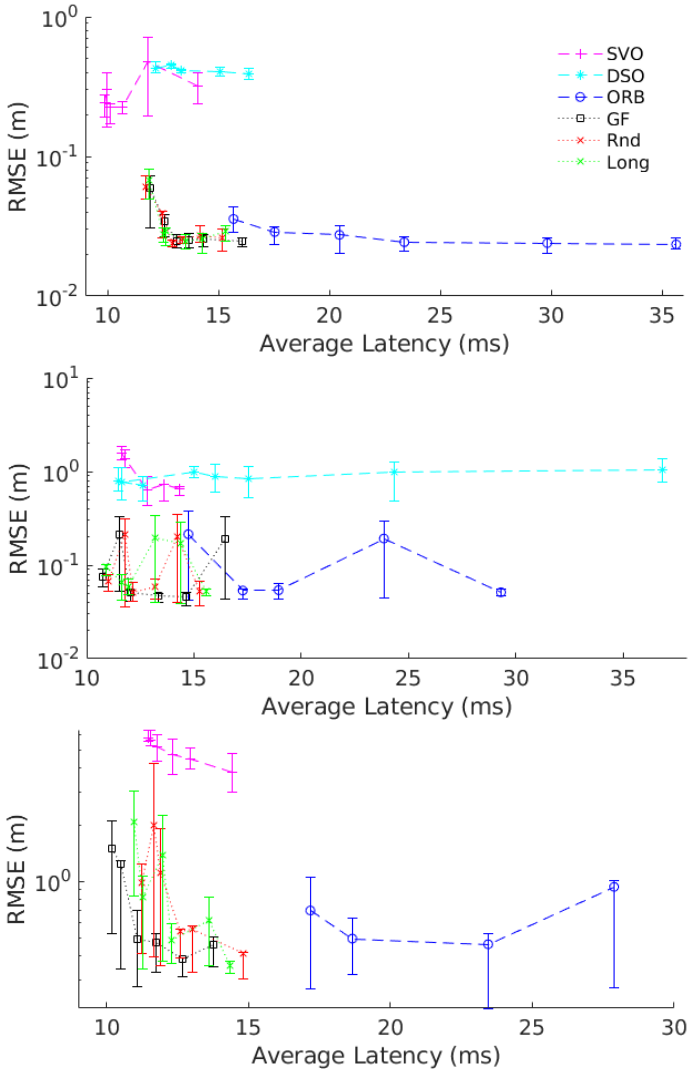


Figure 3.6: Latency vs. accuracy on 3 EuRoC Monocular sequences: *MH 01 easy*, *V2 02 med*, and *MH 04 diff* (from top to bottom). Baseline systems are evaluated with *max feature number* ranging from 150 to 2000; ORB-SLAM variants are evaluated with *good feature number* ranging from 60 to 240, and *max feature number* fixed to 800. Only the configurations with zero failure in 10-run repeat are plotted (e.g. all configurations of DSO fail to track on *MH 04 diff*, therefore omitted in row 3). Same rule applies to latency vs. accuracy figures afterwards.

tracking accuracy (e.g. the 1st blue marker in row 2), or even the risk of track failure (e.g. the first 2 blue markers are omitted in row 3). Meanwhile, better latency-accuracy trade-off is achieved with the proposed *GF* method. According to Fig 3.6, the latency of *GF* is in a similar range as *SVO*, but with the accuracy of *GF* being an order of magnitude better than both *SVO* and *DSO*. Furthermore, the accuracy-preserving property of *GF* is demonstrated

when comparing against reference methods *Rnd* and *Long*. The latency-accuracy curves of *GF* are almost flat and lower than the other two, once a reasonable number of good features are set to be matched (e.g. starting from the 3rd black marker).

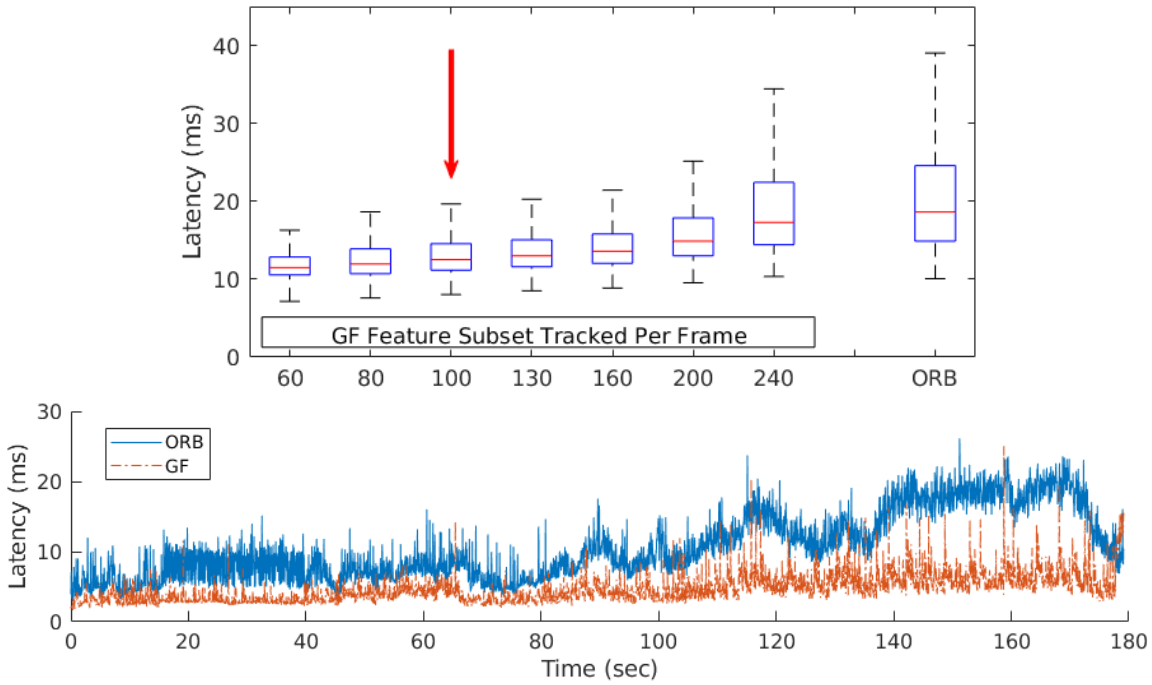


Figure 3.7: Latency vs. *good feature number* on EuRoC sequence *MH 01 easy*. **Top**: box-plots for *GF* and baseline *ORB*. **Bottom**: the latency vs. time trend of *GF* under 100 *good feature number* (marked with red arrow on left) and *ORB* for 1 run.

The latency-reduction of *GF* is further illustrated in Fig 3.7, in which the *max feature number* is set to 800. Compared with *ORB*, the latency of *GF* has lower variance. A good setting for the *good feature number* is 100, as marked by a red arrow in Fig 3.7. The accuracy of *GF* with 100 *good feature number* is on par with *ORB*, as quantified by the 3rd black marker in each row of Fig 3.6.

Last, we report the accuracy and latency of all monocular VSLAM methods under example configurations: the RMSE values are in Table 3.1 (after a *Sim3* alignment to the ground truth), and the latency values in Table 3.2. For the three VSLAM baselines, the *max feature number* is 800. For the three *ORB* variants, the *max feature number* is 800 and the *good feature number* is 100. Results with any tracking failure are omitted from both tables.

Table 3.1: RMSE (m) on EuRoC Monocular Sequences

	VSLAM					
Seq.	SVO	DSO	ORB	GF	Rnd	Long
MH 01 easy	0.227	0.407	0.027	0.025	**0.024**	0.029
MH 02 easy	0.761	-	**0.034**	0.043	0.038	0.040
MH 03 med	0.798	0.751	0.041	0.045	0.041	**0.040**
MH 04 diff	4.757	-	0.699	**0.492**	1.110	1.377
MH 05 diff	3.505	-	0.346	0.464	**0.216**	0.915
VR1 01 easy	0.726	0.950	0.057	0.037	**0.036**	0.037
VR1 02 med	0.808	**0.536**	-	-	-	-
VR1 03 diff	-	-	-	-	-	-
VR2 01 easy	0.277	0.297	0.025	0.024	0.025	**0.023**
VR2 02 med	0.722	0.880	0.053	**0.051**	**0.051**	0.059
VR2 03 diff	-	-	-	-	-	-
All Avg.	1.477	0.637	0.160	**0.147**	0.193	0.315
Int. Avg.	0.550	0.657	0.041	0.036	**0.035**	0.038

GF subset selection does not impact the robustness of ORB-SLAM: it works on all eight sequences that ORB also tracks. The average RMSE for all tracked sequences per method is given (i.e., All Avg.), as well as the average RMSE of 5 sequences that all methods track successfully (i.e., Int. Avg.).

On each EuRoC sequence, the minimum RMSE is noted in bold. Interestingly, *GF* does not just preserve the accuracy and robustness of *ORB*; it further reduces the RMSE on several sequences. On average, *GF* has the lowest RMSE over all evaluated VSLAM methods. Furthermore, *GF* also has better overall accuracy when compared with two reference selection methods. Though *Rnd* seems to have lowest RMSE on multiple sequences, the margin between *Rnd* and *GF* small for them. Meanwhile, both *Rnd* and *Long* lead to large accuracy loss on the difficult sequence *MH 04 diff*, while *GF* improves RMSE.

According to Table 3.2, the average latency of *GF* is the lowest relative to all other methods: *GF* has an average latency 34% lower than *ORB*! Compared with the direct methods, the latency of *GF* has lower variance. The 1st quartile of *GF* latency is higher than direct methods, since feature extraction introduces a constant overhead. However, the 3rd quartile of *GF* latency is lower than direct methods, which might occasionally spend too much time on direct optimization.

Table 3.2: Latency (ms) on EuRoC Monocular Sequences

	VSLAM					
	SVO	DSO	ORB	GF	Rnd	Long
Q_1	7.4	**5.8**	13.9	10.3	10.0	10.0
Avg.	12.6	16.4	18.4	**12.2**	12.3	12.3
Q_3	16.8	19.1	20.7	13.3	13.2	**13.0**

3.6.2 Stereo VSLAM with Latency Reduction

We also evaluate the latency-accuracy trade-off of stereo *GF* against state-of-the-art stereo VSLAM systems. Four baseline stereo systems are included in the evaluation: stereo *SVO*, stereo *DSO* (only on KITTI since no open-source implementation available), canonical stereo ORB-SLAM (*ORB*), and *Lz-ORB*, a sped-up version of stereo ORB-SLAM based on the lazy-stereo pipeline described earlier.

The Good Feature Matching (Alg 3) is integrated into the sped-up ORB-SLAM, *Lz-ORB*. In what follows, we again refer to the good feature enhanced ORB-SLAM as *GF*. As before, two heuristics are integrated into *Lz-ORB* as reference methods, i.e., *Rnd* and *Long*.

The latency-accuracy trade-off of stereo VSLAM on three example EuRoC sequences can be found at Fig 3.8. Among all 3 baseline systems, *Lz-ORB* has the best accuracy, while *SVO* has the lowest latency. Simply lowering the *max feature number* leads to accuracy drop or even track failure in *Lz-ORB*. However, with *GF* the latency of pose tracking can be reduced to the same level as *SVO*, while the RMSE remains a magnitude lower than *SVO*. Two state-of-the-art stereo VINS systems, *OKVIS*[6] [16] and *MSCKF*[7] [10], are evaluated as well. Both VINS systems are assessed under the default parameters, therefore rather than having the full curve only one marker is presented in Fig 3.8. The latency of *GF* is clearly lower than filter-based *MSCKF*, while the accuracy is even better than BA-based *OKVIS*. However, when comparing with two heuristics (*Rnd*, *Long*), the advantage of *GF* is harder to identify than monocular results.

[6] https://github.com/ethz-asl/okvis

[7] https://github.com/KumarRobotics/msckf_vio

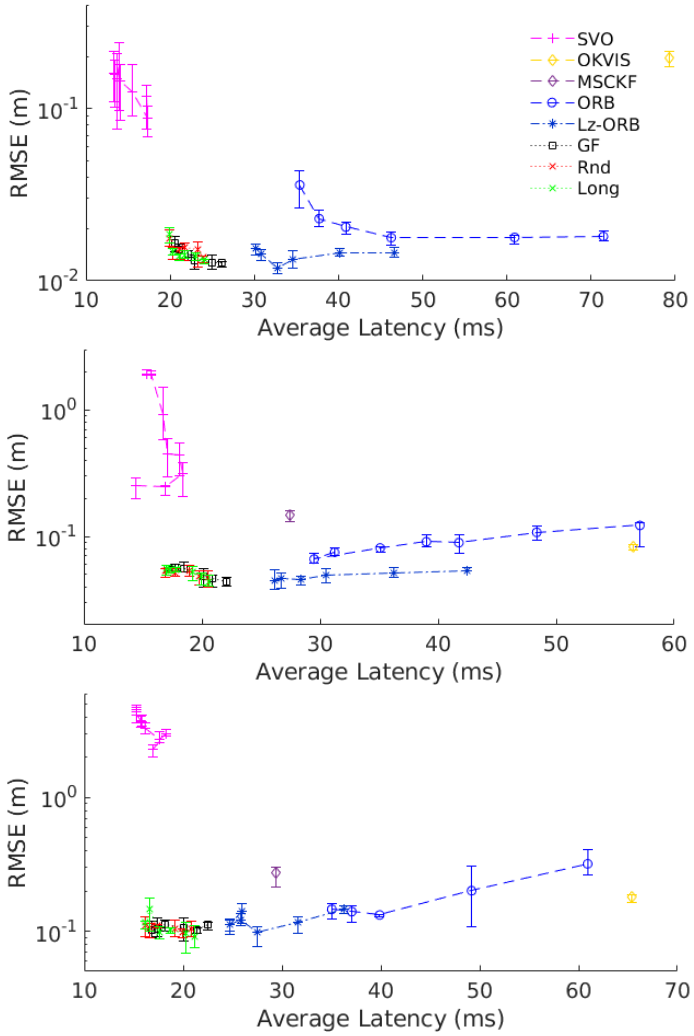


Figure 3.8: Latency vs. accuracy on 3 EuRoC Stereo sequences: *MH 01 easy*, *V2 02 med*, and *MH 04 diff* (from top to bottom). Baseline systems are evaluated with *max feature number* ranging from 150 to 2000; ORB-SLAM variants are evaluated with *good feature number* ranging from 60 to 240, and *max feature number* fixed to 800.

The latency reduction of *GF* is further illustrated in Fig 3.9. The *max feature number* being used in *Lz-ORB* and *ORB* is 800, which balances accuracy and latency. Compared with the two non-GF baselines, the latency of *GF* is has a lower upper bound. A reasonable *good feature number* is 160, since it yields low latency as well as high accuracy (the 3rd black mark to the right, in Fig 3.8).

The RMSEs and latencies of all 6 stereo VSLAM methods under the example configurations (*max feature number* of 800 and *good feature number* of 160) are summarized

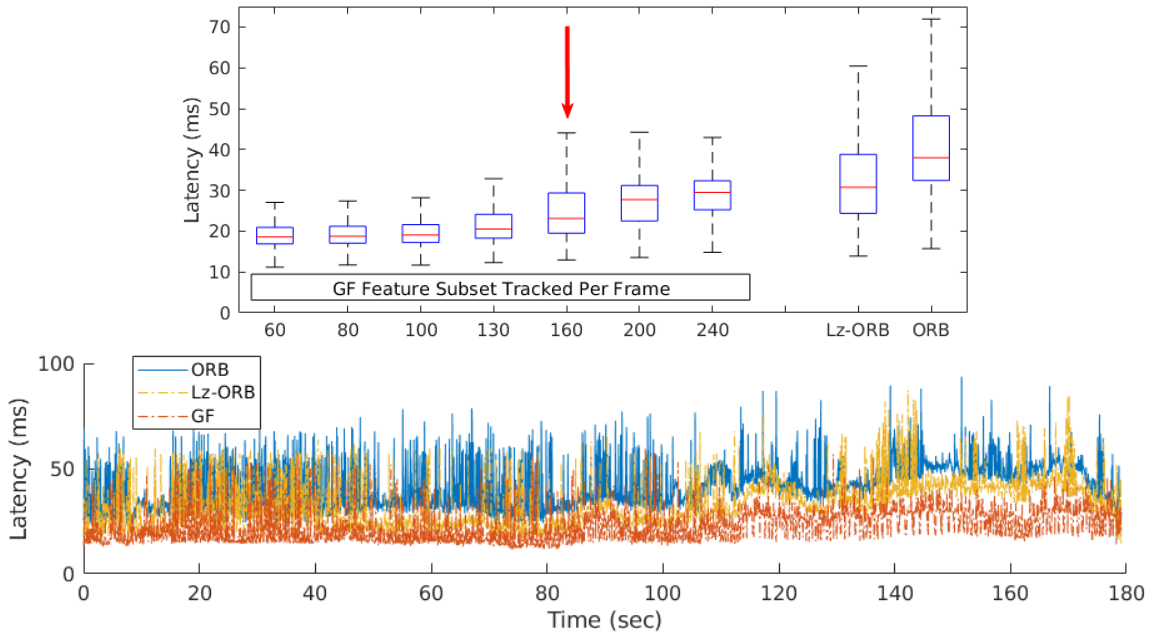


Figure 3.9: Latency vs. *good feature number* on EuRoC sequence *MH 01 easy*. **Top**: latency for *GF* under different *good feature number*, and 2 baselines *Lz-ORB* and *ORB*. **Bottom**: the latency trend of *GF* under 160 *good feature number* (marked with red arrow at the left), *Lz-ORB* and *ORB* in 1 run.

in Table 3.3. The results of 2 stereo VINS systems under default parameters are reported as well. Different from monocular VSLAM, it is expected for stereo systems to estimate scale correctly. Therefore, in each cell of Table 3.8 we report both the RMSE after *Sim3* alignment (as the 1st value) and the scale error percentage (as the 2nd value). The lowest error within each category, i.e., VSLAM or VINS, is highlighted in bold. Similar to the monocular experiment, *GF* is the lowest in terms of average RMSE and average scale error, compared with other stereo VSLAM methods. Furthermore, the accuracy of *GF* is better than the two stereo VINS systems, while the robustness of *GF* is at the same level as stereo VINS (each of them failed on 1 sequence). The advantage of *GF* over *Rnd* and *Long* can be verified as well: both *Rnd* and *Long* failed to track on *MH 02 easy* while *GF* succeed; both the average RMSE and the scale error of *GF* are lower than the other two as well.

The latency of all 8 stereo systems under the same configuration as Table 3.6 are summarized in Table 3.4. The lowest latency is achieved with *SVO*, though the accuracy of *SVO* is an order of magnitude higher than *GF*. The average latency reduction of *GF* is 27.4%

when compared against baseline *Lz-ORB*, and 46.2% when compared with *ORB*.

3.6.3 Real-time Tracking on Low-Power Devices

Last, the proposed *GF* is deployed on three low-power devices with limited processing speed, which serve as on-board processing unit for light weight platforms. The low-power devices being tested include:

1) X200CA: a light-weight laptop with Intel Pentium 2117U processor (passmark score 1662 per thread) and 4 GB of RAM. The processor has 2 cores, and requires 17 W.

2) Jetson TX2: a 64-bit embedded single-board computer system, containing a hybrid processing unit (2 Denver2 + 4 ARM A57) and 8 GB of RAM. The power consumption is 7.5 W.

3) Euclid: a 64-bit embedded single-board computer system, with a Intel Atom x7-Z8700 processor (passmark score 552 per thread) and 4 GB of RAM. The processor has 4 cores, and consumes 4 W of power.

The proposed *GF*, as well as 3 monocular VSLAM baselines, are deployed on these devices, and evaluated with EuRoC monocular sequences. To run *ORB* variants near real-time, the pyramid levels for ORB feature extraction were reduced to 3 from 8, and the *max feature number* set to 400. As a consequence, the robustness performance of the *ORB* variants is worse than the previous EuRoC Mono results. In what follows, we relax the robustness condition slightly, and report results with 1 tracking failure in 10 runs as well (marked with underline).

The RMSEs on all three low-power devices are summarized in Table 3.5, while the latencies are summarized in Table 3.6. The *max feature number* is set to 400, and the *good feature number* is set to 60.

1) When running on X200CA, *GF* has the 2nd lowest average RMSE (23% higher than *ORB*). However, the robustness of *GF* is slightly better than *ORB* and *SVO*: it tracks on 8 sequences without failure, while the other 2 baselines track 7 sequences and with failure.

Table 3.3: RMSE (m) and Scale Error (%) on EuRoC Stereo Sequences

	VSLAM						**VINS**	
Seq.	SVO	ORB	Lz-ORB	GF	Rnd	Long	OKVIS	MSCKF
MH 01 easy	0.179 (0.6)	0.021 (0.7)	**0.012** (**0.5**)	0.013 (**0.5**)	0.016 (**0.5**)	0.014 (**0.5**)	**0.196** (**1.7**)	-
MH 02 easy	-	0.021 (0.3)	**0.018** (**0.1**)	0.021 (**0.1**)	-	-	**0.114** (**1.4**)	0.184 (2.0)
MH 03 med	0.514 (2.3)	0.029 (**0.3**)	**0.024** (0.4)	0.025 (0.4)	0.025 (0.4)	0.025 (0.4)	**0.146** (**0.4**)	0.260 (1.3)
MH 04 diff	3.753 (26.1)	0.140 (1.1)	0.120 (0.6)	0.106 (**0.5**)	0.104 (**0.5**)	**0.102** (0.6)	**0.179** (**0.9**)	0.273 (1.0)
MH 05 diff	1.665 (4.9)	0.096 (**0.2**)	**0.059** (**0.2**)	0.068 (0.3)	0.064 (**0.2**)	0.103 (**0.2**)	**0.266** (**1.2**)	0.356 (2.1)
VR1 01 easy	0.264 (2.3)	**0.033** (0.8)	**0.033** (0.8)	0.035 (**0.7**)	0.035 (0.8)	0.036 (**0.7**)	**0.046** (**0.4**)	0.090 (0.9)
VR1 02 med	0.629 (11.2)	0.064 (**0.4**)	0.047 (0.7)	0.038 (0.7)	**0.032** (0.7)	0.036 (0.7)	**0.068** (0.5)	0.123 (**0.3**)
VR1 03 diff	0.655 (17.4)	0.214 (**2.2**)	0.112 (2.9)	**0.075** (**2.0**)	0.080 (2.1)	0.080 (2.1)	**0.120** (**1.0**)	0.187 (1.1)
VR2 01 easy	0.074 (1.7)	**0.031** (1.1)	0.033 (0.9)	0.044 (**0.5**)	0.041 (0.6)	0.042 (0.6)	**0.053** (0.8)	0.071 (**0.3**)
VR2 02 med	0.447 (3.6)	0.091 (**0.2**)	**0.046** (0.8)	0.049 (0.9)	0.053 (0.9)	0.053 (0.9)	**0.083** (**0.7**)	0.149 (1.0)
VR2 03 diff	**1.618** (**58.7**)	-	-	-	-	-	-	**1.162** (**39.9**)
All Avg.	0.980 (12.9)	0.074 (0.7)	0.050 (0.8)	**0.047** (**0.6**)	0.050 (0.7)	0.054 (0.8)	**0.127** (**0.9**)	0.285 (5.0)
Int. Avg.	1.000 (8.7)	0.087 (0.8)	0.059 (0.9)	0.055 (**0.7**)	**0.054** (0.8)	0.060 (0.8)	**0.120** (**0.7**)	0.189 (1.0)

Table 3.4: Latency (ms) on EuRoC Stereo Sequences

	VSLAM						VINS	
	SVO	ORB	Lz-ORB	GF	Rnd	Long	OKVIS	MSCKF
Q_1	**8.6**	30.0	21.5	14.5	14.2	14.2	50.5	**19.9**
Avg.	**16.4**	38.5	28.5	20.7	19.9	20.1	65.1	**28.3**
Q_3	23.3	44.2	32.1	24.2	**22.5**	22.9	80.3	**36.0**

When comparing on the 7 sequences that *ORB* tracks, *GF* only introduces 14% to average RMSE. The strength of *SVO* is the low-latency; though the average latency of *GF* is 24% less than *ORB*, it is still almost twice as much as the latency of *SVO*.

2) The released binary of *SVO* does not support 64-bit Jetson TX2, therefore only 3 methods are assessed on Jetson. Similar to the X200CA results, *GF* is slightly worse than *ORB* in terms of average RMSE (by 8%). Notice *GF* is also less robust than *ORB*, as it introduces additional tracking failure on sequences *MH 02 easy* and *MH 04 diff*. The latency reduction of *GF* is also small: 11% less than *ORB*.

3) When running on Euclid, *GF* introduces 20% more error in terms of average RMSE. Again, notice that *GF* works on *MH 05 diff* while *ORB* cannot. If we only take the 6 sequences that *ORB* tracks into account, *GF* only introduces 4% to average RMSE. However, the latency reduction of *GF* is smaller than the Jetson results: only 9% time savings. Apart from the 4 monocular VSLAM systems, we also include the VINS results [11] evaluated on a UP Board, which has almost identical hardware specifications as Euclid. The RMSE of the VINS methods, labeled *SVOMSF*[11] and *VIMono*[31], are obtained by *Sim3* alignment to ground truth, which is identical with our evaluation. With additional input from inertial sensors, VINS are clearly more robust than vision-only systems. However, the accuracy of VINS is poorer than vision-only ones (when scale corrected). Furthermore, the latency of the VINS approaches is much higher than vision-only systems, which suggests the scalability of VINS is also poor for low-power devices. Therefore, for VSLAM and VINS, combination of algorithm improvements (e.g. Good Feature) and hardware improvements may be required to achieve low latency and good accuracy on embedded devices.

Table 3.5: RMSE (m) On EuRoC Monocular Systems, Running on Low-power Devices.

	X200CA				**Jetson**			**Euclid**					
Seq.	SVO	DSO	ORB	GF	DSO	ORB	GF	SVO	DSO	ORB	GF	SVOMSF	VIMono
MH 01 easy	0.327	-	0.041	**0.036**	-	**0.033**	0.037	0.244	-	0.044	**0.041**	0.29	0.20
MH 02 easy	-	-	0.053	**0.047**	-	**0.046**	0.135	-	-	**0.044**	0.045	0.31	0.18
MH 03 med	1.14	-	**0.050**	0.056	-	**0.055**	0.059	1.21	-	**0.050**	0.051	0.66	0.17
MH 04 diff	-	-	**0.281**	0.457	-	**0.231**	-	-	-	0.232	0.248	2.02	**0.12**
MH 05 diff	2.54	-	0.289	**0.233**	-	**0.258**	0.340	2.84	-	-	**0.158**	0.87	0.35
VR1 01 easy	0.552	-	**0.036**	**0.036**	0.826	**0.036**	**0.036**	0.645	-	**0.036**	0.040	0.36	0.05
VR1 02 med	**0.730**	-	-	-	-	-	-	**0.857**	-	-	-	0.78	**0.12**
VR1 03 diff	-	-	-	-	-	-	-	-	-	-	-	-	**0.10**
VR2 01 easy	0.397	0.295	0.032	**0.029**	0.288	0.029	**0.027**	0.402	0.300	**0.030**	0.032	0.33	0.08
VR2 02 med	0.634	0.832	-	**0.213**	**0.941**	-	-	0.688	-	-	-	0.59	**0.08**
VR2 03 diff	-	-	-	-	-	-	-	-	-	-	-	-	**0.17**
All Avg.	0.903	0.564	**0.112**	0.138	0.685	**0.098**	0.106	0.984	0.300	**0.073**	0.088	0.69	0.15
Int. Avg.	-	-	**0.112**	0.128	-	**0.076**	0.106	-	-	**0.073**	0.076	0.66	0.13

Table 3.6: Latency (ms) On EuRoC Monocular Systems, Running on Low-power Devices.

	X200CA				**Jetson**			**Euclid**					
	SVO	DSO	ORB	GF	DSO	ORB	GF	SVO	DSO	ORB	GF	SVOMSF	VIMono
Q_1	**8.6**	12.4	19.3	14.5	**21.5**	30.2	25.7	**12.6**	21.5	28.7	24.8	29.8	88.9
Avg.	**9.8**	15.0	24.6	18.7	32.1	35.1	**31.1**	**13.4**	37.3	35.9	32.6	37.5	153.9
Q_3	**12.5**	16.4	28.5	21.0	37.5	38.8	**33.1**	**15.9**	51.1	41.4	39.3	42.1	209.5

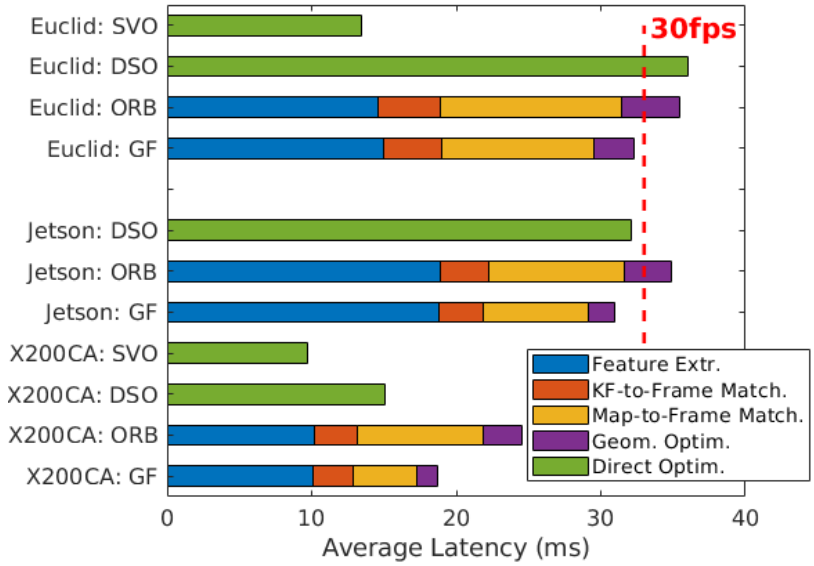


Figure 3.10: Latency breakdown for all modules in pose tracking pipeline, running on low-power devices.

When the computation resources (e.g. processor speed, cache size) are highly limited, the latency reduction of *GF* is less significant. The preservation of accuracy and robustness, on the other hand, scales relatively well on different devices (only with slight drop). The limited scalability to devices such as Jetson and Euclid is mostly due to the sequential nature of the proposed *GF* algorithm. As embedded device hardware specifications improve, in terms of compute power and core quantity, we anticipate that improvements will favor the *GF* variant (as demonstrated on desktop and X200CA). Even on current embedded platforms, the small amount of latency reduced by *GF* could be important: it turns the near real-time *ORB* into a real-time applicable VSLAM system, as illustrated in Fig 3.10.

3.7 Conclusion

This section presents an active map-to-frame feature matching method, Good Feature Matching, which reduces the computational cost (and therefore latency) of VSLAM, while preserving the accuracy and robustness of pose tracking. The feature matching effort is connected to the submatrix selection problem. To that end, the *Max-logDet* matrix revealing metric was shown to perform best via simulated scenarios. For application to active feature

matching, the combination of deterministic selection (greedy) and randomized acceleration (random sampling) is studied. The described Good Feature Matching algorithm is integrated to both monocular and stereo feature-based VSLAM systems, followed by extensive evaluation on multiple benchmarks and computation platforms. Good Feature Matching is shown to be an efficiency enhancement for low-latency VSLAM, while preserving, if not improving, the accuracy and robustness of VSLAM.

In the future, Good Feature Matching can be combined with dedicate hardware (e.g. FPGA) to further boost the performance-efficiency of VSLAM on low-power devices, since the overhead of feature extraction (illustrated in Fig 3.10) can be greatly reduced. Another interesting direction is to combine the advantage of feature-based and direct front-end. Feature matching provides long-baseline associations that are beneficial to the overall accuracy of VSLAM, while direct measurements serve as the short-term constraints for the robustness of VSLAM.

CHAPTER 4
GOOD LINE CUTTING: ACCURACY IMPROVEMENT OF LINE-ASSISTED VSLAM

4.1 Introduction

In this chapter, we present the work of Good Line Cutting, which tackles a key problem in line-assisted VSLAM: accurately solving the least squares pose optimization with unreliable 3D line input. Line features serve as sensible alternatives or additions to point features, given that edges are also fairly abundant in images; especially within man-made environments where sometimes the quantity of points may be lacking to the detriment of VSLAM. The canonical examples being corridors and hallways, whose low-texture degrades the performance of point features methods. Under these circumstances, lines become more reliable constraints versus points.

Adding line features to VSLAM is not a trivial task. Triangulating a 3D line from 2D measurements requires more measurements and is more sensitive to measurement noise, compared to points. Lines are generally weak in constraining the correspondence along its direction of expansion. It is hard to establish reliable point-to-point correspondence between two lines (as segments), which degrades triangulation accuracy. In addition, lines are usually partially-occluded, which brings the challenge of deciding the endpoint correspondence. Examples of 3D lines reconstructed with state-of-the-art line-assisted VSLAM system, PL-SLAM [23], are provided in Fig 4.1. When compared against the ground truth floor plan, the reconstructed 3D lines are clearly off. To solve line-based pose estimation accurately when the 3D line references are potentially erroneous, the low-reliability of triangulated 3D lines has to be resolved.

To reduce the impact of unreliable 3D lines, a common practice is to model the un-

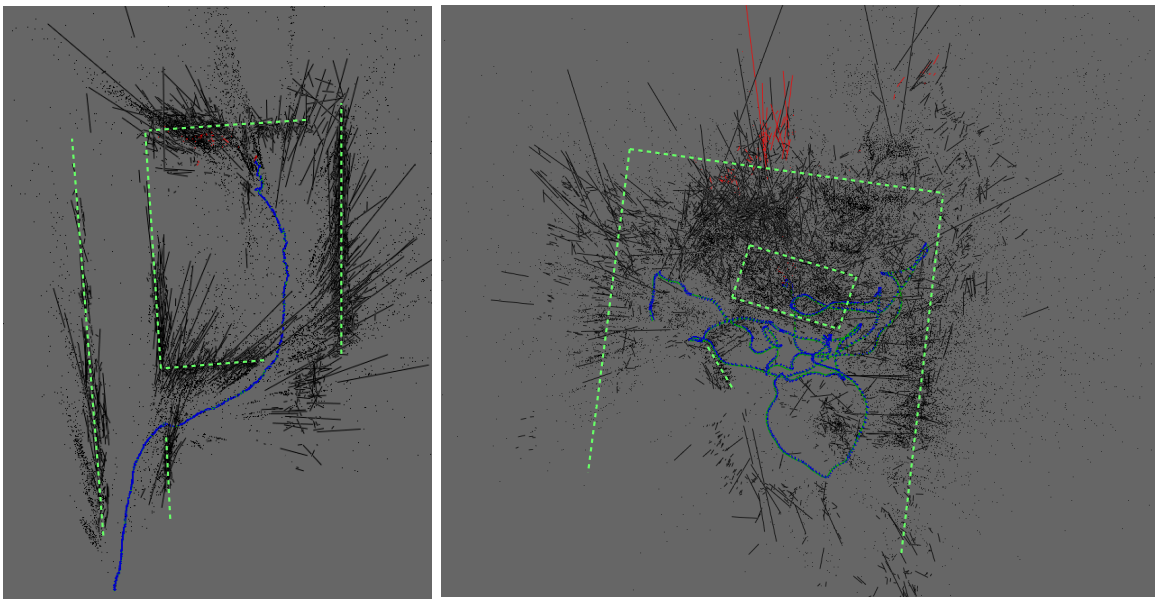


Figure 4.1: The map that includes 3D points and 3D lines estimated with line-assisted PL-SLAM [23]. **Left:** map of gazebo simulate office environment. **Right:** map of EuRoC MAV sequence *V1 01 easy* captured in a room. The 3D lines maintained (and referred) in the map are in black solid lines. The outline of actual floor plan, which serves as the ground truth for 3D lines around, are in green dash lines. Notice the significant error in 3D line map, when compared against the ground truth floor plan.

certainty of the 3D line, and weight the contribution of each line accordingly in pose optimization. The information matrix of the line residual [103, 104, 105, 23] is one of such weighting terms. The residuals of uncertain lines get less weight so that the optimized pose is biased in favor of the certain lines. However, uncertainty of line residual does not immediately imply incorrect pose estimation (though there is some correlation): a certain line residual term might barely contribute to pose estimation, whereby it would make no sense to weight it highly. We posit that, in lieu of the uncertainty of line residual, the uncertainty of pose estimation should be assessed and exploited.

Another way to reduce uncertainty is to simply drop highly-uncertain lines when numerically constructing the pose optimization problem. However, line features are typically low in quantity (e.g. tens of lines). Too much information could be lost by dropping line features. Furthermore, there is a high risk of forming ill-conditioned optimization problem.

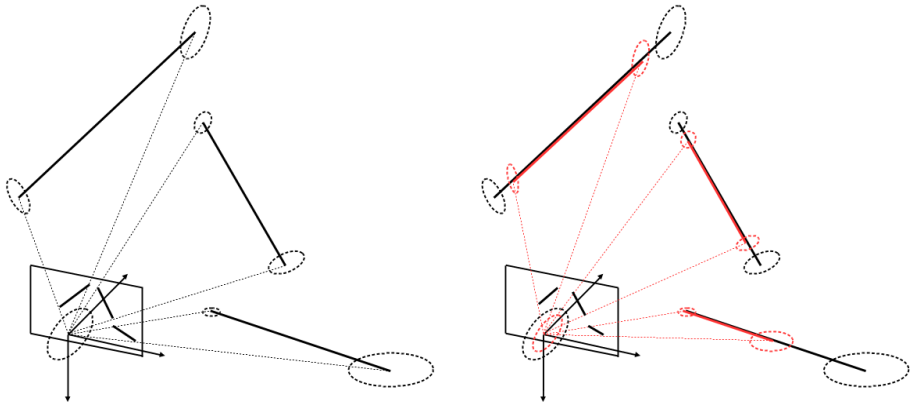


Figure 4.2: A toy case illustrating the proposed Good Line Cutting approach. **Left:** Giving 3 line matchings with confidence ellipsoids (dashed line), the least squares pose estimation has high uncertainty. **Right:** Line-cutting applied to the line-based least squares problem. The cut line segments and their corresponding confidence ellipsoids are in red. The confidence ellipsoid of the new pose estimation improves.

As opposed to line weighting and dropping, this work aims to improve pose optimization through the concept of **Good Line Cutting**. The goal of Good Line Cutting is simple: for each 3D line, find the line segment that contributes the largest amount of information to pose estimation (a.k.a. a good line), and select only those informative segments to solve pose optimization. With line cutting, the conditioning of the optimization problem improves, leading to more accurate pose estimation than the original problem. An illustration of Good Line Cutting can be found at Fig 4.2. To the best of the authors' knowledge, this is the first work discussing the role of line cutting in line-based pose optimization. The contributions of this chapter are:

1) Demonstration that Good Line Cutting improves the overall conditioning of line-based pose optimization;

2) An *efficient algorithm* for real-time applications that approaches the computationally more involved joint optimization solution to Good Line Cutting; and

3) Integration of **Good Line Cutting** algorithm *into a state-of-the-art line-assisted VSLAM* system. When evaluated in two target scenarios (motion blur and low-texture), the proposed line cutting leads to accuracy improvements over line-weighting, while preserving

the robustness of line-assisted pose tracking.

4.2 Background

There are continuous effort investigating line features in the SLAM community. In the early days of visual SLAM, lines features are used to improve the large view change in monocular camera tracking [17, 18]. In [17], the authors integrate lines into a point-based monocular Extended Kalman Filter SLAM (EKF-SLAM). Real-time pose tracking with lines only are demonstrated in [18] by using a Unscented Kalman Filter (UKF). Both methods model 3D lines as endpoint-pairs, project endpoint-pairs to image, then measure the point-wise residual. Alternatively, edges are extracted and utilized [106]. For the convenience of projection, a 12-DOF over-parameterization is used to model 3D edge. Again, the edge residual is measured after 3D-to-2D projection.

Line-assisted VSLAM has been studied with 3D visual sensors, such as RGB-D sensor and stereo camera. In [103], a line-assisted RGB-D odometry system is proposed. It involves parameterizing the 3D lines as 3D endpoint-pairs and minimizing the endpoint residual in $SE(3)$. However, directly working in $SE(3)$ has the disadvantage of being sensitive to inaccurate depth measurements. With the progress in line detectors (e.g. LSD [19]) and descriptors (e.g. LBD [20]), matching and triangulating 3D lines from 2D color image become feasible in real-time VSLAM. As a consequence, stereo [107, 21, 23] and monocular [108, 109, 105] line-assisted VSLAM systems are developed. Though alternative parameterizations have been explored (e.g. Plücker coordinate [22], orthonormal representation [110]), most line-assisted VSLAM continued to use the 3D endpoint-pair parametrization because it conveniently combines with the well-established point-based optimization. Pose estimation typically jointly minimizes the reprojection errors of both point and line matches. For line features, the endpoint-to-line distance is chosen as the reprojection error term, i.e., *the line residual*. To cope with the 3D line uncertainty, a covariance matrix is maintained for each 3D line. Each line residual term is weighted by the

inverse of the covariance matrix obtained by propagating the covariance from the 3D line to the endpoint-to-line distance.

Line-assisted methods building from direct VSLAM have also been developed [109, 105]. Interestingly, neither of them use direct measurements (e.g. photometric error) for line terms in the joint optimization objective. Instead, the line residual is the least squares of endpoint-to-line distance, which is identical to other feature-based approaches.

Research into line-assisted VSLAM is still ongoing. Among the systems described above, there is a set of modules employed in common: 1) 3D lines parameterized as 3D endpoint-pairs; 2) endpoint-to-line distance and variants serve as the line residual; 3) in the optimization objective (pose only and joint), line residuals are weighted by some weighting matrix. The Good Line Cutting approach described in this work expands on these three modules.

4.3 Conditioning of Line-assisted Pose Tracking Objective

Similar to the Good Feature Matching work, we begin with the general least squares objective in line-assisted VSLAM:

$$\hat{x} = \arg\min\{\|\mathbf{p} - \mathbf{h}(x, \mathbf{P})\|^2 + \left\|\mathbf{l}^T\mathbf{h}(x, \mathbf{L})\right\|^2\} \tag{4.1}$$

where $\mathbf{p}$ and $\mathbf{l}$ are stacked matrices of 2D point measurements $\{\mathbf{p_i}\}$ and 2D line coefficients $\{\mathbf{l_i}\}$, respectively. $\mathbf{P}$ is the stacked matrix of 3D points $\{P_i\}$, while $\mathbf{L}$ is the stacked matrix of all endpoints from the 3D line set $\{\mathbf{L_i}\}$. $h(x, \mathbf{P})$ consists of the pose transformation (decided by x) and pin-hole projection. For simplicity, the least squares (4.1) is referred to as line-LSQ problem.

Solving the line-LSQ (4.1) often involves the first-order approximation of the non-linear measurement function. For instance, the endpoint-to-line distance $h(x, \mathbf{L})$ on image plane

can be approximated as,

$$\mathbf{h}(x, \mathbf{L}) = \mathbf{h}(x_0, \mathbf{L}) + \mathbf{H_x}(x - x_0) \tag{4.2}$$

so that the least squares of line residual term can be minimized with Gauss-Newton method, which iteratively updates the pose estimate:

$$\hat{x} = x_0 - (\mathbf{l}^T\mathbf{H_x})^+\mathbf{l}^T(h(x_0, \mathbf{L})) \tag{4.3}$$

Accuracy of $\hat{x}$ is affected by two types of error in line features: 2D line measurement error and 3D line triangulation error. As mentioned earlier, 3D line triangulation is sensitive to noise and less reliable than 3D point triangulation. Therefore, here we only consider the error of 3D line endpoint $\mathbf{L}$ while assuming the 2D measurement $\mathbf{l}$ is accurate. Again, with the first-order approximation of $h(x_0, \mathbf{L})$ at the initial pose x_0 and triangulated 3D endpoint $\mathbf{L_0}$, we may connect the pose optimization error ϵ_x and 3D line endpoint error ϵ_L,

$$\epsilon_x = (\mathbf{l}^T\mathbf{H_x})^+\mathbf{l}^T\mathbf{H_L}\epsilon_L = \mathbf{H}^T\epsilon_L \tag{4.4}$$

where $\mathbf{H}^T = (\mathbf{l}^T\mathbf{H_x})^+(\mathbf{l}^T\mathbf{H_L})$. Here we intentionally ignore the error of point residual term. The reason is, when available, point features are known to be more accurate. Therefore, the main source of error in line-LSQ problem is from 3D line triangulation ϵ_L, which is propagated by $\mathbf{H}$.

Following the common error model of independent distributed zero-mean Gaussian in inverse-depth parametrization, we may derive the pose information matrix $\boldsymbol{\Omega_x}$ from Eq 4.4,

$$\boldsymbol{\Omega_x} = \mathbf{H}^T\boldsymbol{\Omega_L}\mathbf{H} = \sum \mathbf{H_i}^T\boldsymbol{\Omega_{L_i}}\mathbf{H_i} \tag{4.5}$$

where $\mathbf{H_i}$ is the corresponding row block in $\mathbf{H}$ for line $\mathbf{L_i}$, and $\boldsymbol{\Omega_{L_i}}$ is the information matrix of 3D endpoint-pair used to parametrize $\mathbf{L_i}$. $\boldsymbol{\Omega_{L_i}}$ is a block diagonal matrix under

the independent distributed assumption on 3D endpoint error. Set $\mathbf{\Omega}_{\boldsymbol{L_i(0)}}$, $\mathbf{\Omega}_{\boldsymbol{L_i(1)}}$ as the two diagonal blocks of $\mathbf{\Omega}_{\boldsymbol{L_i}}$, and $\mathbf{H_i}(0)$, $\mathbf{H_i}(1)$ the corresponding row block in $\mathbf{H_i}$, then (4.5) can be further broken down into:

$$\begin{aligned}\mathbf{\Omega}_{\boldsymbol{x}} &= \sum [\mathbf{H_i}^T(0)\mathbf{\Omega}_{\boldsymbol{L_i(0)}}\mathbf{H_i}(0) + \mathbf{H_i}^T(1)\mathbf{\Omega}_{\boldsymbol{L_i(1)}}\mathbf{H_i}(1)] \\ &= \sum \mathbf{H_i}^T(\alpha_i)\mathbf{\Omega}_{\boldsymbol{L_i(\alpha_i)}}\mathbf{H_i}(\alpha_i)\end{aligned} \tag{4.6}$$

where we extend the range of i from n lines to $2n$ endpoints, and set $[\alpha_i]$ as a $2n \times 1$ chessboard vector filled with 0 and 1.

As pointed out in the literature of point-feature selection (e.g. [93, 86, 87, 74] and chapter 3), the spectral property of the pose information matrix has strong connection with the error of least squares pose optimization. For example, the worst-case error variance is quantified by the inverse of minimum eigenvalue of $\mathbf{\Omega}_{\boldsymbol{x}}$ [86, 87]. Large min-eigenvalue of $\mathbf{\Omega}_{\boldsymbol{x}}$ is preferred to avoid fatal error in line-LSQ solving. Also, the volume of the confidence ellipsoid in pose estimation can be effectively measured with the log-determinant of $\mathbf{\Omega}_{\boldsymbol{x}}$ [74]. For accurately solving the line-LSQ problem, the large log-determinant of $\mathbf{\Omega}_{\boldsymbol{x}}$ is pursued. In what follows, we quantify the spectral property of $\mathbf{\Omega}_{\boldsymbol{x}}$ with log-determinant, i.e., $\log\det(\mathbf{\Omega}_{\boldsymbol{x}})$.

4.4 Good Line Cutting using Max-LogDet

4.4.1 Intuition of Good Line Cutting

Compared with points that are typically modeled as sizeless entity, lines are modeled to extend along one certain dimension. For a 3D line $\mathbf{L_i}$ defined by endpoint-pair $\mathbf{L_i}(0)$ and $\mathbf{L_i}(1)$ in Euclidean space, the following equations hold for any intermediate 3D point $\mathbf{L_i}(\alpha)$

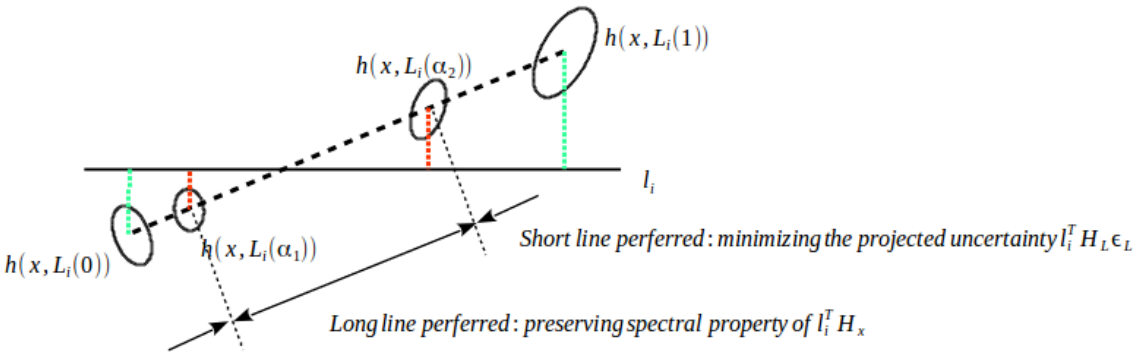


Figure 4.3: Illustration of Good Line Cutting intuition. The final line cutting behavior is jointly determined by two motivations: uncertainty-reduction and information-preservation.

that lies on $\mathbf{L_i}$:

$$\mathbf{L_i}(\alpha) = (1-\alpha)\mathbf{L_i}(0) + \alpha\mathbf{L_i}(1)$$

$$\boldsymbol{\Omega}_{\boldsymbol{L_i(\alpha)}} = \boldsymbol{\Sigma}_{\boldsymbol{L_i(\alpha)}}{}^{-1} = \{(1-\alpha)^2\boldsymbol{\Sigma}_{\boldsymbol{L_i(0)}} + \alpha^2\boldsymbol{\Sigma}_{\boldsymbol{L_i(1)}}\}^{-1},$$

where α is the interpolation ratio, and $\boldsymbol{\Sigma}_{\boldsymbol{L_i(*)}}$ is the covariance matrix of 3D point $\mathbf{L_i}(*)$.

The covariance matrix of the intermediate 3D point, $\boldsymbol{\Sigma}_{\boldsymbol{L_i(\alpha)}}$, is **convex** to the interpolation ratio α, as both $\boldsymbol{\Sigma}_{\boldsymbol{L_i(0)}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{L_i(1)}}$ are positive semi-definite. At some specific $\alpha_m \in [0,1]$, $\boldsymbol{\Sigma}_{\boldsymbol{L_i(\alpha_m)}}$ reaches a global minimum (and $\boldsymbol{\Omega}_{\boldsymbol{L_i(\alpha_m)}}$ a global maximum). In other word, at some intermediate 3D position $\mathbf{L_i}(\alpha_m)$ (both endpoints included) the corresponding 3D uncertainty is minimized. The same conclusion holds when extending from a single 3D point to the 3D point-pair $\langle \mathbf{L_i}(\alpha_1), \mathbf{L_i}(\alpha_2) \rangle$ lying on the 3D line $\mathbf{L_i}$: both 3D points share the least-uncertain position $\mathbf{L_i}(\alpha_m)$. To minimize the amount of uncertainty introduced with 3D line endpoints, the 3D line $\mathbf{L_i}$ will shrink to a single 3D point!

However, the pose information $\boldsymbol{\Omega}_{\boldsymbol{x}}$ is not only dependent on endpoint information matrix $\boldsymbol{\Omega}_{\boldsymbol{L_i(\alpha)}}$, but also the Jacobian term $\mathbf{H_i}(\alpha) = (\mathbf{l_i}^T\mathbf{H_x}(\alpha))^+(\mathbf{l_i}^T\mathbf{H_L}(\alpha))$. Cutting 3D line into smaller segments will affect the corresponding Jacobian term as well. Intuitively, line cutting could hurt the spectral property of measurement Jacobian block $\mathbf{H_x}(\alpha)$: if a 3D line gets cut to a single point, the corresponding measurement Jacobian will degenerate from

rank-2 to rank-1, thereby losing one of the two constraints provided by the original 3D line matching.

Therefore, the objective of Good Line Cutting can be written as follow,

$$\begin{aligned}[\alpha_i] &= \arg\max \log\det(\boldsymbol{\Omega_x}) \\ &= \arg\max \log\det[\sum \mathbf{H_i}^T(\alpha_i)\boldsymbol{\Omega_{L_i(\alpha_i)}}\mathbf{H_i}(\alpha_i) + \boldsymbol{\Omega_x^{pt}}]\end{aligned} \tag{4.7}$$

where we include a constant term $\boldsymbol{\Omega_x^{pt}}$ to capture the information from point features, if applicable. Naturally, this objective can be solved with nonlinear optimization techniques.

4.4.2 Validation of Good Line Cutting

Before describing the optimization of (4.7), we would like to validate the idea of line cutting. One natural question towards line cutting with (4.7) being, is it possible that the Jacobian term $\mathbf{H_i}^T(\alpha_i)$ has much stronger impact towards (4.7) than 3D uncertainty reduction, so that one should always use the **full-length of 3D line**? To address this question, we study the minimal case, **single line cutting**: only one pair of cutting ratio $\langle\alpha_1, \alpha_2\rangle$ can be changed, while the remaining $n-1$ lines are not cut.

It is cumbersome to derive the function from line cut ratio α to Jacobian term $\mathbf{H_i}(\alpha)$: it is highly non-linear, and the Jacobian term varies under different $SE(3)$ parameterizations of camera and 3D lines. Instead, a set of line-LSQ simulation are conducted to validate line cutting.

The testbed is developed based on the simulation framework of [99]. A set of 3D lines that form a cuboid are simulated, under homogeneous-points line (*HPL*) parameterization. To simulate the error in 3D line triangulation, the endpoints of 3D lines are perturbed with zero-mean Gaussians in inverse-depth space, as illustrated with blue lines in Fig. 4.4 left. For the 3D line in red, the optimal line cutting ratio, found through brute-force search, is plotted versus camera pose in Fig. 4.4 right. The boxplots indicate that cutting happens when the 3D line is orthogonal or parallel to the camera frame. In these cases, the

measurement Jacobian of the red 3D line scales poorly with line length. Taking a smaller segment/point is preferred so as to introduce less noise into the least squares problem. According to Fig. 4.4, line cutting adapts to the information and uncertainty of the tracked lines based on the relative geometry.

To visualize the outcomes of different line cutting ratios, we used brute-force sweep to generate the surface of $\log\det(\boldsymbol{\Omega}_{\boldsymbol{x}})$ as a function of the line cutting ratio parameters. Three example surfaces are illustrated in Fig 4.5. In the 1st example, global maximum of $\log\det(\boldsymbol{\Omega}_{\boldsymbol{x}})$ is at $\langle\alpha_1 = 0, \alpha_2 = 1.0\rangle$, which indicates the full-length of 3D line should be used. The 2nd one has global maximum at $\langle\alpha_1 = 0, \alpha_2 = 0.76\rangle$, which encourages cutting out part of the line. In column 3, $\log\det(\boldsymbol{\Omega}_{\boldsymbol{x}})$ is maximized at $\langle\alpha_1 = 0.52, \alpha_2 = 0.52\rangle$, which means the original 3D line should be aggressively cut to a 3D point. To maximize pose information, line cutting is definitely preferred in some cases (e.g. Fig 4.5 columns 2 and 3).

4.5 Efficient Good Line Cutting

4.5.1 Single Line Cutting

To begin with, consider the single line cutting problem as simulated previously. Based on Fig 4.5, we notice the mapping from $\langle\alpha_1, \alpha_2\rangle$ to $\log\det(\boldsymbol{\Omega}_{\boldsymbol{x}})$ is continuous, and is concave within a neighborhood. Therefore by doing gradient ascent in each of the concave regions, the global maximum of $\log\det(\boldsymbol{\Omega}_{\boldsymbol{x}})$ is expected to be found. One possible triplet of initial pairs are: full-length $\langle\alpha_1 = 0, \alpha_2 = 1.0\rangle$, 1st endpoint only $\langle\alpha_1 = 0, \alpha_2 = 0\rangle$, and 2nd endpoint only $\langle\alpha_1 = 1.0, \alpha_2 = 1.0\rangle$.

The effectiveness of the multi-start gradient ascent is demonstrated with 100-run repeated test. Two representative endpoint-pair parameterizations of 3D lines [99] are tested here: homogeneous-points line (*HPL*) and inverse-depth-points line (*IDL*). The error of endpoint estimation is simulated with i.i.d. Gaussian in inverse-depth space (standard deviation of 0.005 and 0.015 unit are used), and propagated to $SE(3)$ space. Five different

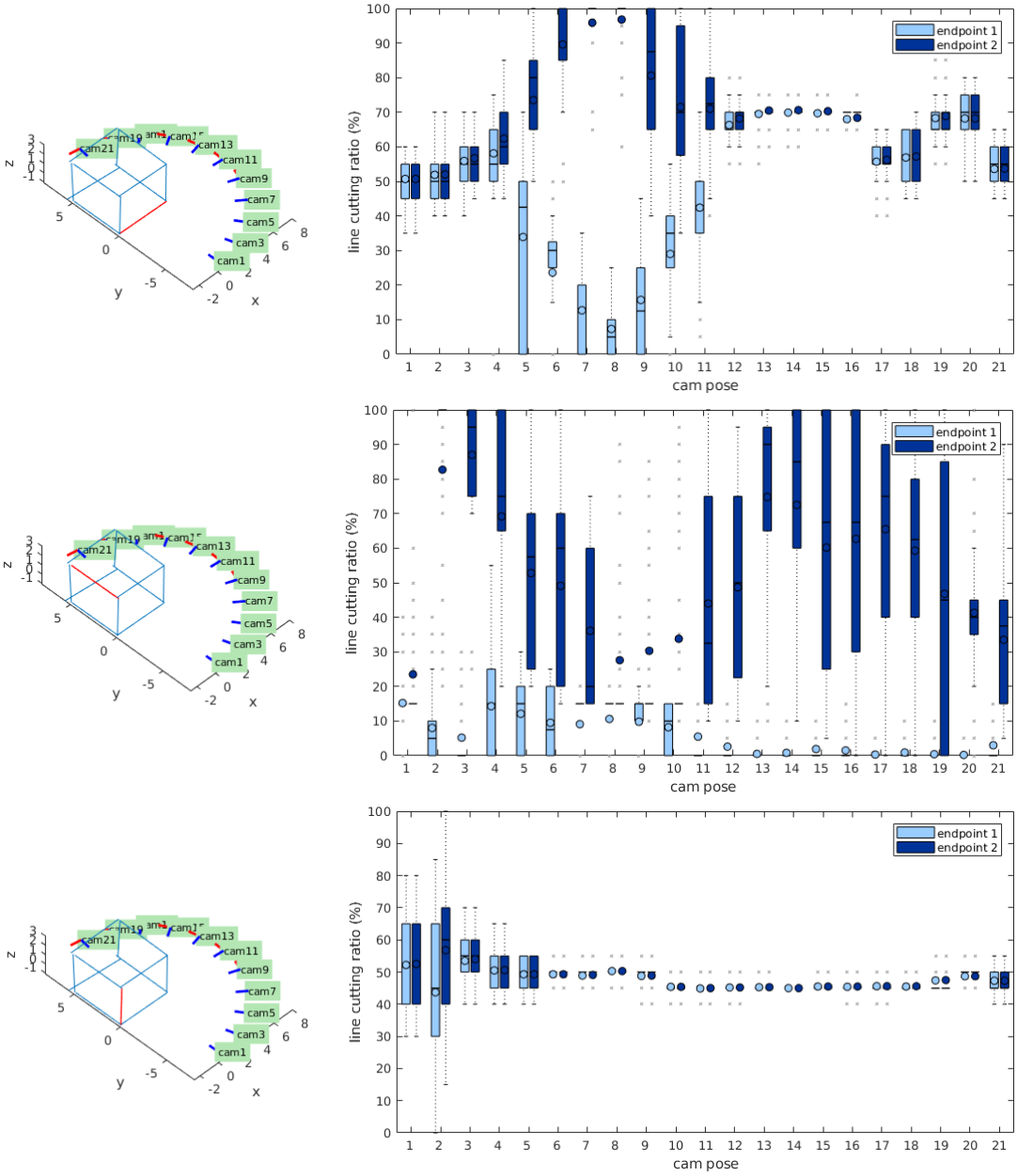


Figure 4.4: Line cutting behavior under different camera poses. A pair $\langle 0, 100 \rangle$ indicates full line selection. Identical ratio pair, e.g. $\langle 45, 45 \rangle$ indicates cutting to a point. At each camera pose, the 3D line in red is cut using the good line cutting objective 4.7. The resulting line cutting ratios are summarized as boxplots to the right side. According to row 1 and 2, line cutting happens when the 3D line is orthogonal or parallel to the camera frame, where the constraint of corresponding line degenerates. In row 3, meanwhile, a consistent cutting outcome of nearly $\langle 50, 50 \rangle$ appears. The outcome is sensible regarding the motion profile (rotation about an axis parallel to the blue dashed line). The line cutting strategy adapts to the information and uncertainty of the tracked lines based on the relative geometry.

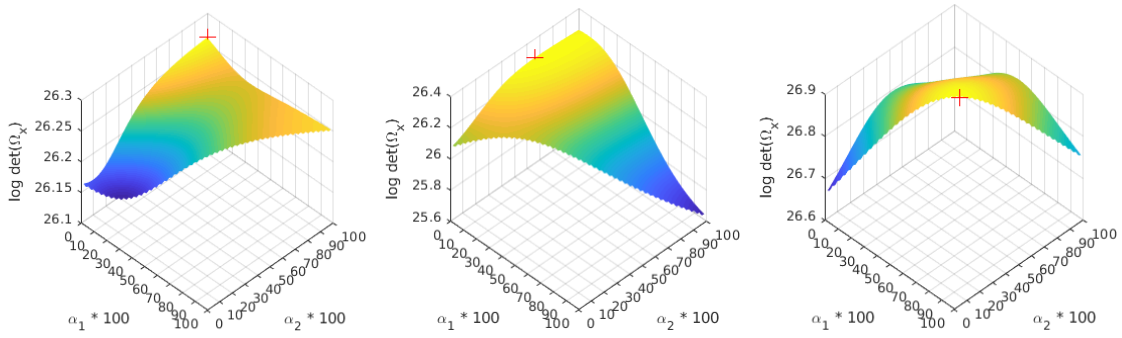


Figure 4.5: Example surfaces of $\log\det(\boldsymbol{\Omega}_{\boldsymbol{x}})$ in single line cutting set-up and *HPL* parameterization. The global maximum of $\log\det(\boldsymbol{\Omega}_{\boldsymbol{x}})$ is marked with red cross.

sizes (6, 10, 15, 20 and 30) of 3D line set are tested. Under both *HPL* and *IDL* parametrization, we compare the best pair from the 3 gradient ascends with the brute-force result. The differences of line cutting ratios are smaller than 0.01 for over 99% of the cases. Therefore, single line cutting problem can be solved effectively using the outcomes from a combination of three gradient ascents.

4.5.2 Joint Line Cutting

Now extend the single line cutting to the complete problem of **joint line cutting**: how to find the line cutting ratios for all n 3D lines, so that the $\log\det$ of pose information matrix generated from n line matchings is maximized?

Naturally, the joint line cutting objective (4.7) can be approached with nonlinear optimizers, e.g. interior-point [111], active-set [112]. Meanwhile, an alternative approach is simple greedy heuristic: instead of optimizing the joint problem (or a smaller subproblem), simply searching for the local maximum for each 3D line as single line cutting problem, and iterating though all n lines. As demonstrated previously, single line cutting can be effectively solved with a combination of 3 gradient ascends. Besides, the 3 independent gradients ascends can execute in parallel. Compared with nonlinear joint optimization that typically requires $\mathcal{O}(\epsilon^{-c})$ iterations of the full problem (c is some constant), the greedy approach has a much well-bounded computation complexity. It takes n iterations to complete,

while at each iteration the single line cutting is solved in $\mathcal{O}(m)$ (m the maximum number of steps in gradient ascend). The efficiency of joint line cutting is crucial, since minimum overhead (e.g. milliseconds) shall be introduced to the real time pose tracking of targeted line-assisted VSLAM applications.

The greedy algorithm for efficient joint line cutting is described in Alg 4. The component of pose information matrix from a full-length line L_i is denoted by $\boldsymbol{\Omega_x}^i(0,1)$, while a line cut from $\langle \alpha_1, \alpha_2 \rangle$ is denoted by $\boldsymbol{\Omega_x}^i(\alpha_1, \alpha_2)$. With the line-LSQ simulation platform, the effectiveness of greedy joint line cutting is demonstrated with 100-run repeated test. The Matlab implementations of interior-point [111], as well as three variants of active-set [112], are chosen to compare against the greedy algorithm. The results are presented as boxplots in Fig 4.6. Under both 3D line parameterizations (*HPL* and *IDL*), greedy algorithm provides the largest increase of $\log\det(\boldsymbol{\Omega_x})$ (on average and in the worst case).

Algorithm 4: Efficient greedy algorithm for joint line cutting.

Data: 3D line set $\{\mathbf{L}(i)\}_n$, 2D measurement set $\{\mathbf{l}(i)\}_n$
Result: $\{\langle \alpha_1(i), \alpha_2(i) \rangle\}_n$

$\Omega_x = \sum \boldsymbol{\Omega_x}^i(0,1)$;
for $i = 1 : n$ **do**
  $\boldsymbol{\Omega_x^r} = \boldsymbol{\Omega_x} - \boldsymbol{\Omega_x^i}(0,1)$;
  $\langle \alpha_1(i), \alpha_2(i) \rangle = \arg\max \log\det(\boldsymbol{\Omega_x^i}(\alpha_1, \alpha_2) + \boldsymbol{\Omega_x^r})$;
  $\boldsymbol{\Omega_x} = \boldsymbol{\Omega_x^r} + \boldsymbol{\Omega_x^i}(\alpha_1(i), \alpha_2(i))$;

4.6 Experiments

4.6.1 Motion Blur Scenarios

The performance improvement of Good Line Cutting to line-assisted VSLAM is assessed on EuRoC MAV dataset. Instead of running on all 11 sequences, only the 6 fast-motion sequences recorded in a Vicon-equipped room (with high potential to exhibit motion blur) are used for motion blur evaluation. Still, the level of motion blur for the original EuRoC sequence is not severe: the shot of each camera is strictly controlled, and the vehicle is only

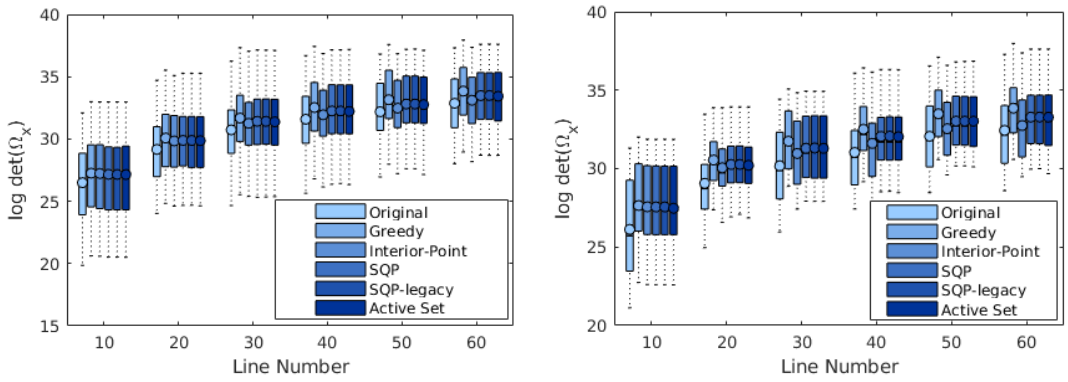


Figure 4.6: Boxplots of joint line cutting with different approaches. **Left:** with *HPL* parametrization. **Right:** with *IDL* parametrization. Boxplots are presented in order: 1) original $\log\det(\boldsymbol{\Omega}_{\boldsymbol{x}})$, 2) after line cutting with greedy approach, 3)-6) after line cutting with nonlinear joint optimizers.

doing fast motion at several moments during the entire sequence. To assess the performance under severe motion blur, we smooth the 6 Vicon sequences with a 5×5 box filter, and include the 6 blurred ones in the evaluation as well.

An open-source stereo line-assisted VSLAM system, PL-SLAM [23], is chosen as the testbed. The Good Line Cutting algorithm is integrated into PL-SLAM in place of the original line-weighting scheme. It takes all feature matchings as input: lines are to be refined with line cutting, while points serve as constant terms in the line cutting objective. After line cutting, all features (points and cut lines) are sent to pose optimization. The loop closing module of PL-SLAM is turned off since the focus of this work is real-time pose tracking.

For comprehensively evaluating the value of line cutting, five variants of the modified PL-SLAM are assessed: 1) point-only SLAM (P), 2) line-only SLAM (L), 3) line-only SLAM with line cutting ($L + Cut$), 4) point and line SLAM (PL), and 5) point and line SLAM with line cutting ($PL + Cut$). Beside the five variants of PL-SLAM, two baselines are evaluated as well: stereo ORB-SLAM [4] (referred as ORB) and stereo SVO [6].

Accuracy of real-time pose tracking is evaluated with three metrics [102] between ground truth track and SLAM estimated track:

1) **Absolute Root-Mean-Square Error (RMSE)**, which captures the absolute error of the

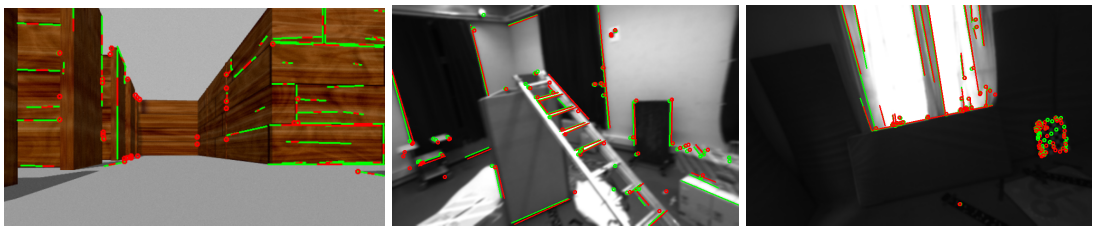


Figure 4.7: Example frames of Line Cutting PL-SLAM running in challenging scenarios: 1) low-texture, 2) motion blur, 3) lighting change. Detected features are in green, while projected are in red. Notice the length of projected line being much shorter than the measurement, after line cutting.

entire trajectory estimated in VSLAM;

2) **Relative Position Error (RPE)**, which captures the average drift of pose tracking in a short period of time;

3) **Relative Orientation Error (ROE)**, which captures the average orientation error of pose tracking with the same estimation pipeline as RPE. Both RPE and ROE are estimated with a fixed time window of 3 seconds.

Due to the fact that most SLAM systems have some level of randomness (e.g. feature extractor, multi-thread), all experiments in the following are repeated with 10 times. For those failed more than 2 times in 10 trials, we ignore the results due to the lack of consistency. For the rest, the average metric values are reported.

The RMSEs on 6 EuRoC sequences with potential motion blur are summarized in the upper hald of Table 4.1. Corresponding RPEs and ROEs are in the upper half of Table 4.2 and Table 4.3, respectively. For each sequence, we compare the line-assisted baseline with the line cutting version, and highlight the better one in bold. Among all 7 methods evaluated here, the one that leads to the lowest error is marked with parentheses.

Compared with the line-assisted baselines (L and PL), the line cutting versions ($L+Cut$ and $PL+Cut$) clearly have better accuracy: both absolute RMSE and relative RPE/ROE are reduced in most rows of Table 4.1, 4.2 and 4.3. Meanwhile, the performance of ORB is not as consistent: when tracking succeed, ORB has the highest accuracy among all 7 methods. However it failed to function reliably on the last 2 sequences. This is not surpris-

Table 4.1: RMSE (m) on EuRoC Sequences with Fast Motion

	Approach						
Sequence	L	$L+Cut$	PL	$PL+Cut$	P	ORB	SVO
V1-01-easy	0.379	**0.205**	0.512	**0.498**	0.988	(0.035)	0.396
V1-02-med	0.525	**0.495**	0.397	**0.345**	0.667	(0.109)	-
V1-03-dif	1.489	**0.890**	1.586	**1.426**	3.748	(0.430)	-
V2-01-easy	0.980	**0.835**	0.639	**0.621**	0.875	(0.047)	0.609
V2-02-med	**1.448**	1.516	0.995	**(0.946)**	-	-	-
V2-03-dif	**3.513**	3.979	**(3.449)**	4.195	-	-	-
V1-01-easy blurred	0.630	**0.346**	0.713	**0.660**	1.051	(0.103)	0.251
V1-02-med blurred	0.525	**(0.375)**	0.474	**0.398**	1.066	0.441	0.565
V1-03-dif blurred	1.623	**1.428**	-	**1.682**	-	-	(0.506)
V2-01-easy blurred	0.934	**0.444**	0.723	**0.557**	0.881	0.307	(0.210)
V2-02-med blurred	1.825	**1.411**	1.612	**1.059**	-	(0.351)	0.473
V2-03-dif blurred	-	-	-	**4.176**	-	-	(1.751)

Table 4.2: RPE (m/s) on EuRoC Sequences with Fast Motion

	Approach						
Sequence	L	$L+Cut$	PL	$PL+Cut$	P	ORB	SVO
V1-01-easy	0.044	**0.043**	0.048	0.048	0.058	(0.041)	0.128
V1-02-med	0.135	**0.059**	0.046	**0.043**	0.072	(0.034)	-
V1-03-dif	0.169	**0.133**	0.164	**0.156**	0.402	(0.108)	-
V2-01-easy	0.100	**0.059**	0.042	**0.030**	0.053	(0.011)	0.109
V2-02-med	0.126	**(0.112)**	0.179	**0.126**	-	-	-
V2-03-dif	0.483	**0.450**	0.431	**(0.364)**	-	-	-
V1-01-easy blurred	0.054	**(0.047)**	0.054	**0.052**	0.062	0.048	0.126
V1-02-med blurred	0.076	**0.068**	0.052	**(0.049)**	0.129	0.178	0.357
V1-03-dif blurred	0.233	**0.206**	-	**(0.148)**	-	-	0.277
V2-01-easy blurred	0.144	**0.054**	**(0.034)**	0.037	0.040	0.049	0.096
V2-02-med blurred	0.166	**0.138**	0.171	**(0.127)**	-	0.162	0.270
V2-03-dif blurred	-	-	-	**0.391**	-	-	(0.289)

ing: when available, point features are known to be more accurate for pose tracking; they are just not as robust as lines under motion blur. Lastly, the direct SVO failed to track on 4 out of 6 sequences, similar to the results reported in [6] (failed on 3 out of 6). It is expected since direct approaches are more sensitive to fast motion and lighting changes (e.g. the 3rd plot in Fig 4.7) than feature-based ones.

The level of motion blur for the original EuRoC sequence is not severe: the shot of each camera is strictly controlled, and the vehicle is only doing fast motion at several moments during the entire sequence. To assess the performance under severe motion blur, we smooth

Table 4.3: ROE (deg/s) on EuRoC Sequences with Fast Motion

	Approach						
Sequence	L	$L+Cut$	PL	$PL+Cut$	P	ORB	SVO
V1-01-easy	0.52	**0.49**	**0.61**	0.63	0.83	(0.43)	4.23
V1-02-med	3.01	**1.52**	0.71	**0.64**	1.71	(0.32)	-
V1-03-dif	4.99	**3.82**	**2.38**	2.85	9.58	(1.96)	-
V2-01-easy	3.58	**2.56**	0.86	**0.77**	0.88	(0.26)	4.49
V2-02-med	**(2.14)**	2.35	4.38	**3.47**	-	-	-
V2-03-dif	12.67	**11.77**	**(10.77)**	12.05	-	-	-
V1-01-easy blurred	0.80	**(0.63)**	0.77	**0.73**	0.95	0.66	4.24
V1-02-med blurred	1.69	**1.62**	0.84	**(0.76)**	3.08	2.63	8.63
V1-03-dif blurred	7.35	**6.65**	-	**(3.17)**	-	-	10.49
V2-01-easy blurred	2.94	**2.08**	**0.99**	1.07	(0.92)	2.25	3.96
V2-02-med blurred	3.47	**2.67**	3.15	**(2.62)**	-	5.48	8.38
V2-03-dif blurred	-	-	-	**10.60**	-	-	(8.58)

the 6 Vicon sequences with a 5×5 box filter, and rerun all 7 VSLAM methods on the blurred ones. Corresponding results are reported in the bottom half of Table 4.1, 4.2 and 4.3.

Under the severe motion blur, point-based approaches (P and ORB) become less accurate than before, while also be prone to loss track. Meanwhile, the line-assisted approaches are more robust to the blur. More importantly, the accuracy of line-assisted approaches are clearly improved with line cutting. Interestingly, direct SVO tracks on all 6 blurred sequences, including 4 sequences that it failed to track originally. The reason is mostly likely due to the blurring applied, which acts to pre-condition the direct objective (original highly non-smooth). The convergence rate of optimizing the direct objective improves and positively impacts the tracking rate.

According to the RMSEs reported in Table 4.1, direct SVO seems having better performance than line-feature VSLAM variants under severe motion blur. However, the relative metrics suggest the opposite: $PL+Cut$ has the lowest RPE and ROE on 3 sequences, while $L+Cut$ has the best relative scores at another sequence. The difference between absolute, global RMSE and relative, local RPE/ROE indicates the proper use case of line features. Instead of incorporating line features to mapping and long-term re-usage, lines are mostly suited as temporal references in short-term pose tracking. Really, line features

should only be used for short and challenging durations that sufficient constraints cannot be obtained with point features only.

Furthermore, we briefly discuss the computation cost of line cutting. Since the baseline PL-SLAM does not maintain covariance matrix for each 3D line, we do so with a simple error model: 1) assume a constant i.i.d. Gaussian at the inverse-depth space of each 3D line endpoint; 2) propagate the endpoint covariance matrix from inverse-depth space of the previous frame to the Euclidean space of current frame. Then we run the greedy line cutting algorithm (Alg 4) with these covariance/information matrices. Most of compute time is spent on the iterative greedy algorithm. When averaged over the EuRoC sequences, the line cutting module takes 3 ms to process 60 lines per frame.

4.6.2 Low-Texture Scenarios

In addition, we evaluate the described approach on low-texture scenario. To the authors' knowledge, no publicly available, low-texture stereo benchmark exists. We synthesized a low-texture stereo sequence with Gazebo for this evaluation. An example frame of the low-texture sequence is provided as the 1st plot in Fig 4.7.

Relative errors are summarized in Table 4.4. After applying line cutting to line-assisted baseline (L and PL), the average relative errors are cut down by almost 40%, as highlighted in bold. The lowest tracking error (i.e., best accuracy) is achieved when combining point and line features, and cutting the lines with the described method ($PL+Cut$). Meanwhile, systems that only utilize point features perform poorly: point-only SLAM (P) has high ROE; ORB-SLAM2 (ORB) failed to track. The direct approach SVO succeeded in tracking the whole low- texture sequence, but has the highest relative errors.

The evaluation results suggest that, line features are valuable for pose tracking in low-texture scenarios. However, simply using the full-length of lines for pose optimization may cause large tracking error. With the described line cutting, the accuracy of line-assisted pose tacking improves.

Table 4.4: Relative Error on Synthetic Low-Texture Sequence

	Approach						
Metric	L	$L+Cut$	PL	$PL+Cut$	P	ORB	SVO
RPE(m/s)	0.246	**0.141**	0.242	**(0.126)**	0.222	-	0.372
ROE(deg/s)	4.78	**3.01**	3.83	**(1.68)**	5.13	-	8.83

4.7 Conclusion

This chapter presents Good Line Cutting, which deals with the uncertain 3D line measurements to be used in line-assisted VSLAM. The goal of Good Line Cutting is to find the (sub-)segment within each uncertain 3D line that contributes the most information towards pose estimation. By only utilizing those informative (sub-)segments, line-based least squares is solved more accurately. We also describe an efficient, greedy algorithm for the joint line cutting problem. With the efficient approximation, line cutting is integrated into a state-of-the-art line-assisted VSLAM system. When evaluated on two target scenarios of line-assisted VSLAM (motion blur; low-texture), accuracy improvements are demonstrated, while robustness is preserved. There are a couple of further directions that can be investigated in the future. First, Good Line Cutting can be extended to infinite parametrizations of 3D lines, such as Plücker coordinates. The combination of point feature selection (i.e., Good Feature Selection) and line cutting (i.e., Good Line Cutting) is worth exploring as well. Last, the computation cost of state-of-the-art line feature extraction algorithm is still limited. Further investigation of active and efficient line extraction is crucial for applicable line-assisted VSLAM.

CHAPTER 5
MAP HASHING: APPEARANCE-ENHANCED COMPACT LOCAL MAP OF FEATURE-BASED VSLAM

5.1 Introduction

In this chapter, we present the work of Map Hashing, which bounds the cardinality of local map with strong appearance prior, therefore improving the long-term performance of VSLAM. Augmentation of the feature matching process of VSLAM systems with a local map matching sub-process aids data association and state optimization [4, 113]. Compared with a global map containing all historical 3D points, the local map includes only the subset of 3D points that are hypothesized to be currently visible. Conducting data association and downstream state optimization on a compact local map is more efficient than for the larger global map. By matching 2D features from the current frame to the local map (which includes 3D points observed at earlier frames), extra long-baseline feature matchings can be extracted and utilized in state optimization; see Figure 5.1 (top-left) depicting a histogram of matched local map points for ORB-SLAM, where the baseline is measured in terms of how long ago the features were seen (as opposed to how far spatially). These long-baseline matchings contribute to the accuracy and robustness of VSLAM. Not surprisingly, VSLAM systems employing a local map [16, 4] tend to be more accurate and robust than systems relying only on frame-to-frame tracking [114, 31, 32].

A compute-economic property to guide the building of the local map with relevant 3D points is *co-visibility*. Co-visibility was introduced for loop closing in VSLAM [115], and later extended to pose tracking [116, 4, 117, 118]. The assumption of co-visibility being: if an earlier keyframe shares many 3D points with a recent keyframe (i.e., co-visible), then all 3D points observed by the earlier keyframe are likely to be seen also. Co-visibility

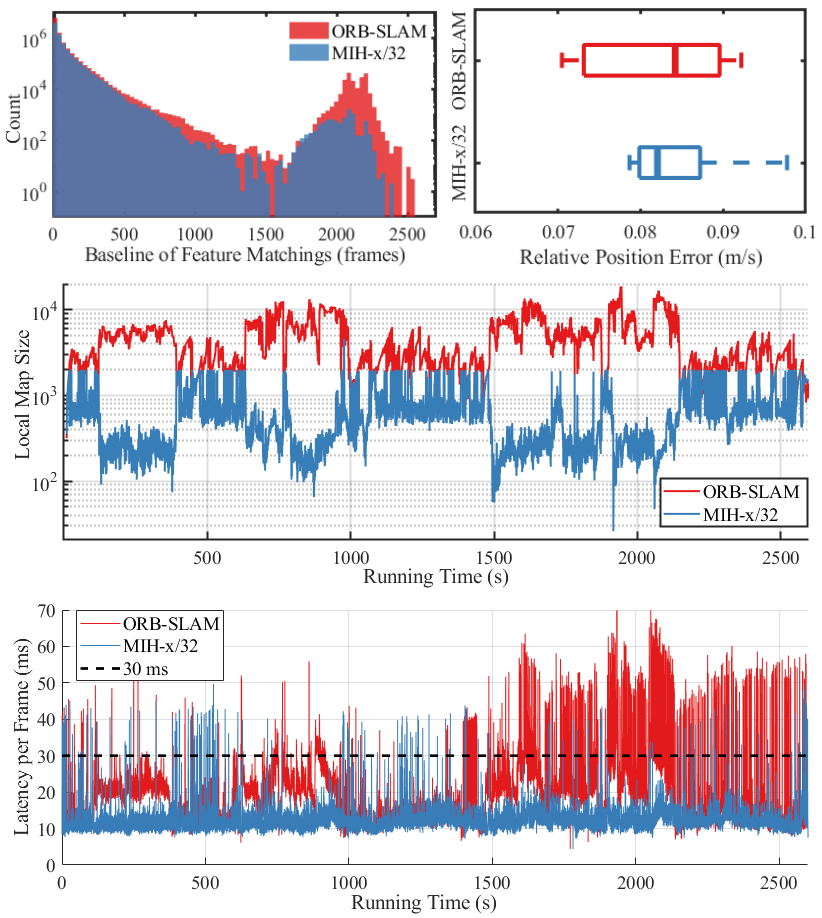


Figure 5.1: Latency reduction of the described Map Hashing algorithm (MIH-x/32), when integrated into a state-of-the-art VSLAM system (ORB-SLAM[4]). **Top-Left:** Histogram of matched features baselines extracted from local map, with and without proposed algorithm **Top-Right:** Accuracy of VSLAM with or without proposed algorithm, measured with RPE (10-sec window). **Middle:** Size of the local map utilized in VSLAM, with or without proposed algorithm. **Bottom:** Latency profile of real-time pose tracking on the long-term *NewCollege* sequence.

information is cheap to obtain as the by-product of earlier data association calculations, therefore it can be considered to be an efficient heuristic for local map building. However, co-visibility only utilizes the relatively-weak temporal prior (i.e., seen before, likely to be seen now). A local map generated with co-visibility could easily grow without bound, and introduce significant latency to VSLAM thereafter. Figure 5.1 (middle row) includes a plot of the ORB-SLAM local map versus time, where it is seen to occasionally grow to be one to two orders of magnitude more than the number of tracked features per frame (typically

on the order of 10^2 to 10^3).

In this work, we propose to enhance the co-visibility local map building step with a strong appearance prior, which will lead to a compact yet relevant local map, as indicated in Figure 5.1 (middle row) where the proposed local map queried is bounded in size and can be up to an order of magnitude lower than ORB-SLAM. The idea is straightforward: only those 3D points that are visually similar to currently extracted features are potentially useful in data association (and state optimization thereafter). To utilize the appearance prior efficiently, we propose to index descriptors of historical 3D points with Multi-Index Hashing (MIH) [119]. By querying historical 3D points from a series of hash tables, we can collect the subset of 3D points that are similar to current measurements in appearance/descriptor space. The visually-similar 3D points are then verified with co-visibility, and put together as the local map for the costly computations, e.g. data association and state optimization.

Furthermore, an online table selection algorithm is developed to choose a subset of hash tables that cover the most relevant 3D points. By only querying 3D points from the subset, the overhead on hash table queries is reduced, while the quality of the local map is preserved, as indicated by comparable RPE in Fig 5.1 (top-right). The table selection process is rooted in the submodular property with regards to the table selection metric (e.g. information gain of feature matchings obtained from each table). Because of the submodular property of table selection metric, a greedy algorithm can achieve near-optimal table selection outcomes with good efficiency properties. Figure 5.1 (bottom row) shows better bounding of the SLAM latency per frame, with fewer outliers, relative to a 30ms threshold.

The described Map Hashing algorithm is generic; it can be easily extended to other visual(-inertial) SLAM systems utilizing a local map, i.e., [16, 120].

5.2 Background

Two closely-related fields are reviewed: Vision-based Localization (VBL) and Visual SLAM (VSLAM). Differences between existing works and the described work are discussed.

VBL aims to retrieve the 6DoF pose of a visual query (image or video) within a huge, pre-built spatial representation, e.g. a 3D point map. One key component of VBL is to index the spatial representation for efficient queries. Co-visibility was introduced to feature-based VBL [121, 122] as a cue to prioritize feature matching efforts. Researchers also proposed alternative indexing methods based on appearance/feature descriptors [123, 124]. Real-valued feature descriptors such as SIFT[125] and SURF [126] are typically indexed offline using a kd-tree. Appearance-based indexing are proven to yield more accurate and robust query results, while co-visibility is more computationally-efficient. Combining both cues was first explored in [127], and further refined in [128, 129]. The work [129] replaced the kd-tree data structure with a faster and more flexible indexing method, inverted multi-index. The appearance-based query results are then filtered with co-visibility. Such a combination scheme is efficient: the VBL system runs real-time on mobile device. Nevertheless, training the inverted index is still an offline process requiring a known 3D map.

Binary feature descriptors such as BRISK [13] and ORB [14] are used in VBL since they are more efficient to extract than real-valued ones. Conventional indexing data structures like kd-trees are better suited to real-valued descriptors, rather than binary ones, motivating the exploration of alternative indexing methods. For example, randomized trees were proposed to index binary descriptors [130], which were trained offline from the pre-built 3D map. Hashing has been proven to be a good indexing solution [131, 132] in binary-descriptor VBL. Coarse-to-fine searching schemes are commonly applied in these VBL systems, where an initial hashing query provides the coarse results that are later refined by a linear scan.

Apart from compatibility with binary descriptors, two other properties of hashing make

it particularly attractive to online and incremental pose estimation problem, e.g. VSLAM. First, hashing index can be updated efficiently for online processes. It is then possible to generate a more compact and relevant index by updating hash tables, e.g., according to changes in the map and the visibility constraints. Second, hashing relaxes the requirement for database pre-training (or prior offline database generation), therefore enabling VSLAM systems to operate in general and unknown environments. Hashing has been applied to modules of VSLAM where real-time performance is not required. In [133], binary descriptors are indexed with Locality Sensitive Hashing (LSH) [134]. Good relocalization performance in a VSLAM system is demonstrated thereafter. Multi-Index Hashing (MIH), which is firstly developed in data query [119], has been introduced to the loop closing module of VSLAM [135].

The described Map Hashing method is based on MIH, but with a key enhancement: an online table selection algorithm is developed to reduce the number of hashing queries, therefore enabling MIH to be used in VSLAM modules with real-time requirements, e.g. pose tracking. The local map queried with appearance/feature descriptors is further tailored with a co-visibility check. The final local map is more compact than the ones generated with either co-visibility or appearance only. Running data association and state optimization on the size-reduced local map is more efficient and leads to significant latency reductions in VSLAM based on a more efficient local map data association step. Furthermore, the quality of the local map (e.g. amount of long-baseline feature matchings) is preserved in the compact local map. Therefore, the performance of VSLAM is preserved. Preliminary quantification of these benefits can be seen in Figure 5.1 for a single sequence.

5.3 Local Map Building with Multi-Index Hashing

A diagram of the proposed local map building method is illustrated in Fig 5.2. The modules of our method are highlighted with shaded boxes, while those in a conventional VSLAM pipeline have clear boxes. This section describes the query and insertion stage of MIH. The

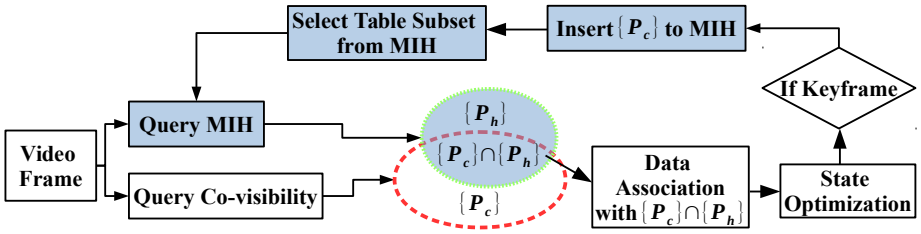


Figure 5.2: Framework of the proposed local map building method. The local map built with co-visibility is the red dashed ellipse, while the one built by querying MIH is the green dashed ellipse. Their intersection defines the local map for downstream processing, i.e., data association and state optimization.

hash table selection algorithm will be introduced in the next section.

5.3.1 Query MIH

Assume that a frame with m binary descriptors extracted is provided and that the MIH contains t hash tables. Each binary descriptor will trigger a MIH query. In a MIH query, the b-bit binary query descriptor is first separated into t disjoint contiguous substrings, as illustrated in Fig 5.3. Each substring gets queried with the corresponding hash table for an exact match. Query results from all t hash tables are put together as the final query result. Repeating the MIH query for all binary descriptors from the input frame, aggregate the 3D point set $\{P_h\}$ that satisfy the appearance prior. The intersection of appearance-based point set $\{P_h\}$ and the 3D point set $\{P_c\}$ collected with conventional co-visibility is the final local map, i.e., $\{P_h\} \cap \{P_c\}$.

5.3.2 Insert to MIH

Updating MIH according to changes in the map and visibility constraints is essential for efficient local map building. As a trade-off between update frequency and computation cost, MIH updates are triggered only for keyframes sent to the mapping thread. Updating MIH in the mapping thread avoids introducing overhead during real-time pose tracking.

For each keyframe, the co-visible 3D points $\{P_c\}$ are inserted into the MIH. Similar to the query process, the b-bit binary descriptor of each 3D point in $\{P_c\}$ is separated into

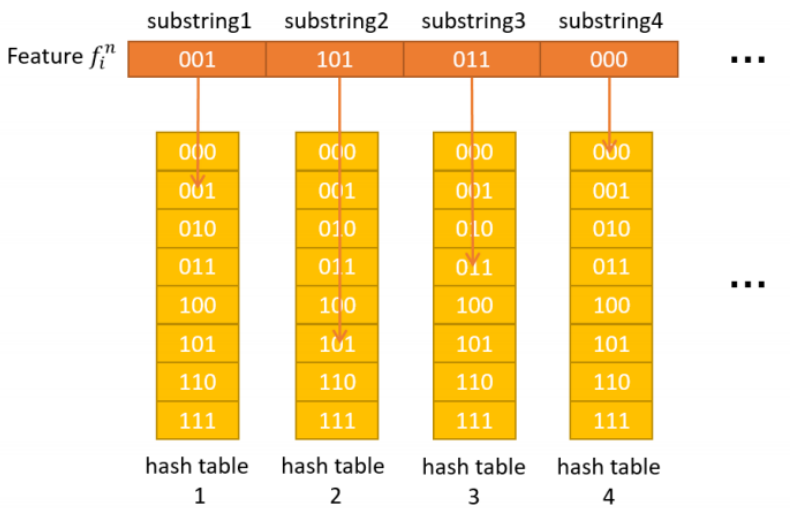


Figure 5.3: An illustration on Multi-Index Hashing (MIH) [119].

t disjoint contiguous substrings, each of which is of length $\lfloor b/t \rfloor$. Each substring is then inserted into a corresponding hash table. For 3D points already in the hash tables, their entries will shift to the front of the bucket, making them more likely to be queried in the future.

5.3.3 Choice of Hash Table Number

The quantity of hash tables t has strong impact on the performance-efficiency of MIH-based local map indexing. Recall the example of a frame with m features extracted. Each feature will trigger a MIH query consisting of t queries to hash tables. Therefore, the MIH-based local map building has a time complexity of $\mathcal{O}(mt)$, i.e., linear in t. Meanwhile, the space complexity of MIH is $\mathcal{O}(tN2^{\lfloor b/t \rfloor})$, where N is the bucket size in each hash table. The space complexity decreases exponentially with table number t. Therefore, only a certain range of t works in practical applications due to time and space complexity limits.

Apart from time and space complexity, the robustness of MIH against perturbations in binary descriptors is largely decided by hash table number t. Assuming ϵ bits of the query descriptor are perturbed under a uniform distribution, the recall probability (i.e., probability that the query succeeds with a perturbed string) is connected to hash table number t as per

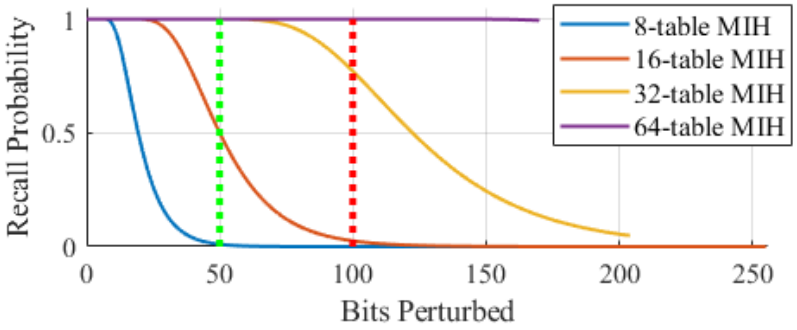


Figure 5.4: Simulation results evaluating the recall probability of hashing (the higher the better) vs. the number of bits perturbed for different numbers of tables in the MIH. For 256-bit descriptors, MIH with 32 tables is preferred: it remains high recall even under significant perturbation (50-100 bits).

[135]:

$$P_{recall}(t, \epsilon) = 1 - t!\ \Theta(\epsilon, t)/t^{\epsilon}, \tag{5.1}$$

where $\Theta(\epsilon, t)$ is the Stirling partition number [136].

When working with 256-bit binary descriptors such as ORB, the relationship described in Eq 5.1 is illustrated in Fig 5.4. The green and red dashed lines indicate example thresholds of bit-wise perturbations in typical SLAM applications. At least 32 tables are needed for high recall probability within the example perturbation levels (vertical dashed lines). Using 64 tables is also possible, but with the drawback of higher overhead due to the linear-growth in time complexity. In the described local map indexing method, 32 hash tables are maintained; each table covers an 8-bit descriptor substring.

5.3.4 Choice of Bucket Size

Another parameter affecting the performance-efficiency of MIH-based local map building is the bucket size N of each hash table. A bucket in MIH is implemented as ring buffer, where only the N most recent 3D points are stored. For the purpose of long-baseline feature matching, it is necessary to keep the entries of 3D points observed earlier in time within the bucket. However, an over-sized bucket will store entries of 3D points that are no longer visible nor relevant. As a consequence, the resulting local map will be less compact and

relevant, introducing overhead to data association. In what follows, the bucket size N is set to 10 based on a parameter sweep.

5.4 Overhead Reduction with Hash Table Selection

For a frame with m features extracted and a 32-table MIH, the number of hash table queries in local map building is $\mathcal{O}(32m)$. While querying all 32 hash tables provides robustness against severe perturbation, querying a subset of hash tables is more efficient when the bit-wise perturbation level is low or medium. We propose an online table selection algorithm to identify the minimum subset of hash tables to be queried, which further improve the compactness of local map without performance degeneration.

5.4.1 Objective Formulation

To begin, the metric used for table selection is introduced. Assume F is the full set of *true* feature matchings between current frame and the full local map built with all 32 hash tables. For each hash table T_i, the *true* feature matchings that can be queried from it form a subset $F_i \subset F$, where $\bigcup_{i=1}^{32} F_i = F$. For each hash table T_i, the contribution towards current state optimization can be assessed with the information matrix of subset F_i.

Following the previous least squares definition of VSLAM pose tracking 2.1, we can derive the information matrix of camera pose $\boldsymbol{\Omega_x}$ as

$$\boldsymbol{\Omega_x} = \sum \mathbf{H}(i)^T \boldsymbol{\Omega_r}(i) \mathbf{H}(i) = \sum \boldsymbol{\Omega_x}(i), \tag{5.2}$$

where $\mathbf{H}(i)$ and $\boldsymbol{\Omega_r}(i)$ are the measurement Jacobian and residual information matrix of corresponding *true* matched features. Denote by $\boldsymbol{\Omega_x}(i)$ the pose information matrix derived from a single feature match i.

As introduced for feature subset selection [74, 62], the *logDet* is especially suited for quantifying the contribution of matched features to VSLAM. Therefore, the value of a hash

table T_i towards current state optimization can be measured with

$$\log\det(\sum_{i\in F_i}\mathbf{\Omega}_{\boldsymbol{x}}(i)). \tag{5.3}$$

There is a certain level of overlap between the *true* matched feature subsets for each hash table. In an ideal scenario without any perturbation to feature descriptor, the full set of *true* feature matchings can be retrieved from any one of the 32 hash tables, i.e., 100% overlapping between subsets, $\forall i,j\ F_i = F_j = F$. In practice perturbations reduce the subset overlap percentage to less than 100%, and each hash table covers a subset of *true* feature matchings F. Therefore, selecting a subset of hash table is equivalent to a problem of maximum coverage, with the objective formulated as:

$$\max_{S\subseteq\{1,2,\ldots,32\},|S|\leq k}\log\det(\sum_{i\in\{\bigcup_{h\in S}F_h\}}\mathbf{\Omega}_{\boldsymbol{x}}(i)), \tag{5.4}$$

where k is the cardinality constraint.

5.4.2 Greedy Table Selection

The maximum coverage problem is studied in the field of computational theory, where it is known to have submodular properties. Recall the proposition 1, that a monotone and submodular problem can be approximated with greedy method with the approximation guarantee of $(1-1/e)$. Furthermore, *logDet* meets both requirements [73]. Solutions to the subset selection problem, and the equivalent hash table selection problem, can be approximated using greedy algorithms. More importantly, a greedy algorithm is guaranteed to be near-optimal, with approximation ratio of $1-1/\epsilon$. Based on this outcome, we present a greedy, online hash table selection algorithm in Alg 5. Two control parameters are fixed after parameter sweeping: cardinality constraint $k=8$, target contribution $d_{thres}=80.0$.

The above discussion assumes that the *true* feature matchings are known before performing hash table selection. In practice, however, we do not know the *true* matchings

Algorithm 5: Online hash table selection algorithm.

Data: feature matching subset from each hash table $\{F_1,\ F_2,\ \ldots,\ F_{32}\}$, cardinality constraint k, target contribution d_{thres}

Result: indices of hash tables selected S

foreach *feature matching* $j \in \bigcup_{i=1}^{32} F_i$ **do**
    collect pose information matrix $\boldsymbol{\Omega}_{\boldsymbol{x}}(j)$;
$S \leftarrow \emptyset$, $d_{acc} = 0$;
while $|S| < k \wedge d_{acc} < d_{thres}$ **do**
    foreach $i \notin S$ **do**
        $d(i) = \log\det(\sum_{i \in \{\bigcup_{h \in S \cup F_i} F_h\}} \boldsymbol{\Omega}_{\boldsymbol{x}}(i))$
    $j \leftarrow \arg\max_i d(i)$;
    $d_{acc} = d(j)$;
    $S \leftarrow S \cup j$;
return S.

beforehand. To mitigate that, we assume that the content of hash tables is a slowly-varying function of time, and execute the hash table subset selection algorithm on keyframes rather than all regular frames. After finishing map-to-frame feature matching for a keyframe, the selection of hash tables gets updated using Alg 5. The updated subset of hash tables is utilized for the incoming regular frames, till another keyframe is taken. The above implementation enables efficient query and construction of the local map, without having noticeable performance loss.

5.5 Experiments

This section evaluates the performance-efficiency trade off of the Map Hashing algorithm on a state-of-the-art VSLAM system, ORB-SLAM [4]. Applying the described algorithm to the real-time tracking thread of ORB-SLAM reduces pose tracking latency. Meanwhile, tracking accuracy is either improved (on short sequences) or remains near the same level as canonical ORB-SLAM (on long sequence), and the robustness is preserved (i.e., avoid tracking failure).

5.5.1 Long-Term VSLAM in Unknown Environment

The latency reduction and strong performance of the Map Hashing algorithm is demonstrated by comparing with other state-of-the-art VSLAM systems on a long-term VSLAM benchmark, *NewCollege* [137]. *NewCollege* contains a 43-minutes stereo sequence collected with a robot traversing a campus and adjacent parks. There are multiple loops/revisits within the sequence. The sequence is well-suited for evaluating the long-term performance and efficiency of VSLAM system (with loop closure). Due to the lack of 6DoF pose ground truth, offline Bundle Adjustment is executed with stereo video, and the jointly optimized camera poses are taken as the ground truth. We only evaluate monocular VSLAM (e.g. with left camera) against the ground truth in this experiment.

The Relative Position Error (RPE) [102, 138] is chosen to evaluate the long-term performance of VSLAM on *NewCollege*. Compared with absolute RMSE, RPE is less sensitive to the inevitable scale drift of monocular VSLAM. Therefore, it is better for evaluating monocular systems on long-term sequences.

The efficiency of VSLAM is evaluated with the latency of real-time pose tracking per frame, which has been described in the experiment section of Chapter 3. Latency of mapping and loop closing is less of a concern in this work due to the relaxed time constraints of those processes. The same configuration as Chapter 3 is applied: 10-run repeat; discarding any track failure; running on Intel i7-7700K quadcore 4.20GHz CPU (passmark score of 2583 per thread).

To demonstrate the benefit of online hash table selection (Alg 5), we performed additional 10-run repeats of MIH-based local map building with a predefined set of fixed hash table subsets, ranging 1 table (*MIH-1/32*) to all 32 tables (*MIH-32/32*). Results of these tests are compared to MIH-based local map building with online hash table selection, i.e., *MIH-x/32* (x = 10).

The latency profiles of different hash table subsets are presented in Fig 5.5. *MIH-x/32* has the lowest latency for data association, when compared to other predefined hash table

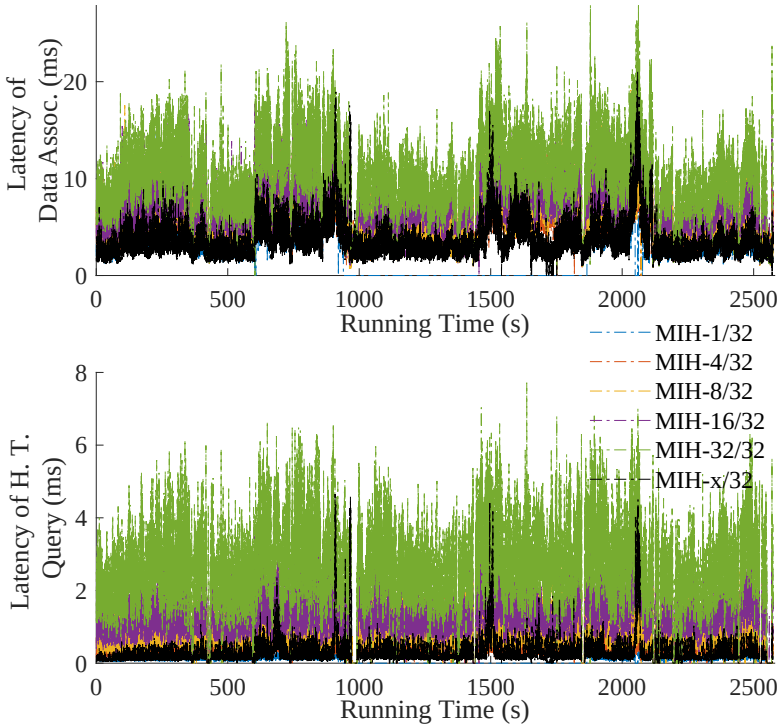


Figure 5.5: **Top:** Latency of data association from 1 run on *NewCollege*. **Bottom:** Latency of hash table query (part of data association) from 1 run on *NewCollege*. The first 5 profiles have predefined hash table subsets, e.g. first 1, first 4, etc. The last profile employs online hash table subset selection.

subsets. The latency of hash table queries is also lower with online hash table selection. Performance evaluation of the methods collected the average RPE (with a 10-sec window), and also logged the average latency of each module in the real-time pose tracking process. Performance (RPE) and efficiency (latency) outcomes are summarized in Fig 5.6. *MIH-x/32* has the lowest latency for pose tracking while preserving the performance of VSLAM relative to the fixed table subsets.

Two state-of-the-art VSLAM systems are chosen as baselines: DSO with loop closure (*LDSO*) [139] and ORB-SLAM (*ORB*) [4]. In addition to the proposed *MIH-x/32*, we integrate two reference methods into ORB-SLAM that enhance co-visibility local map building with simple heuristics. One heuristic is random sampling, i.e., *Rnd*. The other heuristic prioritizes map points with a long track history, denoted as *Long*, since feature points tracked for a long time are more likely to be mapped accurately.

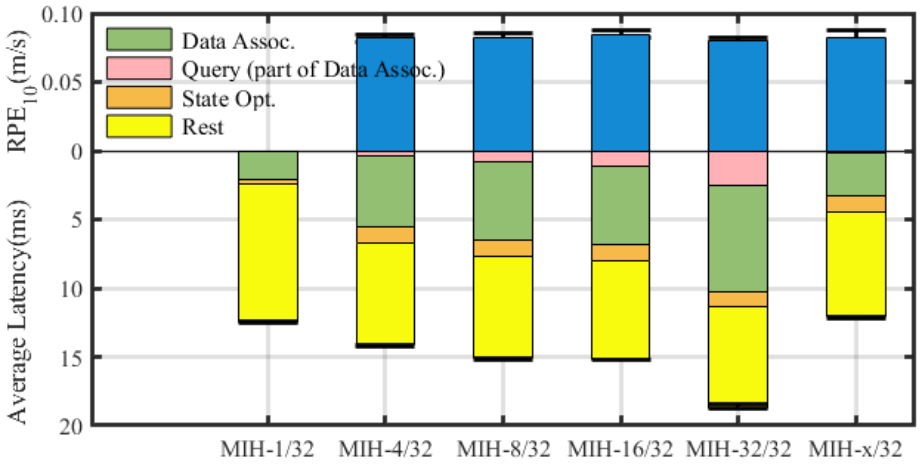


Figure 5.6: RPE and latency for different hash table subsets averaged over 10 runs on *NewCollege*. The first 5 columns are the fixed hash table subset methods, e.g. first 1, first 4, etc. The last column employs online selection. No RPE is reported for the single hash table (*MIH-1/32*) since track loss frequently occurred.

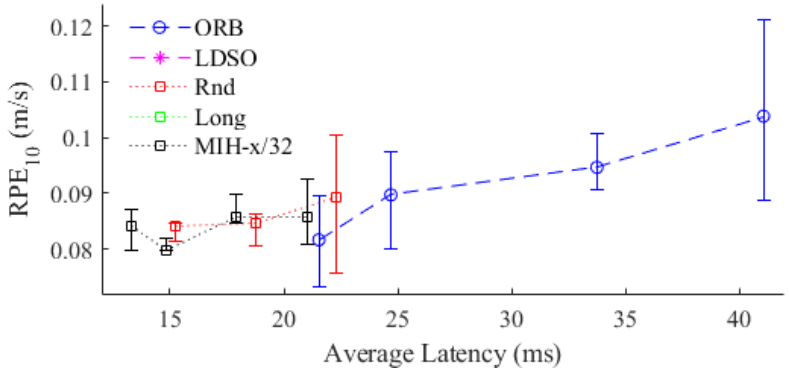


Figure 5.7: Latency vs. accuracy on NewCollege monocular sequence. System evaluation involved a sweep of features per frame: 800, 1000, 1500, 2000.

To capture the performance-efficiency trade off of VSLAM systems, we adjust the number of features/patches extracted per frame. All 5 VSLAM systems are configured to run 10-repeats on *NewCollege*, with feature/patch quantities ranging from 800 to 2000. The RPE under 10-sec window versus the average latency per frame is depicted in Fig 5.7. Relative to ORB-SLAM, the proposed *MIH-x/32* leads to latency reduction for all configurations of feature number. *Rnd* also leads to latency reduction, but not as much as *MIH-x/32*. The *Rnd* case with 800 features leads to track loss, so it is not plotted. Both *LDSO* and *Long* failed to track the full *New College* sequence. The accuracy of *MIH-x/32* is compara-

Table 5.1: RPE (m/s) on NewCollege Sequence

Seq.	*LDSO*	*ORB*	*MIH-x/32*
RPE_3	-	**0.11** (2e-2)	0.12 **(8e-3)**
RPE_{10}	-	**0.08** (8e-3)	**0.08 (6e-3)**
RPE_{30}	-	**0.09 (5e-3)**	0.10 (1e-2)

Table 5.2: Latency (ms) on NewCollege Sequence

Seq.	*LDSO*	*ORB*	*MIH-x/32*
Q_1	-	13.2	**10.4**
Avg.	-	18.3	**12.2**
Q_3	-	21.5	**13.3**

ble to the best performing *ORB* realizations, but with a lower deviation as indicated by the shorter error bars. Lastly, we report the accuracy of the monocular VSLAM systems under the configuration of 800 features per frame in Table 5.1. Corresponding latency averaged per frame is reported in Table 5.2. Three RPE metrics are computed using different sliding windows: 3-sec, 10-sec and 30-sec. In addition to the average RPE over 10-run repeat, the standard deviation (STD) of the RPE is also reported in each cell of Table 5.1. The two heuristics *Rnd* and *Long* are excluded since they both failed to track on the full sequence. The best numbers (lowest average/STD of RPE, lowest latency) are highlighted with bold. The accuracy of *MIH-x/32* remains at similar levels as *ORB* (equal or around 10%), as evaluated on all 3 RPE metrics. More importantly, the latency of described method is lower and more consistent than baseline *ORB*. It is 21%, 33%, and 40% lower for the first quartile, average, and third quartile values.

5.5.2 Long-Term VSLAM in Pre-Mapped Environment

The Map Hashing algorithm is especially suited when huge amount of map points are available. In the presence of pre-built map from previous runs, VSLAM with compact local map is able to track camera pose with low-latency and drift-free.

To demonstrate the applicability of Map Hashing algorithm in map re-using scenarios,

Table 5.3: Sequences Collected in TSRB Office Area

Seq.	Collect Date	Duration (sec)	Frame (stereo pair)
s1	2019-02-05-18-58-08	459	13,798
s2	2019-02-06-18-29-27	377	11,325
s3	2019-02-08-17-16-08	527	15,838
s4	2019-05-03-17-48-01	529	15,877
s5	2019-05-07-19-46-48	1,169	35,086

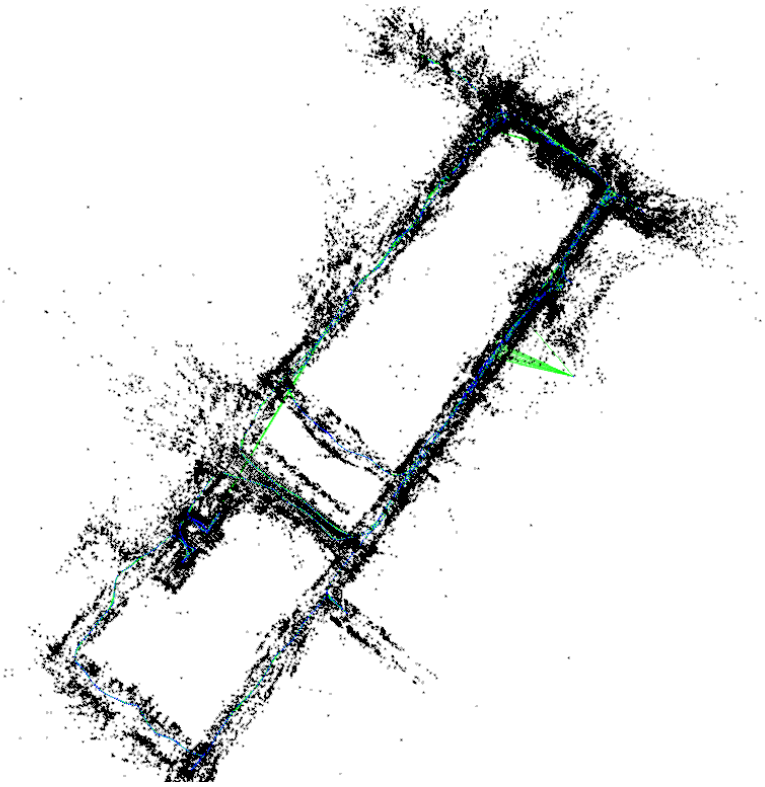


Figure 5.8: 3D view of the pre-build map.

we collect five runs of stereo sequences in an office area. Details of the collected sequences are presented in Table 5.3. The prior map is collected by running VSLAM on the first two sequences, i.e., *s1* and *s2*. The loop closing module, as well as the global pose graph optimization, are activated to improve the global consistency of generated map. The final map is illustrated in Fig 5.8, which contains 1,871 keyframes and 56,150 map points.

The prior map from first two sequences are loaded and utilized as prior when running VSLAMs on the rest sequences: *s3*, *s4* and *s5*. Though the collection date of last two sequences are quite different from the map, majority of the pre-mapped features are suc-

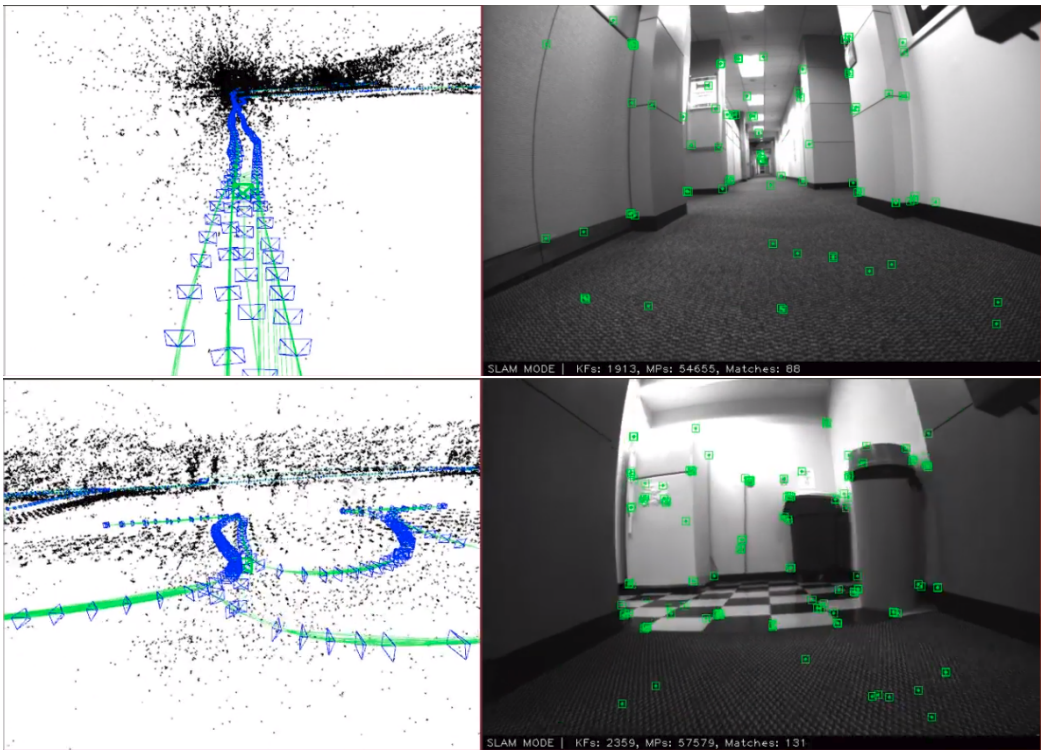


Figure 5.9: Screen shots of *MIH-x/32 + GF* running on sequence *s5*. **Left**: map view. **Right**: image view.

cessfully matched during the evaluation. Three stereo VSLAM methods that support map re-using are evaluated here: canonical ORB-SLAM (*ORB*), the proposed *MIH-x/32*, the combination method *MIH-x/32 + GF* that builds local map with *MIH-x/32* and perform active good feature matching [63]. Some screen shots of *MIH-x/32 + GF* running on sequence *s5* are presented in Fig 5.9. All evaluations are conducted on a desktop equipped with an Intel i7 quadcore 4.20GHz CPU (passmark score of 2583 per thread). Due to the lack of ground truth trajectory, we focus on latency reduction in this evaluation.

The pose tracking latency of three evaluated VSLAM methods are presented in Fig 5.10. Profiles of example runs on three testing sequences are illustrated at the left column; the summarized latency distribution over 3-repeats are illustrated at the right column. Compared with canonical *ORB* that builds local map using co-visibility only, pose tracking latency of proposed *MIH-x/32* is significantly better bounded. The average latency of *MIH-x/32* is around 40ms, which is half the latency of *ORB*. The maximum latency of *MIH-x/32*

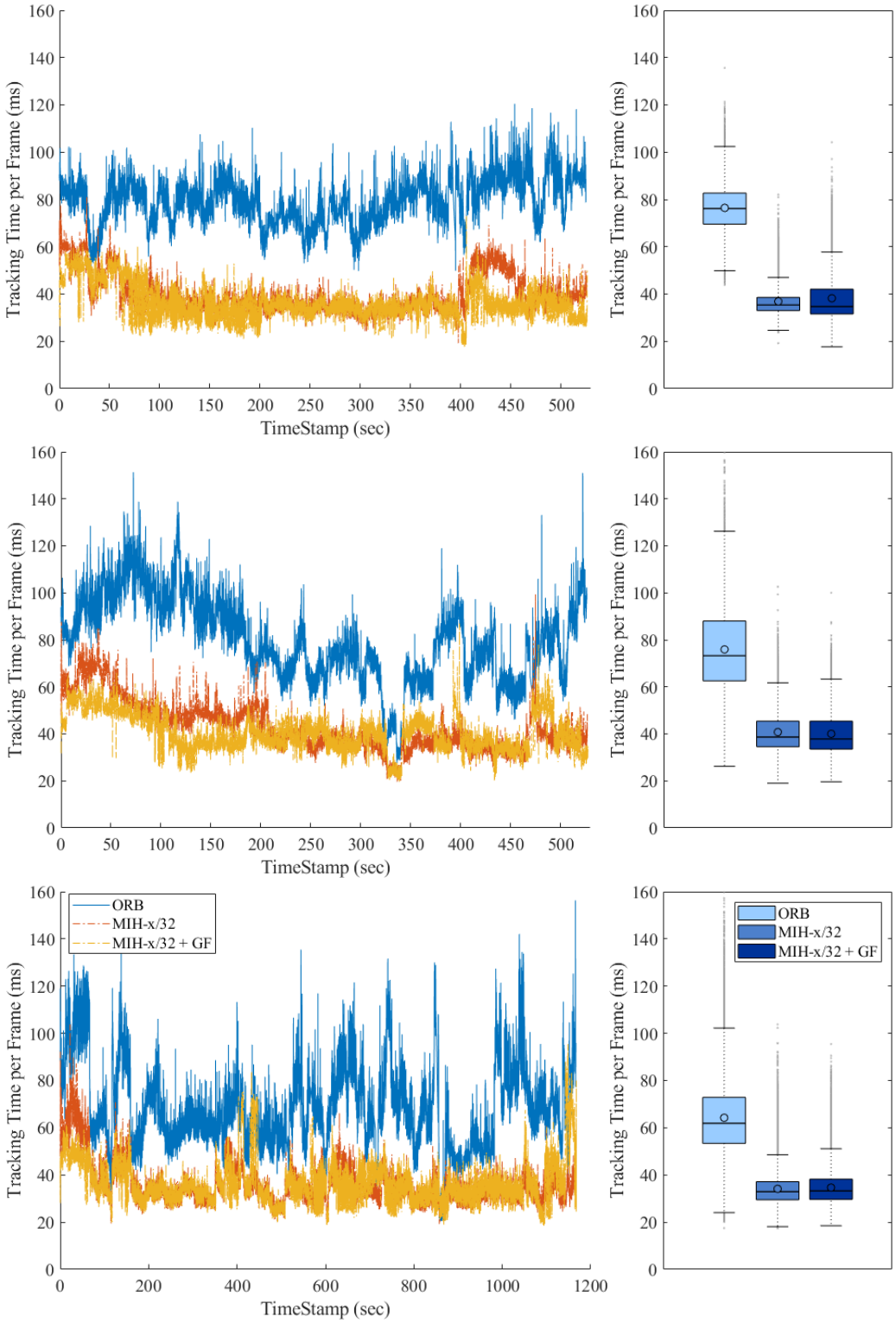


Figure 5.10: Example profiles (**left**) and boxplots (**right**) of pose tracking latency for three VSLAM methods that support map re-using. **Top**: pose tracking latency on sequence *s3*. **Middle**: pose tracking latency on *s4*. **Bottom**: pose tracking latency on *s5*.

is also much less than *ORB*. The combination of *MIH-x/32* and *GF* has slightly improved latency profile than with *MIH-x/32* alone: on *s3* and *s4*, *MIH-x/32 + GF* has fewer peaks than *MIH-x/32* according to latency profiles. Nevertheless, the majority of latency reduction is because of the compact local map constructed with Map Hashing.

5.5.3 Short-Term VO

We also evaluate the Map Hashing algorithm on short-term VO task. The *EuRoC* [25] is used in this evaluation, which contains 11 stereo-inertial sequences comprising 19 minutes of video, recorded in 3 different indoor environments. Compared with *NewCollege*, videos in *EuRoC* are well-suited for evaluating the short-term performance and efficiency of VO (without loop closure). Ground-truth tracks are provided using motion capture systems (Vicon and Leica MS50). We evaluate only monocular VO implementations on *EuRoC*.

The short-term performance of VO on *EuRoC* is evaluated with *absolute root-mean-square error* (RMSE) between ground truth track and real-time VO estimation. Identical with previous VSLAM evaluation, the latency of real-time pose tracking per frame is recorded as well; 10-run repeat is conducted for each configuration, i.e., the benchmark sequence, the VSLAM approach and the parameter (number of features tracked per frame). Results for a tested VSLAM configuration are discarded if at least one run experiences track loss. The experiments are conducted on a desktop equipped with an Intel i7 quadcore 4.20GHz CPU (passmark score of 2583 per thread) running the ROS Indigo environment.

Two state-of-the-art VSLAM baselines are included: *SVO*[6] and *DSO* [7]. For fair comparison, the loop closing module is disabled on all ORB-SLAM variants: canonical *ORB*, *MIH-x/32*, *Rnd*, and *Long*. All VSLAM systems are configured to run 10-repeats on *EuRoC* under example configuration (800 features per frame). The RMSE versus the average latency per frame for two example EuRoC sequences are depicted in Fig 5.11.

RMSE results on all 11 EuRoC sequences are summarized in Table 5.4, while latency results are in Table 5.5. The best value (lowest RMSE, lowest latency) in each row is

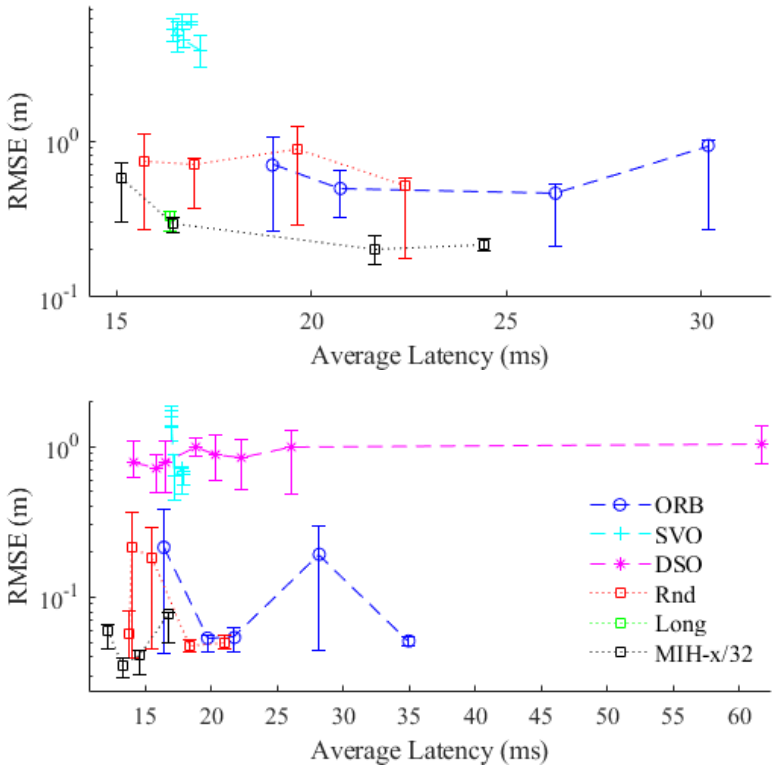


Figure 5.11: Latency vs. accuracy on 2 EuRoC monocular sequence: *MH 04 difficult* (**top**) and *V2 02 medium* (**bottom**). System evaluation involved a sweep of features per frame: 800, 1000, 1500, 2000.

highlighted with bold in Table 5.4 and Table 5.5. According to Table 5.4, *DSO* and the 2 local map building heuristics are not robust enough (e.g. frequent track loss). *SVO* tracks 9 of 11 sequences, but with the highest RMSE over all VSLAM systems. Both *ORB* baseline and proposed *MIH-x/32* track 8 of 11 sequences. Additionally, *MIH-x/32* improves the accuracy relative to baseline *ORB*, with an RMSE average that is 41% lower.

The latency reduction of *MIH-x/32* is less significant for these short-term sequences, when compared with the previous long-term VSLAM evaluations. Nevertheless, *MIH-x/32* has the 2nd lowest average latency among all 6 VSLAM systems, second to *SVO*. When comparing the 3rd quantile of latency, *MIH-x/32* is lower than *SVO* (by 3%), which suggests that tighter latency bounds can be achieved with the Map Hashing algorithm.

Table 5.4: RMSE (m) on EuRoC Sequences

Seq.	*SVO*	*DSO*	*ORB*	*MIH-x/32*	*Rnd*	*Long*
MH 01 easy	0.227	0.407	0.027	0.026	**0.025**	-
MH 02 easy	0.761	-	0.034	**0.031**	0.034	-
MH 03 med	0.798	0.751	0.041	0.086	**0.035**	-
MH 04 diff	4.757	-	0.699	**0.293**	0.746	0.329
MH 05 diff	3.505	-	0.346	**0.197**	-	-
VR1 01 easy	0.726	0.950	0.057	0.040	**0.034**	-
VR1 02 med	0.808	**0.536**	-	-	-	-
VR1 03 diff	-	-	-	-	-	-
VR2 01 easy	0.277	0.297	0.025	0.032	**0.021**	-
VR2 02 med	0.722	0.880	0.053	**0.035**	0.216	-
VR2 03 diff	-	-	-	-	-	-
Avg.	1.477	0.637	0.160	**0.093**	0.159	0.329

Table 5.5: Latency (ms) on EuRoC Sequences

Seq.	*SVO*	*DSO*	*ORB*	*MIH-x/32*	*Rnd*	*Long*
Q_1	7.4	**5.8**	13.9	11.4	12.0	11.3
Avg.	**12.6**	16.4	18.4	15.7	16.0	17.7
Q_3	16.8	19.1	20.7	16.3	**16.1**	21.0

5.6 Conclusion

In this chapter, we demonstrate how an appearance prior can be exploited to build a compact yet relevant local map in VSLAM. Working with the compact local map leads to latency reduction in time-sensitive VSLAM modules, i.e., pose tracking. Meanwhile, the accuracy and robustness of VSLAM is preserved, thanks to the preservation of long-baseline feature associations in the local map. On both long-term VSLAM and short-term VSLAM applications, the described Map Hashing algorithm leads to significant latency reduction in real-time pose tracking, while keeping (if not improving) VSLAM performance relative to the baseline variant and having the best performance relative to other state-of-the-art systems.

CHAPTER 6

GOOD GRAPH SELECTION: COST-EFFECTIVE, BUDGET-AWARE BUNDLE ADJUSTMENT IN VSLAM

6.1 Introduction

In previous chapters, we discussed algorithm improvements for VSLAM front-ends. These algorithm improvements, alongside with recent hardware developments such as FPGA-based feature extraction [26, 27, 28], render highly cost-effective VSLAM front-end reachable. However, the cost-efficiency of VSLAM back-end, especially the local optimization that runs at a high-rate, remains to be a bottleneck for applicable VSLAM.

In VSLAM community, it is favored to use Bundle Adjustment (BA) in local optimization (i.e., local BA), since BA estimates both camera poses and maps with high accuracy and robustness. However, local BA is computationally expensive due to the cubic computational complexity and the iterative computation process. Though recent study starts to explore the usage of FPGA in certain step of BA (e.g. FPGA-based Schur elimination [140]), the rest of BA steps such as re-linearization and factorization still place limitation on the cost-efficiency of BA-based back-end.

Due to the computational cost of local BA, some state-of-the-art VSLAM systems [40, 10] use less expensive filter as back-end solution. The computation complexity of carefully designed EKF variant, i.e., MSCKF [39], is linear to the size of map states. However, MSCKF has the downside of inconsistency and degraded mapping [38]. The majority of state-of-the-art VSLAM systems still utilize BA-based back-end. Several strategies have been developed to reduce of the cost of local BA by only optimizing a scale-limited subset of states. Some systems [6, 7, 31, 45] only take recent states (e.g. camera frames and map points) that stay within a sliding-window. While sliding-window suits for scenarios

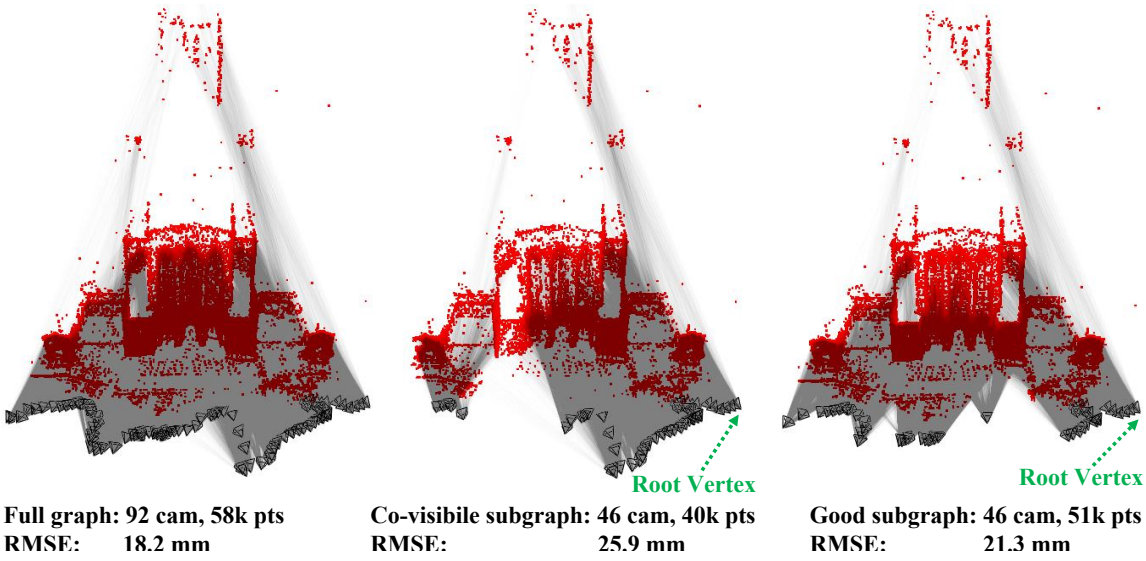


Figure 6.1: BA example on full graph vs. subgraph. **Left**: BA on full graph that has 92 cameras and 58k points. **Middle**: BA on subgraph generated with co-visibility information, which has 46 cameras and 40k points. **Right**: BA on subgraph generated with proposed Good Graph, which has 46 cameras and 51k points. Compare with co-visible subgraph, BA on Good Graph has better accuracy (lower RMSE).

with little re-visit, e.g. infinite-tunnel, it fails to exploit rich historical data when re-visit happens frequently. Other systems [116, 101, 120] use co-visibility graph to organize historical keyframes, and query the co-visible subgraph for local BA. However, the heuristic strategies described above cannot provide any insight on the conditioning of downstream local BA. In the presence of computational limits, the small subset of states selected with these heuristic strategies could form an ill-conditioned local BA, which is slow to converge, or leads to erroneous results.

In this work, we describe a novel, rigorous method to determine the state subset in local BA (i.e., Good Graph), with strong performance guarantee. The theorem backbone of the described Good Graph algorithm is submodular submatrix selection, which is introduced in Chapter 2. Furthermore, the size of desired Good Graph is determined on-the-fly by predicting the amount of valid budget. A small-sized Good Graph is selected for local BA when the budget is tight, e.g. when the camera moves rapidly or computation resource is limited. Otherwise a large-sized Good Graph is selected since the budget can afford

it in local BA. The proposed Good Graph algorithm is integrated into a state-of-the-art VSLAM system [101]. When combined with cost-efficient VSLAM front-end [63], the final VSLAM system achieves superior performance than state-of-the-art VSLAMs under a variety of computational limits. The combined VSLAM system is released [1].

6.2 Background

As pointed out in the pioneering work [38], VSLAM with BA-based back-end has better accuracy and robustness than filter-based ones. Using BA in VSLAM, especially in the high-rate local optimization module, requires careful effort in bounding the scale of states to be optimized. The sliding window strategy has been employed to bound the scale of states in local BA [16, 6, 7, 31, 45]. Only the recent states (camera frames and map points) that stay within the sliding window are optimized in local BA. The older states that are outside the sliding window are either dropped [16, 6] or fixated as linear priors [7, 31, 45]. Though sliding window strategy is applied in visual-inertial odometry that assumes the environment as a infinite tunnel, it is not the optimum solution for SLAM environments with revisits. The ability to reuse historical information that goes outside of sliding window is limited. Fixing historical information as linear prior introduces bias to the optimization, therefore leading to inferior performance when re-visit happens frequently.

Another representative strategy is to bound the scale of the optimization states with co-visibility information. As introduced in [116], co-visibility approximates the amount of mutual information between keyframes. Ideally, a subset of keyframes that have strong co-visibility to each other forms a well-conditioned optimization problem, which can be reliably solved in local BA. For fast query and update, state-of-the-art VSLAM systems [16, 101, 120] typically stores co-visibility information as a graph of historical keyframes, i.e., co-visibility graph. Compared with the sliding window, the co-visibility graph encodes more historical information, which is preferred in general SLAM scenarios. In the pres-

[1] `https://github.com/ivalab/gf_orb_slam2`

ence of revisits, co-visibility graph enables querying and taking early keyframes (and map points) in local BA. Meanwhile in the absence of revisit, the co-visibility graph behaves similar to the sliding window. Nevertheless, co-visibility information is only a rough approximation of frame-to-frame mutual information. Therefore, the actual conditioning of local optimization problem formed with the co-visibility graph is not guaranteed. In practice, local BA with co-visibility graph typically over-selects states, therefore is limited in cost-efficiency.

Apart from bounding the scale of local BA, the incremental nature of SLAM problem has been looked into. Incremental algorithms have been developed to speed up certain matrix manipulations that are compute-intense, such as QR factorization [43, 44], Cholesky factorization [49, 45], and Schur elimination [50]. The method presented in this work is related to the incremental Cholesky factorization work [49]. However, the goal of our work is complementary to these incremental BA algorithms. We are pursuing efficient algorithm to formulate scale-limited BA problem, while the incremental BA algorithms aim at solving a sequence of BA problem efficiently. Cost-efficiency of local BA will be mostly improved by combining proposed BA formulation and incremental solving.

6.3 Good Graph Selection in General BA

Based upon the least squares BA objective (2.1) and linear approximation (2.2) defined in the preliminary chapter, we can write down the normal equation solved in each iteration of non-linear solver:

$$\mathbf{\Lambda}\boldsymbol{\delta} = \boldsymbol{\eta}, \tag{6.1}$$

where $\mathbf{\Lambda} = \mathbf{J^T J}$, $\boldsymbol{\eta} = \mathbf{J^T b}$. The spectral property of system matrix $\mathbf{\Lambda}$ is important: 1) a well-conditioned $\mathbf{\Lambda}$ suggests fast convergence of iterative solving; 2) the volume of $\mathbf{\Lambda}$ is also connected to the information/uncertainty level of corresponding BA problem.

Recall that the BA problem can be equivalently represented as Jacobian matrix or factor

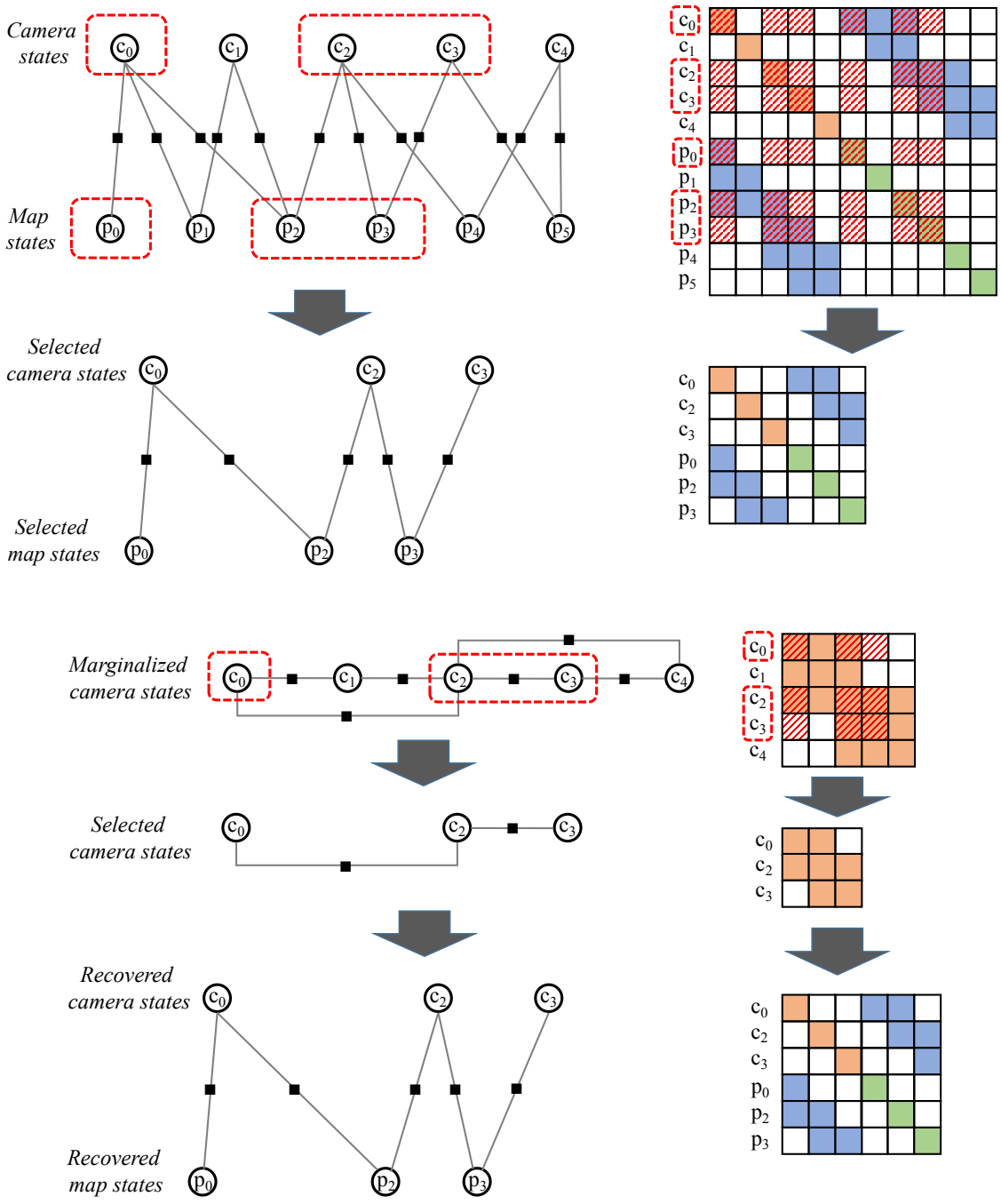


Figure 6.2: Toy example of subgraph selection on complete system vs. camera-only system (best viewed in color). Working on a camera-only system is desired for efficiency purpose. Subgraph selected from camera-only system (and recovered to include map states) will be identical to the one selected from complete system, if all map states that are visible to selected camera subsets are taken.

graph [68]. In the rest of the chapter, following terminology are used: the term *vertex* represents state entity (e.g. camera, map point), *edge* represents measurement, and *graph* represents the BA problem defined by *vertex* and *edge* set.

Due to the cubic complexity of BA solving, working on a subproblem of original BA with smaller scale could be more cost-effective if the full BA solution is not required. Hence we are interested in selecting a subgraph from the full graph (i.e., the full BA problem). As discussed in the Chapter 2, the spectral property of the system matrix is important for BA solving. Naturally, it is desired to select a subgraph with less states, while preserving (if not improving) the spectral property of corresponding system matrix.

Various metrics that measure the spectral property of matrices have been studied in the literature [73, 72]. Similar to Chapter 3, the spectral property of the system matrix is quantified with *logDet* in this chapter, because of the benefit in cost-efficiency. With *logDet* metric, the objective of *Good Graph Selection* can be formulated as submatrix selection problem:

$$\max_{\mathbf{S}\subseteq\{1,2,\ldots,m+n\},|\mathbf{S}|=k} \log\det([\mathbf{\Lambda}(\mathbf{S})]), \tag{6.2}$$

where the complete system matrix $\mathbf{\Lambda}$ contains m camera states and n map states, $\mathbf{S}$ is the index subset of selected camera and map states, $[\mathbf{\Lambda}(\mathbf{S})]$ is the corresponding submatrix, and k is the cardinality constraint. Only the choice of states (vertices) is optimized with submatrix selection objective (6.2), while the choice of non-zero fillings (edges) is conducted implicitly. In other word, we only select a subgraph that has less vertices than the full graph, while the sparsity of the subgraph remains to the same level.

6.3.1 Subgraph Selection on Camera-only System

Now that we have formulated Good Graph Selection as submatrix selection, it is possible to run submatrix selection algorithms on complete system matrix $\mathbf{\Lambda}$. Ideally, this will lead us to a well-conditioned submatrix, as illustrated at the second column of Fig 6.2. The corresponding subgraph, which is presented at the first column of Fig 6.2, should meet

both the size constraint and *logDet* maximization.

In practice, it is undesirable to work on complete system matrix $\mathbf{\Lambda}$. The size of $\mathbf{\Lambda}$ is too large, therefore slows down the submatrix selection drastically. In addition, submatrix selection on the fused system matrix with both camera and map point states may create undesirable behaviors, as indicated in [74]. In BA literature, the map states are marginalized out with Schur elimination. A marginalized matrix that only includes camera states can be obtained:

$$\mathbf{M} = \mathbf{\Lambda_{cc}} - \mathbf{\Lambda_{cp}}\mathbf{\Lambda_{pp}^{-1}}\mathbf{\Lambda_{cp}}^T,\ \mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda_{cc}} & \mathbf{\Lambda_{cp}} \\ \mathbf{\Lambda_{cp}}^T & \mathbf{\Lambda_{pp}} \end{bmatrix}. \tag{6.3}$$

An example of marginalized matrix $\mathbf{M}$ can be found at the top of fourth column of Fig 6.2, while the corresponding camera-only graph is at the top of third column. Notice the size of $\mathbf{M}$ is much smaller than $\mathbf{\Lambda}$.

Selecting a subgraph with k cameras from the marginalized, camera-only matrix $\mathbf{M}$ can be formulated as

$$\max_{\mathbf{S}\subseteq\{1,2,\ldots,m\},|\mathbf{S}|=k} \log\det([\mathbf{M}(\mathbf{S})]). \tag{6.4}$$

More importantly, the corresponding map states can be recovered, by extracting all map points that are visible to the selected camera subsets. An example of recovered subgraph that contains both camera and map states can be found at the bottom of column 3 and 4 of Fig 6.2.

The objective (6.4) for camera-only system is not equivalent to the original objective (6.2). Ideally, the subgraph selected with (6.2) might have better conditioning since both camera and map states can be selected explicitly. However, optimizing (6.2) is both expensive and inconsistent. Map states are selected implicitly in the more efficient and consistent camera-only (6.4): all map points visible to the selected camera subset are taken.

6.3.2 Submatrix Selection with Lazier Greedy

To solve the camera-only objective (6.4) efficiently while limiting the loss in optimality, submodularity of the *logDet* set function is exploited. As described in Chapter 2, the combinatorial optimization objective (6.4) can be approximately solved with greedy methods. Greedy submatrix selection works as follow: starting from submatrix $\mathbf{M}(0)$ of a root camera vertex, e.g. the current keyframe, iteratively searching for the best submatrix that has one more state than $\mathbf{M}(0)$. After $k-1$ iteration, the selected submatrix contains k camera states and the selection stops. Finally, all map states that are visible in the camera subset are included as well.

Further speed-up of greedy selection can be achieved with lazier greedy, as presented in Chapter 3. Compared with greedy method, lazier greedy only evaluate a random subset of candidate states (row and column blocks) at each iteration. The size s of random candidate subset is controlled by decay factor ϵ: $s = \frac{m}{k}\log(\frac{1}{\epsilon})$. Computation complexity of lazier greedy is $\mathcal{O}(\log(\frac{1}{\epsilon})m)$, which is much less than the $\mathcal{O}(km)$ of greedy.

The approximation ratio and computational speed up of lazier greedy hinge on the decay factor ϵ. Lazier greedy with a decay factor of 0 converges to classical greedy, which is with the best approximation ratio and computation cost. Meanwhile lazier greedy with the maximum decay factor (i.e., $e^{-\frac{k}{n}}$) is equivalent to randomized sampling, which is computationally cheap but inconsistent. As discussed in Chapter 3, inconsistent randomized sampling should be avoided in sequential estimation problems such as VSLAM. In this chapter, the decay factor ϵ is fixed to a small positive value 0.05, which enables efficient selection with sub-optimal guarantee. For those interested, a comprehensive evaluation on choice of decay factor can be found in [63].

6.3.3 LogDet with Incremental Cholesky

One bottleneck of lazier greedy algorithm is the cost of computing *logDet* metric. For positive definite square matrix $\mathbf{M}$, efficient computation of *logDet* involves Cholesky fac-

torization $\mathbf{M} = \mathbf{L}\mathbf{L}^T$: $\log\det(\mathbf{M}) = 2\sum\log(diag(\mathbf{L}))$. However, simply plugging the Cholesky-based *logDet* computation in lazier greedy is undesired. As the size of selected submatrix grows during lazier greedy iterations, the cost of Cholesky factorization grows in cubic, therefore affects the cost-efficiency of submatrix selection.

Cost-efficiency of *logDet* computation can be significantly improved with a key observation: Cholesky factorization in iterative submatrix selection is incremental. At each iteration of lazier greedy, system matrix of current selection, dubbed $\mathbf{M}(i)$, only gets updated partially. By re-ordering the row and column blocks, we easily append updated blocks to the bottom right of current submatrix $\mathbf{M}(i)$:

$$\mathbf{M}(i+1) = \begin{bmatrix} \mathbf{M}(i) & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix}. \tag{6.5}$$

Assuming Cholesky factorization of $\mathbf{M}(i)$ is known: $\mathbf{M}(i) = \mathbf{L}(i)\mathbf{L}(i)^T$. According to [141], Cholesky factorization of new submatrix $\mathbf{M}(i+1)$ can be written as:

$$\begin{aligned} \mathbf{L}(i+1) &= \begin{bmatrix} \mathbf{L}(i) & \mathbf{L_1} \\ \mathbf{0} & \mathbf{L_2} \end{bmatrix}, \\ \mathbf{L_1} &= (\mathbf{L}(i)^+)^T\mathbf{B}, \\ \mathbf{L_2} &= chol(\mathbf{D} - \mathbf{L_1}^T\mathbf{L_1}). \end{aligned} \tag{6.6}$$

Computing *logDet* with incremental formula (6.6) avoids redundant Cholesky factorization, therefore improves the cost-efficiency of Good Graph Selection.

6.3.4 Validation of Good Graph Selection

The proposed Good Graph algorithm contains the three improvements described above. In addition, the system matrix $\mathbf{\Lambda}$ is obtained with the analytical approximation of Jacobian [142], which is cheaper to compute than numerical ones. Good Graph algorithm is integrated to a state-of-the-art BA solver, SLAM++ [48]. To validate the cost-efficiency of

Table 6.1: Time Cost Breakdown (ms) of Subgraph BA

Methods		Full	N.C.G.	N.C.L.	N.I.L.	**A.I.L.**
Subgraph	Jacob.	-	588	584	592	**474**
	Schur	-	344	342	**341**	**341**
	Chol.	-	609	105	**12**	**12**
	Misc.	-	121	17	**15**	**15**
	Total	-	1662	1048	960	**842**
Optim.	Jacob.	3410	2183	2184	2199	2180
	Chi2	1042	619	609	609	600
	Linear	10299	4139	4138	4135	4136
	Misc.	31	18	30	19	44
	Total	14782	6959	6961	6962	6960
Total Time		14782	8621	8009	7922	**7802**
Size (cam)		92	46	46	46	46
Diff. (cam)		-	-	3	3	3

Good Graph, BA experiments are conducted on the cathedral dataset, which includes 92 cameras and 58k map points.

Apart from the full BA with all camera and map states, four BA with subgraphs that include 46 cameras are evaluated. Subgraphs are chosen with variants of subgraph selection algorithm: with greedy (G) or lazier greedy (L); with batch (C) or incremental Cholesky (I); with numerical (N) or analytical Jacobian (A). Time cost breakdown of subgraph selection and corresponding BA solving are reported in Table 6.1. The lowest time cost of each subgraph selection step is highlight in bold.

According to Table 6.1, the time consumption of Good Graph Selection (A.I.L.), as well as the total time including downstream subgraph BA, is the lowest among all 4 subgraph BA variants. Each feature described above has clear positive impact to the cost-efficiency of Good Graph Selection. Meanwhile, the difference between efficient Good Graph and slower greedy selection (N.C.G.) is small: only 3 camera states are different within 46 selections. The final RMSE of subgraph BA can be found at Fig 6.1, where the RMSE of Good Graph BA is only slightly higher than that of full BA. In the meantime, full BA takes twice amount of time to compute.

6.4 Budget-Awareness of Local BA in VSLAM

The Good Graph algorithm boosts the cost-efficiency in solving the general BA problem. Compared with general BA, the local BA in VSLAM back-end has more strict budget limits. To provide accurate map points and prevent track failure, local BA has to finish in-time before new measurements accumulate. Similar to general BA, the time cost of the local BA can be adjusted using Good Graph Selection. A general strategy that determines the budget of local BA, and the size k of desired Good Graph, is described in this section.

6.4.1 Predicting Budget of Local BA

The primary role of local BA in VSLAM is to provide accurate map points as localization references for future camera frames. When few map points will be visible in future camera frames, it is necessary to execute local BA at a fast rate so that new map points are fixed in time. When sufficient map points last in future frames, local BA can run at a slower rate, thereby provide a complete and fully-optimized map. Intuitively, the budget of local BA is connected to the amount of persistently visible map points in the future.

Similar to the feature selection work [74], camera poses (with noise) is assumed available for the near future. This assumption is reasonable: for closed-loop systems such as mobile robots, poses in the near future are available from the controller; for open-loop systems such as AR headset, near future poses can be predicted by propagating IMU measurements. If the camera pose at the near future $t + \Delta t$ is available, we can project map points in the predicted camera frame, and count the number of visible map points $N_{t+\delta t}$. Assuming the number of visible map points decays linearly with time, the budget of local BA t_{BA} can be predicted

$$t_{BA} = \Delta t(N_t - N_{min})/(N_t - N_{t+\Delta t}), \tag{6.7}$$

where N_t is the number of points visible at current (key)frame, and N_{min} is the minimum

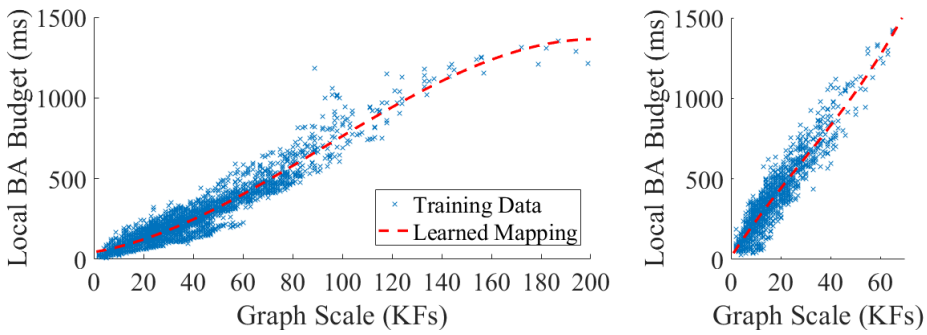


Figure 6.3: Mappings between keyframe number and local BA budget, on two target devices. **Left**: mapping learned on PC with Intel i7 quadcore 4.20GHz CPU. **Right**: mapping learned on Jetson TX2 with ARM SoC (Cortex A57).

number of map points required to remain visible in near future. We set Δt to 0.5 second in the experiments.

The budget of local BA predicted with (6.7) reflects structure and motion information implicitly. A large budget is more likely to appear when the structure is texture-rich (as abundant map points are visible), or when the camera motion is slow (as the parallax of most visible points are limited). A small budget, on the other hand, is typically triggered when the structure has limited texture or the camera is moving rapidly.

6.4.2 Determining the Size of Good Graph

Given a certain budget t_{BA} for local BA, a size-reduced subgraph needs to be selected using Good Graph algorithm, so that the downstream local BA fits within the budget. The key parameter to be sent into good graph selection is the desired size k, characterized by the number of keyframes.

The mapping between keyframe number and local BA budget, dubbed $t_{BA} = f(k)$, is known to be cubic. Coefficients of the actual cubic function, however, vary according to the compute resources available on the target device. For each target device, it is possible to learn the cubic $f(k)$ a priori. Two example mappings used in the experiments are illustrated in Fig 6.3: one for a PC that equips an Intel CPU, the other for an embedded device that has an ARM SoC. The size of desired Good Graph is determined: $k = f^{-1}(t_{BA})$.

6.5 Experiments

This section evaluates the performance of the proposed Good Graph method on a state-of-the-art BA-based stereo VSLAM system, GF-ORB-SLAM (*GF* [63]). Compared with canonical ORB-SLAM (*ORB* [101]), the front-end of *GF* has better cost-efficiency thanks to active feature matching. The back-ends of *GF* and *ORB* are identical: they both use co-visibility to bound local BA, which has limited cost-efficiency. The proposed Good Graph algorithm is integrated to the BA-based back-end of *GF*, dubbed as *GF+GG*. We implement Good Graph algorithm with SLAM++ [48], a state-of-the-art BA solver that supports block matrix manipulation and incremental factorization. The budget-awareness module takes noisy pose prediction (ground truth pose with 10% error) as input. The mapping between local BA budget and desired subgraph size is trained a priori.

Apart from the proposed *GF+GG*, we include two GF-ORB-SLAM variants as evaluation baselines as well. The sliding window strategy is implemented for the BA back-end of *GF*, leading to a combined system *GF+SW*. We also implement an aggressive state selection strategy based on co-visibility: only the top-N co-visible camera states are optimized in local BA. The combined system is referred to as *GF+CV*. Last, the original *ORB* and front-end improved *GF* are evaluated.

Four state-of-the-art visual(-inertial) SLAM systems that support stereo vision are included. *SVO* [6] is a lightweight, visual-only odometry system that has a direct front-end. By skipping explicit feature extraction and matching, *SVO* consumes much less computation than feature-based *GF*. VINS-Fusion [31], or *VIF*, is a visual-inertial SLAM system that tracks sparse optical flow in the front-end and performs sliding window BA in the back-end. ICE-BA [45], or *ICE*, is an incremental and sliding window BA visual-inertial system. A visual-inertial implementation [10] of MSCKF, dubbed as *MSC*, is also included to represent filter-based VSLAM. All three visual-inertial systems, namely *VIF*, *ICE* and *MSC*, track sparse optical flow in the front-end.

6.5.1 Computational Limits Simulation

The goal of this evaluation is to assess the performance of VSLAM under different computational limits. Instead of configuring VSLAM systems on multiple devices with different computational resources, we choose to evaluate VSLAM on the same device, but with different speed in playing back data. Inspired by the idea of *slo-mo* introduced in VSLAM benchmarking [143], we develop *fast-mo* evaluation to simulate different level of computational limits. In *fast-mo*, VSLAM systems are configured on one single device (a PC with an Intel i7 quadcore 4.20GHz CPU), but are evaluated under different rates of visual data input. Performance of a VSLAM running on a PC with 4x real-time data feed serves as the upper bound of its actual real-time performance on a 4-time slower device (with less cache, lower transmission rate, etc.). Five levels of *fast-mo* speeds are evaluated, ranging from 1x to 5x real-time speed. As indicated in [144], low-power CPU can be simulated with 2x and 3x *fast-mo*, while ARM SoC can be simulated with 4x and 5x *fast-mo*.

EuRoC MAV benchmark [25], which contains 11 stereo-inertial sequences recorded in 3 different indoor environments, is used in *fast-mo* evaluation. The performance of each VSLAM system is reflected by the real-time pose tracking output. The accuracy of real-time pose tracking is measured by the *absolute root-mean-square error* (RMSE) [102] between ground truth track and real-time VSLAM output. For each configuration (benchmark sequence, VSLAM system and computational limits), a 10-run repeat is executed. Results are reported only if zero tracking failure occurred during the 10-run repeat. Otherwise results of corresponding configuration are discarded since the VSLAM system cannot work reliably.

Three examples of *fast-mo* results are presented in Fig 6.4. At 1x *fast-mo*, multiple *GF* variants share the best performance (no track failure, lowest RMSE). However, these methods start losing the edge when *fast-mo* speed increases. *GF*, *GF+SW* and *GF+CV* either have quickly increased RMSE or fail to track. Same goes for *ORB*. Two visual-inertial systems with sliding window BA (*ICE* and *VIF*) have significantly higher RMSE,

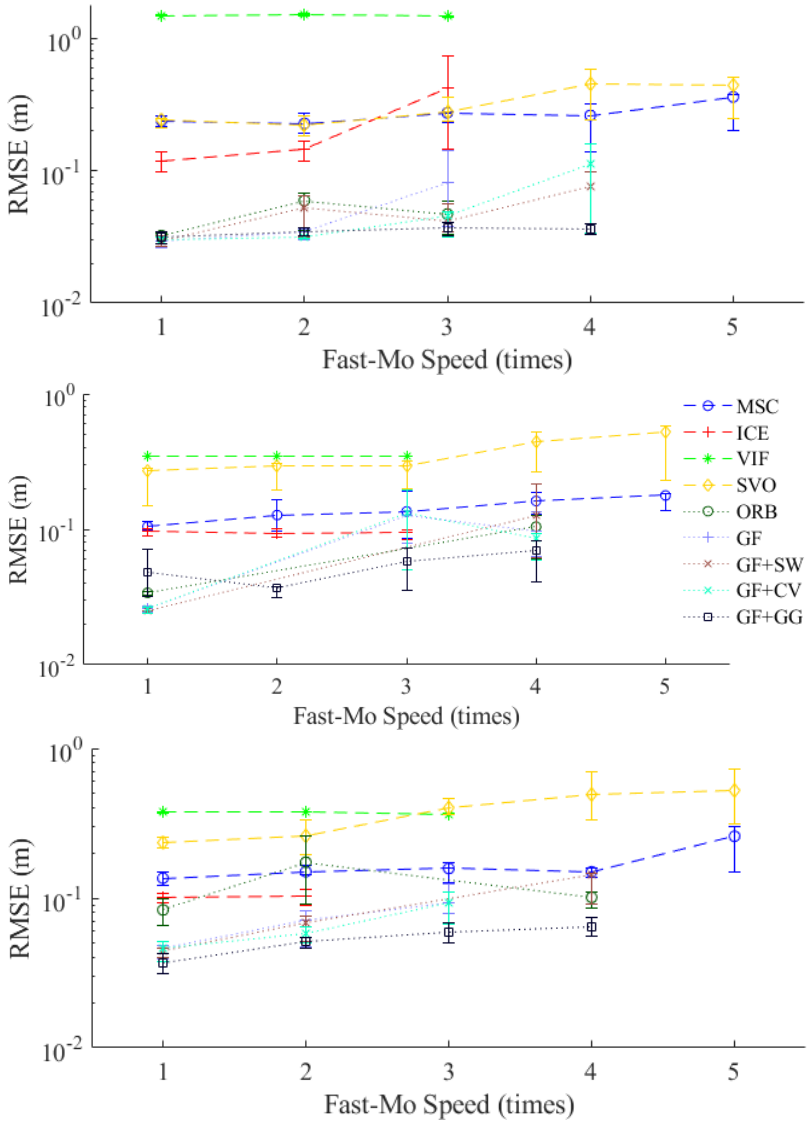


Figure 6.4: *Fast-mo* results on 3 EuRoC sequences: *MH 03 med* (**top**), *V1 02 med* (**middle** and *V2 02 med* (**bottom**). The proposed *GF+GG* tracks on 1x to 4x *fast-mo*, while keeping the best tracking accuracy in all cases (except for 1x on *V1 02 med*). For *V1 02 med*, only *GF+GG* works reliably on 1x to 4x *fast-mo* while other BA-based VSLAM have track failure.

and fail to track on *4x* fast-mo. The two light-weight systems, namely direct *SVO* and filter-based *MSC*, work on all 5 *fast-mo* speeds, yet with high RMSE. The proposed GF+GG consistently has one of the lowest RMSE. The track failure of GF+GG on 5x *fast-mo* is due to the front-end bottleneck, as revealed in the following.

Though the scope of this work is on algorithm improvement for VSLAM, it is still inter-

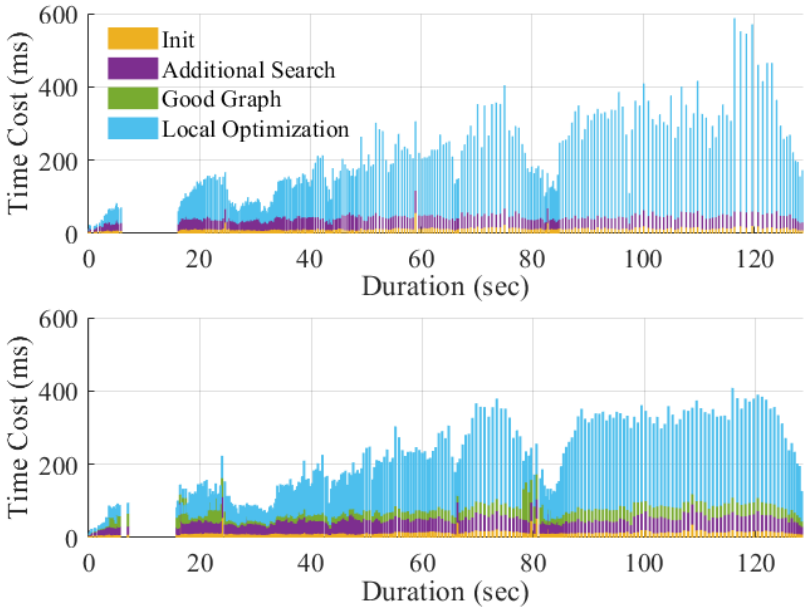


Figure 6.5: Time cost breakdown of BA back-end on sequence *MH 03 med*, under 2x *fast-mo*. **Top**: *GF* that takes all co-visible keyframes into local BA. **Bottom**: *GF+GG* that only optimizes the Good Graph in local BA. Since the camera remains static between 7 sec and 15 sec, no local BA is triggered during that slot.

esting to assess the potential of the Good Graph BA back-end when a hardware-accelerated front-end is available. Again, a simulation is conducted by running four *GF* variants with pre-computed keypoints. In this way the overhead of front-end feature extraction is only several milliseconds. The performance of the actual VSLAM system that contains both algorithm and hardware improvements should fall between the regular *fast-mo* results and the pre-computed results. Both the regular *fast-mo* results and the pre-computed results for are summarized in Table 6.2.

According to Table 6.2, most VSLAM systems except for *MSC* track on all 11 sequences under 1x *fast-mo*. The RMSE of *GF+GG* reaches the lowest, yet only gets improved over *GF* baseline by a small margin. Under 2x and 3x *fast-mo*, the results become interesting: only 3 VSLAM systems, i.e., *SVO*, *VIF* and *GF+GG*, track all 11 sequences. The RMSE of *GF+GG* is an order-of-magnitude lower than the other two. The other 2 *GF* variants with alternative baseline subgraph selection heuristics suffer from track failure. Furthermore, state-of-the-art BA-based VSLAM systems such as *ICE* and *ORB* failed to

track on multiple sequences. Some failures of filter-based *MSC* could be caused by improper initialization; nevertheless the average RMSE of *MSC* is twice that of *GF+GG* on successful sequences. With pre-computed keypoints, *GF+GG* has the lowest RMSE on 2x *fast-mo*, and full track success on 3x *fast-mo*. Though *GF+GG* starts to have track failure on 4x *fast-mo*, it still achieves top-of-the-line robustness (with 3 failed sequences) and accuracy (lowest RMSE on almost all working sequences). We further argue that these track loss are due to front-end processing speed. When coupled with pre-computed keypoints, *GF+GG* tracks on all 11 sequences on 4x *fast-mo*. In general, the performance degradation of *GF+GG* is quite graceful from 2x *fast-mo* to 4x *fast-mo*. Further increasing the *fast-mo* speed to 5x leads to track failure in most VSLAM systems. *SVO* is the only system that works on all 11 sequences in this case, though has quite high RMSE. After releasing the bottleneck of feature extraction in VSLAM font-end with pre-computed keypoints, *GF* variants track most sequences as well. The RMSE of *GF+GG* is the lowest among 4 variants on 8 out of 9 working sequences.

Finally, the time cost breakdown of BA back-end on 2x *fast-mo* simulation is illustrated in Fig. 6.5. Compared with *GF* that takes all co-visible keyframes into local BA, the time cost of local BA in *GF+GG* is better bounded. Furthermore, the time cost of Good Graph algorithm is quite small, compared to the time spent on state optimization.

6.5.2 VSLAM on Low-Power Device

To further validate the conclusion drawn from *fast-mo* simulation, we evaluate VSLAM systems on an embedded device, Jetson TX2 with ARM SoC (Cortex A57). Due to compatibility issues, two VSLAM systems *VIF* and *SVO* are dropped. To resolve the bottleneck on front-end feature extraction, GPU-acceleration is enabled for 5 *ORB*-based systems. Evaluation results on embedded device are summarized in the last block of Table 6.2. Incremental *ICE* tracks on all sequences, while filter-based *MSC* only failed on 1 sequence. However, the RMSE achieved with either methods is quite high. The proposed *GF+GG*,

Table 6.2: RMSE (m) on EuRoC Stereo Sequences

Cfg	Methods	Sequences											All Avg.	ORB Avg.
		MH 01	*MH 02*	*MH 03*	*MH 04*	*MH 05*	*V1 01*	*V1 02*	*V1 03*	*V2 01*	*V2 02*	*V2 03*		
PC 1x	MSC	-	0.139	0.239	0.273	0.245	0.092	0.105	0.168	0.063	0.136	1.290	0.275	-
	ICE	0.115	0.075	0.119	0.233	0.237	0.066	0.096	0.153	0.115	0.101	0.175	0.135	-
	VIF	1.152	1.324	1.490	2.243	2.310	0.345	0.350	0.533	0.474	0.373	0.268	0.987	-
	SVO	0.126	0.102	0.242	1.942	0.292	0.092	0.269	0.359	0.159	0.236	2.471	0.572	-
	ORB	0.030	0.025	0.032	0.166	0.089	0.037	0.034	0.310	0.049	0.083	0.304	0.105	0.105
	GF	0.023	**0.018**	**0.029**	0.119	0.066	**0.036**	0.026	0.053	**0.042**	0.046	0.186	0.059	0.059
	GF+SW	0.023	**0.018**	**0.029**	**0.094**	**0.062**	**0.036**	**0.025**	0.051	0.046	0.045	0.220	0.059	0.059
	GF+CV	**0.021**	0.019	0.030	0.115	0.065	**0.036**	0.026	0.044	0.046	0.046	0.194	0.058	0.058
	GF+GG	0.027	0.019	0.031	0.107	0.073	**0.036**	0.048	**0.030**	**0.042**	**0.036**	**0.161**	**0.056**	**0.056**
PC 1x; Pre	GF	0.025	0.019	0.029	0.109	0.064	**0.035**	**0.024**	0.055	**0.043**	**0.040**	0.228	0.061	0.061
	GF+SW	0.023	**0.018**	0.030	**0.104**	0.064	0.036	0.026	0.047	0.046	**0.040**	0.183	0.056	0.056
	GF+CV	**0.021**	**0.018**	**0.028**	0.113	**0.061**	0.036	0.026	**0.042**	0.046	0.042	**0.160**	**0.054**	**0.054**
	GF+GG	0.024	**0.018**	0.030	0.110	0.070	0.036	0.047	0.058	**0.043**	0.046	0.173	0.060	0.060
PC 2x	MSC	-	0.169	0.228	0.244	0.283	0.102	0.127	0.189	0.077	0.150	-	0.174	-
	ICE	0.116	0.074	0.147	0.276	0.249	0.062	0.094	0.146	0.113	0.102	0.299	0.152	-
	VIF	1.153	1.324	1.505	2.255	2.316	0.345	0.350	0.552	0.467	0.374	0.264	0.991	-
	SVO	0.127	0.106	0.220	2.081	0.438	0.097	0.292	0.409	0.186	0.260	2.391	0.601	-
	ORB	0.032	0.026	0.059	0.190	0.097	0.037	-	-	-	0.174	-	0.088	0.088
	GF	0.028	**0.021**	0.034	0.133	**0.083**	0.037	-	0.202	0.044	0.071	**0.113**	0.077	0.058
	GF+SW	0.030	**0.021**	0.052	**0.108**	0.085	**0.036**	-	-	0.046	0.068	0.117	**0.062**	0.057
	GF+CV	0.025	0.022	**0.032**	-	0.099	**0.036**	-	**0.134**	0.048	0.058	0.123	0.064	**0.045**
	GF+GG	**0.024**	0.022	0.035	0.118	0.084	**0.036**	**0.037**	0.178	**0.041**	**0.051**	0.194	0.075	0.053
PC 2x; Pre	GF	0.025	**0.019**	0.030	**0.107**	**0.064**	0.060	0.043	-	0.045	0.064	**0.144**	**0.060**	0.053
	GF+SW	0.028	**0.019**	**0.028**	0.132	0.067	**0.036**	**0.030**	**0.090**	**0.042**	0.057	0.206	0.067	**0.052**
	GF+CV	0.028	0.020	0.030	0.125	0.069	0.037	0.031	-	0.043	0.052	0.167	**0.060**	**0.052**
	GF+GG	**0.024**	0.020	0.035	0.138	0.079	**0.036**	0.042	0.095	0.043	**0.049**	0.150	0.065	0.054

Table 6.3: RMSE (m) on EuRoC Stereo Sequences (cont'd)

Cfg	Methods	Sequences											All Avg.	ORB Avg.
		MH 01	*MH 02*	*MH 03*	*MH 04*	*MH 05*	*V1 01*	*V1 02*	*V1 03*	*V2 01*	*V2 02*	*V2 03*		
PC 3x	MSC	-	0.151	0.274	-	0.329	0.108	0.136	0.230	0.078	0.158	-	0.183	-
	ICE	0.112	0.261	0.420	0.236	0.232	0.066	0.094	0.190	0.121	-	-	0.192	-
	VIF	1.029	1.042	1.476	2.273	2.313	0.320	0.347	0.549	0.434	0.365	0.262	0.946	-
	SVO	0.117	0.117	0.282	1.872	0.447	0.094	0.293	0.339	0.220	0.399	2.514	0.609	-
	ORB	0.029	0.026	0.047	-	0.095	-	-	-	0.131	-	-	**0.066**	0.066
	GF	0.031	0.024	0.082	0.174	**0.090**	0.040	0.127	-	**0.048**	0.093	-	0.079	0.055
	GF+SW	0.033	**0.022**	0.042	**0.151**	0.099	0.038	-	-	0.050	-	**0.153**	0.073	0.049
	GF+CV	0.035	0.027	0.046	0.160	0.095	0.039	0.133	-	0.061	0.094	-	0.077	0.053
	GF+GG	**0.027**	0.023	**0.037**	0.154	0.098	**0.037**	**0.059**	**0.152**	**0.048**	**0.059**	0.169	0.078	**0.047**
PC 3x; Pre	GF	0.033	**0.021**	0.039	0.155	0.083	0.038	0.091	-	0.043	0.088	**0.158**	0.075	0.044
	GF+SW	0.029	0.022	**0.031**	**0.138**	0.083	0.045	0.102	-	**0.042**	-	0.195	0.076	0.041
	GF+CV	0.026	0.023	0.036	0.145	**0.072**	0.052	0.084	-	0.045	-	0.159	**0.071**	**0.040**
	GF+GG	**0.023**	**0.021**	0.035	0.156	0.087	**0.036**	**0.059**	**0.169**	**0.042**	**0.059**	0.198	0.080	0.042
PC 4x	MSC	-	0.223	0.262	1.319	0.317	0.149	0.162	-	0.096	0.150	-	0.335	-
	ICE	0.096	0.808	-	0.258	0.225	0.070	-	-	0.113	-	-	0.262	-
	VIF	-	-	-	-	-	-	-	-	-	-	-	-	-
	SVO	0.122	0.114	0.457	2.173	0.400	0.093	0.445	0.485	0.183	0.493	2.311	0.661	-
	ORB	**0.031**	0.024	-	0.210	0.118	**0.039**	0.105	-	0.210	0.101	-	0.105	0.661
	GF	-	**0.022**	-	0.189	0.123	-	0.098	-	0.065	-	-	0.100	0.099
	GF+SW	-	**0.022**	0.075	**0.178**	0.142	-	0.127	-	0.113	0.145	-	0.136	0.121
	GF+CV	-	0.023	0.112	0.201	0.115	-	0.085	-	0.081	-	-	0.103	0.101
	GF+GG	-	0.024	**0.036**	**0.178**	**0.095**	-	**0.069**	**0.230**	**0.047**	**0.064**	-	**0.093**	**0.080**
PC 4x; Pre	GF	0.035	0.028	0.068	**0.147**	0.088	0.050	0.111	-	**0.043**	-	0.202	0.086	0.072
	GF+SW	0.037	0.025	0.035	0.166	0.089	0.039	**0.092**	-	0.048	-	**0.144**	0.075	0.071
	GF+CV	0.032	**0.023**	**0.034**	0.182	0.103	0.039	0.120	-	**0.043**	-	-	**0.072**	0.077
	GF+GG	**0.027**	**0.023**	0.038	0.161	**0.082**	**0.036**	0.113	**0.231**	0.045	**0.063**	0.202	0.093	**0.069**

Table 6.4: RMSE (m) on EuRoC Stereo Sequences (cont'd)

Cfg	Methods	Sequences											All Avg.	ORB Avg.
		MH 01	*MH 02*	*MH 03*	*MH 04*	*MH 05*	*V1 01*	*V1 02*	*V1 03*	*V2 01*	*V2 02*	*V2 03*		
PC 5x	MSC	-	-	**0.361**	-	0.817	0.129	**0.178**	-	-	**0.259**	**1.341**	**0.514**	-
	ICE	-	-	-	-	-	-	-	-	-	-	-	-	-
	VIF	-	-	-	-	-	-	-	-	-	-	-	-	-
	SVO	**0.106**	**0.112**	0.446	**1.883**	**0.355**	**0.094**	0.524	**0.475**	**0.216**	0.518	2.363	0.645	-
	ORB	-	-	-	-	-	-	-	-	-	-	-	-	-
	GF	-	-	-	-	-	-	-	-	-	-	-	-	-
	GF+SW	-	-	-	-	-	-	-	-	-	-	-	-	-
	GF+CV	-	-	-	-	-	-	-	-	-	-	-	-	-
	GF+GG	-	-	-	-	-	-	-	-	-	-	-	-	-
PC 5x; Pre	GF	0.039	0.025	0.060	0.181	0.104	0.051	-	-	0.078	0.112	**0.200**	0.094	-
	GF+SW	0.046	**0.024**	0.062	0.182	0.117	0.043	0.114	-	**0.048**	0.095	0.206	0.094	-
	GF+CV	0.048	0.025	0.083	0.202	0.112	0.048	0.130	-	0.067	0.132	-	0.094	-
	GF+GG	**0.028**	0.026	**0.037**	**0.173**	**0.100**	**0.038**	**0.110**	-	**0.048**	**0.067**	-	**0.070**	-
Embedded 1x	MSC	-	0.136	0.249	**0.228**	0.355	0.087	0.105	0.179	0.065	**0.119**	0.970	0.249	-
	ICE	0.100	0.320	1.174	0.247	**0.224**	0.067	0.107	**0.146**	0.136	0.155	**0.538**	0.292	-
	ORB	-	-	-	-	-	0.442	-	-	-	-	-	0.442	-
	GF	-	0.045	-	-	-	**0.048**	-	-	**0.058**	-	-	**0.050**	**0.048**
	GF+SW	-	0.042	-	-	0.458	0.050	-	-	0.063	-	-	0.153	0.050
	GF+CV	0.053	-	-	-	0.346	0.066	-	-	0.059	-	-	0.131	0.066
	GF+GG	**0.049**	**0.033**	**0.102**	-	0.234	0.052	**0.099**	-	0.062	-	-	0.090	0.052

Table 6.5: RMSE (m) on UZH-FPV Stereo Sequences

Methods	**Sequences**						
	if 3	*if 5*	*if 6*	*if 7*	*if 9*	*if 10*	**Avg.**
MSC	5.36	8.11	**1.59**	**1.38**	**2.31**	**2.08**	3.47
ICE	-	-	-	-	-	-	-
VIF	-	5.27	7.46	4.51	-	-	5.75
SVO	9.43	5.25	8.41	6.68	3.95	3.95	6.28
ORB	2.03	**2.34**	1.95	-	-	-	**2.11**
GF	**1.91**	2.92	-	-	3.33	3.77	2.98
GF+SW	-	2.81	-	-	-	3.84	3.32
GF+CV	-	2.92	-	-	3.62	3.65	3.40
GF+GG	2.28	2.42	-	-	2.57	2.91	2.55

on the other hand, has the lowest RMSE. The clear performance improvement from *GF* to *GF+GG* suggests that the bottleneck of BA back-end has been largely tackled. We further conjure that, when working with a FPGA-based front-end, the 4 failure cases shall be resolved for *GF+GG*.

6.5.3 VSLAM on Agile Camera Motion

Apart from compute limits, another challenging scenario for state-of-the-art VSLAM is the agile camera motion. To track the camera reliably, VSLAM systems have to process new visual measurements and fix new map points within a small budget. All 9 VSLAM systems mentioned above are evaluated on the UZH FPV benchmark [145], which is recorded with a racing quadrotor with max speed of 12.8 m/s. Six indoor sequences collected with front-facing fisheye camera and full ground truth coverage are selected in this evaluation. Ground truth trajectories are collected with Vicon motion tracking.

Similar to the previous evaluation, a 10-run repeat is executed for each configuration, and zero tracking failure is allowed. The RMSEs of VSLAM real-time tracking output are summarized in Table 6.5. Only 2 VSLAM systems manage to track all 6 agile-motion sequences without any failure: filter-based *MSC* and direct *SVO*. However, the RMSE of *SVO* is really high. *MSC* seems to perform quite well on multiple sequences; yet it does

Table 6.6: RPE (m/s) on UZH-FPV Stereo Sequences

Methods	Sequences						Avg.
	if 3	*if 5*	*if 6*	*if 7*	*if 9*	*if 10*	
MSC	1.94	2.06	2.04	1.09	0.66	0.68	1.41
ICE	-	-	-	-	-	-	-
VIF	-	0.61	1.04	**0.49**	-	-	0.71
SVO	1.19	1.39	1.65	0.88	0.81	0.94	1.14
ORB	0.34	**0.41**	**0.44**	-	-	-	0.40
GF	**0.30**	0.51	-	-	0.39	0.51	0.43
GF+SW	-	0.50	-	-	-	0.53	0.51
GF+CV	-	0.51	-	-	0.43	0.51	0.48
GF+GG	0.34	0.43	-	-	**0.31**	**0.45**	**0.38**

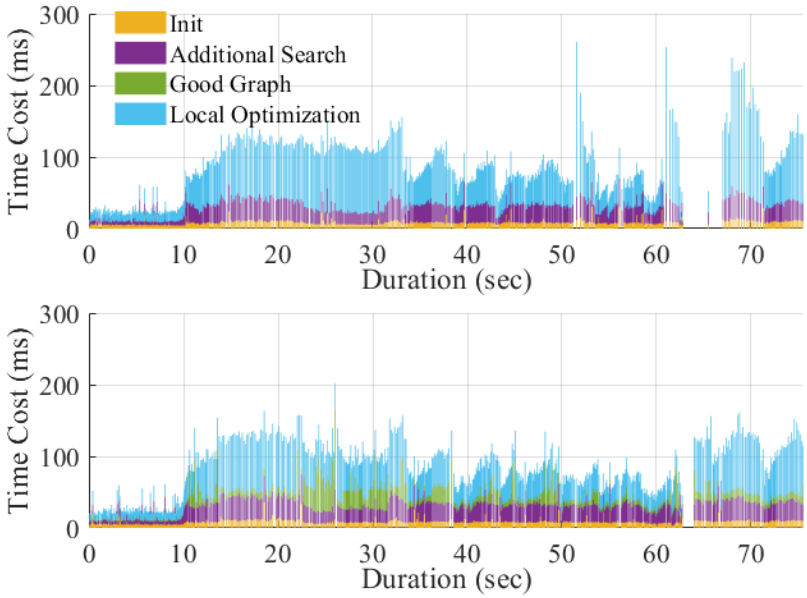


Figure 6.6: Time cost breakdown of BA back-end on agile motion sequence *indoor forward 10*. **Top**: *GF* that takes all co-visible keyframes into local BA. **Bottom**: *GF+GG* that only optimizes the Good Graph in local BA. Several frames are dropped at around 63 sec for both methods; but they are able to recover quickly afterwards.

really bad on certain sequences such as *if 3* and *if 5*. The proposed *GF+GG*, on the other hand, tracks on 4 out of 6 sequences with relative stable accuracy: the RMSEs are around 2.50 meter. Other BA-based VSLAM systems either fail to track on most of the sequences (e.g. *ICE*, *VIF*, *ORB*, *GF+SW* and *GF+CV*), or have high RMSE (e.g. *GF*).

The RMSEs in Table 6.5 are much higher than those in EuRoC evaluation due to the challenging agile camera motion. As an alternative metric, we compute the relative posi-

tion error (RPE) of agile-motion results using 10-second sliding window. The RPEs are summarized in Table 6.6. Interestingly, the proposed *GF+GG* has the lowest RPE on average, which suggests *GF+GG* tracks camera motion well at local scale. Furthermore, the performance of *GF+GG* could be further improved with inertial measurements; currently only visual data is used in *GF+GG*.

The time cost breakdown of the BA back-end on an agile motion sequence is illustrated in Fig. 6.6. Similar to the evaluation under computational limits, *GF+GG* bounds the time cost of local BA better than *GF*; the time cost of Good Graph algorithm is also small.

6.6 Conclusion

This chapter describes a novel, rigorous method to improve the cost-efficiency of BA-based VSLAM back-end, which is essential for SLAM applications with compute limits. An efficient algorithm is developed to select a size-reduced graph for local BA with conditioning preservation. The budget of local BA, as well as the desired size of selected graph, are determined with budget-awareness. The proposed algorithm is integrated into a state-of-the-art VSLAM system. Superior performance is achieved under a variety of computational limits, when compared against state-of-the-art VSLAM systems.

CHAPTER 7
CLOSED-LOOP NAVIGATION WITH ROBUST, LOW-LATENCY VISUAL INERTIAL SLAM

7.1 Introduction

In this chapter, we study the performance of VSLAM in a representative robotics application, i.e., closed-loop navigation. Traditional benchmarking of VSLAM employs open-loop analysis (i.e., isolating VSLAM from rest of the autonomous system). Open-loop evaluation fails to fully address the impact of noise or measurement error on navigation performance. Therefore, it is hard to gain insights on VSLAM from published benchmark scores. Furthermore, open-loop evaluation does not measure the impact of latency in online VSLAM applications. In this chapter, we first describe the VSLAM computational components essential to high performance closed-loop navigation. Then we present a reproducible benchmarking simulation for closed-loop VSLAM evaluation.

The proposed approach employs a loosely-coupled visual-inertial SLAM setup. Visual and inertial sensors provide complementary constraints in state estimation. The visual sensor provides accurate, yet sparse and delayed measurements of absolute landmarks in the environment. Estimation drift is mitigated by observing and matching landmarks with a long but potentially intermittent measurement history during state optimization. The inertial sensor provides high-rate, almost-instantaneous, yet drifting measurements of robot motion. Inertial measurements constrain the unobservable scale of monocular vision and compensate for short duration visual feature loss (e.g. in texture poor settings). Closed-loop navigation benefits from visual-inertial sensor fusion when these properties are simultaneously leveraged; more so when the state estimation rate admits feedback control.

A critical characteristic of visual-inertial state estimation in closed-loop navigation is

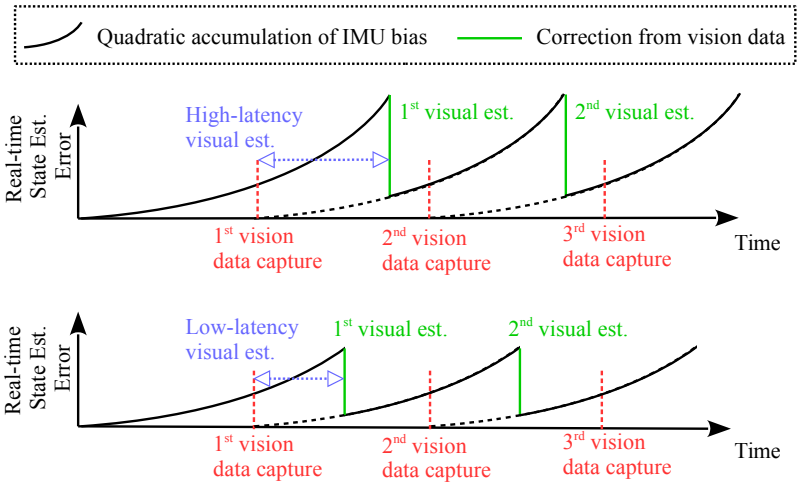


Figure 7.1: Impact of visual processing latency in visual-inertial SLAM. Assuming 100% correct visual estimation and purely-random IMU noise, the only source of error in visual-inertial state estimation is accumulated IMU bias (quadratic in time). **Top:** visual-inertial state estimation trend when visual estimation takes 75% of the visual processing budget. **Bottom:** Trend of visual-inertial state estimation when visual estimation takes 50% of the budget. Reduced latency yields a reduced state estimation error.

the latency of visual processing (e.g. feature extraction, data association, BA, etc.). As illustrated in Fig 7.1, a slight latency-reduction on visual processing end could improve the accuracy of fused visual-inertial state estimate due to the quadratic (in time) nature of accumulated IMU bias. Therefore it is important that VSLAM exhibit the smallest latency possible. As demonstrated in Chapter 3, an active feature matching algorithm, namely Good Feature Matching (*GF*), improves the cost-efficiency of feature matching significantly. Through open-loop evaluation, the feature-based VSLAM system with Good Feature Matching achieves accurate and low-latency pose tracking. In this chapter, we integrate *GF* VSLAM to a multi-sensor fusion framework [146]. The resulting visual-inertial SLAM system has low latency and high performance. The impact of the system is demonstrated in simulated closed-loop navigation scenarios. When deployed on a navigation planner and controller system, the navigation performance of the described visual-inertial SLAM outperforms state-of-the-art direct and feature-based visual(-inertial) SLAM systems.

7.2 Background

In this section, we first review existing works on visual-inertial state estimation for closed-loop navigation. The term **VI-SLAM** will be used to indicate both visual-inertial odometry (VIO) and visual-inertial SLAM. We also review the evaluation methods for closed-loop navigation.

7.2.1 VI-SLAM in Closed-Loop Navigation

Pioneering works developing VI-SLAM for closed-loop navigation tasks on ground [147] and aerial vehicle navigation [148, 149] utilize EKF-based visual inertial state estimation. Due to the cubic computation cost of the EKF in state dimension, only the pose is fully tracked. There is no long term mapping in these EKF solutions, which limits the performance of state estimation and closed-loop navigation.

Closed-loop navigation with full VI-SLAM running off-board was demonstrated in [150], based on the ground breaking PTAM [3]. Two important ideas were raised in [3]: 1) non-linear bundle adjustment (BA) can achieve better performance than filters in state estimation; 2) pose tracking and mapping can be parallelized and function semi-independently. In [151], a customized version of monocular PTAM was fit into the on-board budget of a Micro Aerial Vehicle (MAV). Still, PTAM with full BA is computationally expensive; when working with more than 1 camera, estimation is shifted off-board to meet real-time needs [152].

To reduce the on-board computational load of VI-SLAM, alternatives have been explored in both the image processing front-end and state optimization back-end. The combination of sparse optical flow (e.g. KLT [88]) and the linear-complexity filter (e.g. MSCKF [39]) has been chosen as an efficient VI-SLAM solution [153, 10, 154]. MSCKF-based VI-SLAM with both monocular [153] and stereo [10] vision runs in real-time, while still leaving room for other navigation modules, e.g. planning and control. One downside of

MSCKF is the low mapping quality. To compensate, a BA-based mapper was integrated into the monocular MSCKF-based VI-SLAM in [154].

Apart from the linear complexity filter, the sliding-window filter (SWF) [155] has also been used for efficient VI-SLAM. SWF keeps a finite set of historical information (keyframes and landmarks) in the window for performing BA within it. Compared to full BA, SWF has better bounding on the computation footprint while still allowing re-linearization of historical information. Representative works using the SWF include feature-based OKVIS [16] and KLT-based VINS-Fusion [156]. Closed-loop navigation with OKVIS has been demonstrated on both ground [157] and aerial robots [158]. For a high-rate feedback signal to the controller, OKVIS was enhanced with visual-inertial fusion [146]. In [156], a KLT-based image processing front-end is developed to further cut down the cost of feature extraction and matching. Full navigation has been demonstrated with VINS-Fusion on a MAV [159].

The efficiency of direct VI-SLAM has also been studied. Instead of extracting and matching features explicitly between visual frames, the direct method jointly solves data association and state optimization by optimizing a direct objective on raw image readings. Direct VI-SLAM systems such as SVO [6] and ROVIO [33] have been integrated into closed-loop navigation systems, e.g. monocular system [160], stereo system [161], and IR system [162]. While both KLT and direct VI-SLAM are computationally cheaper than feature-based VI-SLAM, they are more sensitive to navigation-based conditions: e.g. they require accurate pose prediction (from inertial) and constant light condition. Furthermore, both KLT and direct methods are mostly suited for extracting short-baseline feature matchings. Feature descriptor matching, on the other hand, can find reliable long-baseline feature matches that are of great value to state optimization (Fig 1.2).

7.2.2 Evaluation of Closed-Loop Navigation

Open-loop evaluation of different VI-SLAM methods has been extensively conducted in the literature, e.g. on multiple public benchmarks [63], on multiple computation devices [11, 163], and on multiple synthetic environments [164, 165]. Closed-loop evaluation of different VI-SLAM methods in navigation tasks, however, has not been pursued till very recently. One big challenge towards closed-loop evaluation is that closed-loop navigation is not just a software problem; the performance of the full system can be affected by sensor choice, computational resources, system kinematics and target environment. All these factors need to be determined and experimentally controlled to comprehensively evaluate the performance of closed-loop navigation using VI-SLAM.

One way to conduct comprehensive and repeatable closed-loop evaluation is via simulation. Several existing simulators are commonly used in the robotics community. Gazebo [166] is one of the best simulators in robotics field, with MAV-specific extensions such as RotorS [167]. AirSim [168] from Microsoft Research is another choice, with photorealistic renderings of visual data via Unreal Engine. A more recent development incorporates hardware in the loop [169]. By capturing the trajectory of the actual robot on the fly, while rendering virtual visual data on a remote workstation, [169] collects both actual data under real physics and virtual data from an easy-to-extend renderer. To the best of our knowledge, there is no comprehensive evaluation of different VI-SLAM methods for closed-loop navigation. In this paper, we provide evaluation results on closed-loop navigation using Gazebo.

7.3 Closed-Loop Navigation System Design

As illustrated in Fig 7.2, our closed-loop navigation system consists of three major subsystems: 1) a feature-based Visual SLAM subsystem taking stereo vision data to generate sparse yet accurate state estimates; 2) a sensor fusion subsystem taking both sparse visual

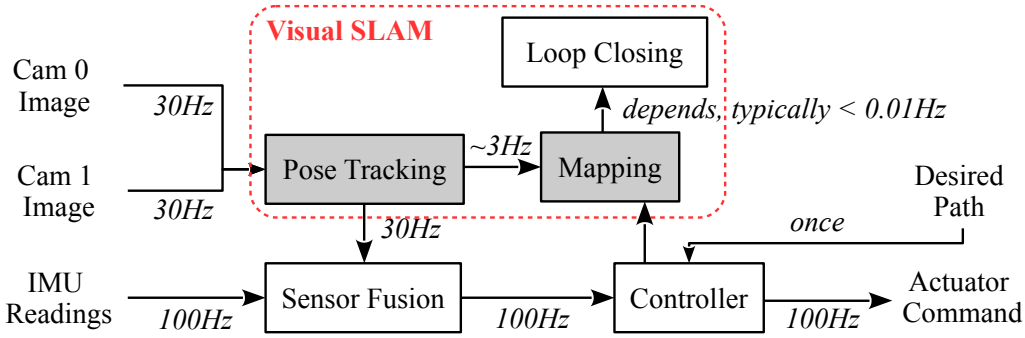


Figure 7.2: Overview of the closed-loop navigation system. Algorithmic improvements proposed in Chapter 3 and 6 are included to the key modules, pose tracking and mapping.

estimates and high-rate inertial readings for high-rate and accurate positioning; and 3) a controller taking high-rate output from sensor fusion to generate actuator commands. It falls into the category of loosely-coupled stereo inertial SLAM. Though tightly-coupled VI-SLAM systems tend to have better performance than loosely coupled ones, we choose loosely-coupled VI-SLAM in our navigation system because it is more open to future enhancement by additional sensing modules, such as wheel odometry and magnetic position. Though this paper only discusses a stereo implementation, the described SLAM system also supports an RGB-D camera as the visual sensor.

The visual SLAM subsystem is based on ORB-SLAM [4], which consists of three cascaded modules running as separate threads: pose tracking, mapping, and loop closing. The pose tracking module estimates the current pose at the same rate as visual sensory input (e.g. 30Hz). The mapping module accumulates the output of pose tracking, i.e., feature matchings and current pose estimate, and performs a lower-than-frame-rate (e.g. 3Hz) BA on historical measurements within a window. The loop closing module is only activated when the robot revisits previously explored places. It typically triggers at an extremely low rate, e.g. 0.01Hz.

The efficiency of pose tracking is essential in visual-inertial state estimation and therefore in closed-loop navigation. Pose tracking results are fused with high-rate inertial readings in EKF-based sensor fusion [146]. The fused high-rate position signal is fed into

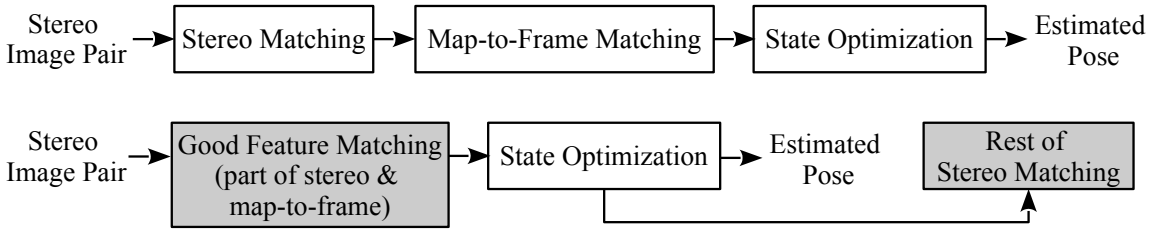


Figure 7.3: Comparison between different pose tracking pipelines. **Top:** canonical pose tracking pipeline, e.g. in ORB-SLAM [4]. **Bottom:** low-latency pose tracking pipeline, where modifications are highlighted in shade.

position controller [170] to drive the robot. As mentioned earlier (e.g. in Fig 7.1), latency in pose tracking significantly impacts the performance of visual-inertial state estimation. Cost-efficiency of BA in mapping thread is also crucial, since it affects the quality and quantity of map points available in pose tracking. As described in Chapter 6, the budget of local BA is anticipated with controller feedback. Next we will briefly recap the algorithm improvements.

7.4 Robust, Low-Latency Stereo VSLAM

7.4.1 Low-Latency Pose Tracking

The pose tracking thread in stereo VSLAM systems like ORB-SLAM [4] consists of three sequential steps (top row of Fig 7.3): stereo matching, map-to-frame matching, and state optimization. Stereo matching provide disparity/depth information for the current measured features. Map-to-frame matching associates these measurements to features in the 3D map (assumed to be static). State optimization recovers the current pose of the camera/robot based on matched and tracked features. Following this pipeline, all valid feature matchings are guaranteed to inform pose optimization. However, it is often unnecessary to use all valid feature matchings in pose optimization, which is already over-determined [62]. With the Good Feature Matching algorithm described in Chapter 3, it is possible to prioritize and re-sequence the feature matching effort (bottom row of Fig 7.3). As a consequence, pose tracking can have a lower latency than before, while preserve the accuracy

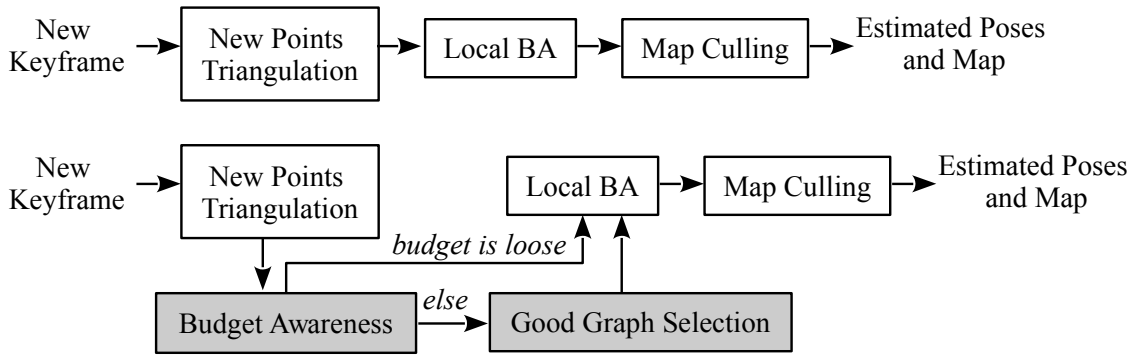


Figure 7.4: Comparison between different pose tracking pipelines. **Top:** canonical pose tracking pipeline, e.g. in ORB-SLAM [4]. **Bottom:** low-latency pose tracking pipeline, where modifications are highlighted in shade.

and robustness.

7.4.2 Cost-Effective Local BA

The mapping thread in stereo VSLAM systems like ORB-SLAM [4] consists of three sequential steps (top row of Fig 7.4): new points triangulation, local BA, and map culling. The computational cost of local BA, which is cubic to the scale of states optimized, is the bottleneck of mapping task, In online applications such as closed-loop navigation, the budget for local BA varies according to multiple factors: the camera / robot motion, the texture level of working environment, the computational resources available, etc. The on-the-fly budget awareness module described in Chapter 6 is added to the original mapping pipeline (bottom row of Fig 7.4). When the budget is loose, all valid states are optimized by local BA, which is identical to original mapping method. When the budget is tight due to camera motion or computational limits, the Good Graph algorithm described in Chapter 6 is activated to select a subset of states in local BA with strong performance guarantee.

7.5 Feedback Control

The desired path $d^*(t) \in \mathbb{R}^2$ is constructed from a series of specified waypoints using splines. An exponentially stabilizing trajectory tracking controller for Hilare-style robots

[170] generates a kinematically realistic trajectory for the robot to follow. In the following discussion, constraints on accelerations and velocities are omitted for clarity.

Let the robot pose as a function of time $g(t) \in SE(2)$ follow the mixed first- and second-order control equations of motion,

$$\dot{g} = g \cdot \begin{bmatrix} \nu \\ 0 \\ \omega \end{bmatrix} \quad \text{and} \quad \begin{aligned} \dot{\nu} &= u^1 \\ \dot{\omega} &= u^2 \end{aligned} \tag{7.1}$$

where ν is the forward velocity and ω is the angular velocity, both in the body frame. The signal $u = (u^1, u^2)^T$ coordinates are the forward and angular acceleration (in body frame).

This controller relies on the differential flatness of the robot kinematics to achieve exponential stabilization of a virtual point in front of the robot by λ. Define the λ-adjusted rotation matrix and angular velocity matrix to be

$$\mathbf{R}_\lambda = \mathbf{R} \cdot \mathrm{diag}(1, \lambda) \quad \text{and} \quad \hat{\boldsymbol{\omega}}(\lambda, \dot{\lambda}) = \begin{bmatrix} 0 & -\lambda\omega \\ \frac{1}{\lambda}\omega & \frac{\dot{\lambda}}{\lambda} \end{bmatrix}, \tag{7.2}$$

where $\mathbf{R}$ is the orientation from g. For e_1 the unit body $\hat{x}$-vector in the world frame, the trajectory tracking control is

$$\begin{aligned} u =& c_p {\mathbf{R}_\lambda}^{-1} \left(d^* - d - \lambda * \mathbf{R} e_1\right) + c_d \left({\mathbf{R}_\lambda}^{-1} \dot{d^*} - V\right) \\ & - c_d \dot{\lambda} e_1 - \hat{\boldsymbol{\omega}}(\lambda, \dot{\lambda}) V - (\hat{\boldsymbol{\omega}}(\lambda, \dot{\lambda}) - c_\lambda \mathbf{I}) \dot{\lambda} e_1, \end{aligned} \tag{7.3}$$

where c_p, c_d, c_λ are feedback gains and $V = \begin{bmatrix} \nu \,|\, \omega \end{bmatrix}^T$. The additional offset dynamics are

$$\dot{\lambda} = -c_\lambda(\lambda - \epsilon), \text{ where } \lambda(0) > \epsilon > 0, \; c_\lambda > 0. \tag{7.4}$$

The dynamical system represented by Eqs 7.1-7.4 yields a reference trajectory of robot

poses $g^*(t)$ and body velocity components $V^*(t)$ for tracking the desired path $d^*(t)$. The offset variable $\lambda^*(t)$ can be ignored.

The real time trajectory controller drives the robot to track the reference trajectory based on feedback of the robot's state (a $SE(2)$ substate of the $SE(3)$ state estimation). These control commands are:

$$\begin{aligned} \nu_{cmd} &= k_x * \widetilde{x} + \nu^* \\ w_{cmd} &= k_\theta * \widetilde{\theta} + k_y * \widetilde{y} + \omega^* \end{aligned} \tag{7.5}$$

where $[\widetilde{x}, \widetilde{y}, \widetilde{\theta}]^T \simeq \widetilde{g} = g^{-1}g^*$ is the relative pose error between the current state g and the desired state g^* in body frame. In the absence of error, the control signal is $V^*(t)$.

7.6 Experiments

This section describes a simulated closed-loop navigation environment for testing VI-SLAM systems using Gazebo/ROS. Experimental results show the proposed solution outperforms state-of-the-art VI-SLAM systems.

7.6.1 Simulation Setup and Baseline Methods

A virtual office world is created for robot navigation (Fig 7.5). The world is based on the floor-plan of an actual office, with texture-mapped surfaces. The walls are placed 1m above the ground plane since collision checking and path planning is outside the scope of this chapter. Within this world, the differential drive robot TurtleBot2 [171] maneuvers. A 30fps stereo camera with an 11cm baseline is mounted to the TurtleBot. An IMU is placed at the base of TurtleBot. Two commonly-used IMUs are simulated: a high-end ADIS16448 and a low-end MPU6000. Data streams from both the stereo camera and IMU are input to the VI-SLAM which then outputs $SE(3)$ state estimates. The position controller described previously uses the $SE(2)$ subspace of the $SE(3)$ estimate to drive the TurtleBot to follow the desired path.

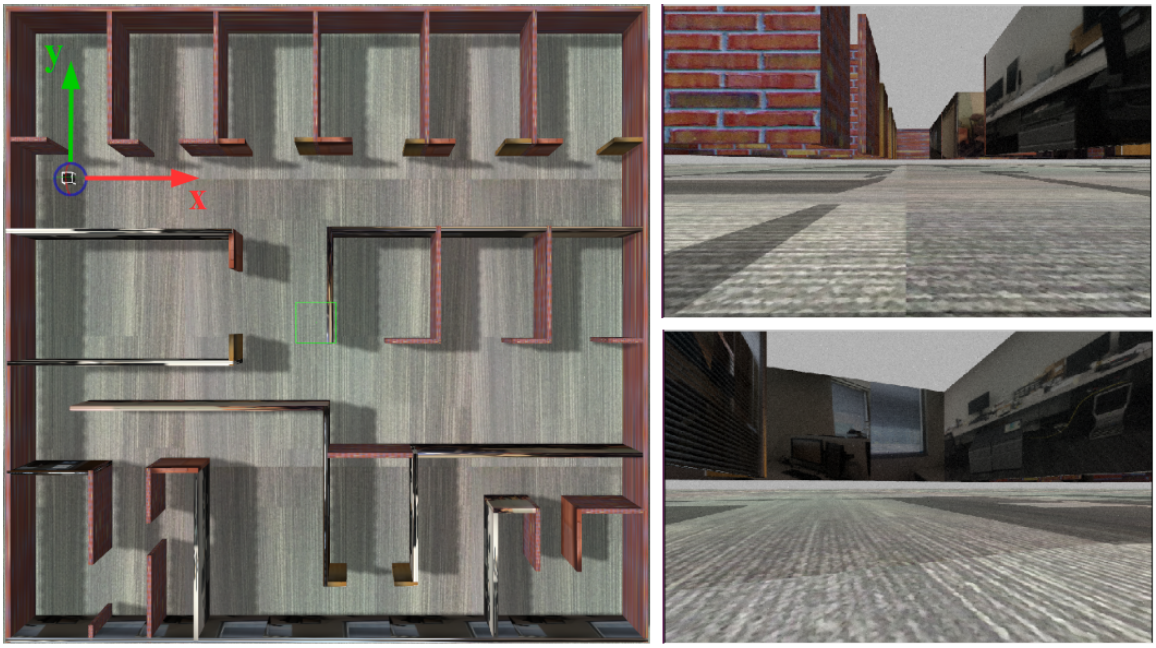


Figure 7.5: The virtual office world. **Left:** Top-down view. The robot starts at the top-left corner, facing the long corridor. **Right:** Example images captured by on-board stereo camera (left camera).

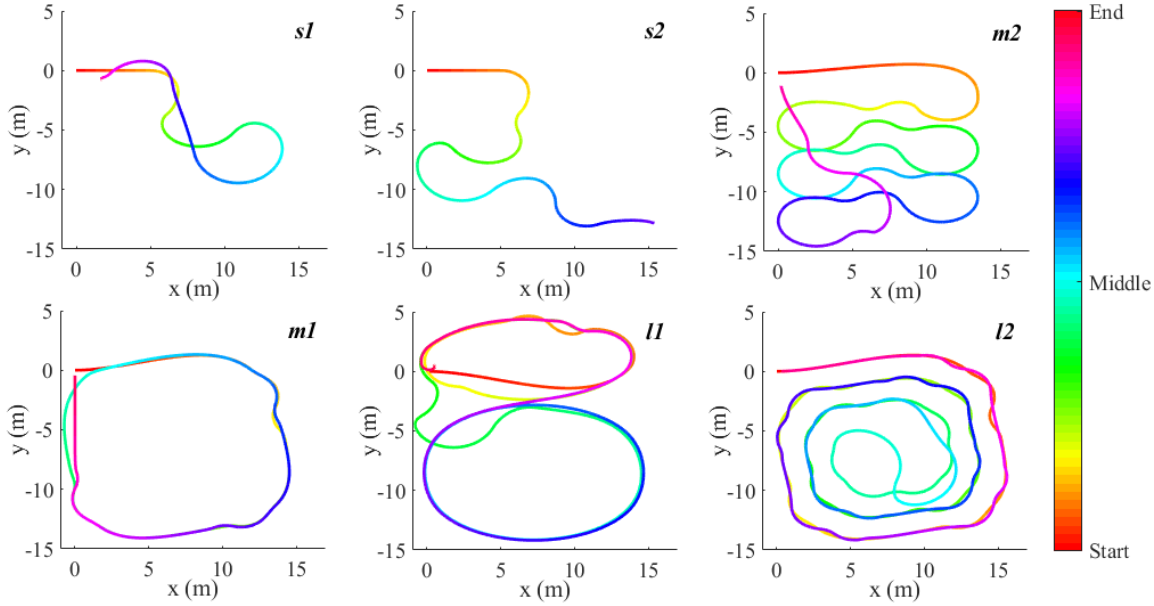


Figure 7.6: All 6 desired paths used in closed-loop navigation experiments. Each desired path is color-coded to show the direction of travel.

Six test paths were created for the closed-loop navigation experiments, each with different characteristics (Fig 7.6). The first 2 paths are relatively short (∼50m), with few to none re-visits. The 3rd and 4th paths are both of medium length (∼120m) with many to

few re-visits. The last 2 paths are long (∼240m) with many re-visits. All paths have the same start point for the robot, the origin of the world. Three desired linear velocities are tested: 0.5m/s, 1.0m/s, and 1.5m/s.

The methods tested were the following VI-SLAM systems:

1. *MSC*: MSCKF-VIO [10] is a tightly-coupled VIO system, with KLT-based front-end and MSCKF back-end. EKF-based sensor fusion [146] is used to densify the low-rate estimation output from MSC, before sending it to controller.

2. *VIF*: VINS-Fusion [156, 172] is a tightly-coupled VI-SLAM with KLT-based front-end and BA-based back-end (SWF). *VIF* has a large latency due to the SWF BA. It does provide a low-latency, high-rate IMU propagation signal, which is sent to controller.

3. *SVO*: SVO + MSF [173]. A loosely-coupled VIO system with SVO [6], a light-weight direct method.

4. *ORB*: ORB-SLAM + MSF. ORB-SLAM [4] has a feature-based front-end and BA-based back-end. *ORB* is computationally costly, so the latency is large.

5. *GF*: ORB-SLAM with GF front-end + MSF. As described in Chapter 3, a loosely-coupled *ORB*, with the low-latency Good Feature pose tracking.

6. *GF+GG*: ORB-SLAM with GF front-end and GG back-end + MSF. The combination of Good Feature pose tracking (in Chapter 3) and cost-efficient Good Graph mapping (in Chapter 6).

The methods with "+ MSF" are loosely-coupled systems with fusion via MSF [146]. All 6 VI-SLAM systems are set to reasonably good parameters found by parameter sweep, the raw data of which is released online [1]. For each test configuration (desired path, desired

[1] `https://github.com/ivalab/FullResults_ClosedNav`

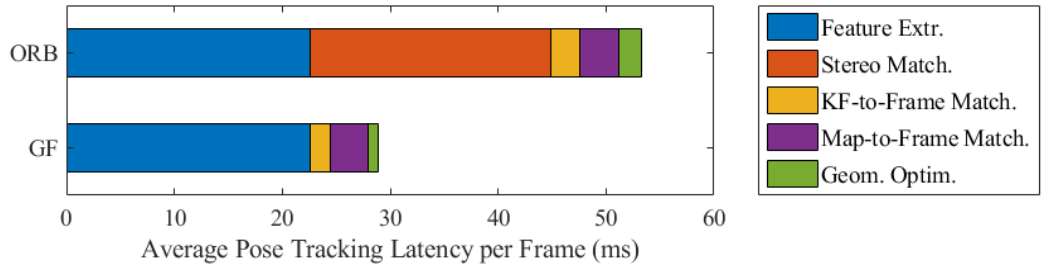


Figure 7.7: Pose tracking profiling of *ORB* and *GF*.

linear velocity, VI-SLAM method, and IMU), we repeat each closed-loop navigation run 5 times.

7.6.2 Simulation on Low-Power Laptop

An Intel Xeon E5-2680 CPU workstation (passmark score 1661 per thread) is chosen to conduct Gazebo simulation and graph rendering. The computations of closed-loop navigation, include VSLAM, visual-inertial fusion and feedback control, are conducted on a laptop with low-power Intel Core i7-8550U CPU. The total power consumption of the low-power navigation laptop is 15W. The computing speed of the low-power laptop can be quantified with passmark score, which is 2140 per thread. For reference, most published closed-loop navigation systems [151, 158, 154, 160, 10, 159, 161] employ an Intel NUC whose CPUs score between 1900-2300 per thread. The communication between simulating workstation and navigation laptop is based on Ethernet.

First, we show that the proposed system does indeed reduce the image to pose latency. Fig 7.7 provides the time breakdown of the pose tracking computation, generated by averaging the latency across all test runs (180 in total). Compared with *ORB*, *GF* removes the overhead of stereo matching. The latency of map-to-frame matching is similar for both, which includes some stereo matching for *GF*. The Good Feature Matching and Lazy Stereo modifications lead to a 50% reduction in latency.

Next, we review the high-end IMU (ADIS16448) outcomes. The actual trajectories traveled by the robot with 0.5m/s desired linear velocity are illustrated in Fig 7.8, read

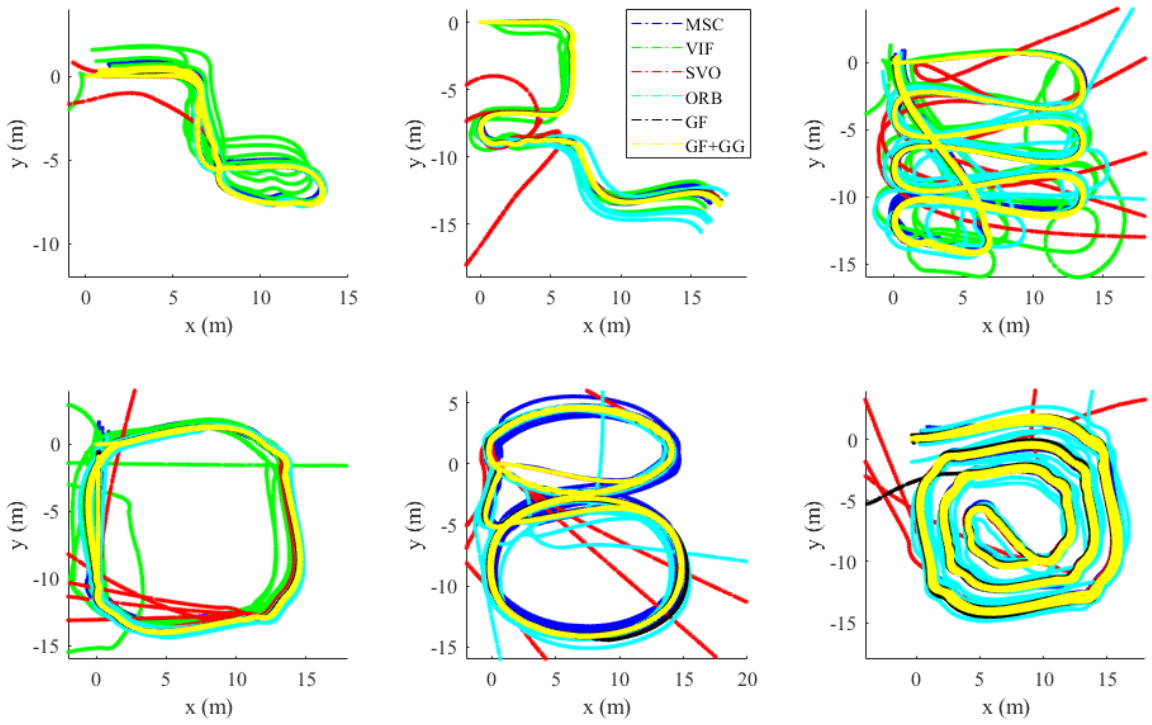


Figure 7.8: Trajectories the robot traveled for each desired path, color-coded by method. Desired velocity is 0.5m/s and IMU is simulated as a high-end ADIS16448. Navigation-related computations are conducted on a low-power laptop.

from left to right, top to bottom row. When following short and mid-term paths (plots 1-4), *VIF* and *SVO* have the largest tracking error (red + green). The trajectory of *ORB* gets off when the length of desired path is medium or long (plots 3-6). The performance of *MSC* (blue) also degrades when following long-term paths (plots 5-6). Trajectories of the two low-latency VI-SLAM, namely *GF* (black) and *GF+GG* (yellow) are consistent and accurate (to the desired path). The trajectories for the low-end MPU6000 have similar outcomes, as illustrated in Fig 7.9. Though *VIF* appears better, many runs leave the figure bounds. Both *SVO* and *ORB* lead to large errors for the mid-term and long-term paths. On medium and long paths (plot 3-6), trajectories of *GF* also deviate from the desired ones. The best path following performance seems to be achieved with two methods: the light-weight VIO system *MSC*, and the low-latency BA-based *GF+GG*.

Navigation performance is quantified in Tables 7.1 and 7.2. The navigation perfor-

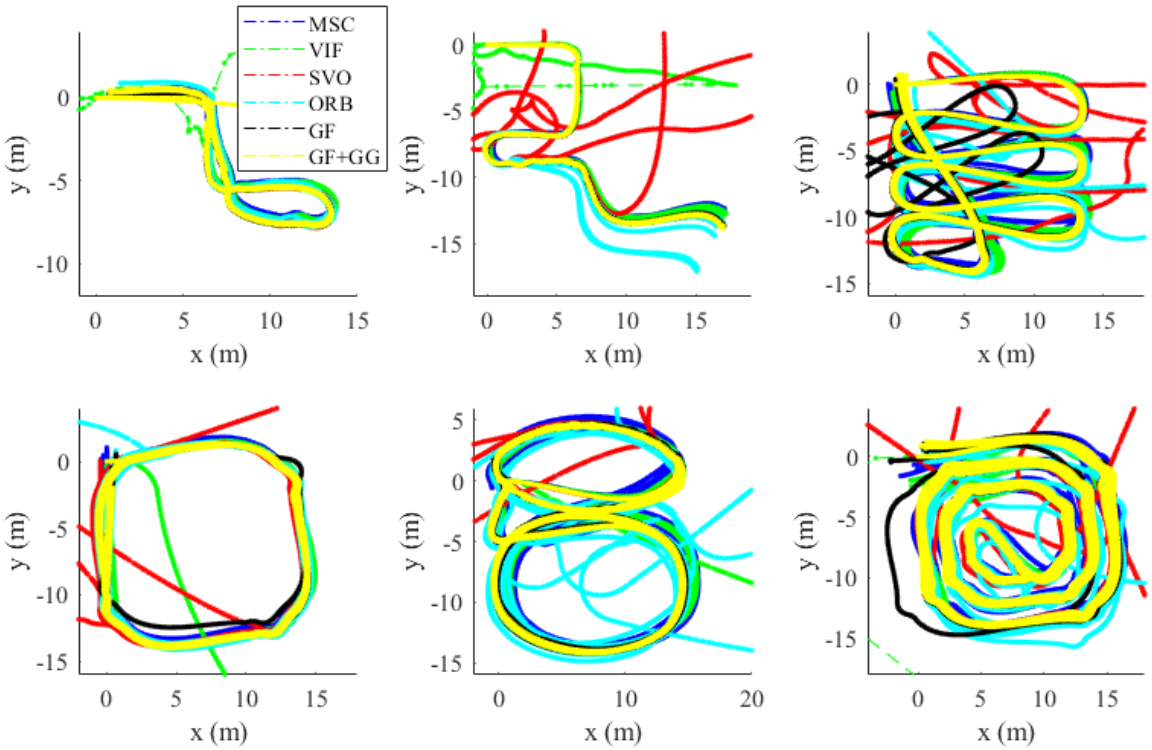


Figure 7.9: Trajectories the robot traveled for each desired path, color-coded by method. Desired velocity is 0.5m/s and IMU is simulated as a low-end MPU6000. Navigation-related computations are conducted on a low-power laptop.

mance metric is the root-mean-square (RMS) error between the desired path and the actual path, averaged over the 5-run repeats. Cases with average RMS over 10m are considered navigation failures and omitted (the dashes). For each sequence and desired linear velocity, the VI-SLAM system with the lowest navigation error is in bold. For reference, navigation performance with perfect visual estimation (no error or latency) is also presented under the column *GT* for each configuration.

According to Table 7.1, *SVO*, *VIF* and *ORB* fail under multiple configurations. The latency of visual estimation, as described earlier, contributes to these failures. Filter-based *MSC*, low-latency *GF* and *GF+GG* succeed in most configurations. However *MSC* either has slightly more track loss or worse RMSE when compared with BA-based *GF* and *GF+GG*. The performance of *GF+GG* is further improved over *GF* when the desired velocity is greater or equal to 1.0 m/s. From Table 7.2, the similar observation can be made

Table 7.1: Closed-loop Navigation Error (RMS; in m) on Laptop, with High-end IMU ADIS16448

	0.5m/s							**1.0m/s**							**1.5m/s**						
Seq.	GT	SVO	MSC	VIF	ORB	GF	GF+GG	GT	SVO	MSC	VIF	ORB	GF	GF+GG	GT	SVO	MSC	VIF	ORB	GF	GF+GG
s1	0.13	2.00	0.34	-	0.16	**0.12**	0.14	0.14	1.06	0.26	5.59	0.24	-	**0.14**	0.15	0.54	-	5.40	0.48	0.23	**0.21**
s2	0.12	-	0.35	0.64	0.82	0.12	**0.11**	0.11	8.49	0.45	-	0.17	0.12	**0.11**	0.13	1.52	0.39	7.80	0.36	0.19	**0.17**
m1	0.13	-	0.50	-	-	**0.18**	0.21	0.15	-	0.53	-	2.40	0.24	**0.23**	0.17	-	0.52	0.58	0.40	0.65	**0.26**
m2	0.13	-	0.62	-	0.32	**0.12**	0.14	0.14	-	0.67	0.88	-	0.20	**0.19**	0.16	-	0.51	-	0.82	**0.21**	0.23
l1	0.13	-	0.56	-	-	**0.22**	0.23	0.15	-	0.85	-	-	0.42	**0.33**	0.17	-	0.76	-	0.58	**0.33**	0.39
l2	0.13	-	0.58	-	5.29	-	**0.32**	0.15	-	0.87	-	-	**0.36**	0.46	0.17	-	-	-	0.82	0.44	**0.27**
Avg.	0.13	2.00	0.49	0.64	1.65	**0.15**	0.19	0.14	4.78	0.61	3.24	0.94	0.27	**0.24**	0.16	0.88	0.55	4.59	0.58	0.34	**0.26**

Table 7.2: Closed-loop Navigation Error (RMS; in m) on Laptop, with Low-end IMU MPU6000

	0.5m/s							**1.0m/s**							**1.5m/s**						
Seq.	GT	SVO	MSC	VIF	ORB	GF	GF+GG	GT	SVO	MSC	VIF	ORB	GF	GF+GG	GT	SVO	MSC	VIF	ORB	GF	GF+GG
s1	0.15	0.17	0.32	-	0.25	**0.13**	-	0.16	0.98	-	-	0.22	**0.17**	**0.17**	0.17	0.79	0.31	-	0.43	-	**0.20**
s2	0.16	-	0.34	-	1.18	**0.14**	0.17	0.13	-	0.41	-	0.18	**0.15**	0.19	0.15	5.32	0.36	-	0.43	0.25	**0.19**
m1	0.15	-	0.54	0.53	-	1.02	**0.33**	0.17	-	0.43	-	0.44	**0.21**	0.31	0.19	-	0.59	-	0.36	0.63	**0.32**
m2	0.16	-	0.43	-	-	0.36	**0.29**	0.19	-	0.39	-	-	**0.14**	0.48	0.20	-	0.51	-	1.02	**0.17**	**0.17**
l1	0.15	-	0.84	-	-	**0.36**	0.47	0.16	-	0.56	-	-	0.42	**0.34**	0.19	-	0.66	-	-	0.44	**0.33**
l2	0.16	-	0.63	-	-	0.56	**0.53**	0.17	-	0.62	-	-	**0.34**	0.49	0.19	-	-	-	-	**0.40**	0.42
Avg.	0.16	**0.17**	0.52	0.53	0.72	0.43	0.36	0.16	0.98	0.48	-	0.28	**0.24**	0.33	0.18	3.06	0.53	-	0.56	0.38	**0.27**

for the low-end IMU. Navigation performance degrades for most configurations. *SVO*, *VIF* and *ORB* still fail under multiple configurations. The performance of filter-based *MSC* is quite consistent; while *GF* and *GF+GG* are with lower RMSE, especially when the desired linear velocity is 1.0 or 1.5m/s.

To summarize, low-latency *GF* and *GF+GG* outperform the other 4 state-of-the-art VI-SLAM systems in a closed-loop navigation simulation, with *MSC* having next best performance. The navigation error of *GF+GG* is further reduced when the desired linear velocity is 1.0 or 1.5m/s. The improvement of navigation performance is significant in most configurations, with both high-end and low-end IMUs. Still, there are some cases that *GF* and *GF+GG* are prune to tack loss. A tighter integration of visual-inertial estimation is desired to improve the robustness of VI-SLAM state estimation.

7.7 Conclusion

This chapter tests two variants of latency-reduction VSLAM, *GF* and *GF+GG*, in closed-loop navigation. The relative performance of ORB-SLAM versus those of *GF* and *GF+GG* suggests a connection between visual estimation latency and navigation performance. In a comprehensive and repeatable simulated evaluation, the navigation performance of low-latency *GF* outperforms state-of-the-art direct and feature-based VI-SLAM systems. With *GG* enhancement, the navigation performance is further improved, especially in medium or high velocity configurations.

CHAPTER 8

CONCLUSION AND FUTURE RESEACH

This thesis investigates the applicability of VSLAM on target applications in robotics and AR. The performance-efficiency trade-off of VSLAM is significantly improved by leveraging theorems on submodular submatrix selection. Multiple algorithmic components of modern VSLAM system are investigated and improved in this thesis, each of which pushed the applicability of VSLAM forward a little bit. The key contributions are summarized:

- **Good Feature Matching: Low-Latency Front-End of Feature-based VSLAM**. The concept of submodular submatrix selection is introduced to a compute-intense module in the VSLAM front-end, i.e. map-to-frame feature matching. The feature selection problem is formulated and tackled with submodular submatrix selection. It is then combined with active feature matching, leading to a low-latency, performance guaranteed feature matching algorithm, dubbed Good Feature Matching.

- **Good Line Cutting: Accuracy Improvement of Line-Assisted VSLAM**. The idea of point feature selection is extended to line features. A specific property of lines, i.e. extending along a specific direction, is exploited to enable line feature refinement. The underlying optimization objective of line refinement is a convex optimization. An efficient, multi-start algorithm for generating sub-optimal solutions, dubbed Good Line Cutting, is described and evaluated.

- **Map Hashing: Appearance-Enhanced Compact Local Map of Feature-based VSLAM**. An appearance-based enhancement is developed to construct, populate, and query the local map. Local map is a critical accuracy-improving sub-component of the VSLAM front-end. An efficient hashing technique is applied to store and query appearance prior. Furthermore, submodular submatrix selection provides a means to

reduce the quantity of hash queries through active, online table selection, thereby reducing the overhead of local map construction.

- **Good Graph Selection: Cost-Effective, Budget-Aware Bundle Adjustment in VSLAM**. The concept of submodular submatrix selection is introduced to general BA problem, which is frequently solved in the BA-based VSLAM back-end. A novel, rigorous method to determine the state subset in BA with strong performance guarantees is proposed, dubbed Good Graph Selection. Furthermore, we explore the potential of budget-awareness to determine the size of Good Graph on-the-fly.

- **Closed-Loop Navigation with Robust, Low-Latency Visual Inertial SLAM**. Two algorithmic improvements described in this thesis, namely the Good Feature Matching (Chapter 3) and the Good Graph Selection (Chapter 6), are applied to a visual-inertial fusion framework. The accurate and low-latency of described visual SLAM method is revealed in the closed-loop navigation scenario investigated. A reproducible benchmarking simulation for closed-loop VSLAM evaluation is presented, which supports comprehensive evaluation of VSLAM in closed-loop navigation tasks.

There are several research directions for the future work. From the algorithmic perspective, the combination of feature-based and direct front-end is promising: feature matching provides long-baseline associations that are beneficial to the overall accuracy of VSLAM, while direct measurements serve as the short-term constraints for the robustness of VSLAM. The combination of Good Graph and incremental BA solver could further improve the cost-efficiency of BA-based VSLAM back-end. Deep integration of dedicated hardwares such as FPGA and the efficient algorithms such as Good Feature is an interesting topic as well, which could have great impact to applications with computational limits. Finally, semantic clues extracted from visual input using deep learning algorithms (e.g. R-CNN) could be exploited to further improve the robustness of VSLAM. Studies on combining VSLAM and deep learning, however, is still under-going.

REFERENCES

[1] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual SLAM: Applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.

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

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

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

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

[6] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.

[7] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 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 International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

[9] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[10] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[11] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," *IEEE International Conference on Robotics and Automation*, vol. 10, p. 20, 2018.

[12] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[13] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *IEEE International Conference on Computer Vision*, 2011, pp. 2548–2555.

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

[15] P Vandergheynst, R. Ortiz, and A. Alahi, "FREAK: Fast retina keypoint," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 510–517.

[16] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[17] P. Smith, I. Reid, and A. Davison, "Real-time monocular SLAM with straight lines," in *British Machine Vision Conference*, 2006, pp. 17–26.

[18] A. P. Gee and W. Mayol-Cuevas, "Real-time model-based SLAM using line segments," in *International Symposium on Visual Computing*, 2006, pp. 354–363.

[19] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A line segment detector," *Image Processing On Line*, vol. 2, pp. 35–55, 2012.

[20] L. Zhang and R. Koch, "Line matching using appearance similarities and geometric constraints," *Pattern Recognition*, vol. 7476, pp. 236–245, 2012.

[21] R. Gomez-Ojeda and J. Gonzalez-Jimenez, "Robust stereo visual odometry through a probabilistic combination of points and line segments," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 2521–2526.

[22] B. Přibyl, P. Zemčík, and M. Čadík, "Camera pose estimation from lines using Plücker coordinates," in *British Machine Vision Conference*, 2015, pp. 1–12.

[23] 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 Transactions on Robotics*, 2019.

[24] F. Zheng, G. Tsai, Z. Zhang, S. Liu, C.-C. Chu, and H. Hu, "Trifo-VIO: Robust and efficient stereo visual inertial odometry using points and lines," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 3686–3693.

[25] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[26] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 431–437.

[27] M. Quigley, K. Mohta, S. S. Shivakumar, M. Watterson, Y. Mulgaonkar, M. Arguedas, K. Sun, S. Liu, B. Pfrommer, V. Kumar, *et al.*, "The open vision computer: An integrated sensing and compute system for mobile robots," *ArXiv preprint arXiv:1809.07674*, 2018.

[28] R. Liu, J. Yang, Y. Chen, and W. Zhao, "ESLAM: An energy-efficient accelerator for real-time ORB-SLAM on FPGA platform," in *Design Automation Conference*, 2019, 193:1–193:6.

[29] C. Tomasi and T. Kanade, "Detection and tracking of point features," *Tech. Rep. CMU-CS-91-132, School of Computer Science, Carnegie Mellon Univ. Pittsburgh*, 1991.

[30] H. Zhang, "Quantitative evaluation of feature extractors for visual SLAM," in *Canadian Conference on Computer and Robot Vision*, 2007, pp. 157–164.

[31] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[32] K. Mohta, K. Sun, S. Liu, M. Watterson, B. Pfrommer, J. Svacha, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Experiments in fast, autonomous, GPS-denied quadrotor flight," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 7832–7839.

[33] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback,"

The International Journal of Robotics Research, vol. 36, no. 10, pp. 1053–1072, 2017.

[34] M. K. Paul, K. Wu, J. A. Hesch, E. D. Nerurkar, and S. I. Roumeliotis, "A comparative analysis of tightly-coupled monocular, binocular, and stereo VINS," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 165–172.

[35] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2878–2885, 2018.

[36] D. Schubert, N. Demmel, V. Usenko, J. Stuckler, and D. Cremers, "Direct sparse odometry with rolling shutter," in *European Conference on Computer Vision*, Springer, 2018, pp. 682–697.

[37] S. Thrun and Y. Liu, "Multi-robot SLAM with sparse extended information filers," in *The International Symposium Robotics Research*, 2005, pp. 254–266.

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

[39] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572.

[40] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[41] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment-a modern synthesis," in *International Workshop on Vision Algorithms*, 1999, pp. 298–372.

[42] M. I. Lourakis and A. A. Argyros, "SBA: A software package for generic sparse bundle adjustment," *ACM Transactions on Mathematical Software*, vol. 36, no. 1, p. 2, 2009.

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

[44] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "ISAM2: Incremental smoothing and mapping using the bayes tree," *International Journal of Robotics Research*, vol. 31, pp. 217–236, 2012.

[45] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao, "ICE-BA: Incremental, consistent and efficient bundle adjustment for visual-inertial SLAM," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1974–1982.

[46] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[47] S. Agarwal, K. Mierle, and et al., *Ceres solver*, `http://ceres-solver.org`.

[48] V. Ila, L. Polok, M. Solony, and P. Svoboda, "SLAM++-a highly efficient and temporally scalable incremental SLAM framework," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 210–230, 2017.

[49] L. Polok, V. Ila, M. Solony, P. Smrz, and P. Zemcik, "Incremental block Cholesky factorization for nonlinear least squares in robotics.," in *Robotics: Science and Systems*, 2013, pp. 328–336.

[50] V. Ila, L. Polok, M. Solony, and K. Istenic, "Fast incremental bundle adjustment with covariance recovery," in *IEEE International Conference on 3D Vision*, 2017, pp. 175–184.

[51] A. J. Davison, "FutureMapping: The computational structure of spatial AI systems," *ArXiv preprint arXiv:1803.11288*, 2018.

[52] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[53] M. W. Mahoney *et al.*, "Randomized algorithms for matrices and data," *Foundations and Trends® in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.

[54] A. Miller, *Subset selection in regression*. Chapman and Hall/CRC, 2002.

[55] C. Couvreur and Y. Bresler, "On the optimality of the backward greedy algorithm for the subset selection problem," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 3, pp. 797–808, 2000.

[56] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[57] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, 10. Springer series in statistics New York, 2001, vol. 1.

[58] R. Hocking and R. Leslie, "Selection of the best subset in regression analysis," *Technometrics*, vol. 9, no. 4, pp. 531–540, 1967.

[59] G. M. Furnival and R. W. Wilson, "Regressions by leaps and bounds," *Technometrics*, vol. 16, no. 4, pp. 499–511, 1974.

[60] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, no. 1, pp. 80–86, 2000.

[61] R. Tibshirani, "Regression shrinkage and selection via the lasso: A retrospective," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 3, pp. 273–282, 2011.

[62] Y. Zhao and P. Vela, "Good feature selection for least squares pose optimization in VO/VSLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 3569–3574.

[63] Y. Zhao and P. A. Vela, "Good feature matching: Towards accurate, robust VO/VSLAM with low latency," *Submitted to IEEE Transactions on Robotics*, 2019.

[64] Y. Zhao and P. A. Vela, "Good line cutting: Towards accurate pose tracking of line-assisted VO/VSLAM," in *European Conference on Computer Vision*, Springer, 2018, pp. 516–531.

[65] Y. Zhao, W. Ye, and P. Vela, "Low-latency visual SLAM with appearance-enhanced local map building," in *IEEE International Conference on Robotics and Automation*, 2019.

[66] Y. Zhao, J. S. Smith, and P. A. Vela, "Good graph to optimize: Cost-effective, budget-aware bundle adjustment in visual SLAM," *Submitted to IEEE Transactions on Robotics*, 2020.

[67] Y. Zhao, J. S. Smith, and P. A. Vela, "Robust, low-latency, feature-based visual-inertial SLAM for improved closed-loop navigation," *Submitted to IEEE Conference on Robotics and Automation*, 2020.

[68] F. Dellaert, M. Kaess, *et al.*, "Factor graphs for robot perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[69] M. Gu and S. C. Eisenstat, "Efficient algorithms for computing a strong rank-revealing QR factorization," *SIAM Journal on Scientific Computing*, vol. 17, no. 4, pp. 848–869, 1996.

[70] C. Boutsidis, M. W. Mahoney, and P. Drineas, "An improved approximation algorithm for the column subset selection problem," in *ACM-SIAM Symposium on Discrete Algorithms*, 2009, pp. 968–977.

[71] T. H. Summers, F. L. Cortesi, and J. Lygeros, "On submodularity and controllability in complex dynamical networks," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 1, pp. 91–101, 2016.

[72] S. T. Jawaid and S. L. Smith, "Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems," *Automatica*, vol. 61, pp. 282–288, 2015.

[73] M. Shamaiah, S. Banerjee, and H. Vikalo, "Greedy sensor selection: Leveraging submodularity," in *IEEE Conference on Decision and Control*, 2010, pp. 2572–2577.

[74] L. Carlone and S. Karaman, "Attention and anticipation in fast visual-inertial navigation," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 1–20, 2019.

[75] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[76] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization techniques*, Springer, 1978, pp. 234–243.

[77] R. A. Horn, R. A. Horn, and C. R. Johnson, *Matrix analysis*. Cambridge university press, 1990.

[78] P. Drineas, M. W. Mahoney, and S Muthukrishnan, "Relative-error CUR matrix decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 844–881, 2008.

[79] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff, "Fast approximation of matrix coherence and statistical leverage," *Journal of Machine Learning Research*, vol. 13, no. Dec, pp. 3475–3506, 2012.

[80] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, "Near-optimal column-based matrix reconstruction," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 687–717, 2014.

[81] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause, "Lazier than lazy greedy.," in *AAAI Conference on Artificial Intelligence*, 2015, pp. 1812–1818.

[82] A. Hassidim and Y. Singer, "Robust guarantees of stochastic greedy algorithms," in *International Conference on Machine Learning*, 2017, pp. 1424–1432.

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

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

[85] I. Cvišić and I. Petrović, "Stereo odometry based on careful feature selection and tracking," in *European Conference on Mobile Robots*, 2015, pp. 1–6.

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

[87] ——, "Good features to track for visual SLAM," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1373–1382.

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

[89] P. Sala, R. Sim, A. Shokoufandeh, and S. Dickinson, "Landmark selection for vision-based navigation," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 334–349, 2006.

[90] Z. Shi, Z. Liu, X. Wu, and W. Xu, "Feature selection for reliable data association in visual SLAM," *Machine Vision and Applications*, pp. 1–16, 2013.

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

[92] S. Zhang, L. Xie, and M. D. Adams, "Entropy based feature selection scheme for real time simultaneous localization and map building," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1175–1180.

[93] R. Lerner, E. Rivlin, and I. Shimshoni, "Landmark selection for task-oriented navigation," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 494–505, 2007.

[94] F. A. Cheein, G. Scaglia, F. di Sciasio, and R. Carelli, "Feature selection criteria for real time EKF-SLAM algorithm," *International Journal of Advanced Robotic Systems*, vol. 6, no. 3, p. 21, 2009.

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

[96] M. Chli and A. J. Davison, "Active matching," in *European Conference on Computer Vision*, Springer, 2008, pp. 72–85.

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

[98] C. Zach, "Robust bundle adjustment revisited," in *European Conference on Computer Vision*, Springer, 2014, pp. 772–787.

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

[100] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.

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

[102] J. Sturm, W. Burgard, and D. Cremers, "Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark," in *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems*, 2012.

[103] Y. Lu and D. Song, "Robust RGB-D odometry using point and line features," in *IEEE International Conference on Computer Vision*, 2015, pp. 3934–3942.

[104] A. Vakhitov, J. Funke, and F. Moreno-Noguer, "Accurate and linear time pose estimation from points and lines," in *European Conference on Computer Vision*, Springer, 2016, pp. 583–599.

[105] S. Yang and S. Scherer, "Direct monocular odometry using points and lines," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 3871–3877.

[106] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *European Conference on Computer Vision*, Springer, 2008, pp. 802–815.

[107] T. Koletschka, L. Puig, and K. Daniilidis, "MEVO: Multi-environment stereo visual odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4981–4988.

[108] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "PL-SLAM: Real-time monocular visual SLAM with points and lines," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 4503–4508.

[109] R. Gomez-Ojeda, J. Briales, and J. Gonzalez-Jimenez, "PL-SVO: Semi-direct monocular visual odometry by combining points and line segments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 4211–4216.

[110] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-D line-based map using stereo SLAM," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1364–1377, 2015.

[111] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban, "An interior algorithm for nonlinear optimization that combines line search and trust region steps," *Mathematical Programming*, vol. 107, no. 3, pp. 391–408, 2006.

[112] M. J. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," in *Numerical Analysis*, 1978, pp. 144–157.

[113] R. Mur-Artal and J. D. Tards, "Visual-inertial monocular SLAM with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[114] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 5303–5310.

[115] C. Mei, G. Sibley, and P. Newman, "Closing loops without places," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3738–3744.

[116] H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *IEEE International Conference on Computer Vision*, 2011, pp. 2352–2359.

[117] M. Bürki, I. Gilitschenski, E. Stumm, R. Siegwart, and J. Nieto, "Appearance-based landmark selection for efficient long-term visual localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 4137–4143.

[118] M. A. Nitsche, G. I. Castro, T. Pire, T. Fischer, and P. De Cristóforis, "Constrained-covisibility marginalization for efficient on-board stereo SLAM," in *European Conference on Mobile Robots*, Springer, 2017, pp. 1–6.

[119] D. Greene, M. Parnas, and F. Yao, "Multi-index hashing for information retrieval," in *35th Annual Symposium on Foundations of Computer Science*, IEEE, 1994, pp. 722–731.

[120] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular SLAM with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[121] Y. Li, N. Snavely, and D. P. Huttenlocher, "Location recognition using prioritized feature matching," in *European Conference on Computer Vision*, Springer, 2010, pp. 791–804.

[122] S. Choudhary and P. Narayanan, "Visibility probability structure from SfM datasets and applications," in *European Conference on Computer Vision*, Springer, 2012, pp. 130–143.

[123] T. Sattler, B. Leibe, and L. Kobbelt, "Fast image-based localization using direct 2D-to-3D matching," in *International Conference on Computer Vision*, 2011, pp. 667–674.

[124] H. Lim, S. N. Sinha, M. F. Cohen, and M. Uyttendaele, "Real-time image-based 6-DoF localization in large-scale environments," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1043–1050.

[125] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[126] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *European Conference on Computer Vision*, Springer, 2006, pp. 404–417.

[127] T. Sattler, B. Leibe, and L. Kobbelt, "Improving image-based localization by active correspondence search," in *European Conference on Computer Vision*, Springer, 2012, pp. 752–765.

[128] ——, "Efficient & effective prioritized matching for large-scale image-based localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1744–1756, 2017.

[129] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization.," in *Robotics: Science and Systems*, 2015.

[130] Y. Feng, L. Fan, and Y. Wu, "Fast localization in large-scale environments using supervised indexing of binary features," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 343–358, 2016.

[131] J. Cheng, C. Leng, J. Wu, H. Cui, and H. Lu, "Fast and accurate image matching with cascade hashing for 3D reconstruction," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1–8.

[132] N.-T. Tran, D.-K. Le Tan, A.-D. Doan, T.-T. Do, T.-A. Bui, M. Tan, and N.-M. Cheung, "On-device scalable image-based localization via prioritized cascade search and fast one-many RANSAC," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1675–1690, 2019.

[133] J. Straub, S. Hilsenbeck, G. Schroth, R. Huitl, A. Möller, and E. Steinbach, "Fast relocalization for visual odometry using binary features," in *IEEE International Conference on Image Processing*, 2013, pp. 2548–2552.

[134] L. Paulevé, H. Jégou, and L. Amsaleg, "Locality sensitive hashing: A comparison of hash function types and querying mechanisms," *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1348–1358, 2010.

[135] L. Han and L. Fang, "MILD: Multi-index hashing for appearance based loop closure detection," in *IEEE International Conference on Multimedia and Expo*, 2017, pp. 139–144.

[136] R. L. Graham, D. E. Knuth, O. Patashnik, and S. Liu, "Concrete mathematics: A foundation for computer science," *Computers in Physics*, vol. 3, no. 5, pp. 106–107, 1989.

[137] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The New College vision and laser data set," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 595–599, 2009.

[138] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[139] X. Gao, R. Wang, N. Demmel, and D. Cremers, "LDSO: Direct sparse odometry with loop closure," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2198–2204, 2018.

[140] S. Qin, Q. Liu, B. Yu, and S. Liu, "PI-BA: Bundle adjustment acceleration on embedded FPGAs with co-observation optimization," *ArXiv preprint arXiv:1905.02373*, 2019.

[141] M. A. Osborne, "Bayesian Gaussian processes for sequential prediction, optimisation and quadrature," PhD thesis, Oxford University, UK, 2010.

[142] J. Sola, "Quaternion kinematics for the error-state Kalman filter," *ArXiv preprint arXiv:1711.02508*, 2017.

[143] W. Ye, Y. Zhao, and P. A. Vela, "Characterizing SLAM benchmarks and methods for the robust perception age," *Workshop on Dataset Generation and Benchmarking*

of SLAM Algorithms for Robotics and VR/AR at the IEEE International Conference on Robotics and Automation, 2019.

[144] E. Blem, J. Menon, and K. Sankaralingam, "A detailed analysis of contemporary ARM and x86 architectures," *UW-Madison Technical Report*, 2013.

[145] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset," in *IEEE International Conference on Robotics and Automation*, 2019.

[146] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3923–3929.

[147] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3946–3952.

[148] Y. Watanabe, C. Lesire, A. Piquereau, P. Fabiani, M. Sanfourche, and G. Le Besnerais, "System development and flight experiment of vision-based simultaneous navigation and tracking," in *AIAA Infotech@ Aerospace*, 2010, p. 3422.

[149] G. Chowdhary, E. N. Johnson, D. Magree, A. Wu, and A. Shein, "GPS-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft," *Journal of Field Robotics*, vol. 30, no. 3, pp. 415–438, 2013.

[150] J. Engel, J. Sturm, and D. Cremers, "Accurate figure flying with a quadrocopter using onboard visual and inertial sensing," *IMU*, vol. 320, no. 240,

[151] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, *et al.*, "Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in GPS-denied environments," *IEEE Robotics & Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[152] A. Harmat, M. Trentini, and I. Sharf, "Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 2, pp. 291–317, 2015.

[153] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2017.

[154] S. Paschall and J. Rose, "Fast, lightweight autonomy through an unknown cluttered environment," in *IEEE Aerospace Conference*, 2017, pp. 1–8.

[155] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *Journal of Field Robotics*, vol. 27, no. 5, pp. 587–608, 2010.

[156] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," *ArXiv preprint arXiv:1901.03638*, 2019.

[157] F. Blochliger, M. Fehr, M. Dymczyk, T. Schneider, and R. Siegwart, "Topomap: Topological mapping and navigation based on visual SLAM maps," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 1–9.

[158] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard MAVs in unknown environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 1872–1878.

[159] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *Journal of Field Robotics*, vol. 35, no. 1, pp. 23–51, 2018.

[160] I. Cvišic, J. Cesic, I. Markovic, and I. Petrovic, "Soft-SLAM: Computationally efficient stereo visual SLAM for autonomous UAVs," *Journal of Field Robotics*, 2017.

[161] H. Oleynikova, Z. Taylor, A. Millane, R. Siegwart, and J. Nieto, "A complete system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments," *ArXiv preprint arXiv:1812.03892*, 2018.

[162] C. Papachristos, S. Khattak, and K. Alexis, "Autonomous exploration of visually-degraded environments using aerial robots," in *IEEE International Conference on Unmanned Aircraft Systems*, 2017, pp. 775–780.

[163] S. Saeedi, B. Bodin, H. Wagstaff, A. Nisbet, L. Nardi, J. Mawer, N. Melot, O. Palomar, E. Vespa, T. Spink, *et al.*, "Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality," *Proceedings of the IEEE*, no. 99, pp. 1–20, 2018.

[164] A. Antonini, W. Guerra, V. Murali, T. Sayre-McCord, and S. Karaman, "The blackbird dataset: A large-scale dataset for UAV perception in aggressive flight," in *International Symposium on Experimental Robotics*, 2018.

[165] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger, "Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset," in *British Machine Vision Conference*, 2018.

[166] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2004, pp. 2149–2154.

[167] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS: A modular Gazebo MAV simulator framework," in *Robot Operating System*, Springer, 2016, pp. 595–625.

[168] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, Springer, 2018, pp. 621–635.

[169] T. Sayre-McCord, W. Guerra, A. Antonini, J. Arneberg, A. Brown, G. Cavalheiro, Y. Fang, A. Gorodetsky, D. McCoy, S. Quilter, *et al.*, "Visual-inertial navigation algorithm development using photorealistic camera simulation in the loop," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 2566–2573.

[170] R. Olfati-Saber, "Near-identity diffeomorphisms and exponential epsi-tracking and epsi-stabilization of first-order nonholonomic se(2) vehicles," in *IEEE American Control Conference*, vol. 6, 2002, pp. 4690–4695.

[171] W. Garage, "Turtlebot," *Website: Https://www.turtlebot.com/turtlebot2/*, pp. 11–25, 2011.

[172] T. Qin, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," *ArXiv preprint arXiv:1901.03642*, 2019.

[173] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle," *Journal of Field Robotics*, vol. 33, no. 4, pp. 431–450, 2016.