UNIVERSITY OF CALIFORNIA

Los Angeles

Fast and Adaptive Geometric Robot Perception

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical and Computer Engineering

by

Kenny Jieyou Chen

2023

ABSTRACT OF THE DISSERTATION

Fast and Adaptive Geometric Robot Perception

by

Kenny Jieyou Chen
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2023
Professor Brett Thomas Lopez, Co-Chair
Professor Jonathan Chau-Yan Kao, Co-Chair

Mobile robots rely on state estimation and mapping to perceive, plan, and navigate in the real-world, but they face significant challenges when operating in unstructured environments. Factors such as low lighting, the presence of particulates (e.g., dust or fog), uneven terrain, and other environmental disturbances can severely affect their ability to localize and map its surrounding environment—leading to errors in the robot's movements, inability to complete tasks, and potentially even causing damage to itself or its surroundings. While current state-of-the-art algorithms may work well in controlled environments, they quickly break down in unstructured scenarios due to their brittle architecture, strong environmental assumptions, and high computational complexity, limiting their applicability to the real-world. To this end, this research aims to address these limitations and proposes several novel methods for fast and reliable, domain-agnostic geometric perception through innovative algorithmic design grounded by first principles. Towards this, through three novel algorithms which leverage precise depth measurements from LiDAR technology, we propose several unique algorithmic innovations which increase localization accuracy, computational speed, and overall

operational reliability. The first algorithm introduces a lightweight LiDAR odometry (LO) solution that enables the use of dense point clouds for fast and accurate localization with an adaptive keyframing approach and data structure recycling. The second proposes a new condensed LiDAR-inertial odometry (LIO) architecture with a fast coarse-to-fine method for continuous-time motion correction, providing a technique for parallelizable point-wise deskewing with a constant jerk and angular acceleration motion model. In the third, we present a reliable LiDAR SLAM algorithm that prioritizes operational reliability and real-world efficacy by strategically placing proactive safe-guards against common failure points in both the front-end and back-end subsystems. The perspectives gained from this dissertation provide better insight into developing a general-purpose perception framework for autonomous mobile robots operating in a diverse set of environments in-the-wild.

The dissertation of Kenny Jieyou Chen is approved.

Jason L. Speyer

Sriram Narasimhan

Jonathan Chau-Yan Kao, Committee Co-Chair

Brett Thomas Lopez, Committee Co-Chair

University of California, Los Angeles

2023

Just keep swimming

Dory, from *Finding Nemo*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I'd like to extend my sincerest gratitude to the members of my committee, Brett Lopez, Jonathan Kao, Sriram Narasimhan, and Jason Speyer. Your critical insights, thoughtful discussions, and probing questions have all played a vital role in refining my work and in my development as a researcher.

I would like to specifically extend my deepest appreciation to my advisor Brett Lopez. Words cannot express how grateful I am for you and for taking me under your wing and believing in me and my potential when others overlooked me. Your intellectual rigor and attention to detail have set a high standard that I continually strive to meet, and your insightful feedback has not only improved the quality of my work, but has also honed my critical thinking skills and has made me a better researcher. I owe these amazing past years to you and am so grateful to have learned from you. Beyond academia though, thank you for always being so genuine and for always looking out for me, my career goals, and my well-being. And finally, thank you for being a great friend that I can always trust in.

I'd like to also express my heartfelt gratitude to my parents for their unwavering love and support. Mom, your understanding and empathy have created a safe space where I could freely express my both my triumphs and frustrations, and your comforting words have given me the strength to persevere when the weight of academic pressure seemed almost too overwhelming. Dad, you have imparted invaluable life lessons that have shaped my character and taught me the importance of integrity, resilience, and compassion, and I know that the qualities you have instilled in me will continue to shape my path long after my academic journey. And thank you, to you both, for sacrificing the comfort of your hometown in China and proximity to your family to immigrate to the States with nothing but each other and a leap of faith. In doing so, you provided me with the tremendous opportunity to pursue my dreams, and I hope I have made you two proud and will continue doing so in the future.

To my brother Kellen Chen, I consider myself incredibly fortunate to have you as my

older brother and my closest friend, and you have had a profound influence on my life both on the way I think and as my best friend and mentor in navigating the complexities of life. Your wisdom has helped me grow tremendously as a person, and you have taught me the importance of embracing the journey and nurturing the relationships that bring me happiness. Thanks for always being there for me for literally my entire life, and I'm so thankful that we're so close and that we constantly call each other to catch up and talk about everything, even when we're in different parts of the country.

To my incredible partner Emily Zhang, thank you for all your support and the countless ways you keep me sane and happy. You bring me incredible happiness and joy, and you have taught me the importance of finding balance—especially when I get too obsessive over my research. From hiking ten miles on a Saturday morning to going on foodie adventures around LA and trying a variety of tasty dishes, it's always so much fun going on adventures with you. I can't wait to see what's in store for us next.

Finally, I am immensely grateful to all my current and former labmates, including Aaron Sabu, Alex Zhen, Cat Nystrom, David Thorne, Helene Levy, Jacob Sayono, James Tseng, Ryan Nemiroff, Sam Gessow, Sarah Enayati, and Yanlong Ma from the VECTR lab, and Alexie Pogue, Amir Omidfar, Jasper Liu, Pehuen Moure, Tsang-Kai Chang, Wenzhong Yan, Zhaoliang Zheng, and Zida Wu from the LEMUR lab. I am so thankful to have had such wonderful and fun labmates to hangout with both inside and outside of the lab. A special shoutout to my former colleagues at NASA JPL's Team CoSTAR, including Ali Agha, Amanda Bouman, Antoni Rosinol, David Fan, Kyle Coble, Luca Carlone, Micah Corah, Patrick Spieler, Seyed Fakoorian, and Thomas Touma, for the wonderful friendships we've built during our participation in the DARPA SubT Challenge. And thank you, from the bottom of my heart, to all my friends, coworkers, and the countless individuals who have played a significant role in shaping the person that I am today, including Aashish Patel, Abdullah Choudhry, Alex Lin, Alex Schperberg, Alex Thoms, Alexandre Binet, Ankur Mehta, Duy Tran, Eric Peltola, Erin Maher, Ethan Co, Flora Zhang, Gabe Musen, Jeanette

Nguyen, Jonathan Bunton, Jonathan Chen, Julie Vu, Justin Lee, Kevin Kawabata, Kevin Le, Levon Markossian, Maggie Xiao, Manie Tadayon, Maxine Yang, Melody Tsolmonkhuu, Michael Yip, Michelle Lee, Minh Chau, Miranda Mangahas, Monica Liu, Nikolay Atanasov, Paolo Gabriel, Pat Kimball, Patrick Chen, Rajan Bhattacharyya, Rio Sano, Ronnie Saxena, Sam Vinyard, Sid Paladugu, Stephanie Tsuei, Stephen Bauer, Thomas An, Tiffany Kim, Tim O'Donnell, Troy Prejusa, Vikash Gilja, and Wahab Alasfour. I am truly fortunate to have each and everyone one of you in my life, and I am forever grateful for the impact that you all have had on my personal and professional growth.

VITA

2019 - 2023 Ph.D. (Expected), Electrical and Computer Engineering
University of California, Los Angeles – Samueli School of Engineering

2020 - 2021 Research Intern
NASA Jet Propulsion Laboratory

2018 - 2019 Software Engineer - Computer Vision
HRL Laboratories

2017 - 2018 M.S., Electrical Engineering
University of California, San Diego – Jacobs School of Engineering

2013 - 2017 B.S., Electrical Engineering
University of California, San Diego – Jacobs School of Engineering

PUBLICATIONS

K. Chen, R. Nemiroff, and B.T. Lopez, "Direct LiDAR-Inertial Odometry and Mapping: Perceptive and Connective SLAM," *arXiv preprint arXiv:2305.01843*, 2023. (Under Review)

R. Nemiroff, K. Chen, and B.T. Lopez, "Joint On-Manifold Gravity and Accelerometer Intrinsics Estimation," *arXiv preprint arXiv:2303.03505*, 2023. (Under Review)

K. Chen, R. Nemiroff, and B.T. Lopez, "Direct LiDAR-Inertial Odometry: Lightweight LIO with Continuous-Time Motion Correction," *IEEE International Conference on Robotics and Automation (ICRA)*, ExCel, London, May 2023.

A. Bouman, J. Ott, S.K. Kim, **K. Chen**, M. Kochenderfer, B.T. Lopez, A. Agha-mohammadi, and J. Burdick, "Adaptive Coverage Path Planning for Efficient Exploration of Unknown Environments," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan, Oct 2022, pp. 11916-11923, doi: 10.1109/IROS47612.2022.9982287.

K. Chen, B.T. Lopez, A. Agha-mohammadi, and A. Mehta, "Direct LiDAR Odometry: Fast Localization with Dense Point Clouds," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 2, pp. 2000-2007, April 2022, doi: 10.1109/LRA.2022.3142739. Presented at ICRA 2022, Philadelphia, PA.

T.K. Chang, **K. Chen**, A. Mehta, "Resilient and Consistent Multirobot Cooperative Localization with Covariance Intersection," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 197-208, Feb. 2022, doi: 10.1109/TRO.2021.3104965.

K. Chen, A. Pogue, B.T. Lopez, A. Agha-mohammadi, and A. Mehta, "Unsupervised Monocular Depth Learning with Integrated Intrinsics and Spatio-Temporal Constraints," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic, Oct 2021, pp. 2451-2458, doi: 10.1109/IROS51168.2021.9636030.

A. Schperberg*, **K. Chen***, S. Tsuei, M. Jewett, J. Hooks, S. Soatto, A. Mehta, and D. Hong, "Risk-Averse MPC via Visual-Inertial Input and Recurrent Networks for Online Collision Avoidance," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, Oct 2020, pp. 5730-5737, doi: 10.1109/IROS45743.2020.9341070.
*Indicates equal contribution

CHAPTER 1

Introduction

1.1 Overview

The field of robotic perception stemmed from the convergence of ideas from computer vision, machine learning, state estimation, and numerical optimization. In general, "robot perception" includes any system in which raw input data from one or more sensors is abstracted into models or patterns to produce information about the robot itself and/or the surrounding environment. These sensors can either be *proprioceptive*, which produce internal measurements of the robot itself relative to the environment (e.g., inertial measurement unit, wheel encoders, temperature), or *exteroceptive*, which measure environmental parameters external to the system (e.g., monocular/stereo cameras, infrared, LiDAR, radar, tactile). Tasks ranging from state estimation and environmental mapping, to object recognition, semantic segmentation, scene reconstruction, and even image restoration all aim to provide an embodied system with visual intelligence in order to make sense of the physical world.

Simultaneous localization and mapping, or SLAM, is the computational problem of concurrently estimating a robot's *state* (i.e., position, velocity, and/or sensor biases) and its operating environment's *map* (i.e., an internal representation of the environment) via proprioceptive and/or exteroceptive sensors. The basic idea of SLAM is to use one or more sensors and process/fuse their (often noisy) measurements in such a way so that the robot can "back-out" a map of the world in addition to its position within the constructed map. The importance and fundamental need for SLAM in autonomous robotics is supported by

not only its substantial research history since the genesis of the problem in 1986 [SC86], but also the astonishing progress made since then and its continual popularity in the robotics community to this day [DB06, BD06]. The depth and complexity of this problem can be further observed by the flexibility in sensor selection and the lack of convergence by researchers towards an "ideal" sensor as they explored various sensor combinations for this task. Proposed architectures have ranged from primarily using one or more visual (camera) sensors to interpret 3D scene geometry, to infrared sensors, radar, thermal, tactile, and event cameras [CCC16]. Metrically-accurate depth measurements are required for the robot to understand environmental scale, and while there has been much progress in accurately regressing depth from a set of images, active time-of-flight sensors such as LiDAR have emerged as a promising solution to address the limitations of visual odometry and cameras by providing direct depth measurements of the surrounding environment—making them an attractive option for state estimation and mapping in challenging environments.

Despite the tremendous progress made in SLAM and other robotic perception tasks, there are still many challenges that remain to be tackled. One such challenge is the ability of robots to operate in unstructured and dynamic environments, where the information provided by sensors may be noisy or incomplete. Another challenge is the ability of robots to adapt to changing environmental conditions, such as lighting or weather, without the need for recalibration or retraining. These challenges are particularly important for applications such as autonomous driving, where the safety of passengers and pedestrians depends on the accurate and reliable perception of the surrounding environment, and more generally autonomous robots, where an agent needs an accurate representation of its operating environment to reliably plan and navigate without injuring itself or other entities. In this dissertation, we aim to address these challenges by developing new algorithms and techniques for state estimation and mapping that can operate reliably and efficiently in challenging real-world scenarios.

1.2 Motivation & Research Statement

While current methods may work well in known, structured environments (i.e., lab settings, human-made buildings, etc.), they can be unreliable outside of their intended target environments due to their inabilities to generalize (therefore inducing errors in data association), and from their high computational-complexity which fail to keep up with the high data throughput from modern sensors. Therefore, this dissertation aims to address these known deficiencies in modern state-of-the-art SLAM algorithms through two orthogonal thrusts. The first is the development of **fast and efficient** algorithms with **low computational complexity** and **high scalability** through innovative algorithmic techniques to reduce the reliance on significant compression or downsampling without sacrificing estimation accuracy. The second is designing **reliable architectures** with **long-term operational reliability** through **online adaptability** to generalize across a diverse set of structured and unstructured environments without manually hand-tuning parameters or heuristics. By addressing the deficiencies of modern SLAM algorithms through the proposed two-pronged approach, this dissertation aims to significantly improve the accuracy and reliability of autonomous robots operating in unstructured environments. The resulting algorithms and architectures will be able to generalize across a diverse set of environments without the need for manual tuning, enabling robots to operate with greater efficiency and reliability in the face of challenging real-world conditions.

1.3 Related Work

1.3.1 LiDAR Odometry

Geometric LiDAR odometry (LO) algorithms rely on aligning point clouds by solving a non-linear least-squares problem that minimizes the error across corresponding points and/or planes. To find point/plane correspondences, methods such as the iterative closest point

(ICP) algorithm [BM92, CM92] or Generalized-ICP (GICP) [SHT09] recursively match entities until alignment converges to a local minimum. Slow convergence time is often observed when determining correspondences for a large set of points, so *feature*-based methods [ZS14, SE18, SEM20, SER21b, PXH21, XCH21, NYC21, YCL19] attempt to extract only the most salient data points, e.g., corners and edges, in a scan to decrease computation time. These works aim to translate insights gained from visual odometry (VO) techniques into the 3D domain. However, adding this step increases computational overhead and risks discarding data points which could help with better correspondence matching for odometry accuracy. In addition, feature-based methods contain a strong assumption on the world, since these works explicitly look for edges and corners which may not exist in unstructured environments. Conversely, *dense* methods [PMT20, TTC20, CLA22a, XCH22, RPM22] directly align acquired scans but often rely heavily on aggressive voxelization—a process that can alter important data correspondences—to achieve real-time performance. Direct methods are much more accurate and do not contain any assumption on the outside world, but are usually infeasible for real-time applications due to the sheer number of points in a cloud.

A second stage immediately following scan alignment between adjacent clouds has been shown to reduce global drift by increasing pose estimation consistency with previous past scans [PMT20, ECP20]. In the scan-to-map stage, a scan-to-scan transformation is further refined by aligning the current point cloud with an existing in-memory map; this submap is typically derived by retrieving nearby map points within some radius of the robot's current position. However, this search in "point-space" can quickly explode in computational expense due to the sheer number of operations needed to retrieve the nearest neighbor data points. While there exists techniques to mitigate this such as only incrementally storing map data at keyed locations [SEM20], this search still involves thousands of calculations which can increase overall processor load and hence the potential to drop frames.

1.3.2 LiDAR-Inertial Odometry

LiDAR odometry approaches can also be broadly classified according to their method of incorporating other sensing modalities into the estimation pipeline. *Loosely*-coupled methods [ZS14, SE18, PMT20, TTC20, CLA22a] process data sequentially. For example, IMU measurements are used to augment LiDAR scan registration by providing an optimization prior. These methods are often quite reliable due to the precision of LiDAR measurements, but localization results can be less accurate as only a subset of all available data is used for estimation. *Tightly*-coupled methods [SEM20, XCH22, YCL19, NYC21], on the otherhand, a.k.a. LiDAR-inertial odometry (LIO), can offer improved accuracy by jointly considering measurements from all sensing modalities. These methods commonly employ either graph-based optimization [SEM20, YCL19, NYC21, ZKS16] or a stochastic filtering framework, e.g., Kalman filter [XCH21, XCH22]. However, compared to geometric observers [BMT07, VSO08], these approaches possess minimal convergence guarantees even in the most ideal settings which can result in significant localization error from inconsistent sensor fusion and map deformation from incorrect scan placement.

Incorporating additional sensors can also aid in correcting motion-induced point cloud distortion. For example, LOAM [ZS14] compensates for spin distortion by iteratively estimating sensor pose via scan-matching and a loosely-coupled IMU using a constant velocity assumption. Similarly, LIO-SAM [SEM20] formulates LiDAR-inertial odometry atop a factor graph to jointly optimize for body velocity, and in their implementation, points were subsequently deskewed by linearly interpolating rotational motion. FAST-LIO [XCH21] and FAST-LIO2 [XCH22] instead employ a back-propagation step on point timestamps after a forward-propagation of IMU measurements to produce relative transformations to the scan-end time. However, these methods (and others [RSS20, DBK21]) all operate in *discrete*-time which may induce a loss in precision, leading to a high interest in *continuous*-time methods. Elastic LiDAR Fusion [PMK18], for example, handles scan deformation by optimizing for a continuous linear trajectory, whereas Wildcat [RKC22] and [DB18] instead iteratively fit

a cubic B-spline to remove distortion from surfel maps. More recently, CT-ICP [DDJ22] and ElasticLiDAR++ [PMW22] use a LiDAR-only approach to define a continuous-time trajectory parameterized by two poses per scan, which allows for elastic registration of the scan during optimization. However, these methods can still be too simplistic in modeling the trajectory under highly dynamical movements or may be too computationally costly to work reliably in real-time.

Another crucial component in LIO systems is submapping, which involves extracting a smaller point cloud from the global map for efficient processing and to increase pose estimation consistency. Rather than processing the entire map on every iteration which is often computationally intractable, a *submap* instead contains only a subset of all available data points to be considered. However, the efficacy of a submapping strategy depends on its ability to extract only the most relevant map points for scan-matching to avoid any wasted computation when constructing corresponding data structures (i.e., kdtrees, normals). One common approach is to use a sliding window such that the submap consists of a set of recent scans [YCL19, LZ21]. However, this method assumes a strong temporal correspondence between points which may not always be the case (i.e., revisiting a location) and may not perform well under significant drift. To mitigate this, radius-based approaches [PMT20, SEM20] extract points nearby the current position by directly working with point clouds and continually adding points to an internal octree data structure. This, however, results in unbounded growth in the map [VGM23a] and therefore an explosion in computational expense through the large number of nearest neighbor calculations required, which is infeasible for real-time usage. Keyframe-based methods [SEM20, CLA22a], on the other hand, link keyed locations in space to its corresponding scan, and therefore reduces the search space required to extract a comprehensive submap. However, previous methods [CLA22a, CNL23b] have still implicitly assumed that nearby keyframes contain the most relevant data points for a submap and do not explicitly compute a metric of relevancy per keyframe, risking the extraction of keyframes which may not be used at all.

More recently, researchers have been interested in building new methods of algorithmic resiliency into odometry pipelines to ensure reliable localization across a diverse set of environments. While the field of adaptive localization is still in its infancy, early works have pioneered the idea of adaptivity in unstructured and/or extreme environments. For instance, to provide resiliency against LiDAR slippage in geometrically-degenerate environments, LION [TTC20], DARE-SLAM [EPW21], and DAMS-LIO [FHW23] proposed using the condition number of the scan-matching Hessian as an observability score in order to switch to a different state-estimation algorithm (e.g., visual-inertial). However, these works assume the availability of other odometry paradigms on-board which may not always be available. On the other hand, works such as [KYO21a] and [KYO21b] attempt to automatically tune system hyperparameters based on trajectory error modeling, but these require an expensive offline training procedure to retrieve optimal parameter values and do not generalize to other environments outside of the training set. Similarly, KISS-ICP [VGM23a] adaptively tunes maximum correspondence distance for ICP, but their method scales the metric according to robot acceleration which does not generalize across differently-sized environments. AdaLIO [LKK23], on the other hand, automatically tunes voxelization, search distance, and plane residual parameters in detected degenerate cases. Other methods, such as [MBG22], include providing formal guarantees for the LiDAR scan-matching process. Methods of global loop closure detection can also aid in reducing drift in the map through place recognition and relocalization, using descriptors and detectors such as ScanContext [KK18, KCK21] and Segregator [YYC23]. More recently, [ZKS16] and X-ICP [TNN22] propose innovative online methods to mitigate degeneracy by analyzing the geometric structure of the optimization constraints in scan-to-scan and scan-to-map, respectively, but these works only target a specific module in an entire, complex SLAM pipeline.

1.4 Outline of Dissertation

This dissertation aims to address deficiencies in current state-of-the-art SLAM algorithms which struggle to achieve long-term operational reliability in the unstructured real world. To this end, this dissertation presents three main contributions which leverage LiDAR technology to enable fast, reliable localization and mapping.

1.4.1 Contribution 1: Fast Localization with Dense Point Clouds

LiDAR odometry has received tremendous attention as an alternative to vision-based methods due to the precise depth measurements and larger field-of-view of the sensors; however, modern LiDAR sensors have posed a challenge to current odometry algorithms due to their high data throughput. To address this challenge, this contribution presents a novel frontend localization solution that enables the direct use of dense point cloud scans without significant preprocessing. The main contribution of this research is a custom speed-oriented pipeline that accurately estimates robot pose in real-time using minimally-preprocessed LiDAR scans and an optional IMU on consumer-grade processors. The approach is based on three core innovations: an adaptive keyframing system that efficiently captures significant environmental information using a novel spaciousness metric, a fast keyframe-based submapping approach via convex optimization that generates permissive local submaps for global pose refinement, and NanoGICP, a custom iterative closet point solver for lightweight point cloud scan-matching with data structure recycling to eliminate redundant calculations. The key insight of this contribution is the relationship between algorithmic speed and accuracy, which enables the development of a highly efficient and accurate frontend localization solution.

1.4.2 Contribution 2: Parallelizable Continuous-Time Motion Correction

Loosely-coupled LiDAR odometry methods, while reliable, lack the high-rate output and additional state estimates provided by tightly-coupled LiDAR-inertial odometry methods.

Moreover, point cloud deskewing can be infeasible without properly calibrated IMU biases for double-integration of accelerometer data. To address these limitations, this contribution presents a fast and reliable odometry algorithm that delivers accurate localization and detailed 3D mapping through three core innovations. First, a novel coarse-to-fine technique is proposed for constructing continuous-time trajectories, leveraging a set of analytical equations with a constant jerk and angular acceleration motion model for efficient and parallelizable point-wise motion correction. Second, a condensed architecture that integrates motion correction and prior construction into one step, performing direct scan-to-map registration is introduced to significantly reduce the overall computational overhead of the algorithm. Third, a new nonlinear geometric observer with strong performance guarantees is employed in the pipeline to reliably generate accurate estimates of the robot's full state with minimal computational complexity. Extensive experimental results using multiple datasets against state-of-the-art approaches demonstrate the effectiveness of our proposed method.

1.4.3 Contribution 3: Perceptive and Connective SLAM

Building proactive safe-guards against common failure points in SLAM can provide reliable and failure-tolerant localization perceptive to a wide range of operating environments for long-term reliability. Therefore, this contribution proposes four new techniques to LiDAR-based SLAM systems which address several deficiencies in both the front-end and back-end. Our ideas target different scales in the data processing pipeline to comprehensively increase localization resiliency and mapping accuracy. First, an adaptive scan-matching method via a novel point cloud sparsity metric for consistent registration in both large and small environments. Second, a method for slip-resistant keyframing by detecting the onset of scan-matching slippage during abrupt scene changes via a global and sensor-agnostic degeneracy metric. Third, A method which generates explicitly-relevant local submaps with maximum coverage by computing the relative 3D Jaccard index for each keyframe for scan-to-map registration. Finally, a method to increase local mapping accuracy and global loop closure

resiliency via connectivity factors and keyframe-based loop closures.

1.4.4 Structure of Chapters

The following chapters of this dissertation are arranged as follows.

- Chapter 2 covers the Direct LiDAR Odometry (DLO) algorithm [CLA22b], a fast and lightweight LiDAR-only solution for localization and mapping with dense point clouds capable of real-time processing on computationally-limited platforms, which was the first of its kind.

- Chapter 3 details the Direct LiDAR-Inertial Odometry (DLIO) algorithm [CNL23b], which extends our DLO algorithm with an inertial measurement unit (IMU) for fast point cloud motion correction through a condensed architecture with strong convergence properties.

- Chapter 4 presents the Direct LiDAR-Inertial Odometry and Mapping (DLIOM) algorithm [CNL23a], a reliable LiDAR SLAM algorithm that prioritizes operational reliability and real-world efficacy by strategically placing proactive safe-guards against common failure points in both the front-end and back-end subsystems.

- Chapter 5 discusses the work presented in this dissertation, examines its implications to the field of robot perception, and sets a concrete future plan for achieving the philosophies presented in this dissertation to further push our perception capabilities in autonomous robotics.

Other works that the author was involved in but have been omitted from this dissertation include [SCT20], [CCM22], [CPL21], [BGA20], and [NCL23].

CHAPTER 2

Direct LiDAR Odometry: Fast Localization with Dense Point Clouds

Field robotics in perceptually-challenging environments require fast and accurate state estimation, but modern LiDAR sensors quickly overwhelm current odometry algorithms. To this end, this chapter presents a lightweight frontend LiDAR odometry solution with consistent and accurate localization for computationally-limited robotic platforms. Our Direct LiDAR Odometry (DLO) method includes several key algorithmic innovations which prioritize computational efficiency and enables the use of dense, minimally-preprocessed point clouds to provide accurate pose estimates in real-time. This is achieved through a novel keyframing system which efficiently manages historical map information, in addition to a custom iterative closest point solver for fast point cloud registration with data structure recycling. Our method is more accurate with lower computational overhead than the current state-of-the-art and has been extensively evaluated in multiple perceptually-challenging environments on aerial and legged robots as part of NASA JPL Team CoSTAR's research and development efforts for the DARPA Subterranean Challenge.

2.1 Overview

Accurate state estimation and mapping in large, perceptually-challenging environments have become critical capabilities for autonomous mobile robots. Whereas typical visual SLAM approaches often perform poorly in dust, fog, or low-light conditions, LiDAR-based methods

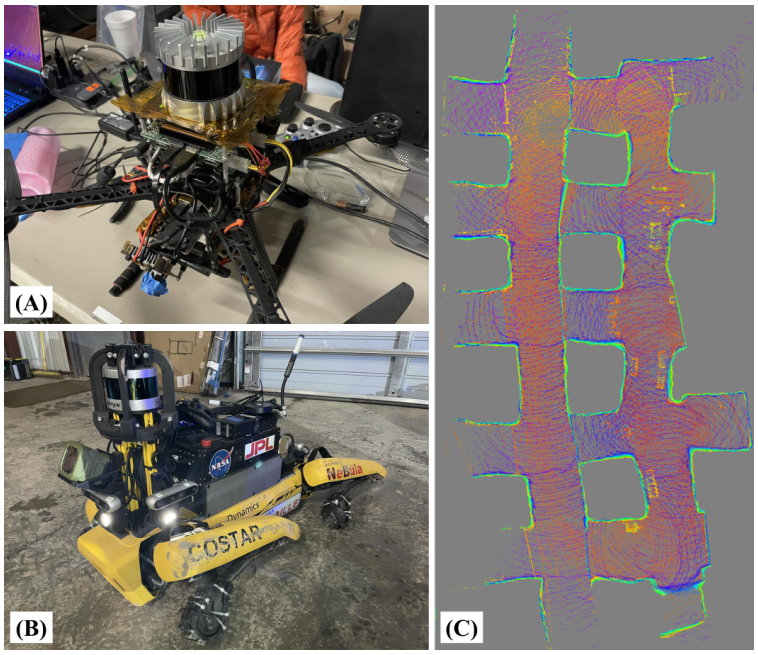

Figure 2.1: **Fast and Lightweight LiDAR Odometry.** Two of Team CoSTAR's robotic platforms which have limited computational resources. (A) Our custom quadrotor platform which features an Ouster OS1 LiDAR sensor on top. (B) A Boston Dynamics Spot robot with a mounted custom payload and a Velodyne VLP-16 with protective guards. (C) Top-down view of a mapped limestone mine using our lightweight odometry method on these robots during testing and integration for the DARPA Subterranean Challenge.

can provide more reliable localization due to the superior range and accuracy of direct depth measurements [CCC16]. However, recent work on LiDAR odometry (LO) have revealed the challenges of processing the large number of depth returns generated by commercial LiDAR sensors in real-time for high-rate state estimation [SE18, ECP20]. This chapter presents several algorithmic innovations that make real-time localization with dense LiDAR scans feasible while also demonstrating the superiority of our method in terms of accuracy and computational complexity when compared to the state-of-the-art.

Current LO algorithms estimate a robot's egomotion in two stages: first, by performing a "scan-to-scan" alignment between adjacent LiDAR frames to recover an immediate motion guess, followed by a "scan-to-map" registration between the current scan and past environmental knowledge to increase global pose consistency. Unfortunately, the large number

of data points per scan from modern LiDARs quickly overwhelms computationally-limited processors and bottlenecks performance during alignment, which can induce frame drops and ultimately cause poor pose estimation. More specifically, scan-to-scan alignment requires a registration of corresponding points between two clouds, but this process often involves a nearest-neighbor search which grows exponentially with the number of points per scan. Feature-based methods [SE18, SER21a, YCL19, SEM20] attempt to mitigate this by using only the most salient points, but these methods employ an often computationally-intensive feature extraction step and may accidentally discard data which could otherwise help improve the quality of downstream registration. Moreover, in scan-to-map alignment, keyed environmental history (which consists of all or a subset of past points) grows rapidly in size as new scans are acquired and stored in memory. While aligning with a submap (rather than the full history of scans) helps increase computational efficiency, the perpetual addition of points still significantly expands the nearest-neighbor search space for typical submap extraction methods. Tree-based data structures have been shown to decrease this nearest-neighbor search cost significantly [Bha10], but the extraction of a local submap still involves too many points after just a few keyframes, thus preventing consistent performance for long-term navigation.

In this chapter, we present our Direct LiDAR Odometry (DLO) algorithm, a high-speed and computationally-efficient frontend localization solution which permits the direct use of dense point cloud scans without significant preprocessing. The main contribution of this chapter is a custom speed-first pipeline which accurately resolves robot egomotion in real-time using minimally-preprocessed LiDAR scans and an optional IMU on consumer-grade processors. A key insight of our work is the link between algorithmic speed and accuracy, and our approach is comprised of three core innovations. First, an adaptive keyframing system which efficiently captures significant environmental information through a novel spaciousness metric. Second, a fast keyframe-based submapping approach via convex optimization which generates permissive local submaps for global pose refinement. Third, NanoGICP, a custom

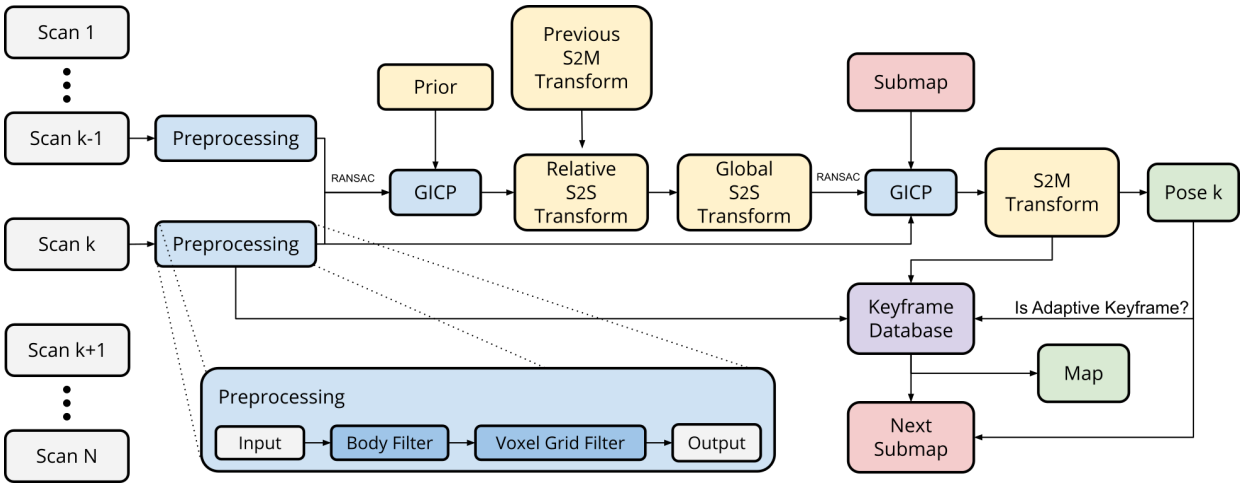

Figure 2.2: **LiDAR Odometry Architecture.** Our system first retrieves a relative transform between two temporally-adjacent scans of times k and $k-1$ through scan-to-scan (S2S) matching with RANSAC outlier rejection and an optional rotational prior from IMU. This initial estimate is then propagated into the world frame and used as the initialization point for our secondary GICP module for scan-to-map optimization (S2M), which scan-matches the current point cloud $\mathcal{P}_k$ with a derived submap $\mathcal{S}_k$ consisting of scans from nearby and boundary keyframes. The output of this is a globally-consistent pose estimate which is subsequently checked against several metrics to determine if the current pose should be stored as a new keyframe.

iterative closet point solver for lightweight point cloud scan-matching with data structure recycling to eliminate redundant calculations. Our method has been extensively evaluated in numerous challenging environments on computationally-limited robotic platforms as part of Team CoSTAR's research and development efforts for the DARPA Subterranean Challenge, and we have open-sourced our code for benefit of the community[1].

2.2 System Description

Our DLO algorithm is built around a "speed-first" philosophy to permit the use of minimally-preprocessed point clouds and provide accurate pose estimates even for robots with limited computational resources (Fig. 2.2). The key contribution of our work lies in how we efficiently

[1] https://github.com/vectr-ucla/direct_lidar_odometry

derive our submap for global refinement in scan-to-map matching. That is, rather than extracting points within a local vicinity of a robot's current position as most works do, DLO instead searches in keyframe-space by associating a scan's set of points with its corresponding keyframe position. The submap is subsequently constructed by concatenating the clouds from a subset of historic keyframes derived from nearby keyframes and those which make up the convex hull; this provides the current scan with both nearby and distant points in the submap to anchor to. In addition, a custom GICP solver enables extensive reuse of data structures across multiple solver instantiations to eliminate redundant operations across the two-stage process. Our system also optionally accepts an initialization prior from an IMU in a loosely-coupled fashion to further improve accuracy during aggressive rotational motions. The reliability of our approach is demonstrated through extensive tests on several computationally-limited robotic platforms in multiple challenging environments. This algorithm was part of Team CoSTAR's research and development efforts for the DARPA Subterranean Challenge in support of NASA Jet Propulsion Laboratory's Networked Belief-aware Perceptual Autonomy (NeBula) framework [AOM21], in which DLO was the primary state estimation component for our fleet of autonomous aerial vehicles (Fig. 2.1A).

2.2.1 Notation

A point cloud, $\mathcal{P}$, is composed of a set of points $p \in \mathcal{P}$ with Cartesian coordinates $p_i \in \mathbb{R}^3$. We denote $\mathcal{L}$ as the LiDAR's coordinate system, $\mathcal{B}$ as the robot's coordinate system located at the IMU frame, and $\mathcal{W}$ as the world coordinate system which coincides with $\mathcal{B}$ at the initial position. Note that in this work we assume $\mathcal{L}$ and $\mathcal{B}$ reference frames coincide. Submap, covariance, and kdtree structures are denoted as $\mathcal{S}$, $\mathcal{C}$ and $\mathcal{T}$, respectively. We adopt standard convention such that x points forward, y points left, and z points upward, and our work attempts to address the following problem: given adjacent point clouds scans $\mathcal{P}_k$ and $\mathcal{P}_{k-1}$ at time k, estimate the robot's current pose $\hat{\mathbf{X}}_k^{\mathcal{W}} \in \mathbb{SE}(3)$ and map $\mathcal{M}_k$ in $\mathcal{W}$.

2.2.2 Preprocessing

Our system assumes an input of 3D point cloud data gathered by a 360° LiDAR such as an Ouster OS1 (20Hz) or a Velodyne VLP-16 (10Hz). To minimize information loss from the raw sensor data, only two filters are used during preprocessing: first, we remove all point returns that may be from the robot itself through a box filter of size $1m^3$ around the origin. This is especially important if an aerial robot's propellers (Fig. 2.1A) or protective guards (Fig. 2.1B) are in the LiDAR's field of view. The resulting cloud is then sent through a 3D voxel grid filter with a resolution of 0.25m to lightly downsample the data for subsequent tasks while maintaining dominate structures within the surrounding environment. Note that in this algorithm we do not correct for motion distortion since non-rigid transformations can be computationally burdensome, and we directly use the dense point cloud rather than extracting features as most works do. On average, each cloud contains ∼10,000 points after preprocessing.

2.3 Lightweight LO Architecture

2.3.1 Scan Matching via Generalized-ICP

LiDAR-based odometry can be viewed as the process of resolving a robot's egomotion by means of comparing successive point clouds and point clouds in-memory to recover an $\mathbb{SE}(3)$ transformation, which translates to the robot's 6-DOF motion between consecutive LiDAR acquisitions. This process is typically performed in two stages, first to provide a best instantaneous guess, which is subsequently refined to be more globally consistent with previous keyframe locations.

Algorithm 1: Direct LiDAR Odometry

input: $\mathcal{P}_k$, $\hat{\mathbf{X}}^{\mathcal{W}}_{k-1}$; **initialize:** $\hat{\mathbf{X}}^{\mathcal{W}}_{k-1} = \mathbf{I}$ or gravityAlign()
output: $\hat{\mathbf{X}}^{\mathcal{W}}_{k}$, $\mathcal{M}_k$
while $\mathcal{P}_k \neq \emptyset$ **do**
    // preprocessing
    $\bar{\mathcal{P}}_k \leftarrow \text{preprocessPoints}(\mathcal{P}_k)$;
    computeAdaptiveParameters($\bar{\mathcal{P}}_k$) ;
    // initialization
    if $k = 0$ **then**
        $\mathcal{T}^{\mathrm{t}_1}_k, \mathcal{C}^{\mathrm{t}_1}_k \leftarrow \text{NanoGICP}_1\text{.build}(\bar{\mathcal{P}}_k)$;
        $\mathcal{K}_k \leftarrow \text{updateKeyframeDatabase}(\hat{\mathbf{X}}^{\mathcal{W}}_{k-1}, \bar{\mathcal{P}}_k)$;
        continue;
    end
    // prior
    if *IMU* **then** $\tilde{\mathbf{X}}^{\mathcal{L}}_k \leftarrow \tilde{\mathbf{X}}^{\mathcal{B}}_k$; **else** $\tilde{\mathbf{X}}^{\mathcal{L}}_k \leftarrow \mathbf{I}$;
    // scan-to-scan
    $\mathcal{T}^{\mathrm{s}_1}_k, \mathcal{C}^{\mathrm{s}_1}_k \leftarrow \text{NanoGICP}_1\text{.build}(\bar{\mathcal{P}}_k)$;
    $\hat{\mathbf{X}}^{\mathcal{L}}_k \leftarrow \text{NanoGICP}_1\text{.align}(\mathcal{T}^{\mathrm{s}_1}_k, \mathcal{T}^{\mathrm{t}_1}_k, \mathcal{C}^{\mathrm{s}_1}_k, \mathcal{C}^{\mathrm{t}_1}_k, \tilde{\mathbf{X}}^{\mathcal{L}}_k)$;
    $\tilde{\mathbf{X}}^{\mathcal{W}}_k \leftarrow \hat{\mathbf{X}}^{\mathcal{W}}_{k-1}\, \hat{\mathbf{X}}^{\mathcal{L}}_k$;
    // scan-to-map
    $\mathcal{Q}_k \leftarrow \text{getKeyframeNeighbors}(\hat{\mathbf{X}}^{\mathcal{W}}_{k-1}, \mathcal{K}_k)$;
    $\mathcal{H}_k \leftarrow \text{getKeyframeHulls}(\hat{\mathbf{X}}^{\mathcal{W}}_{k-1}, \mathcal{K}_k)$;
    $\mathcal{S}_k \leftarrow \mathcal{Q}_k \oplus \mathcal{H}_k$;
    if $\mathcal{S}_k \neq \mathcal{S}_{k-1}$ **then** $\mathcal{T}^{\mathrm{t}_2}_k \leftarrow \text{NanoGICP}_2\text{.build}(\mathcal{S}_k)$; **else** $\mathcal{T}^{\mathrm{t}_2}_k \leftarrow \mathcal{T}^{\mathrm{t}_2}_{k-1}$;
    $\mathcal{T}^{\mathrm{s}_2}_k \leftarrow \mathcal{T}^{\mathrm{s}_1}_k$; $\mathcal{C}^{\mathrm{s}_2}_k \leftarrow \mathcal{C}^{\mathrm{s}_1}_k$; $\mathcal{C}^{\mathrm{t}_2}_k \leftarrow \sum_n^N \mathcal{C}^{\mathcal{S}}_{k,n}$;
    $\hat{\mathbf{X}}^{\mathcal{W}}_k \leftarrow \text{NanoGICP}_2\text{.align}(\mathcal{T}^{\mathrm{s}_2}_k, \mathcal{T}^{\mathrm{t}_2}_k, \mathcal{C}^{\mathrm{s}_2}_k, \mathcal{C}^{\mathrm{t}_2}_k, \tilde{\mathbf{X}}^{\mathcal{W}}_k)$;
    // update keyframe database and map
    $\mathcal{K}_k \leftarrow \text{updateKeyframeDatabase}(\hat{\mathbf{X}}^{\mathcal{W}}_k, \bar{\mathcal{P}}_k)$;
    $\mathcal{M}_k \leftarrow \mathcal{M}_{k-1} \oplus \{\mathcal{K}_k \setminus \mathcal{K}_{k-1}\}$;
    // propagate data structures
    $\mathcal{T}^{\mathrm{t}_1}_k \leftarrow \mathcal{T}^{\mathrm{s}_1}_k$; $\mathcal{C}^{\mathrm{t}_1}_k \leftarrow \mathcal{C}^{\mathrm{s}_1}_k$;
    return $\hat{\mathbf{X}}^{\mathcal{W}}_k$, $\mathcal{M}_k$
end

2.3.1.1 Scan-to-Scan

In the first stage, the scan-to-scan matching objective is to compute a relative transform $\hat{\mathbf{X}}_k^{\mathcal{L}}$ between a source $\mathcal{P}_k^{\mathrm{s}}$ and a target $\mathcal{P}_k^{\mathrm{t}}$ (where $\mathcal{P}_k^{\mathrm{t}} = \mathcal{P}_{k-1}^{\mathrm{s}}$) captured in $\mathcal{L}$ where

$$\hat{\mathbf{X}}_k^{\mathcal{L}} = \underset{\mathbf{X}_k^{\mathcal{L}}}{\arg\min}\ \mathcal{E}\left(\mathbf{X}_k^{\mathcal{L}}\mathcal{P}_k^{\mathrm{s}}, \mathcal{P}_k^{\mathrm{t}}\right) . \tag{2.1}$$

The residual error $\mathcal{E}$ from GICP is defined as

$$\mathcal{E}\left(\mathbf{X}_k^{\mathcal{L}}\mathcal{P}_k^{\mathrm{s}}, \mathcal{P}_k^{\mathrm{t}}\right) = \sum_i^N d_i^\top \left(\mathcal{C}_{k,i}^{\mathrm{t}} + \mathbf{X}_k^{\mathcal{L}}\mathcal{C}_{k,i}^{\mathrm{s}}\mathbf{X}_k^{\mathcal{L}^\top}\right)^{-1} d_i \,, \tag{2.2}$$

such that the overall objective for this stage is

$$\hat{\mathbf{X}}_k^{\mathcal{L}} = \underset{\mathbf{X}_k^{\mathcal{L}}}{\arg\min} \sum_i^N d_i^\top \left(\mathcal{C}_{k,i}^{\mathrm{t}} + \mathbf{X}_k^{\mathcal{L}}\mathcal{C}_{k,i}^{\mathrm{s}}\mathbf{X}_k^{\mathcal{L}^\top}\right)^{-1} d_i \,, \tag{2.3}$$

for N number of corresponding points between point clouds $\mathcal{P}_k^{\mathrm{s}}$ and $\mathcal{P}_k^{\mathrm{t}}$, where $d_i = p_i^{\mathrm{t}} - \mathbf{X}_k^{\mathcal{L}} p_i^{\mathrm{s}}$, $p_i^{\mathrm{s}} \in \mathcal{P}_k^{\mathrm{s}}, p_i^{\mathrm{t}} \in \mathcal{P}_k^{\mathrm{t}}, \forall i$, and $\mathcal{C}_{k,i}^{\mathrm{s}}$ and $\mathcal{C}_{k,i}^{\mathrm{t}}$ are the corresponding estimated covariance matrices associated with each point i of the source or target cloud, respectively. As will be further discussed in Section 2.3.1.3, we can initialize the above objective function with a prior supplied by external sensors in an attempt to push the convergence towards a global minimum. That is, for Eq. (2.3), if a prior $\tilde{\mathbf{X}}_k^{\mathcal{B}}$ is available by means of IMU preintegration, we can set the initial guess $\tilde{\mathbf{X}}_k^{\mathcal{L}} = \tilde{\mathbf{X}}_k^{\mathcal{B}}$ to create a loosely-coupled system. If a prior is not available however, the system reverts to pure LiDAR odometry in which $\tilde{\mathbf{X}}_k^{\mathcal{L}} = \mathbf{I}$ and relies solely on point cloud correspondence matching for this step.

2.3.1.2 Scan-to-Map

After recovering an initial robot motion estimate, a secondary stage of scan-to-map matching is performed and follows a similar procedure to that of scan-to-scan. However, rather than

computing a relative transform between two instantaneous point clouds, the objective here is to further refine the motion estimate from the previous step to be more globally-consistent by means of matching with a local submap. In other words, the task here is to compute an optimal transform $\hat{\mathbf{X}}_k^{\mathcal{W}}$ between the current source cloud $\mathcal{P}_k^{\mathrm{s}}$ and some derived submap $\mathcal{S}_k$ such that

$$\hat{\mathbf{X}}_k^{\mathcal{W}} = \underset{\mathbf{X}_k^{\mathcal{W}}}{\arg\min}\ \mathcal{E}\left(\mathbf{X}_k^{\mathcal{W}} \mathcal{P}_k^{\mathrm{s}}, \mathcal{S}_k\right) . \tag{2.4}$$

After similarly defining the residual error $\mathcal{E}$ from GICP as in Eq. (2.2), the overall objective function for scan-to-map is

$$\hat{\mathbf{X}}_k^{\mathcal{W}} = \underset{\mathbf{X}_k^{\mathcal{W}}}{\arg\min} \sum_j^M d_j^\top \left(\mathcal{C}_{k,j}^{\mathcal{S}} + \mathbf{X}_k^{\mathcal{W}} \mathcal{C}_{k,j}^{\mathrm{s}} \mathbf{X}_k^{\mathcal{W}^\top}\right)^{-1} d_j \,, \tag{2.5}$$

for M number of corresponding points between point cloud $\mathcal{P}_k^{\mathrm{s}}$ and submap $\mathcal{S}_k$, where $\mathcal{C}_{k,j}^{\mathcal{S}}$ is the corresponding scan-stitched covariance matrix for point j in the submap as defined later in Section 2.4. Eq. (2.5) is initialized using the propagated result from scan-to-scan in the previous section from $\mathcal{L}$ to $\mathcal{W}$, i.e. $\tilde{\mathbf{X}}_k^{\mathcal{W}} = \hat{\mathbf{X}}_{k-1}^{\mathcal{W}}\ \hat{\mathbf{X}}_k^{\mathcal{L}}$, so that this prior motion can be compared against historical map data for global consistency. The output of this stage $\hat{\mathbf{X}}_k^{\mathcal{W}}$ is the final estimated robot pose used for downstream modules.

We note here that a key innovation of this algorithm is how we derive and manage our submap for this stage. Whereas previous works create a submap by querying the locality of each individual point in a stored map, we associate scans to keyframes and search rather in keyframe-space to stitch point clouds together and create $\mathcal{S}_k$. The implications of this include a far faster and more consistent generation of a local submap, which is additionally more permissive as compared to a radius-based search and will be further discussed in Section 2.3.2.

2.3.1.3 Optimization Prior

Eq. (2.3) describes the scan-to-scan nonlinear optimization problem and can be initialized with a prior to reduce the chances of converging into a sub-optimal local minima. This prior represents an initial guess of the relative motion between two LiDAR frames and can come from integrating angular velocity measurements from an inertial measurement unit (IMU). More specifically, angular velocity measurements $\hat{\boldsymbol{\omega}}_k$ is defined as $\hat{\boldsymbol{\omega}}_k = \boldsymbol{\omega}_k + \mathbf{b}_k^{\boldsymbol{\omega}} + \mathbf{n}_k^{\boldsymbol{\omega}}$ measured in $\mathcal{B}$ with static bias $\mathbf{b}_k^{\boldsymbol{\omega}}$ and zero white noise $\mathbf{n}_k^{\boldsymbol{\omega}}$ for convenience. After calibrating for the bias, a relative rotational motion of the robot's body between two LiDAR frames can be computed via gyroscopic propagation of the quaternion kinematics $\mathbf{q}_{k+1} = \mathbf{q}_k + (\frac{1}{2}\mathbf{q}_k \otimes \boldsymbol{\omega}_k)\Delta t$. Here, $\mathbf{q}_k$ is initialized to identity prior to integration, Δt is the difference in time between IMU measurements in seconds, and only gyroscopic measurements found between the current LiDAR scan and the previous one are used. Note that we are only concerned with a rotational prior during IMU preintegration and leave the retrieval of a translational prior via the accelerometer for future work. The resulting quaternion of this propagation is converted to an $\mathbb{SE}(3)$ matrix with zero translational component to be used as $\tilde{\mathbf{X}}_k^{\mathcal{B}}$, the scan-to-scan prior.

2.3.2 Fast Keyframe-Based Submapping

A key innovation of this algorithm lies in how our system manages map information and derives the local submap in scan-to-submap matching for global egomotion refinement. Rather than working directly with point clouds and storing points into a typical octree data structure, we instead keep a history of *keyframes* to search within, in which each keyframe is linked to its corresponding point cloud scan in a key-value pair. The resulting local submap used for scan-to-submap matching is then generated by concatenating the corresponding point clouds from a subset of the keyframes, rather than directly retrieving local points within some radius of the robot's current position.

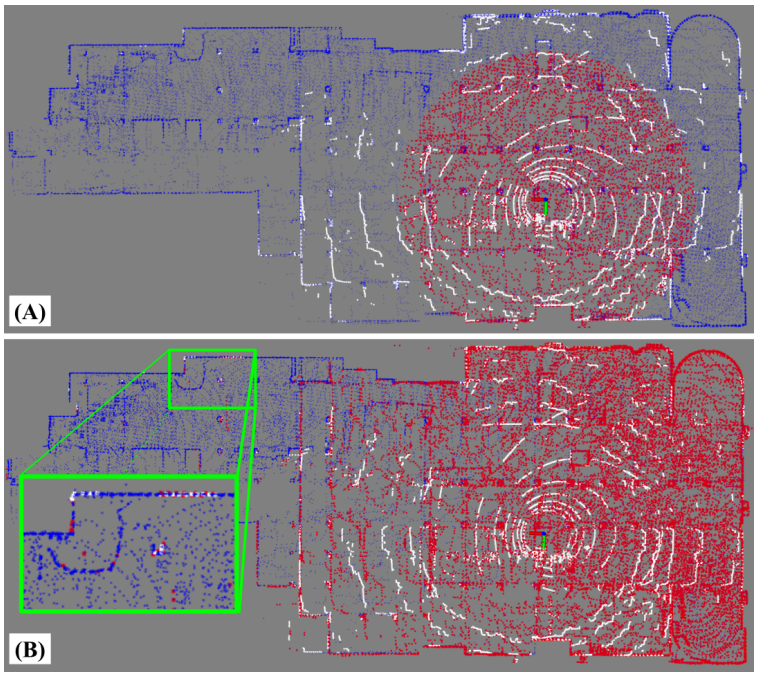

Figure 2.3: **Keyframe-Based Submapping.** A comparison between the different submapping approaches, visualizing the current scan (white), the derived submap (red), and the full map (blue). (A) A common radius-based submapping approach of $r = 20$m retrieved in point cloud-space. (B) Our keyframe-based submapping approach, which concatenates a subset of keyed scans and helps anchor even the most distant points in the current scan (green box) during the scan-to-map stage.

The implication of this design choice is twofold: first, by searching in "keyframe-space" rather than "point cloud-space," a much more computationally tractable problem is obtained. Radius-based searches within a cumulative point cloud map can require distance calculations against hundreds of thousands of points — a process that quickly becomes infeasible even with an incremental octree data structure. Searching against keyframes, however, typically involves only a few hundred points even after long traversals and provides much more consistent computational performance, reducing the chances of dropping frames. Additionally, a keyframe-based approach constructs a much more permissive submap as compared to range-based methods. That is, since the size of a submap derived from keyframe point clouds relies solely on the LiDAR sensor's range rather than a predetermined distance, the derived submap can have a larger overlap with the current scan; this is illustrated in Fig. 2.3. In

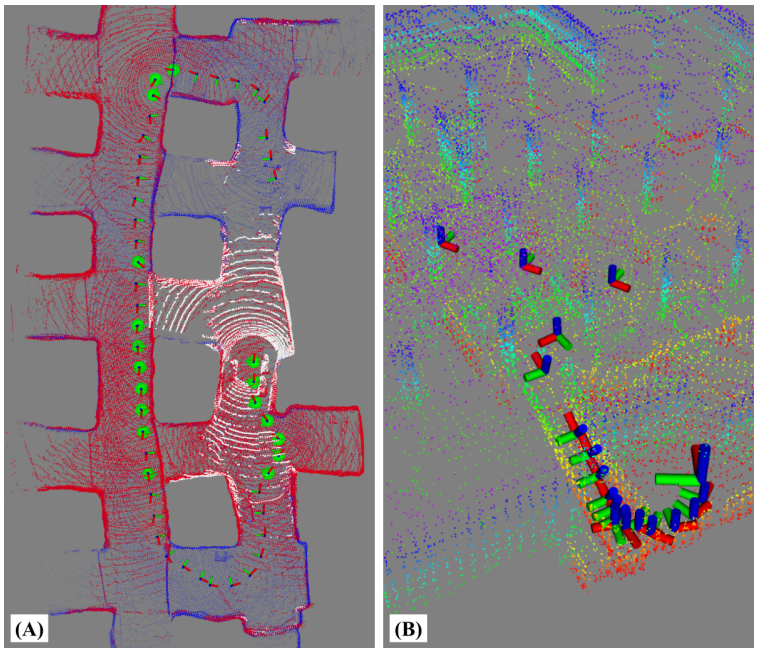

Figure 2.4: **Keyframe Selection and Adaptive Thresholds.** (A) Our method's submap (red) is generated by concatenating the scans from a subset of keyframes (green spheres), which consists of K nearest neighbor keyframes and those that construct the convex hull of the keyframe set. (B) An illustration of adaptive keyframing. In this scenario, the threshold decreases when traversing down a narrow ramp to better capture small-scale details.

this example, a submap of fixed radius $r = 20$m insufficiently overlaps with the current scan and can introduce drift over time due to containing only spatially-nearby points; however, a keyframe-based approach covers most of the current scan which helps with better scan-to-map alignment. Expanding the radius size may help increase this overlap for radius-based methods, but doing so would significantly slowdown subsequent tasks such as the GICP covariance calculations.

2.3.3 Keyframe Selection via kNN and Convex Hull

To construct the submap $\mathcal{S}_k$, we concatenate the corresponding point clouds from a selected subset of environmental keyframes. Let $\mathcal{K}_k$ be the set of all keyframe point clouds such that $\mathcal{S}_k \subseteq \mathcal{K}_k$. We define submap $\mathcal{S}_k$ as the concatenation of K nearest neighbor keyframe

scans $\mathcal{Q}_k$ and L nearest neighbor convex hull scans $\mathcal{H}_k$ such that $\mathcal{S}_k = \mathcal{Q}_k \oplus \mathcal{H}_k$, where the indices which specify the convex hull are defined by the set of keyframes which make up the intersection of all convex sets containing the keyframes which compose $\mathcal{K}_k$.

The result of this is illustrated in Fig. 2.4A, in which the keyframes highlighted in green are those that compose the extracted submap, indicated in red. Intuitively, extracting nearest neighbor keyframes aims to help with overlap of nearby points in the current scan, while those from the convex hull — which contain boundary map points — increase the overlap with more distant points in the scan. This combination reduces overall trajectory drift by maximizing scan-to-map overlap and provides the system with multiple scales of environmental features to align with. Note that keyframes which are classified as both a nearest neighbor and a convex hull index are only used once in the submap.

2.3.4 Adaptive Keyframing

The location of keyframes affects the derived submap and can subsequently influence accuracy and robustness of the odometry. Keyframe nodes are commonly dropped using fixed thresholds (e.g., every 1m or 10° of translational or rotational change) [PMT20, SEM20, SER21a], but the optimal position can be highly dependent on a surrounding environment's structure. More specifically, in large-scale settings, features captured by the point cloud scan are much more prominent and can be depended on for longer periods of time. Conversely, for narrow or small-scale environments, a smaller threshold is necessary to continually capture the small-scale features (i.e., tight corners) in the submap for better localization. Thus, we choose to scale the translational threshold for new keyframes according to the "spaciousness" in the instantaneous point cloud scan, defined as $m_k = \alpha m_{k-1} + \beta M_k$, where M_k is the median Euclidean point distance from the origin to each point in the preprocessed point cloud, $\alpha = 0.95$, $\beta = 0.05$, and m_k is the smoothed signal used to scale the keyframe threshold th_k

at time k such that

$$th_k = \begin{cases} 10\text{m} & \text{if } m_k > 20\text{m} \\ 5\text{m} & \text{if } m_k > 10\text{m} \,\&\, m_k \leq 20\text{m} \\ 1\text{m} & \text{if } m_k > 5\text{m} \,\&\, m_k \leq 10\text{m} \\ 0.5\text{m} & \text{if } m_k \leq 5\text{m} \end{cases} \tag{2.6}$$

with rotational threshold held fixed at 30°. Fig. 2.4B illustrates the effects of this adaptive thresholding, which helps with robustness to changing environmental dimension.

2.4 Algorithmic Implementation

2.4.1 Scan-Stitched Submap Normals

Generalized-ICP involves minimizing the plane-to-plane distance between two clouds, in which these planes are modeled by a computed covariance for each point in the scan. Rather than computing the normals for each point in the submap on every iteration (which can be infeasible for real-time operation), we assume that the set of submap covariances $\mathcal{C}_k^{\mathcal{S}}$ can be approximated by concatenating the normals $\mathcal{C}_{k,n}^{\mathcal{S}}$ from N keyframes which populate the submap such that $\mathcal{C}_k^{\mathcal{S}} \approx \sum_n^N \mathcal{C}_{k,n}^{\mathcal{S}}$. As a consequence, each submap's set of normals need not be explicitly computed, but rather just reconstructed by stitching together those calculated previously.

2.4.2 Data Structure Recycling

Expanding on the above, several algorithmic steps in current LiDAR odometry pipelines can benefit from data structure sharing and reuse, drastically reducing overall system overhead by removing unnecessary and redundant operations. As summarized in Table 2.1, the system requires eight total elements to successfully perform scan-to-scan and scan-to-map matching. This includes kdtrees $\mathcal{T}_k$ used to search for point correspondences and covariance matrices

Table 2.1: Summary of Data Structure Recycling

Element	Scan-to-Scan	Scan-to-Map
$\mathcal{T}_k^{\text{source}}$	build	$\xrightarrow{\text{reuse from S2S}}$
$\mathcal{T}_k^{\text{target}}$	$\mathcal{T}_{k-1}^{\text{source}}$	build when $\mathcal{S}_k \neq \mathcal{S}_{k-1}$
$\mathcal{C}_k^{\text{source}}$	compute	$\xrightarrow{\text{reuse from S2S}}$
$\mathcal{C}_k^{\text{target}}$	$\mathcal{C}_{k-1}^{\text{source}}$	$\sum_n^N \mathcal{C}_{k,n}^{\mathcal{S}}$

$\mathcal{C}_k$ for GICP alignment for both source and target clouds in each scan-matching process.

Out of the four required kdtrees data structures, only two need to be built explicitly. That is, the tree for the source (input) cloud $\mathcal{T}_k^{\text{source}}$ can be built just once per scan acquisition and shared between both modules (as the same scan is used for both sources). For the scan-to-scan target tree $\mathcal{T}_k^{\text{target}}$, this is simply just the previous iteration's scan-to-scan source tree $\mathcal{T}_{k-1}^{\text{source}}$ and thus can be propagated. The scan-to-map target tree needs to be built explicitly, but since the submap is derived from a set of keyframes, this build only needs to be performed when the set of selected keyframes via our kNN and convex hull strategy changes from one iteration to the next, such that $\mathcal{S}_k \neq \mathcal{S}_{k-1}$. Otherwise, the data structure can just be reused again for additional computational savings. Point covariances $\mathcal{C}_k$ needed for GICP, on the other hand, only need to be computed once per scan aquisition, and its data can be shared directly in the other three instances.

2.4.2.1 Dual NanoGICP

To facilitate the cross-talking of data between scan-matching modules, we developed NanoGICP, a custom iterative closest point solver which combines the FastGICP [KYO21c] and NanoFLANN [BR14] open-source packages with additional modifications for data structure sharing as described before. In particular, NanoGICP uses NanoFLANN to efficiently build kdtree data structures, which are subsequently used for point cloud correspondence matching by FastGICP. In practice, data structure sharing is performed between two separate NanoG-

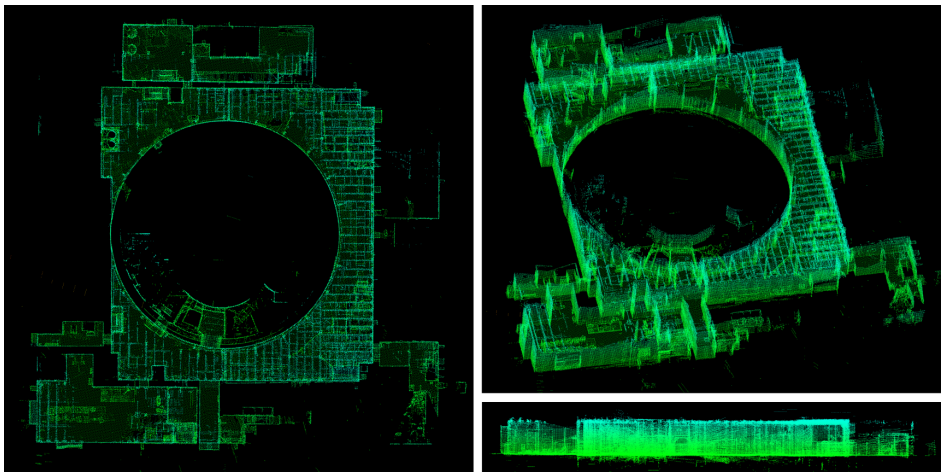

Figure 2.5: **Alpha Course Map.** Different views and angles of the dense 3D point cloud map generated using our DLO algorithm on the Urban Alpha dataset. Estimated positions at each timestamp were used to transform the provided scan into a world frame; this was performed for all scans across the dataset and concatenated / voxel filtered to generated the above images.

ICP instantiations with different hyperparameters — one to target each scan-matching problem — and done procedurally as detailed in Algorithm 1.

2.5 Experimental Results

2.5.1 Component Evaluation

To investigate the impact of our system's components, including keyframe-based submapping, submap normal approximation, and the reuse of data structures, we compare each component with its counterpart using the Alpha Course dataset from the Urban circuit of the DARPA Subterranean Challenge. This dataset contains LiDAR scans from a Velodyne VLP-16 sensor, in addition to IMU measurements from a VectorNav VN-100, collected across 60 minutes in an abandoned powerplant located in Elma, WA which contains multiple perceptual challenges such as large or self-similar scenes (Fig. 2.5). For these component-wise evaluations, data was processed using a 4-core Intel i7 1.30GHz CPU.

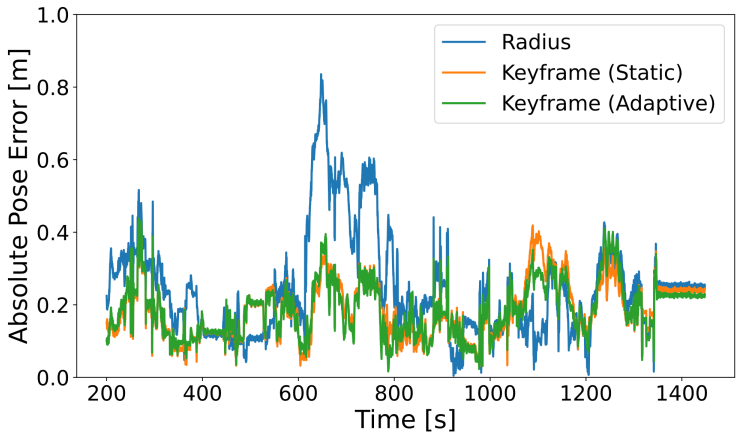

Figure 2.6: **Error Comparison.** The absolute pose error plotted across a 1200s window of movement, showing the difference between radius and keyframe submapping schemes. Keyframe-based approaches do not have the range restriction that radius-based approaches inherently contain, which directly translates to a lower error in odometry due to more perceptive submapping. Note that adaptive keyframing primarily helps with reliability against changes in environmental dimension (Fig. 2.9).

2.5.1.1 Keyframe-Based Submapping

We compared the absolute pose error (APE), processing time, and CPU load across three submapping schemes, including: radius-based ($r = 10$m), keyframe-based with a 1m static threshold, and keyframe-based with adaptive thresholding. For keyframe-based variants, we used 10 nearest-neighbor and 10 convex hull keyframes for submap derivation. From Fig. 2.6, the influence of our approach is clear: submapping in keyframe-space can significantly reduce positional error by considering more distant points that would otherwise be outside the scope of a radius-based approach. These additional points influence the outcome of the GICP optimization process as they are considered during error minimization for the optimal transform; this is especially important in purely frontend-based odometry, since any additional error in pose can quickly propagate over time due to drift. Processing time and CPU load showed similar trends: radius-based processed each scan notably slower at 74.2ms per scan with an average of 37.5% CPU load as compared to 21.6ms / 10.2% and 19.1ms / 9.1% for static and adaptive schemes, respectively.

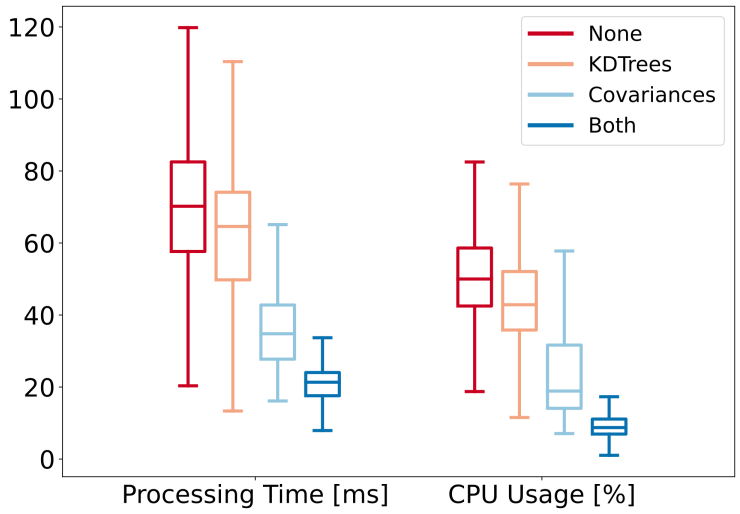

Figure 2.7: **Ablation Study of Data Recycling Schemes.** Box plots of the processing time and CPU usage for four different data recycling schemes, ranging from no data structure reuse to partial reuse and full reuse.

Table 2.2: Dropped LiDAR Scans per Recycling Scheme

	None	KDTrees	Covariances	Both
% Scans	9.37%	4.51%	0.00%	0.00%

2.5.1.2 Data Structure Recycling

To evaluate the effectiveness of data reusage, we measured and compared the processing time and CPU usage between different recycling schemes via a box plot (Fig. 2.7) and percentage of dropped scans over the dataset (Table 2.2). In a naive system which explicitly calculates each kdtree and cloud covariance, computation time exceeded LiDAR rate (10Hz for Velodyne) with a high average of 69.8ms per scan and nearly 10% of scans dropped due to high processing time. Recycling kdtrees but not covariances provides a slight improvement in processing time and CPU percentage, while recycling covariances but not kdtrees provides a more prominent performance boost; this is reasonable since our covariance recycling scheme is more aggressive than kdtree reusage. Finally, using the full scheme as detailed in Table 2.1 significantly decreases both metrics, with an average processing time of 21.9ms and 9.5%

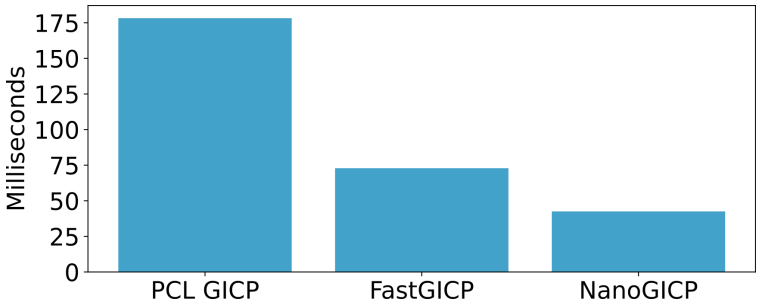

Figure 2.8: **Average Convergence Time.** A comparison of average convergence times across 100 benchmark alignments for each algorithm, including our NanoGICP solver and two other open-source GICP packages.

CPU load, which prevents any LiDAR frames from dropping.

2.5.1.3 NanoGICP

To compare NanoGICP with the state-of-the-art, we use FastGICP's [KYO21c] benchmark alignment code found in the authors' open-source repository. This benchmark measures the average convergence time to align two LiDAR scans across 100 runs, and we compare against PCL's [RC11] GICP implementation as well as FastGICP's multithreaded implementation. Note that we do not compare against the voxelized FastGICP variant, since this method approximates planes with groups of planes and decreases overall accuracy. All tested algorithms were initialized with an identity prior, and as shown in Fig. 2.8, we observed that NanoGICP converged faster on average (42.53ms) when compared to FastGICP (72.88ms) and PCL's GICP (178.24ms).

2.5.2 Benchmark Results

The odometry accuracy and CPU load of DLO was compared to several LiDAR and LiDAR-IMU odometry methods — including BLAM [Nel], Cartographer [HKR16], LIO-Mapping [YCL19], LOAM [ZS14], and LOCUS [PMT20] — using the Alpha and Beta course dataset from the Urban Circuit of the Subterranean Challenge (numbers and ground truth retrieved

Table 2.3: Comparison on Benchmark Datasets

Method	Alpha Course (757.4m)				Beta Course (631.5m)				CPU Usage	
	APE [m]			ME [m]	APE [m]			ME [m]	No. of Cores	
	max	mean	std	rmse	max	mean	std	rmse	max	mean
BLAM [Nel]	3.44	1.01	0.94	0.43	3.89	2.27	0.89	1.27	1.14	0.93
Cartographer [HKR16]	5.84	2.91	1.60	1.05	2.64	1.37	0.67	0.31	1.75	0.88
LIO-Mapping [YCL19]	2.12	0.99	0.51	0.45	1.60	1.18	0.22	0.61	1.80	1.53
LOAM [ZS14]	4.33	1.38	1.19	0.60	2.58	2.11	0.44	0.99	1.65	1.41
LOCUS [PMT20]	0.63	0.26	0.18	0.28	1.20	0.58	0.39	0.48	3.39	2.72
DLO	**0.40**	**0.18**	**0.06**	**0.19**	**0.50**	**0.16**	**0.09**	**0.19**	**0.92**	**0.62**

from [PMT20]). We note that LIO-SAM [SEM20] and LVI-SAM [SER21a], two state-of-the-art tightly-coupled approach, could not be tested at the time due to their sensitive calibration procedure and strict input data requirements. We observed that our method's CPU load was measured to be far lower than any other algorithm, using less than one core both on average and at its peak. This is likely a result how our system derives its submap, in addition to the extensive reuse of internal data structures. This observation can also explain DLO's much lower absolute pose error (APE) and mean error (ME), with similar trends in the relative pose error. With this faster processing time, our method outperformed all other methods in both Alpha and Beta courses, having more than twice the accuracy in the Beta course for max, mean and standard deviation, even without motion distortion correction. In addition to our more permissive submapping approach, we are less likely to drop frames than other methods and have the processing capital to match the dense point clouds at a higher resolution.

2.5.3 Field Experiments

We additionally tested and implemented our solution on several custom robotic platforms for real-world field operation. Specifically, we integrated DLO onto an aerial vehicle (Fig. 2.1A) with an Ouster OS1 and a Boston Dynamics Spot (Fig. 2.1B) with a Velodyne VLP-16. Both systems contained a VectorNav VN-100 IMU rigidly mounted below the base of the LiDAR and processed data on an Intel NUC Board NUC7i7DNBE 1.9GHz CPU. We conducted both manual and autonomous traversals in two perceptually-challenging environments: in an underground limestone cave in Lexington, KY and at an abandoned subway in Los Angeles, CA (Fig. 2.9). Both locations contained environmental properties which often challenge perceptual systems, including poor lighting conditions, featureless corridors, and the presence of particulates such as dust or fog. Despite traversing over 850m across three different levels in the abandoned subway, our system reported only a 10cm end-to-end drift, largely owing to DLO's robust keyframing scheme which adapted to large and small spaces. Our tests in

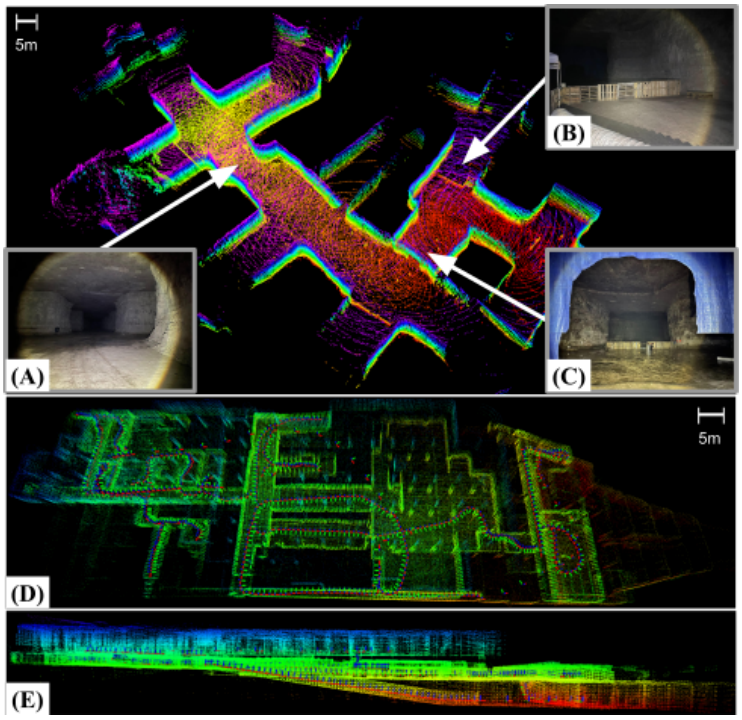

Figure 2.9: **Extreme Environments.** *Top:* A section of an underground mine in Lexington, KY mapped autonomously using our custom drone while running DLO. This environment contained challenging conditions such as: (A) low illuminance, (B) object obstructions, and (C) wet and muddy terrain. *Bottom:* Top-down (D) and side (E) views of the three levels of an abandoned subway located in Downtown Los Angeles, CA mapped via DLO using a Velodyne VLP-16 on a quadruped. In this run, we manually tele-operated the legged robot to walk up, down, and around each floor for a total of 856m.

the underground mine showed similar promise: while this environment lacked any external lighting deep within the cave, DLO could still reliably track our aerial vehicle across 348m of autonomous flight. These results demonstrate the real-world reliability of our method.

2.6 Discussion

This chapter presented Direct LiDAR Odometry (DLO), a lightweight and accurate front-tend localization solution with minimal computational overhead for long-term traversals in extreme environments. A key innovation which distinguishes our work from others is how

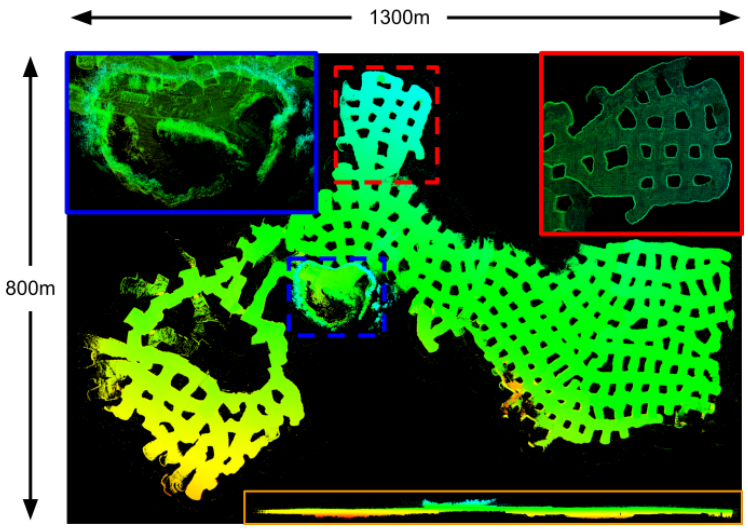

Figure 2.10: **Mega Cavern.** Different views of the Mega Cavern in Louisville, KY mapped by our DLO algorithm, with a total estimated trajectory of 9057.66m. Data is courtesy of Team Explorer.

we efficiently derive a local submap for global pose refinement using a database of keyframe-point cloud pairs. This in turn permits a substantial number of solver data structures to be shared and reused between system modules, all of which is facilitated using our custom NanoGICP cloud registration package. We demonstrate the reliability of our approach through benchmarks and extensive field experiments on multiple platforms operating in large-scale perceptually-challenging environments, and we invite others to use and evaluate our open-source code. DLO was developed for and used on NASA JPL's Team CoSTAR's fleet of quadrotors in the DARPA Subterranean Challenge (Fig. 2.10), and in the future we are interested in tighter IMU integration as well as motion distortion correction.

CHAPTER 3

Direct LiDAR-Inertial Odometry: Lightweight LIO with Continuous-Time Motion Correction

Aggressive motions from agile flights or traversing irregular terrain induce motion distortion in LiDAR scans that can degrade state estimation and mapping. Some methods exist to mitigate this effect, but they are still too simplistic or computationally costly for resource-constrained mobile robots. To this end, this chapter presents Direct LiDAR-Inertial Odometry (DLIO), a lightweight LiDAR-inertial odometry algorithm with a new coarse-to-fine approach in constructing continuous-time trajectories for precise motion correction. The key to our method lies in the construction of a set of analytical equations which are parameterized solely by time, enabling fast and parallelizable point-wise deskewing. This method is feasible only because of the strong convergence properties in our nonlinear geometric observer, which provides provably correct state estimates for initializing the sensitive IMU integration step. Moreover, by simultaneously performing motion correction and prior generation, and by directly registering each scan to the map and bypassing scan-to-scan, DLIO's condensed architecture is nearly 20% more computationally efficient than the current state-of-the-art with a 12% increase in accuracy. We demonstrate DLIO's superior localization accuracy, map quality, and lower computational overhead as compared to four state-of-the-art algorithms through extensive tests using multiple public benchmark and self-collected datasets.

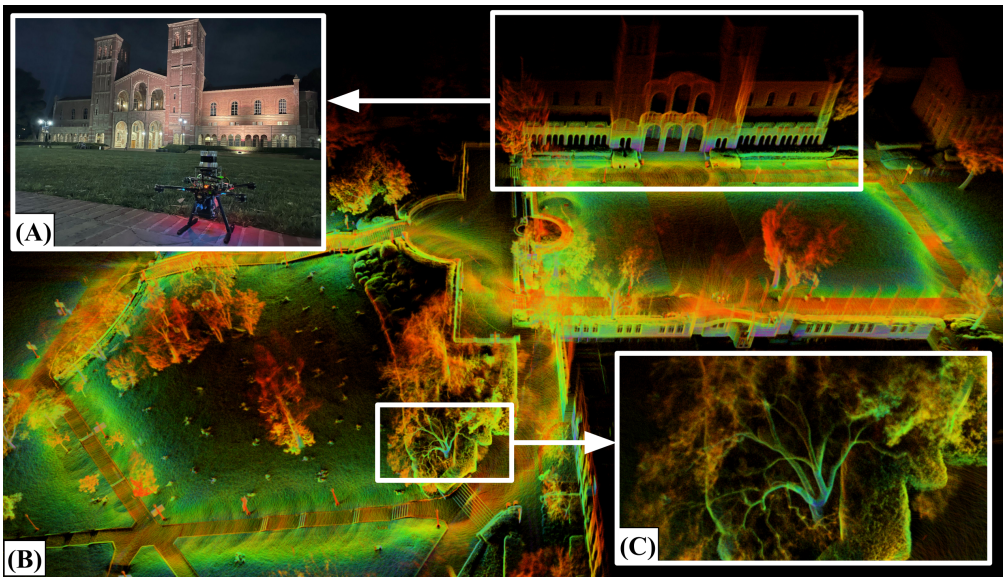

Figure 3.1: **Real-time Localization and Dense Mapping.** DLIO generates detailed maps by reliably estimating robot pose, velocity, and sensor biases in real-time. (A) Our custom aerial vehicle next to UCLA's Royce Hall. (B) A bird's eye view of Royce Hall and its surroundings generated by DLIO. (C) A close-up of a tree, showcasing the fine detail that DLIO is able to capture in its output map. Color denotes intensity of point return.

3.1 Overview

Accurate real-time state estimation and mapping are necessary capabilities for mobile robots to perceive, plan, and navigate through unknown environments. LiDAR-based localization has recently become a viable option for many mobile platforms, such as drones, due to more compact and accurate sensors. As a result, researchers have developed several new LiDAR odometry (LO) and LiDAR-inertial odometry (LIO) algorithms which often outperform vision-based approaches due to the superior range and depth measurement accuracy of a LiDAR. However, there are still fundamental challenges in developing reliable and accurate LiDAR-centric algorithms [CCC16], especially for robots that execute agile maneuvers or traverse uneven terrain. In particular, such aggressive movements can induce significant distortion in the point cloud which corrupts the scan-matching process, resulting in severe or catastrophic localization error and map deformation.

Existing algorithms which attempt to compensate for this effect may work well in structured environments for non-holonomic systems (e.g., autonomous driving), but their performance can degrade under irregular conditions due to simplistic motion models, loss in precision from discretization, and/or computational inefficiencies. For instance, works such as [ZS14, SE18, SEM20] assume constant velocity during scan acquisition which may work well for simple, predictable trajectories, but this quickly breaks down under significant acceleration. On the other hand, [XCH21] and [XCH22] use a back-propagation technique to mitigate distortion for each point, but their method may induce a loss in precision from accumulating integration error over time. More recently, continuous-time methods attempt to fit a smooth trajectory over a set of control points [PMK18, RKC22] or augment scan-matching optimization with additional free variables [DDJ22], but such methods still hold strong assumptions on the trajectory (i.e., smooth movement) or may be too computationally costly for weight-limited platforms.

To this end, we present Direct LiDAR-Inertial Odometry (DLIO), which proposes a fast, coarse-to-fine approach to construct each inter-sweep trajectory for accurate motion correction. A discrete set of poses is first computed via numerical integration on IMU measurements, and smooth trajectories between measurement samples are subsequently built via analytical, continuous-time equations to query each unique per-point deskewing transform. Our approach is fast in that we solve a set of analytical equations rather than an optimization problem (e.g., spline-fitting), parameterized solely by the timestamp of the point which can be easily parallelized. Our approach is also accurate in that we use a higher-order motion model to represent the underlying system dynamics which can capture high-frequency movements that may otherwise be lost in methods that attempt to fit a smooth trajectory to a set of control points. This approach is built into a simplified LIO architecture which performs motion correction and GICP prior construction in one shot, in addition to performing scan-to-map alignment directly without the intermediary scan-to-scan; this is all possible through the strong convergence guarantees of our novel geometric observer with provably

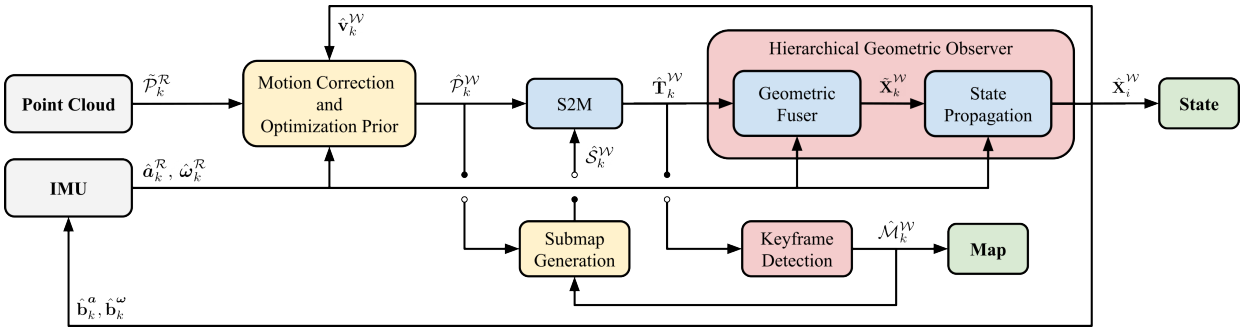

Figure 3.2: **System Architecture.** DLIO's lightweight architecture combines motion correction and prior construction into a single step, in addition to removing the scan-to-scan module previously required for LiDAR-based odometry. Point-wise continuous-time integration in $\mathcal{W}$ ensures maximum fidelity of the corrected cloud and is registered onto the robot's map by a custom GICP-based scan-matcher. The system's state is subsequently updated by a nonlinear geometric observer with strong convergence properties [Lop23], and these estimates of pose, velocity, and bias then initialize the next iteration.

correct state estimates.

DLIO is a fast and reliable odometry algorithm that provides accurate localization and detailed 3D mapping (Fig. 3.1) with four main contributions. First, we develop a new coarse-to-fine technique for constructing continuous-time trajectories, in which a set of analytical equations with a constant jerk and angular acceleration motion model is derived for fast and parallelizable point-wise motion correction. Second, a novel condensed architecture is presented which combines motion correction and prior construction into one step and directly performs scan-to-map registration, significantly reducing overall computational overhead of the algorithm. Third, we leverage a new nonlinear geometric observer [Lop23] that possesses strong performance guarantees—critical for achieving the first two contributions—in the pipeline to reliably generate accurate estimates of the robot's full state with minimal computational complexity. Finally, the efficacy of our approach is verified through extensive experimental results using multiple datasets against the state-of-the-art.

3.2 System Description

DLIO is a lightweight LIO algorithm that generates robot state estimates and geometric maps through a unique architecture that contains two main components with three innovations (Fig. 3.2). The first is a fast scan-matcher which registers dense, motion-corrected point clouds onto the robot's map by performing alignment with an extracted local submap. Point-wise continuous-time integration in $\mathcal{W}$ ensures maximum image fidelity of the corrected cloud while simultaneously building in a prior for GICP optimization. In the second, a nonlinear geometric observer [Lop23] updates the system's state with the first component's pose output to provide high-rate and provably correct estimates of pose, velocity, and sensor biases which converge globally. These estimates then initialize the next iteration of motion correction, scan-matching, and state update.

3.2.1 Notation

Let the point cloud for a single LiDAR sweep initiated at time t_k be denoted as $\mathcal{P}_k$ and indexed by k. The point cloud $\mathcal{P}_k$ is composed of points $p_k^n \in \mathbb{R}^3$ that are measured at a time Δt_k^n relative to the start of the scan and indexed by $n = 1, \ldots, N$ where N is the total number of points in the scan. The world frame is denoted as $\mathcal{W}$ and the robot frame as $\mathcal{R}$ located at its center of gravity, with the convention that x points forward, y left, and z up. The IMU's coordinate system is denoted as $\mathcal{B}$ and the LiDAR's as $\mathcal{L}$, and the robot's state vector $\mathbf{X}_k$ at index k is defined as the tuple

$$\mathbf{X}_k = \left[\, \mathbf{p}_k^{\mathcal{W}}, \, \mathbf{q}_k^{\mathcal{W}}, \, \mathbf{v}_k^{\mathcal{W}}, \, \mathbf{b}_k^{\boldsymbol{a}}, \, \mathbf{b}_k^{\boldsymbol{\omega}} \, \right]^\top , \tag{3.1}$$

where $\mathbf{p}^{\mathcal{W}} \in \mathbb{R}^3$ is the robot's position, $\mathbf{q}^{\mathcal{W}}$ is the orientation encoded by a four vector quaternion on $\mathbb{S}^3$ under Hamilton notation, $\mathbf{v}^{\mathcal{W}} \in \mathbb{R}^3$ is the robot's velocity, $\mathbf{b}^{\boldsymbol{a}} \in \mathbb{R}^3$ is the accelerometer's bias, and $\mathbf{b}^{\boldsymbol{\omega}} \in \mathbb{R}^3$ is the gyroscope's bias. Measurements $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{\omega}}$ from an

IMU are modeled as

$$\hat{\boldsymbol{a}}_i = (\boldsymbol{a}_i - \boldsymbol{g}) + \mathbf{b}_i^{\boldsymbol{a}} + \mathbf{n}_i^{\boldsymbol{a}} \,, \tag{3.2}$$

$$\hat{\boldsymbol{\omega}}_i = \boldsymbol{\omega}_i + \mathbf{b}_i^{\boldsymbol{\omega}} + \mathbf{n}_i^{\boldsymbol{\omega}} \,, \tag{3.3}$$

and indexed by $i = 1, \ldots, M$ for M measurements between clock times $t_{k\text{-}1}$ and t_k. With some abuse of notation, indices k and i occur at LiDAR and IMU rate, respectively, and will be written this way for simplicity unless otherwise stated. Raw sensor measurements $\boldsymbol{a}_i$ and $\boldsymbol{\omega}_i$ contain bias $\mathbf{b}_i$ and white noise $\mathbf{n}_i$, and $\boldsymbol{g}$ is the rotated gravity vector. In this chapter, we address the following problem: given an accumulated point cloud $\mathcal{P}_k$ from a LiDAR and measurements $\boldsymbol{a}_i$ and $\boldsymbol{\omega}_i$ sampled between each received scan by an IMU, estimate the robot's state $\hat{\mathbf{X}}_i^{\mathcal{W}}$ and the geometric map $\hat{\mathcal{M}}_k^{\mathcal{W}}$.

3.2.2 Preprocessing

The inputs to DLIO are a dense 3D point cloud collected by a modern 360° mechanical LiDAR, such as an Ouster or a Velodyne (10-20Hz), in addition to time-synchronized linear acceleration and angular velocity measurements from a 6-axis IMU at a much higher rate (100-500Hz). Prior to downstream tasks, all sensor data is transformed to be in $\mathcal{R}$ located at the robot's center of gravity via extrinsic calibration. For IMU, effects of displacing linear acceleration measurements on a rigid body must be considered if the sensor is not coincident with the center of gravity; this is done by considering all contributions of linear acceleration at $\mathcal{R}$ via the cross product between angular velocity and the offset of the IMU. To minimize information loss, we do not preprocess the point cloud except for a box filter of size 1m^3 around the origin to remove points that may be from the robot itself, and a light voxel filter for higher resolution clouds. This distinguishes our work from others that either attempt to detect features (e.g., corners, edges, or surfels) or heavily downsamples the cloud through a voxel filter.

Algorithm 2: Direct LiDAR-Inertial Odometry

input: $\hat{\mathbf{X}}_{k-1}^{\mathcal{W}}$, $\mathcal{P}_k^{\mathcal{L}}$, $\boldsymbol{a}_k^{\mathcal{B}}$, $\boldsymbol{\omega}_k^{\mathcal{B}}$; **output:** $\hat{\mathbf{X}}_i^{\mathcal{W}}$, $\hat{\mathcal{M}}_k^{\mathcal{W}}$

// LiDAR Callback Thread

while $\mathcal{P}_k^{\mathcal{L}} \neq \emptyset$ **do**

  // initialize points and transform to $\mathcal{R}$

  $\tilde{\mathcal{P}}_k^{\mathcal{R}} \leftarrow$ initializePointCloud($\mathcal{P}_k^{\mathcal{L}}$) (Sec. 3.2.2);

  // continuous-time motion correction

  for $\hat{\boldsymbol{a}}_i^{\mathcal{R}}, \hat{\boldsymbol{\omega}}_i^{\mathcal{R}}$ *between* t_{k-1} *and* t_k **do**

    $\hat{\mathbf{p}}_i, \hat{\mathbf{v}}_i, \hat{\mathbf{q}}_i \leftarrow$ discreteInt($\hat{\mathbf{X}}_{k-1}^{\mathcal{W}}$, $\hat{\boldsymbol{a}}_{i-1}^{\mathcal{R}}$, $\hat{\boldsymbol{\omega}}_{i-1}^{\mathcal{R}}$) (3.4);

    $\hat{\mathbf{T}}_i^{\mathcal{W}} = [\, \hat{\mathbf{R}}(\hat{\mathbf{q}}_i) \,|\, \hat{\mathbf{p}}_i \,]$;

  end

  for $p_k^n \in \tilde{\mathcal{P}}_k^{\mathcal{R}}$ **do**

    $\hat{\mathbf{T}}_n^{\mathcal{W}*} \leftarrow$ continuousInt($\hat{\mathbf{T}}_i^{\mathcal{W}*}$, t_n) (3.5);

    $\hat{p}_k^n = \hat{\mathbf{T}}_n^{\mathcal{W}*} \otimes p_k^n$; $\hat{\mathcal{P}}_k^{\mathcal{W}}$.append($\hat{p}_k^n$);

  end

  // scan-to-map registration

  $\hat{\mathcal{S}}_k^{\mathcal{W}} \leftarrow$ generateSubmap($\hat{\mathcal{M}}_k^{\mathcal{W}}$) [CLA22a]

  $\hat{\mathbf{T}}_k^{\mathcal{W}} \leftarrow$ GICP($\hat{\mathcal{P}}_k^{\mathcal{W}}$, $\hat{\mathcal{S}}_k^{\mathcal{W}}$) (3.6);

  // geometric observer: state update

  $\hat{\mathbf{X}}_k^{\mathcal{W}} \leftarrow$ update($\hat{\mathbf{T}}_k^{\mathcal{W}}$, Δt_k^+) (Sec. 3.3.3);

  // update keyframe map

  if $\hat{\mathcal{P}}_k^{\mathcal{W}}$ *is a keyframe* **then** $\hat{\mathcal{M}}_k^{\mathcal{W}} \leftarrow \hat{\mathcal{M}}_{k-1}^{\mathcal{W}} \oplus \hat{\mathcal{P}}_k^{\mathcal{W}}$;

  return $\hat{\mathbf{X}}_k^{\mathcal{W}}$, $\hat{\mathcal{M}}_k^{\mathcal{W}}$

end

// IMU Callback Thread

while $\boldsymbol{a}_i^{\mathcal{B}} \neq \emptyset$ *and* $\boldsymbol{\omega}_i^{\mathcal{B}} \neq \emptyset$ **do**

  // apply biases and transform to $\mathcal{R}$

  $\hat{\boldsymbol{a}}_i^{\mathcal{R}}, \hat{\boldsymbol{\omega}}_i^{\mathcal{R}} \leftarrow$ initializeImu($\boldsymbol{a}_i^{\mathcal{B}}$, $\boldsymbol{\omega}_i^{\mathcal{B}}$) (Sec. 3.2.2);

  // geometric observer: state propagation

  $\hat{\mathbf{X}}_i^{\mathcal{W}} \leftarrow$ propagate($\hat{\mathbf{X}}_k^{\mathcal{W}}$, $\hat{\boldsymbol{a}}_i^{\mathcal{R}}$, $\hat{\boldsymbol{\omega}}_i^{\mathcal{R}}$, Δt_i^+) (Sec. 3.3.3);

  return $\hat{\mathbf{X}}_i^{\mathcal{W}}$

end

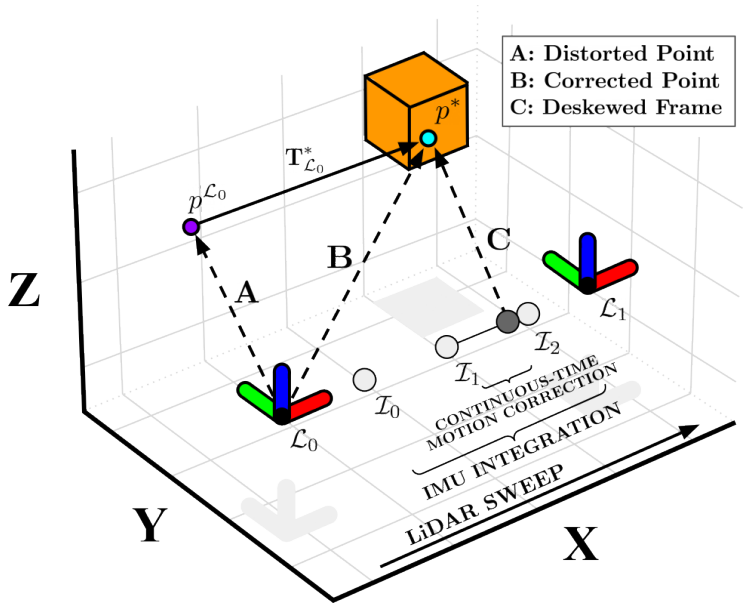

Figure 3.3: **Coarse-to-Fine Point Cloud Deskewing.** A distorted point $p^{\mathcal{L}_0}$ (A) is deskewed through a two-step process which first integrates IMU measurements between scans, then solves for a unique transform in continuous-time (C) for the original point which deskews $p^{\mathcal{L}_0}$ to p^* (B).

3.3 Condensed LIO Architecture

3.3.1 Continuous-Time Motion Correction with Joint Prior

Point clouds from spinning LiDAR sensors suffer from motion distortion during movement due to the rotating laser array collecting points at different instances during a sweep. Rather than assuming simple motion (i.e., constant velocity) during sweep that may not accurately capture fine movement, we instead use a more accurate constant jerk and angular acceleration model to compute a unique transform for each point via a two-step coarse-to-fine propagation scheme. This strategy aims to minimize the errors that arise due to the sampling rate of the IMU and the time offset between IMU and LiDAR point measurements. Trajectory throughout a sweep is first coarsely constructed through numerical IMU integration [FCD16], which is subsequently refined by solving a set of analytical continuous-time equations in $\mathcal{W}$ (Fig. 3.3).

Let t_k be the clock time of the received point cloud $\mathcal{P}_k^{\mathcal{R}}$ with N number of accumulated points within the time period, and let $t_k + \Delta t_k^n$ be the timestamp of a point p_k^n in the cloud. To approximate each point's location in $\mathcal{W}$, we first integrate IMU measurements between $t_{k\text{-}1}$ and $t_k + \Delta t_k^N$ via

$$\begin{aligned}
\hat{\mathbf{p}}_i &= \hat{\mathbf{p}}_{i\text{-}1} + \hat{\mathbf{v}}_{i\text{-}1}\Delta t_i + \tfrac{1}{2}\hat{\mathbf{R}}(\hat{\mathbf{q}}_{i\text{-}1})\hat{\boldsymbol{a}}_{i\text{-}1}\Delta t_i^2 + \tfrac{1}{6}\hat{\boldsymbol{j}}_i\Delta t_i^3\,, \\
\hat{\mathbf{v}}_i &= \hat{\mathbf{v}}_{i\text{-}1} + \hat{\mathbf{R}}(\hat{\mathbf{q}}_{i\text{-}1})\hat{\boldsymbol{a}}_{i\text{-}1}\Delta t_i\,, \\
\hat{\mathbf{q}}_i &= \hat{\mathbf{q}}_{i\text{-}1} + \tfrac{1}{2}(\hat{\mathbf{q}}_{i\text{-}1} \otimes \hat{\boldsymbol{\omega}}_{i\text{-}1})\Delta t_i + \tfrac{1}{4}(\hat{\mathbf{q}}_{i\text{-}1} \otimes \hat{\boldsymbol{\alpha}}_i)\Delta t_i^2\,,
\end{aligned} \tag{3.4}$$

for $i = 1, \ldots, M$ for M number of IMU measurements between two scans, where $\hat{\boldsymbol{j}}_i = \frac{1}{\Delta t_i}(\hat{\mathbf{R}}(\hat{\mathbf{q}}_i)\hat{\boldsymbol{a}}_i - \hat{\mathbf{R}}(\hat{\mathbf{q}}_{i\text{-}1})\hat{\boldsymbol{a}}_{i\text{-}1})$ and $\hat{\boldsymbol{\alpha}}_i = \frac{1}{\Delta t_i}(\hat{\boldsymbol{\omega}}_i - \hat{\boldsymbol{\omega}}_{i\text{-}1})$ are the estimated linear jerk and angular acceleration, respectively. The set of homogeneous transformations $\hat{\mathbf{T}}_i^{\mathcal{W}} \in \mathbb{SE}(3)$ that correspond to $\hat{\mathbf{p}}_i$ and $\hat{\mathbf{q}}_i$ then define the coarse, *discrete*-time trajectory during a sweep. Then, an analytical, *continuous*-time solution from the nearest preceding transformation to each point p_k^n recovers the point-specific deskewing transform $\hat{\mathbf{T}}_n^{\mathcal{W}*}$, such that

$$\begin{aligned}
\hat{\mathbf{p}}^*(t) &= \hat{\mathbf{p}}_{i\text{-}1} + \hat{\mathbf{v}}_{i\text{-}1}t + \tfrac{1}{2}\hat{\mathbf{R}}(\hat{\mathbf{q}}_{i\text{-}1})\hat{\boldsymbol{a}}_{i\text{-}1}t^2 + \tfrac{1}{6}\hat{\boldsymbol{j}}_i t^3\,, \\
\hat{\mathbf{q}}^*(t) &= \hat{\mathbf{q}}_{i\text{-}1} + \tfrac{1}{2}(\hat{\mathbf{q}}_{i\text{-}1} \otimes \hat{\boldsymbol{\omega}}_{i\text{-}1})t + \tfrac{1}{4}(\hat{\mathbf{q}}_{i\text{-}1} \otimes \hat{\boldsymbol{\alpha}}_i)t^2\,,
\end{aligned} \tag{3.5}$$

where $i{-}1$ and i correspond to the closest preceding and successive IMU measurements, respectively, t is the timestamp between point p_k^n and the closest preceding IMU, and $\hat{\mathbf{T}}_n^{\mathcal{W}*}$ is the transformation corresponding to $\hat{\mathbf{p}}^*$ and $\hat{\mathbf{q}}^*$ for p_k^n (Fig. 3.4). Note that (3.5) is parameterized only by t and therefore a transform can be queried for any desired time to construct a continuous-time trajectory.

The result of this two-step procedure is a motion-corrected point cloud that is also approximately aligned with the map in $\mathcal{W}$, which therefore inherently incorporates the optimization prior used for GICP (Sec. 3.3.2). Importantly, (3.4) and (3.5) depend on the accuracy of $\hat{\mathbf{v}}_0^{\mathcal{W}}$, the initial estimate of velocity, $\mathbf{b}_k^a$ and $\mathbf{b}_k^\omega$, the estimated IMU biases, in addition to

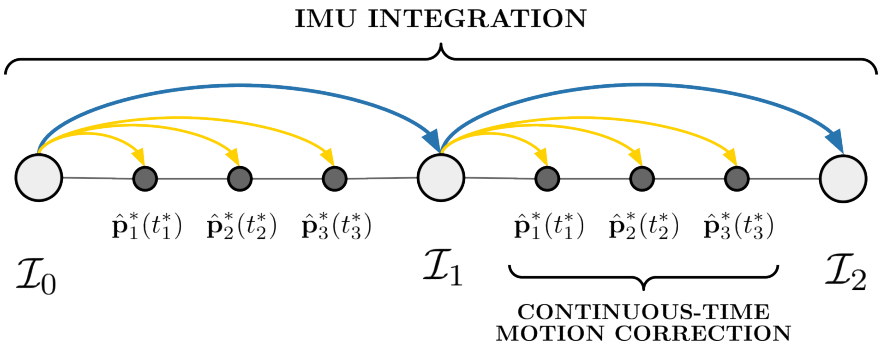

Figure 3.4: **Continuous-Time Motion Correction.** For each point in a cloud, a unique transform is computed by solving a set of closed-form motion equations initialized at the closest preceeding IMU measurement. This provides accurate and parallelizable continuous–time motion correction.

an accurate initial body orientation $\hat{\mathbf{q}}_0$ (to properly compensate for the gravity vector) at the time of motion correction. We therefore emphasize that, a key to the reliability of our approach is the *guaranteed global convergence* of these terms by leveraging DLIO's nonlinear geometric observer [Lop23], provided that scan-matching returns an accurate solution.

3.3.2 Scan-to-Map Registration

By simultaneously correcting for motion distortion and incorporating the GICP optimization prior into the point cloud, DLIO can directly perform scan-to-map registration and bypass the scan-to-scan procedure required in previous methods. This registration is cast as a nonlinear optimization problem which minimizes the distance of corresponding points/planes between the current scan and an extracted submap. Let $\hat{\mathcal{P}}_k^{\mathcal{W}}$ be the corrected cloud in $\mathcal{W}$ and $\hat{\mathcal{S}}_k^{\mathcal{W}}$ be the extracted keyframe-based submap via [CLA22a]. Then, the objective of scan-to-map optimization is to find a transformation $\Delta\hat{\mathbf{T}}_k$ which better aligns the point cloud such that

$$\Delta\hat{\mathbf{T}}_k = \underset{\Delta\mathbf{T}_k}{\arg\min}\ \mathcal{E}\left(\Delta\mathbf{T}_k\hat{\mathcal{P}}_k^{\mathcal{W}},\ \hat{\mathcal{S}}_k^{\mathcal{W}}\right), \tag{3.6}$$

where the GICP residual error $\mathcal{E}$ is defined as

$$\mathcal{E}\left(\Delta\mathbf{T}_k\hat{\mathcal{P}}_k^{\mathcal{W}}, \hat{\mathcal{S}}_k^{\mathcal{W}}\right) = \sum_{c\in\mathcal{C}} d_c^\top \left(C_{k,c}^{\mathcal{S}} + \Delta\mathbf{T}_k C_{k,c}^{\mathcal{P}} \Delta\mathbf{T}_k^\top\right)^{-1} d_c\,,$$

for a set of $\mathcal{C}$ corresponding points between $\hat{\mathcal{P}}_k^{\mathcal{W}}$ and $\hat{\mathcal{S}}_k^{\mathcal{W}}$ at timestep k, $d_c = \hat{s}_k^c - \Delta\mathbf{T}_k\hat{p}_k^c$, $\hat{p}_k^c \in \hat{\mathcal{P}}_k^{\mathcal{W}}$, $\hat{s}_k^c \in \hat{\mathcal{S}}_k^{\mathcal{W}}$, $\forall c \in \mathcal{C}$, and $C_{k,c}^{\mathcal{P}}$ and $C_{k,c}^{\mathcal{S}}$ are the estimated covariance matrices for point cloud $\hat{\mathcal{P}}_k^{\mathcal{W}}$ and submap $\hat{\mathcal{S}}_k^{\mathcal{W}}$, respectively. Then, following [SHT09], this point-to-plane formulation is converted into a plane-to-plane optimization by regularizing covariance matrices $C_{k,c}^{\mathcal{P}}$ and $C_{k,c}^{\mathcal{S}}$ with $(1, 1, \epsilon)$ eigenvalues, where ϵ represents the low uncertainty in the surface normal direction. The resulting $\Delta\hat{\mathbf{T}}_k$ represents an optimal correction transform which better globally aligns the prior-transformed scan $\hat{\mathcal{P}}_k^{\mathcal{W}}$ to the submap $\hat{\mathcal{S}}_k^{\mathcal{W}}$, so that $\hat{\mathbf{T}}_k^{\mathcal{W}} = \Delta\hat{\mathbf{T}}_k\hat{\mathbf{T}}_M^{\mathcal{W}}$ (where $\hat{\mathbf{T}}_M^{\mathcal{W}}$ is the last point's IMU integration) is the globally-refined robot pose which is used for map construction and as the update signal for the nonlinear geometric observer.

3.3.3 Geometric Observer

The transformation $\hat{\mathbf{T}}_k^{\mathcal{W}}$ computed by scan-to-map alignment is fused with IMU measurements to generate a full state estimate $\hat{\mathbf{X}}_k$ via a novel hierarchical nonlinear geometric observer. A full analysis of the observer can be found in [Lop23], but in summary, one can show that $\hat{\mathbf{X}}$ will globally converge to $\mathbf{X}$ in the deterministic setting with minimal computation. The proof utilizes contraction theory to first prove that the quaternion estimate converges exponentially to a region near the true quaternion. The orientation estimate then serves as an input to another contracting observer that estimates translation states. This architecture forms a contracting hierarchy that guarantees the estimates converge to their true values. The strong convergence property of the observer is the main advantage over other fusion schemes, e.g., filtering or pose graph optimization that possess minimal convergence guarantees, and will be important for future theoretical studies on the advantages of our LiDAR

odometry and mapping pipeline. From a practical viewpoint, the observer generates smooth estimates in real-time so it output is also suitable for control. The observer used here is a special case of that in [Lop23].

Let $\gamma_{\ell\in\{1,\dots,5\}}$ be positive constants and Δt_k^+ be the time between GICP poses. If $\mathbf{q}_e \coloneqq (q_e^\circ,\ \vec{q}_e) = \hat{\mathbf{q}}_i^* \otimes \hat{\mathbf{q}}_k$ and $\mathbf{p}_e = \hat{\mathbf{p}}_k - \hat{\mathbf{p}}_i$ (errors between propagated and measured poses) then the state correction takes the form

$$
\begin{aligned}
\hat{\mathbf{q}}_i &\leftarrow \hat{\mathbf{q}}_i + \Delta t_k^+ \, \gamma_1 \, \hat{\mathbf{q}}_i \otimes \begin{bmatrix} 1 - |q_e^\circ| \\ \operatorname{sgn}(q_e^\circ)\, \vec{q}_e \end{bmatrix}, \\
\hat{\mathbf{b}}_i^{\boldsymbol{\omega}} &\leftarrow \hat{\mathbf{b}}_i^{\boldsymbol{\omega}} - \Delta t_k^+ \, \gamma_2 \, q_e^\circ \vec{q}_e \,, \\
\hat{\mathbf{p}}_i &\leftarrow \hat{\mathbf{p}}_i + \Delta t_k^+ \, \gamma_3 \, \mathbf{p}_e \,, \\
\hat{\mathbf{v}}_i &\leftarrow \hat{\mathbf{v}}_i + \Delta t_k^+ \, \gamma_4 \, \mathbf{p}_e \,, \\
\hat{\mathbf{b}}_i^{\boldsymbol{a}} &\leftarrow \hat{\mathbf{b}}_i^{\boldsymbol{a}} - \Delta t_k^+ \, \gamma_5 \, \hat{\mathbf{R}}(\hat{\mathbf{q}}_i)^\top \mathbf{p}_e \,.
\end{aligned}
\tag{3.7}
$$

Note (3.7) is hierarchical as the attitude update (first two eqs.) is completely decoupled from the translation update (last three eqs.). Also, (3.7) is a fully nonlinear update which allows one to guarantee the state estimates are accurate enough to directly perform scan-to-map registration solely with an IMU prior without the need for scan-to-scan.

3.4 Experimental Results

DLIO was evaluated using the Newer College benchmark dataset [RWC20] and data self-collected around the UCLA campus. We compare accuracy and efficiency against four state-of-the-algorithms, namely DLO [CLA22a], CT-ICP [DDJ22], LIO-SAM [SEM20], and FAST-LIO2 [XCH22]. Each algorithm employs a different degree and method of motion compensation, therefore creating an exhaustive comparison to the current state-of-the-art. Aside from extrinsics, default parameters at the time of writing for each algorithm were used

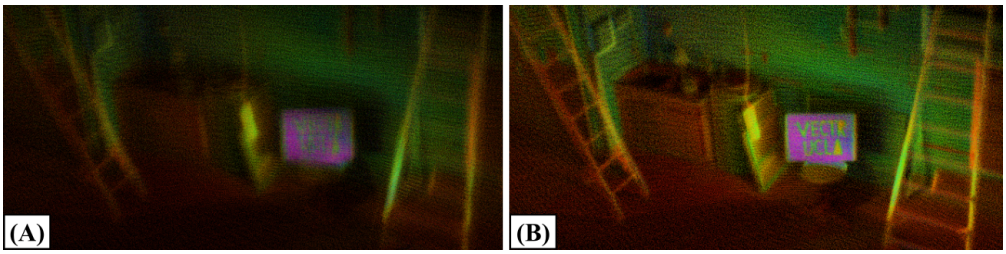

Figure 3.5: **Deskewing Comparison.** Map generated from aggressive maneuvers without (A) and with (B) our motion correction method.

in all experiments unless otherwise noted. Specifically, loop-closures were enabled for LIO-SAM and online extrinsics estimation disabled for FAST-LIO2 to provide the best results of each algorithm. For CT-ICP, voxelization was slightly increased and data playback was slowed down otherwise the algorithm would fail due to significant frame drops. All tests were conducted on a 16-core Intel i7-11800H CPU.

3.4.1 Ablation Study and Comparison of Motion Correction

To investigate the impact of our proposed motion correction scheme, we first conducted an ablation study with varying degrees of deskewing in DLIO using the Newer College dataset [RWC20]. This study ranged from no motion correction (None), to correction using only nearest IMU integration via (3.4) (Discrete), and finally to full continuous-time motion correction via both (3.4) and (3.5) (Continuous) (Table 3.1). Particularly of note is the Dynamic dataset, which contained highly aggressive motions with rotational speeds up to 3.5 rad/s. With no correction, error was the highest among all algorithms at 0.1959 RMSE. With partial correction, error significantly reduced due to scan-matching with more accurate and representative point clouds; however, using the full proposed scheme, we observed an error of only 0.0612 RMSE—the lowest among all tested algorithms. With similar trends for all other datasets, the superior tracking accuracy granted by better motion correction is clear: constructing a unique transform in continuous-time creates a more authentic point cloud than previous methods, which ultimately affects scan-matching and therefore trajectory accuracy.

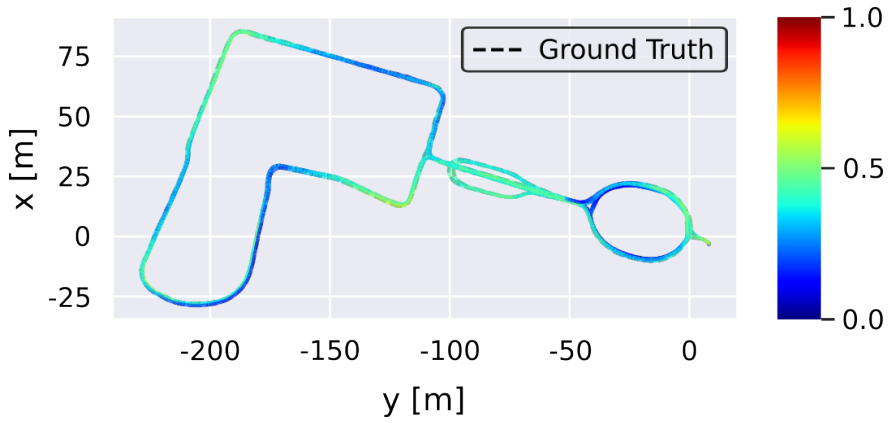

Figure 3.6: **Trajectory of Long Experiment.** DLIO's generated trajectory for the Newer College - Long Experiment. Color indicates absolute pose error.

Fig. 3.5 showcases this empirically: DLIO can capture minute detail that is otherwise lost with simple or no motion correction.

3.4.2 Benchmark Results

3.4.2.1 Newer College Dataset

Trajectory accuracy and average per-scan time of all algorithms were also compared using the original Newer College benchmark dataset [RWC20] via evo [Gru17]. For these tests, we used data from the Ouster's IMU (100Hz) alongside LiDAR data (10Hz) to ensure accurate time synchronization between sensors. For certain Newer College datasets, the first 100 poses were excluded from computing FAST-LIO2's RMSE due to slippage at the start in order to provide a fair comparison. We also compared using the recent extension of the Newer College dataset [ZCF21] and observed similar results, but those results have been omitted due to space constraints. The results are shown in Table 3.1, in which we observed DLIO to produce the lowest trajectory RMSE and lowest overall per-scan computational time (averaged across all five datasets) as compared to the state-of-the-art. Fig. 3.6 illustrates DLIO's low trajectory error compared to ground truth for the Newer College - Long Experiment dataset even after over three kilometers of travel.

Table 3.1: Comparison with Newer College Dataset

Algorithm	Type	Absolute Trajectory Error (RMSE) [m]					Avg Comp. [ms]
		Short (1609.40m)	Long (3063.42m)	Quad (479.04m)	Dynamic (97.20m)	Park (695.68m)	
DLO [CLA22a]	LO	0.4633	0.4125	0.1059	0.1954	0.1846	48.10
CT-ICP [DDJ22]	LO	0.5552	0.5761	0.0981	0.1426	0.1802	412.27
LIO-SAM [SEM20]	LIO	0.3957	0.4092	0.0950	0.0973	0.1761	179.33
FAST-LIO2 [XCH22]	LIO	0.3775	0.3324	0.0879	0.0771	0.1483	42.86
DLIO (None)	LIO	0.4299	0.3988	0.1117	0.1959	0.1821	34.88
DLIO (Discrete)	LIO	0.3803	0.3629	0.0943	0.0798	0.1537	**34.61**
DLIO (Continuous)	LIO	**0.3606**	**0.3268**	**0.0837**	**0.0612**	**0.1196**	35.74

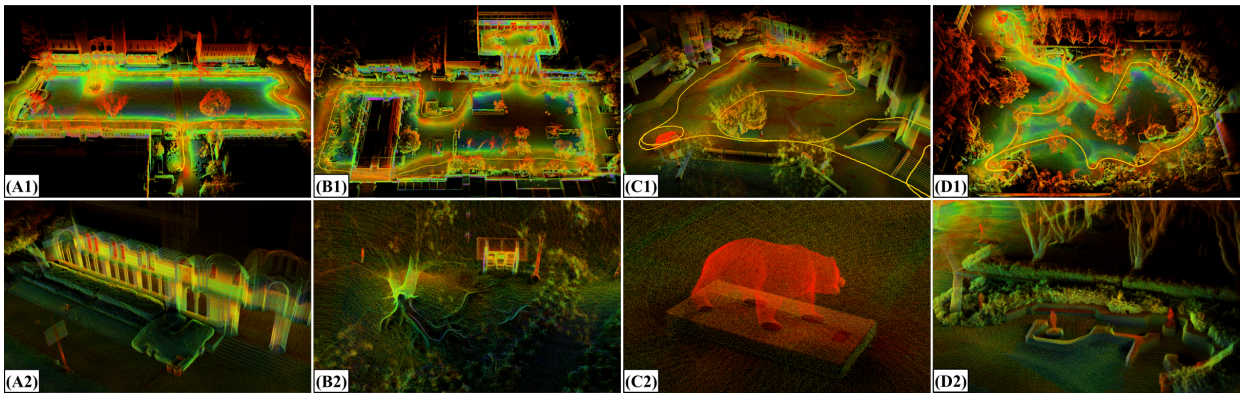

Figure 3.7: **UCLA Campus.** Detailed maps of locations around UCLA in Los Angeles, CA generated by DLIO, including (A) Royce Hall in Dickson Court, (B) Court of Sciences, (C) Bruin Plaza, and (D) the Franklin D. Murphy Sculpture Garden, with both (1) a bird's eye view and (2) a close-up to demonstrate the level of fine detail DLIO can generate. The trajectory taken to generate these maps is shown in yellow in the first row.

3.4.2.2 UCLA Campus Dataset

We additionally collected four large-scale datasets at UCLA for additional comparison (Fig. 3.7). These datasets were gathered by hand-carrying our aerial platform (Fig. 3.1) over 2261.37m of total trajectory. Our sensor suite included an Ouster OS1 (10Hz, 32 channels recorded with a 512 horizontal resolution) and a 6-axis InvenSense MPU-6050 IMU located approximately 0.1m below it. We note here that this IMU can be purchased for approximately $10, demonstrating that LIO algorithms need not require high-grade IMU sensors that previous works have used. Note that a comparison of absolute trajectory error was not possible due to the absence of ground truth, so as is common practice, we compute end-to-end translational error as a proxy metric (Table 3.2). In these experiments, DLIO outperformed all others across the board in both end-to-end translational error and per-scan efficiency. DLIO's resulting maps can capture fine detail in the environment which ultimately provides more intricate information cues for autonomous mobile robots such as terrain traversability.

Table 3.2: Comparison with UCLA Campus Dataset

Algorithm	Type	End-to-End Translational Error [m]				Avg. Comp. [ms]			
		A (652.66m)	B (526.58m)	C (551.38m)	D (530.75m)	A	B	C	D
DLO [CLA22a]	LO	0.0216	1.2932	0.0375	0.0178	20.40	20.77	21.18	21.62
CT-ICP [DDJ22]	LO	0.0387	0.0699	0.0966	0.0253	351.85	342.76	334.15	370.19
LIO-SAM [SEM20]	LIO	0.0216	0.0692	0.0936	0.0249	33.21	29.14	39.04	48.94
FAST-LIO2 [XCH22]	LIO	0.0454	0.0353	0.0363	0.0229	15.39	12.25	14.84	15.01
DLIO	LIO	**0.0105**	**0.0233**	**0.0301**	**0.0082**	**10.45**	**8.37**	**8.66**	**10.96**

3.5 Discussion

This chapter presented Direct LiDAR-Inertial Odometry (DLIO), a highly reliable LIO algorithm that yields accurate state estimates and detailed maps in real-time for resource-contrained mobile robots. The key innovation that distinguishes DLIO from others is its fast and parallelizable coarse-to-fine approach in constructing continuous-time trajectories for point-wise motion correction. This approach is built into a simplified LIO architecture which performs motion correction and prior construction in one shot, in addition directly performing scan-to-map alignment for reduced computational overhead. This is all feasible due to our observer's strong convergence guarantees which reliably initializes pose, velocity, and biases for accurate IMU integration. Our experimental results demonstrate DLIO's improved localization accuracy, map clarity, and algorithmic efficiency as compared to the state-of-the-art, and future work includes closed-loop flight tests and adding loop closures.

CHAPTER 4

Direct LiDAR-Inertial Odometry and Mapping: Perceptive and Connective SLAM

This chapter presents Direct LiDAR-Inertial Odometry and Mapping (DLIOM), a reliable SLAM algorithm with an explicit focus on computational efficiency, operational reliability, and real-world efficacy. DLIOM contains several key algorithmic innovations in both the front-end and back-end subsystems to design a resilient LiDAR-inertial architecture that is perceptive to the environment and produces accurate localization and high-fidelity 3D mapping for autonomous robotic platforms. Our ideas spawned after a deep investigation into modern LiDAR SLAM systems and their inabilities to generalize across different operating environments, in which we address several common algorithmic failure points by means proactive safe-guards to provide long-term operational reliability in the unstructured real world. We detail several important innovations to localization accuracy and mapping resiliency distributed throughout a typical LiDAR SLAM pipeline to comprehensively increase algorithmic speed, accuracy, and reliability. In addition, we discuss insights gained from our ground-up approach while implementing such a complex system for real-time state estimation on resource-constrained systems, and we experimentally show the increased performance of our method as compared to the current state-of-the-art on both public benchmark and self-collected datasets.

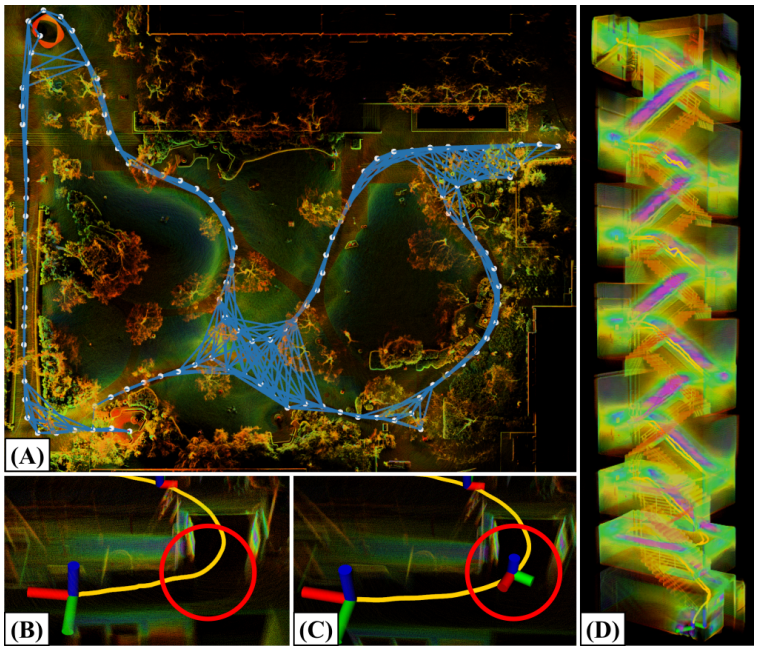

Figure 4.1: **Dense Connective Mapping with Resilient Localization.** Our novel DLIOM algorithm contains several proactive safe-guards against common failure points in LiDAR odometry to create a resilient SLAM framework that adapts to its operating environment. (A) A top-down view of UCLA's Sculpture Garden mapped by DLIOM, showcasing the algorithm's derived pose graph with interkeyframe constraints for local accuracy and global resiliency. (B & C) An example of DLIOM's slip-resistant keyframing which helps anchor scan-to-map registration, in which abrupt scenery changes (e.g., traversal through a door) that normally cause slippage (B) are mitigated by scene change detection (C). (D) A map of an eight-story staircase generated by DLIOM, showcasing the difficult environments our algorithm can track in.

4.1 Overview

Accurate real-time state estimation and mapping are fundamental capabilities that are the backbone for autonomous robots to perceive, plan, and navigate through unknown environments. Long-term operational reliability of such capabilities require algorithmic resiliency against off-nominal conditions, such as the presence of particulates (e.g., dust or fog), low-lighting, difficult or unstructured landscape, and other external factors. While visual SLAM approaches may work in well-posed environments, they quickly break down in-the-wild from their strong environmental assumptions, brittle architecture, or high computational com-

plexity. LiDAR-based methods, on the other hand, have recently become a viable option for many mobile platforms due to lighter and cheaper sensors. As a result, researchers have recently developed several new LiDAR odometry (LO) and LiDAR-inertial odometry (LIO) systems which often outperform vision-based localization due to the sensor's superior range and depth measurement accuracy. However, there are still several fundamental challenges in developing reliable, long-term LiDAR-centric SLAM solutions, especially for autonomous robots that explore unknown environments, execute agile maneuvers, or traverse uneven terrain [CCC16].

Algorithmic resiliency by means of building proactive safe-guards against common failure points in SLAM can provide reliable and failure-tolerant localization across a wide range of operating environments for long-term reliability. While existing algorithms may work well in structured environments that pose well-constrained problems for the back-end optimizer, their performance can quickly degrade under irregular conditions—yielding slow, brittle perception systems unsuitable for real-world use. Of the few recent approaches that do have the ability to adapt to different conditions on-the-fly, they either rely on switching to other sensing modalities in degraded environments [TTC20], require complex parameter tuning procedures based on manually specified heuristics [KYO21a,KYO21b], or focus solely on the scan-matching process in an effort to better anchor weakly-constrained registration problems [TNN22].

To this end, this chapter presents the Direct LiDAR-Inertial Odometry and Mapping (DLIOM) algorithm, a reliable, real-time SLAM system with several key innovations that provide increased resiliency and accuracy for both localization and mapping (Fig. 4.1). The main contributions of this chapter are five-fold, each targetting a specific module in a typical LiDAR SLAM architecture (bolded) to comprehensively increase algorithmic speed, accuracy, and reliability. First, a method for slip-resistant keyframing by detecting the onset of scan-matching slippage during abrupt scene changes via a global and sensor-agnostic degeneracy metric. Second, a method which generates explicitly-relevant local submaps with

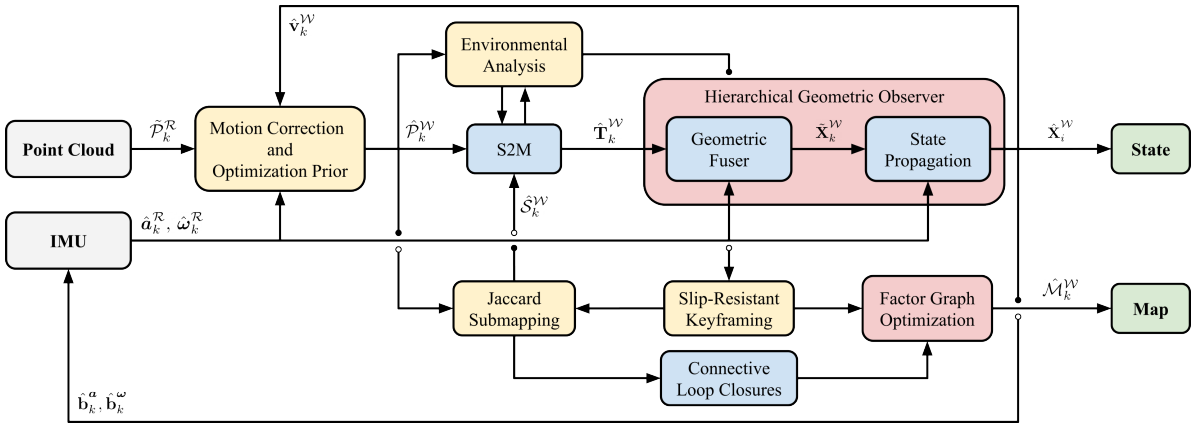

Figure 4.2: **System Architecture.** DLIOM's two-pronged architecture contains several key innovations to provide a comprehensive SLAM pipeline with real-world operational reliability. Point-wise continuous-time integration in $\mathcal{W}$ ensures maximum fidelity of the corrected cloud and is registered onto the robot's map by a custom GICP-based scan-matcher. An analysis on the environmental structure and health of scan-matching provides several system metrics for adaptively tuning maximum correspondence distance, in addition to slip-resistant keyframing. Additionally, a 3D Jaccard index for each keyframe is computed against the current scan to maximize submap coverage and therefore scan-matching correspondences. The system's state is subsequently updated by a nonlinear geometric observer with strong convergence properties, and these estimates of pose, velocity, and bias then initialize the next iteration. This system state is also subsequently sent to a background mapping thread, which places pose graph nodes at keyframe locations and builds a connective graph via interkeyframe constraints for local accuracy and global resiliency.

maximum coverage by computing the relative 3D Jaccard index for each keyframe for scan-to-map registration. Third, a method to increase local mapping accuracy and global loop closure resiliency via connectivity factors and keyframe-based loop closures. Fourth, an adaptive scan-matching method via a novel point cloud sparsity metric for consistent registration in both large and small environments. Finally, fifth, a new coarse-to-fine technique for fast and parallelizable point-wise motion correction, in which a set of analytical equations with a constant jerk and angular acceleration motion model is derived for constructing continuous-time trajectories.

DLIOM proposes several new techniques to LiDAR-based SLAM systems which address several deficiencies in both the front-end and back-end. Our ideas target different scales

in the data processing pipeline to progressively increase localization resiliency and mapping accuracy. Motion-corrected clouds are incrementally registered against an extracted submap via an adaptive scan-matching technique, which tunes the maximum correspondence distance based on the current cloud's sparsity for consistent registration across different environments. Each extracted submap is explicitly generated by computing each environmental keyframe's relevancy towards the current scan-matching problem via a relative 3D Jaccard index; this is done to maximize submap coverage and therefore data association between the scan and submap. To prevent slippage, scan-matching health is continually monitored through a novel sensor-agnostic degeneracy metric, which inserts a new keyframe when optimization is too weakly-constrained during rapid scene changes. Finally, to increase local mapping accuracy and global loop closure resiliency, we compute interkeyframe overlap to provide additional factors to our keyframe-based factor graph mapper.

4.2 System Description

DLIOM is a reliable SLAM algorithm with a specific focus on localization resiliency, mapping accuracy, and real-world operational reliability (Fig. 4.2). The architecture contains two parallel threads which process odometry estimation and global mapping in real-time. In the first, LiDAR scans are consecutively motion-corrected and then registered against a keyframe-based submap to provide an accurate update signal for integrated IMU measurements. This accuracy is ensured by an adaptive maximum correspondence distance for consistent scan-matching, in addition to how we explicitly derive the local submap, whereby maximum submap coverage of the current scan is enforced by computing a 3D Jaccard index for each environmental keyframe. This helps increase reliability against errors in data association during GICP optimization. A novel method for computing environmental degeneracy provides a global notion of scan-matching slippage and continually monitors optimization health status, placing a new keyframe right before the onset of slippage, and a nonlinear

geometric observer fuses LiDAR scan-matching and IMU preintegration to provide high-rate state estimation with certifiable convergence guarantees.

In the second thread, keyframes continually build upon an internal factor graph, whereby each keyframe is represented as a node in the graph and various constraints between pairs of keyframes are factors between nodes. "Sequential" factors between adjacent keyframes provide a strong backbone to the pose graph and is feasible due to our system's accurate local odometry; "connective" factors between overlapping keyframes (via their 3D Jaccard index) provide local map accuracy and global map resiliency against catastrophically incorrect loop closures. Frame offsets after such loop closures are carefully managed in order to prevent discontinuities in estimated velocity for safe robot control. Our algorithm is completely built from the ground-up to decrease computational overhead and increase algorithmic failure-tolerance and real-world reliability.

4.2.1 Notation

Let the point cloud for a single LiDAR sweep initiated at time t_k be denoted as $\mathcal{P}_k$ and indexed by k. The point cloud $\mathcal{P}_k$ is composed of points $p_k^n \in \mathbb{R}^3$ that are measured at a time Δt_k^n relative to the start of the scan and indexed by $n = 1, \ldots, N$ where N is the total number of points in the scan. The world frame is denoted as $\mathcal{W}$ and the robot frame as $\mathcal{R}$ located at its center of gravity, with the convention that x points forward, y left, and z up. The IMU's coordinate system is denoted as $\mathcal{B}$ and the LiDAR's as $\mathcal{L}$, and the robot's state vector $\mathbf{X}_k$ at index k is defined as the tuple

$$\mathbf{X}_k = \left[\, \mathbf{p}_k^{\mathcal{W}}, \, \mathbf{q}_k^{\mathcal{W}}, \, \mathbf{v}_k^{\mathcal{W}}, \, \mathbf{b}_k^{\boldsymbol{a}}, \, \mathbf{b}_k^{\boldsymbol{\omega}} \, \right]^\top , \tag{4.1}$$

where $\mathbf{p}^{\mathcal{W}} \in \mathbb{R}^3$ is the robot's position, $\mathbf{q}^{\mathcal{W}}$ is the orientation encoded by a four vector quaternion on $\mathbb{S}^3$ under Hamilton notation, $\mathbf{v}^{\mathcal{W}} \in \mathbb{R}^3$ is the robot's velocity, $\mathbf{b}^{\boldsymbol{a}} \in \mathbb{R}^3$ is the accelerometer's bias, and $\mathbf{b}^{\boldsymbol{\omega}} \in \mathbb{R}^3$ is the gyroscope's bias. Measurements $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{\omega}}$ from an

Algorithm 3: DLIOM: Odometry Thread (LiDAR)

input: $\hat{\mathbf{X}}_{k\text{-}1}^{\mathcal{W}}$, $\hat{\mathcal{M}}_{k}^{\mathcal{W}}$, $\mathcal{P}_{k}^{\mathcal{L}}$, $\boldsymbol{a}_{k}^{\mathcal{B}}$, $\boldsymbol{\omega}_{k}^{\mathcal{B}}$; **output:** $\hat{\mathbf{X}}_{k}^{\mathcal{W}}$

while $\mathcal{P}_{k}^{\mathcal{L}} \neq \emptyset$ **do**

  // initialize points and transform to $\mathcal{R}$

  $\tilde{\mathcal{P}}_{k}^{\mathcal{R}} \leftarrow$ initializePointCloud($\mathcal{P}_{k}^{\mathcal{L}}$); (3.2.2)

  // continuous-time motion correction

  for $\hat{\boldsymbol{a}}_{i}^{\mathcal{R}}, \hat{\boldsymbol{\omega}}_{i}^{\mathcal{R}}$ *between* $t_{k\text{-}1}$ *and* t_k **do**

    $\hat{\mathbf{p}}_i, \hat{\mathbf{v}}_i, \hat{\mathbf{q}}_i \leftarrow$ discreteInt($\hat{\mathbf{X}}_{k\text{-}1}^{\mathcal{W}}$, $\hat{\boldsymbol{a}}_{i\text{-}1}^{\mathcal{R}}$, $\hat{\boldsymbol{\omega}}_{i\text{-}1}^{\mathcal{R}}$); (3.4)

    $\hat{\mathbf{T}}_{i}^{\mathcal{W}} = [\, \hat{\mathbf{R}}(\hat{\mathbf{q}}_i) \,|\, \hat{\mathbf{p}}_i \,]$;

  end

  for $p_{k}^{n} \in \tilde{\mathcal{P}}_{k}^{\mathcal{R}}$ **do**

    $\hat{\mathbf{T}}_{n}^{\mathcal{W}*} \leftarrow$ continuousInt($\hat{\mathbf{T}}_{i}^{\mathcal{W}*}$, t_n); (3.5)

    $\hat{p}_{k}^{n} = \hat{\mathbf{T}}_{n}^{\mathcal{W}*} \otimes p_{k}^{n}$; $\hat{\mathcal{P}}_{k}^{\mathcal{W}}$.append($\hat{p}_{k}^{n}$);

  end

  // environmental analysis: compute spaciousness and cloud sparsity

  $m_k, z_k \leftarrow$ computeAdaptiveParams($\hat{\mathcal{P}}_{k}^{\mathcal{W}}$); [CLA22a], (4.9)

  // construct submap via 3D Jaccard index

  for $\mathcal{K}_{j}^{\mathcal{W}} \in \hat{\mathcal{M}}_{k}^{\mathcal{W}}$ **do**

    $J(\hat{\mathcal{P}}_{k}^{\mathcal{W}}, \mathcal{K}_{j}^{\mathcal{W}}) \leftarrow |\hat{\mathcal{P}}_{k}^{\mathcal{W}} \cap \mathcal{K}_{j}^{\mathcal{W}}| \,/\, |\hat{\mathcal{P}}_{k}^{\mathcal{W}} \cup \mathcal{K}_{j}^{\mathcal{W}}|$; (4.7)

    if $J(\hat{\mathcal{P}}_{k}^{\mathcal{W}}, \mathcal{K}_{j}^{\mathcal{W}}) \geq thresh_{jaccard}$ **then**

      $\hat{\mathcal{S}}_{k}^{\mathcal{W}}$.append($\hat{\mathcal{P}}_{k}^{\mathcal{W}}$);

    end

  end

  // adaptive scan-to-map registration

  $\hat{\mathbf{T}}_{k}^{\mathcal{W}} \leftarrow$ GICP($\hat{\mathcal{P}}_{k}^{\mathcal{W}}$, $\hat{\mathcal{S}}_{k}^{\mathcal{W}}$, z_k); (4.8)

  // slip-resistant keyframing

  if $\hat{\mathcal{P}}_{k}^{\mathcal{W}}$ *is a keyframe via (4.5) or [CLA22a]* **then**

    $\mathcal{K}_{k}^{\mathcal{W}} \leftarrow \hat{\mathcal{P}}_{k}^{\mathcal{W}}$;

    // compute new keyframe overlap and append to connectivity matrix

    for $\mathcal{K}_{j}^{\mathcal{W}} \in \hat{\mathcal{M}}_{k}^{\mathcal{W}}$ **do**

      $\mathbf{C}_{kj} \leftarrow J(\mathcal{K}_{k}^{\mathcal{W}}, \mathcal{K}_{j}^{\mathcal{W}})$; (4.4.1)

    end

    // send new keyframe and connectivity matrix to mapping thread

    odom2map($\mathcal{K}_{k}^{\mathcal{W}}$, $\mathbf{C}$);

  end

  // geometric observer: state update

  $\hat{\mathbf{X}}_{k}^{\mathcal{W}} \leftarrow$ updateState($\hat{\mathbf{T}}_{k}^{\mathcal{W}}$, Δt_{k}^{+}); (3.3.3)

  return $\hat{\mathbf{X}}_{k}^{\mathcal{W}}$

end

Algorithm 4: DLIOM: Odometry Thread (IMU)

input: $\hat{\mathbf{X}}_k^{\mathcal{W}}$, $\boldsymbol{a}_k^{\mathcal{B}}$, $\boldsymbol{\omega}_k^{\mathcal{B}}$; **output:** $\hat{\mathbf{X}}_i^{\mathcal{W}}$

while $\boldsymbol{a}_i^{\mathcal{B}} \neq \emptyset$ ***and*** $\boldsymbol{\omega}_i^{\mathcal{B}} \neq \emptyset$ **do**

    // apply biases and transform to $\mathcal{R}$

    $\hat{\boldsymbol{a}}_i^{\mathcal{R}}, \hat{\boldsymbol{\omega}}_i^{\mathcal{R}} \leftarrow$ initializeImu($\boldsymbol{a}_i^{\mathcal{B}}$, $\boldsymbol{\omega}_i^{\mathcal{B}}$); (3.2.2)

    // geometric observer: state propagation

    $\hat{\mathbf{X}}_i^{\mathcal{W}} \leftarrow$ propagateState($\hat{\mathbf{X}}_k^{\mathcal{W}}, \hat{\boldsymbol{a}}_i^{\mathcal{R}}, \hat{\boldsymbol{\omega}}_i^{\mathcal{R}}, \Delta t_i^+$); (3.3.3)

    return $\hat{\mathbf{X}}_i^{\mathcal{W}}$

end

IMU are modeled as

$$\hat{\boldsymbol{a}}_i = (\boldsymbol{a}_i - \boldsymbol{g}) + \mathbf{b}_i^{\boldsymbol{a}} + \mathbf{n}_i^{\boldsymbol{a}} \,, \tag{4.2}$$

$$\hat{\boldsymbol{\omega}}_i = \boldsymbol{\omega}_i + \mathbf{b}_i^{\boldsymbol{\omega}} + \mathbf{n}_i^{\boldsymbol{\omega}} \,, \tag{4.3}$$

and indexed by $i = 1, \ldots, M$ for M measurements between clock times $t_{k\text{-}1}$ and t_k. With some abuse of notation, indices k and i occur at LiDAR and IMU rate, respectively, and will be written this way for simplicity unless otherwise stated. Raw sensor measurements $\boldsymbol{a}_i$ and $\boldsymbol{\omega}_i$ contain bias $\mathbf{b}_i$ and white noise $\mathbf{n}_i$, and $\boldsymbol{g}$ is the rotated gravity vector. In this chapter, we address the following problem: given a distorted point cloud $\mathcal{P}_k$ from a LiDAR and $\boldsymbol{a}_i$ and $\boldsymbol{\omega}_i$ from an IMU, estimate the robot's state $\hat{\mathbf{X}}_i^{\mathcal{W}}$ and the geometric map $\hat{\mathcal{M}}_k^{\mathcal{W}}$.

4.2.2 Preprocessing

The inputs to DLIOM are a dense 3D point cloud collected by a modern 360° mechanical LiDAR, such as an Ouster or a Velodyne (10-20Hz), in addition to time-synchronized linear acceleration and angular velocity measurements from a 6-axis IMU at a much higher rate (100-500Hz). To minimize information loss, we do not preprocess the point cloud except for a box filter of size 1m^3 around the origin which removes points that may be from the robot itself, and a light voxel filter for higher resolution clouds. This distinguishes our work

Algorithm 5: DLIOM: Mapping Thread

input: $\mathcal{K}_k^{\mathcal{W}}$, $\mathbf{C}$, $\mathcal{G}_k$; **output:** $\hat{\mathcal{M}}_k^{\mathcal{W}}$
initialize: $\mathcal{G}_k$.addPriorFactor($\mathcal{K}_0^{\mathcal{W}}$)

// Keyframe Callback Thread
while $\mathcal{K}_k^{\mathcal{W}} \neq \emptyset$ ***and*** $\mathbf{C} \neq \emptyset$ **do**
  // add factor to previous keyframe
  $\mathcal{G}_k$.addBetweenFactor($\mathcal{K}_{k-1}^{\mathcal{W}}, \mathcal{K}_k^{\mathcal{W}}$);
  // add factors to connective keyframes
  for $\mathcal{K}_j^{\mathcal{W}} \in \hat{\mathcal{M}}_k^{\mathcal{W}}$ **do**
    if $\mathbf{C}_{kj} \geq thresh_{conn}$ **then**
      $\mathcal{G}_k$.addBetweenFactor($\mathcal{K}_k^{\mathcal{W}}, \mathcal{K}_j^{\mathcal{W}}$);
    end
  end
  // submap-based loop closures
  for $\mathcal{K}_j^{\mathcal{W}} \in \hat{\mathcal{M}}_k^{\mathcal{W}}$ **do**
    if $\mathcal{K}_j^{\mathcal{W}}$ *is a loop candidate via 4.4.2* **then**
      $\mathcal{L}_k^{\mathcal{W}} \leftarrow \mathcal{L}_k^{\mathcal{W}} \oplus \mathcal{K}_j^{\mathcal{W}}$
    end
  end
  $\hat{\mathbf{T}} \leftarrow$ GICP($\mathcal{L}_k^{\mathcal{R}}$, $\hat{\mathcal{K}}_k^{\mathcal{R}}$, z_k); (4.8)
  if $fitnessScore \geq thresh_{loop}$ **then**
    $\mathcal{G}_k$.addBetweenFactor($\mathcal{K}_k^{\mathcal{W}}, \mathcal{K}_{\mathcal{L}}^{\mathcal{W}}$);
  end
  // optimize the factor graph
  $\mathcal{G}_k$.optimize($\mathcal{K}_k^{\mathcal{W}}$);
  // update map after optimization
  $\hat{\mathcal{M}}_k^{\mathcal{W}} \leftarrow$ updateKeyframes($\mathcal{G}_k$);
  // send to odometry thread
  map2odom($\hat{\mathcal{M}}_k^{\mathcal{W}}$);
  return $\hat{\mathcal{M}}_k^{\mathcal{W}}$
end

from others that either attempt to detect features (e.g., corners, edges, and/or surfels) or aggressively downsamples the input cloud. On average, the point clouds used in this work on our custom platform contained ∼16,000 points per scan.

In addition, prior to downstream tasks, all sensor data is transformed to be in $\mathcal{R}$ located at the robot's center of gravity via extrinsic calibration. For LiDAR, each acquired scan is rotated and shifted via ${}^{\mathcal{R}}_{\mathcal{L}}\mathbf{T} \in \mathbb{SE}(3)$ such that $[\,p^{\mathcal{R}}\;1\,]^\top = {}^{\mathcal{R}}_{\mathcal{L}}\mathbf{T}\,[\,p^{\mathcal{L}}\;1\,]^\top$ for each point in the scan. For IMU, effects of displacing linear acceleration measurements on a rigid body must be considered if the sensor is not located exactly at the center of gravity. This is compensated for by considering all contributions of linear acceleration at $\mathcal{R}$ via the cross product of angular velocity with the displacement between the IMU and center of gravity, such that for raw linear acceleration $\boldsymbol{a}_i^{\mathcal{B}}$ measured in the IMU's frame, the corresponding linear acceleration in frame $\mathcal{R}$ is, assuming a constant displacement,

$$\hat{\boldsymbol{a}}_i^{\mathcal{R}} = \hat{\boldsymbol{a}}_i^{\mathcal{B}} + \left[\dot{\hat{\boldsymbol{\omega}}}_i^{\mathcal{R}} \times {}^{\mathcal{R}}_{\mathcal{B}}\mathbf{t}\right] + \left[\hat{\boldsymbol{\omega}}_i^{\mathcal{R}} \times \left(\hat{\boldsymbol{\omega}}_i^{\mathcal{R}} \times {}^{\mathcal{R}}_{\mathcal{B}}\mathbf{t}\right)\right] \tag{4.4}$$

where ${}^{\mathcal{R}}_{\mathcal{B}}\mathbf{t}$ is the translational distance from $\mathcal{B}$ to $\mathcal{R}$, and $\hat{\boldsymbol{\omega}}_i^{\mathcal{R}}$ is $\hat{\boldsymbol{\omega}}_i^{\mathcal{B}}$ but only rotated to be in axis convention since angular velocity is equivalent at all points on a rigid body.

4.3 Reliable & Perceptive Localization

4.3.1 Slip-Resistant Keyframing via Sensor-Agnostic Degeneracy

Convergence of (4.8) into a sub-optimal local minima can occur when correspondences for GICP plane-to-plane registration are sparse or insufficient. Such weak data correspondences can subsequently lead to poor or diverging localization due to the estimate "escaping" a shallow gradient around the local minima. This phenomenon, often referred to as *LiDAR slippage*, often occurs when the surrounding environment is featureless or otherwise geometrically degenerate (e.g., long tunnels or large fields) and is a result of incorrect data asso-

ciation between the source and target point clouds. Ill-constrained optimization problems can also arise in *keyframe-based* LIO when the extracted submap insufficiently represents the surrounding environment and which results in a low number of data correspondences for scan-to-map matching. This can happen when there is an abrupt change in the environment (e.g., walking through a door or up a stairwell) but there are no nearby keyframes which describe the new environment. While previous works have used the condition number of the Hessian [TTC20, EPW21, FHW23] to identify environmental degeneracy, such that $\kappa(\mathbf{H_{tt}}) = |\lambda_{\max}(\mathbf{H_{tt}})| \,/\, |\lambda_{\min}(\mathbf{H_{tt}})|$, this metric informs a system only of *relative* slippage with respect to the most constrained and least constrained directions of the problem. The condition number is affected by the size of the environment and the density of points, and therefore a more reliable approach is to compute a more consistent metric of degeneracy across differently-sized environments and sensor configurations. This enables a global, sensor-agnostic metric in which we use to detect when a new keyframe should be inserted into the environment.

Let $\mathcal{C}$ be the set of all corresponding points between $\hat{\mathcal{P}}_k^{\mathcal{W}}$ and $\hat{\mathcal{S}}_k^{\mathcal{W}}$, and therefore $|\mathcal{C}|$ is the total number of corresponding points, and let $\mathcal{E}$ be the total error between all correspondences after convergence of a nonlinear least squares solver (such as Levenberg-Marquardt) as described previously in (4.8). Also, let $\mathbf{H} \in \mathbb{R}^{6x6}$ be the Hessian of GICP, and let $\mathbf{H_{tt}} \in \mathbb{R}^{3x3}$ be the submatrix corresponding to the translational portion of $\mathbf{H}$, with eigenvalues $\lambda_{\max}(\mathbf{H_{tt}}) \geq \cdots \geq \lambda_{\min}(\mathbf{H_{tt}})$ which provide information regarding the local gradient of the nonlinear optimization after convergence. Note that $\mathbf{H} \approx \mathbf{J}^\top \mathbf{J}$ for computational efficiency and $\mathbf{J}$ is the Jacobian. Then, the *global degeneracy* d_k of the system is the maximum value after scaling each of the eigenvalues $\lambda(\mathbf{H_{tt}})$, such that

$$d_k = \max \left[\frac{m_k^2}{\lambda(\mathbf{H_{tt}}) \sqrt{z_k}} \right], \tag{4.5}$$

where m_k is the computed *spaciousness* [CLA22a], defined as $m_k = \alpha m_{k-1} + \beta M_k$, where

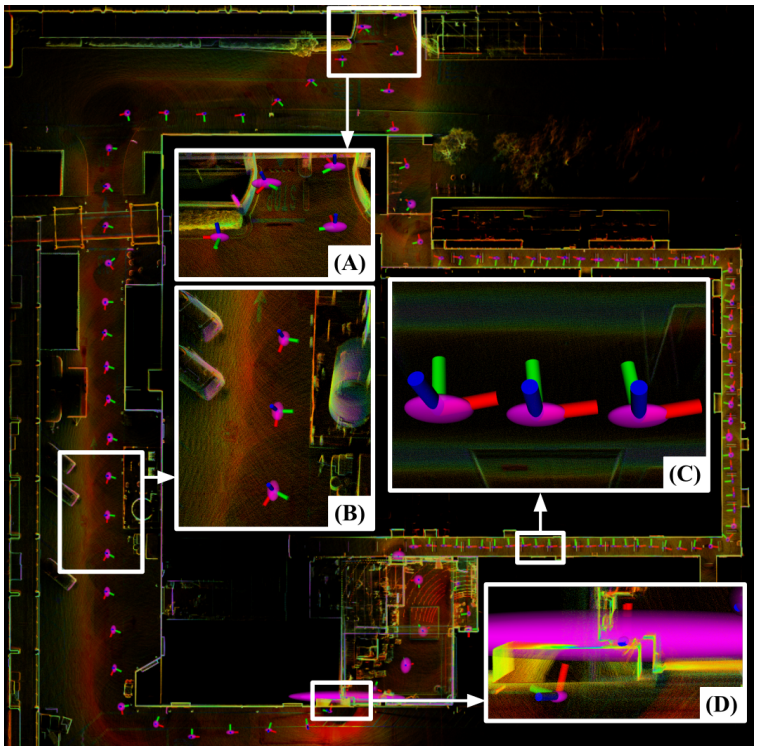

Figure 4.3: **Sensor-Agnostic Degeneracy.** Uncertainty ellipsoids (purple) for each keyframe computing using our generalized degeneracy metric in (A & B) outdoor environments, (C) a narrow hallway, and (D) through a doorway. Our metric is global in that the ellipsoids are consistent in size in both indoor and outdoor environments; our metric is also sensor-agnostic in that it accounts for the density of the cloud (which can vary across different LiDAR sensors and voxelization leaf sizes). Note that these ellipsoids usually on the millimeter-scale but have been enlarged for visualization clarity.

M_k is the median Euclidean point distance from the origin to each point in the preprocessed point cloud (with constants $\alpha = 0.95$, $\beta = 0.05$), and z_k is the cloud *sparsity* as defined above in (4.9). This degeneracy is computed for each incoming scan and saved in-memory for each new keyframe. If the difference between the current degeneracy and degeneracy at the location of the previous keyframe is sufficiently large, a new keyframe is inserted to provide the scan-to-map module with new information.

The intuition behind (4.5) lies in how each scaling factor (i.e., m_k and z_k) affects $\lambda(\mathbf{H_{tt}})$. In particular, while computing the condition number $\kappa(\mathbf{H_{tt}})$ can provide an idea of how ellipsoidal the local gradient is (and therefore how long it may take to converge to a local

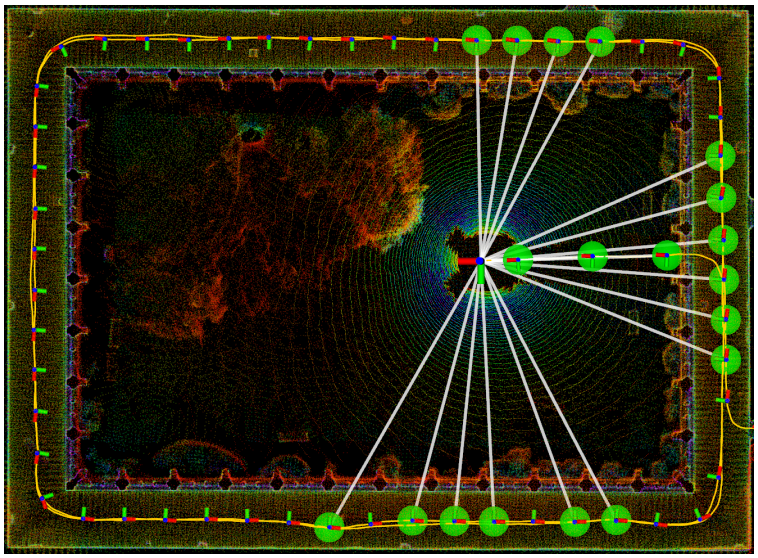

Figure 4.4: **Submapping via Jaccard Index.** Submap generation for the scan-to-map stage using the Newer College Dataset Extension - Cloister in Collection 2 [ZCF21]. For each newly acquired scan, we compute its Jaccard index against each environmental keyframe (axes) and extract only those which have a significant overlap with the current scan (green circles & white lines). The point clouds associated to the overlapping keyframes are then concatenated, alongside their in-memory covariances, for accurate scan-to-map registration. A threshold of at least 20% overlap was used in this example.

minimum), an elongated gradient does not necessarily indicate the onset of slippage from being poorly constrained. In other words, $\kappa(\mathbf{H_{tt}})$ is a *relative* metric of how well the optimization problem is constrained, since it only computes the relative ratio between the steepest and shallowest directions. To get a more accurate idea of when slippage may occur, (4.5) directly looks at (the inverse of) each individual eigenvalue. By rewarding sensors which provide less information about the environment via z_k, and by penalizing larger environments since measurements are less accurate with increasing distance, these various scaling factors allow d_k to be more consistent across different sensors and differently sized environments, which enables a more reliable metric of slip-detection (Fig. 4.3).

4.3.2 Submap Generation via 3D Jaccard Index

A key innovation of the DLIOM algorithm is how it explicitly derives its keyframe-based submap for scan-to-map registration. Ideally, the full history of all observed points would be matched against to ensure that there is no absence of important environmental information during scan-matching. Unfortunately, this is far too computationally intractable due to the sheer number of nearest-neighbor operations required for aligning against such a large map. Whereas previous approaches either naively assume that the closest points in map-space are those which are most relevant, or they implicitly compute keyframe relevancy via nearest neighbor and convex hull extraction in keyframe-space [CLA22a], we develop a new method for deriving the local submap that explicitly maximizes coverage between the current scan and the submap by computing the *Jaccard index* [Jac12] between the current scan and each keyframe.

Let the intersection between two point clouds $\mathcal{P}_1 \cap \mathcal{P}_2$ be a set $\mathcal{C}_{1,2}$ which contains all corresponding points between the two clouds in a common reference frame (within some corresponding distance), and therefore let $|\mathcal{C}_{1,2}|$ be the total number of corresponding points. In addition, let the union between two point clouds $\mathcal{P}_1 \cup \mathcal{P}_2$ be defined as the set $\mathcal{U}_{1,2}$ which contains all non-intersecting points between the two point clouds, in addition to the mean of each pair of corresponding points in $\mathcal{C}_{1,2}$, such that the total number of points in $\mathcal{U}_{1,2}$ equates to

$$|\mathcal{U}_{1,2}| = (|\mathcal{P}_1| \oplus |\mathcal{P}_2|) \setminus |\mathcal{C}_{1,2}| . \tag{4.6}$$

Then, for each newly acquired LiDAR scan at time k, we compute the 3D Jaccard index between the scan $\hat{\mathcal{P}}_k^{\mathcal{W}}$ and each j^{th} keyframe $\mathcal{K}_j^{\mathcal{W}}$, defined as

$$J(\hat{\mathcal{P}}_k^{\mathcal{W}}, \mathcal{K}_j^{\mathcal{W}}) = \frac{|\hat{\mathcal{P}}_k^{\mathcal{W}} \cap \mathcal{K}_j^{\mathcal{W}}|}{|\hat{\mathcal{P}}_k^{\mathcal{W}} \cup \mathcal{K}_j^{\mathcal{W}}|} , \tag{4.7}$$

or, in otherwords, the equivalent of the "intersection over union" similarity measurement

in the 3D domain. If $J(\hat{\mathcal{P}}_k^{\mathcal{W}}, \mathcal{K}_j^{\mathcal{W}})$ surpasses a set threshold for the j^{th} keyframe (i.e., a keyframe is sufficiently similar), then that keyframe is included within the submap to be used for scan-to-map registration. In contrast to previous methods which derive the submap through a series of heuristics (such as directly retrieving local points within a certain radius of the current position or assuming that nearby keyframes contain relevant points) our method explicitly computes each keyframes' relevancy to the current environment to ensure the scan-to-map optimization is well-constrained with maximum coverage between the scan and the submap. In addition, by using only keyframe scans that contain significant overlap with the current scan, this guarantees that there are no wasted operations when building normals or the kdtree data structure for the submap (Fig. 4.4).

4.3.3 Adaptive Scan-Matching via Cloud Sparsity

By simultaneously correcting for motion distortion and incorporating the GICP optimization prior into the point cloud, DLIOM can directly perform scan-to-map registration and bypass scan-to-scan required in previous methods. This registration is cast as a nonlinear optimization problem which minimizes the distance of corresponding points/planes between the current scan and an extracted local submap. Let $\hat{\mathcal{P}}_k^{\mathcal{W}}$ be the corrected cloud in $\mathcal{W}$ and $\hat{\mathcal{S}}_k^{\mathcal{W}}$ be the extracted submap. Then, the objective of scan-to-map optimization is to find a transformation $\Delta\hat{\mathbf{T}}_k$ which better aligns the point cloud, where

$$\Delta\hat{\mathbf{T}}_k = \underset{\Delta\mathbf{T}_k}{\arg\min}\ \mathcal{E}\left(\Delta\mathbf{T}_k\hat{\mathcal{P}}_k^{\mathcal{W}},\, \hat{\mathcal{S}}_k^{\mathcal{W}}\right), \tag{4.8}$$

such that the GICP residual error $\mathcal{E}$ is defined as

$$\mathcal{E}\left(\Delta\mathbf{T}_k\hat{\mathcal{P}}_k^{\mathcal{W}}, \hat{\mathcal{S}}_k^{\mathcal{W}}\right) = \sum_{c\in\mathcal{C}} d_c^{\top}\left(C_{k,c}^{\mathcal{S}} + \Delta\mathbf{T}_k C_{k,c}^{\mathcal{P}}\Delta\mathbf{T}_k^{\top}\right)^{-1} d_c\,,$$

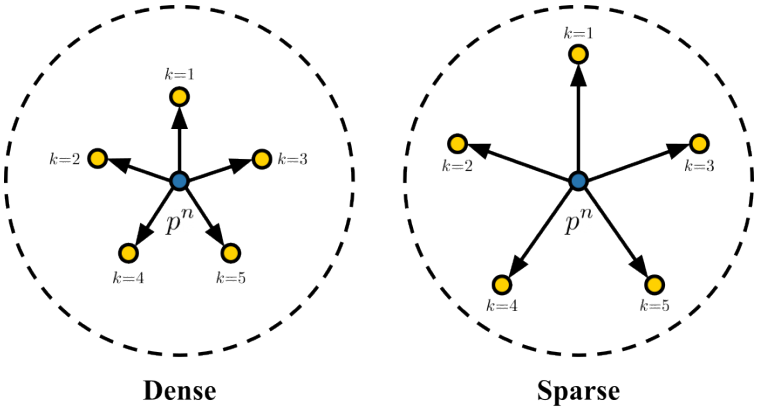

Figure 4.5: **Adaptive Scan-Matching via Cloud Sparsity.** For each motion-corrected point cloud, we compute its *sparsity*, defined as the average per-point Euclidean distance across K nearest neighbors (4.9) (K=5 in this example). This metric is used to scale the scan-to-map module's maximum correspondence distance for adaptive registration. A scan within a small-scale environment will contain points much closer together (left), so a small movement will have a small effect on point displacement. On the otherhand, a large environment will have points much more spread out (right) and will require a larger search distance during GICP for correct data association.

for a set $\mathcal{C}$ of corresponding points between $\hat{\mathcal{P}}_k^{\mathcal{W}}$ and $\hat{\mathcal{S}}_k^{\mathcal{W}}$ at timestep k, $d_c = \hat{s}_k^c - \Delta\mathbf{T}_k\hat{p}_k^c$, $\hat{p}_k^c \in \hat{\mathcal{P}}_k^{\mathcal{W}}$, $\hat{s}_k^c \in \hat{\mathcal{S}}_k^{\mathcal{W}}$, $\forall c \in \mathcal{C}$, and $C_{k,c}^{\mathcal{P}}$ and $C_{k,c}^{\mathcal{S}}$ are the estimated covariance matrices for point cloud $\hat{\mathcal{P}}_k^{\mathcal{W}}$ and submap $\hat{\mathcal{S}}_k^{\mathcal{W}}$, respectively. Then, following [SHT09], this point-to-plane formulation is converted into a plane-to-plane optimization by regularizing covariance matrices $C_{k,c}^{\mathcal{P}}$ and $C_{k,c}^{\mathcal{S}}$ with $(1, 1, \epsilon)$ eigenvalues, where ϵ represents the low uncertainty in the surface normal direction. The resulting $\Delta\hat{\mathbf{T}}_k$ represents an optimal correction transform which better globally aligns the prior-transformed scan $\hat{\mathcal{P}}_k^{\mathcal{W}}$ to the submap $\hat{\mathcal{S}}_k^{\mathcal{W}}$, so that $\hat{\mathbf{T}}_k^{\mathcal{W}} = \Delta\hat{\mathbf{T}}_k\hat{\mathbf{T}}_M^{\mathcal{W}}$ (where $\hat{\mathbf{T}}_M^{\mathcal{W}}$ is the last point's IMU integration) is the globally-refined robot pose which is used for map construction and as the update signal for the nonlinear geometric observer.

An important parameter that is often overlooked is the maximum distance at which corresponding points or planes should be considered in the optimization. This parameter is often hand-tuned by the user but should scale with the environmental structure for consistency and computational efficiency. For example, in small-scale environments (e.g., a lab

room), points in the LiDAR scan are much closer together so a small movement has a small effect on the displacement of a given point. In contrast, the same point in a more open environment (e.g., a point on a distant tree outside) will be displaced farther with a small rotational movement due to a larger distance and therefore needs a greater search radius for correct correspondence matching (Fig. 4.5). Thus, we set the GICP solver's maximum correspondence search distance between two point clouds according to the "sparsity" of the current scan, defined as $z_k = \alpha z_{k-1} + \beta D_k$, where

$$D_k = \frac{1}{|\mathcal{P}|N} \sum_{n=1}^{N} D_k^n \tag{4.9}$$

is the normalized per-point sparsity, D_k^n is the average Euclidean distance to K nearest neighbors for point n, and $\alpha = 0.95$ and $\beta = 0.05$ are smoothing constants to produce z_k, the filtered signal set as the max correspondence distance. Intuitively, this is the average inter-point distance in the current scan; the larger the environment, higher the number of sparse points (i.e., points further away), driving this number up. By adapting the corresponding distance according to the sparsity of points, the efficacy of scan-matching can be more consistent across differently-sized environments.

4.4 Connective Mapping

Factor graphs are widely used in SLAM as they are a powerful tool to estimate a system's full state by combining pose estimates from various modalities via *pose graph optimization* [SEM20]. Such works model relative pose constraints as a maximum a posteriori (MAP) estimation problem with a Gaussian noise assumption, and they typically view mapping as an afterthought as a result of refining the trajectory. However, such a unimodal noise model is far too simplistic for the complex uncertainty distribution that can arise from LiDAR scan-matching and IMU pre-integration. Moreover, graph-based optimization for odometry possesses minimal convergence guarantees and can often result in significant localization error

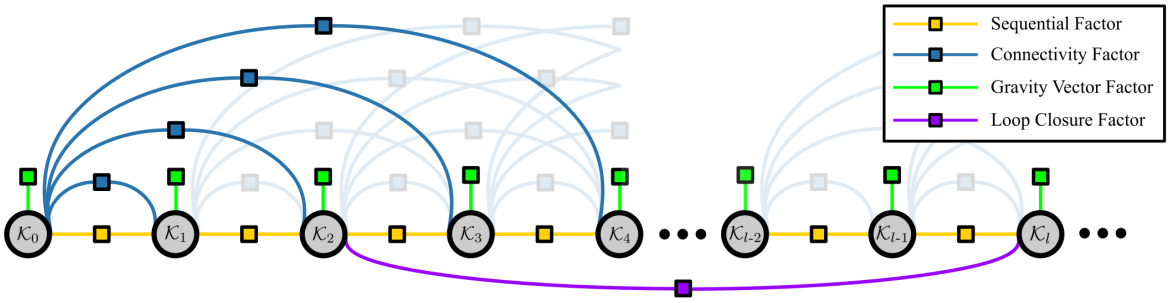

Figure 4.6: **Keyframe-based Factor Graph Mapping.** Our mapper adds a node to its factor graph for each new keyframe and adds relative constraints through either sequential factors (yellow), connectivity factors (blue), gravity factors (green), or loop closure factors (purple). Sequential factors provide a strong "skeleton" for the graph with low uncertainty between adjacent keyframes, while connectivity factors scale depending on the overlap between pairs of keyframes. Loop closure factors enable global consistency after long-term drift from pure odometry.

and map deformation from inconsistent sensor fusion. To this end, we instead employ a factor graph not for odometry (which is instead handled by our geometric observer), but rather to explicitly represent the environment. A new node is added to the graph for every incoming keyframe (as determined by DLIOM's odometry thread), and various factors between nodes contribute to the global consistency of the map (Fig. 4.6). In addition to the factors detailed in this section, we additionally add a gravity factor to locally constrain the direction of each keyframe as described in [NCL23].

4.4.1 Connective Keyframe Factors

Relative constraints between nodes in a factor graph are typically added sequentially (i.e., factors are added between adjacent nodes [SEM20]), but the relationship between non-adjacent keyframes can provide additional information to the graph which helps to create a more accurate and globally consistent map. These additional constraints are what we call *connective* factors, which are determined by pairs of keyframes having sufficient overlap in $\mathcal{W}$ as computed in Sec. 4.3.2. That is, for K number of total keyframes, the connectivity between $K_i^{\mathcal{W}}$ and $K_j^{\mathcal{W}}$ is defined as the 3D Jaccard index (4.7) and encoded in a symmetric matrix

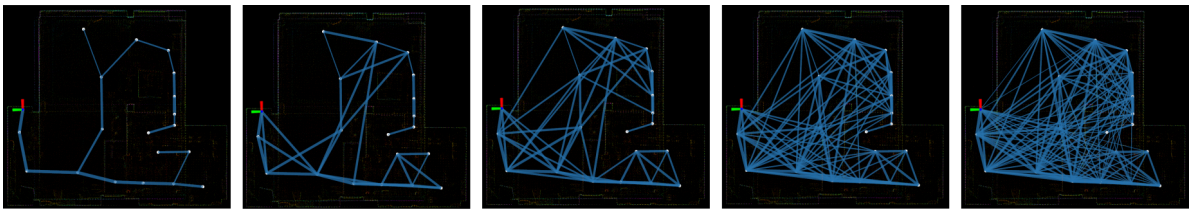

Figure 4.7: **Environmental Connectivity.** Example of increasing graph strength (left to right) by reducing the threshold for connective factors. A weak graph (left) is less locally accurate but allows for more compliancy when adding loop closures to the graph, while a strong graph (right) is more locally accurate from its higher number of interkeyframe factors, which are computed according to keyframe-to-keyframe overlap.

$\mathbf{C} \in \mathbb{R}^{K \text{x} K}$, such that

$$\mathbf{C}_{ij} = \begin{cases} \dfrac{|\hat{\mathcal{K}}_i^{\mathcal{W}} \cap \mathcal{K}_j^{\mathcal{W}}|}{|\hat{\mathcal{K}}_i^{\mathcal{W}} \cup \mathcal{K}_j^{\mathcal{W}}|} & \text{for } i > j \\ \dfrac{|\hat{\mathcal{K}}_j^{\mathcal{W}} \cap \mathcal{K}_i^{\mathcal{W}}|}{|\hat{\mathcal{K}}_j^{\mathcal{W}} \cup \mathcal{K}_i^{\mathcal{W}}|} & \text{for } i < j \\ 1 & \text{for } i = j \end{cases} \tag{4.10}$$

where the diagonal contains all 1's by definition of (4.7). A new factor is added between two keyframes if $\mathbf{C}_{ij}$ is above a set threshold, and the noise for this factor is computed by $\zeta(1 - \mathbf{C}_{ij})$, where ζ is a tunable scaling parameter that controls the strength of the environmental graph (Fig. 4.7).

4.4.2 Keyframe-based Loop Closures

Ideally, place recognition modules would search across all seen data for loop closures and add corresponding graph factors accordingly. However, storing all historical scans is computationally infeasible, so scans are stored in-memory incrementally as *keyframes*. In this lens, such keyframes can be understood as the subset of all historical point clouds which maximize information about the environment and contain data about the most significant locations. However, individual scans can be quite sparse (depending on the selected sensor) and may not contain enough points for data association for accurate detection via scan-matching.

Therefore, rather than iterating through all keyframes individually or reusing the submap constructed from the frontend which is only optimal for odometry [WZS22], we instead build an additional submap optimized for the backend mapper which consists of all candidate loop closure keyframes.

After adding the new keyframe to the factor graph and the associated connectivity constraints as described above, prior to optimization we search for and perform loop closure detection through a three-step process. First, we extract keyframes that are within some radius of the current position, in addition to those that contain some overlap with the current keyframe. The corresponding point clouds are then concatenated into a *loop cloud* $\mathcal{L}_k^{\mathcal{W}}$ and transformed back into $\mathcal{R}$, and GICP scan-matching is performed between this and the new keyframe. If the fitness score between $\hat{\mathcal{K}}_k^{\mathcal{R}}$ and $\mathcal{L}_k^{\mathcal{R}}$ is sufficiently low (i.e., the average Euclidean error across all corresponding points is small), a new loop-closure factor is added between the current keyframe and the closest keyframe in $\mathcal{W}$ used to build the loop cloud. Crucially, the registration is primed with a prior equal to the distance between these two keyframes in $\mathcal{W}$ with a sufficiently large plane-to-plane search distance. This process is fast since we do not rebuild the covariances required for GICP, as individual keyframe normals are concatenated instead [CLA22a]. This idea of reconstructing a submap for loop closure detection can easily be extend to using other place recognition modules (e.g., [KK18, KCK21]) for further robustness.

4.5 Algorithmic Implementation

This section highlights three important implementation details of our system for small lightweight platforms: sensor time synchronization, resource management for consistent computational load, and velocity-consistent loop closures.

4.5.1 Sensor Synchronization

Time synchronization is a critical element in odometry algorithms which utilize sensors that have their own internal clock. This is necessary as it permits time-based data association to temporally align IMU measurements and LiDAR scans. There are three clock sources in DLIOM: one each for the LIDAR, IMU, and processing computer. Hardware-based time synchronization—where the acquisition of a LiDAR scan is triggered from an external source—is not compatible with existing spinning LiDARs since starting and stopping the rotation assembly can lead to inconsistent data acquisition and timing. As a result, we developed a software-based approach that compensates for the offset between the LiDAR (IMU) clock and the processing computer clock. When the first LiDAR (IMU) packet is received, the processing computer records its current clock time ${}^{c}t_0$ and the time the measurement was taken on the sensor ${}^{s}t_0$. Then, each subsequent k^{th} measurement has a time ${}^{c}t_k$ with respect to the processing computer clock given by ${}^{c}t_k = {}^{c}t_0 + ({}^{s}t_k - {}^{s}t_0)$, where ${}^{s}t_k$ is the time the measurement was taken on the sensor. This approach was found to work well in practice despite its inability to observe the transportation delay of sending the first measurement over the wire. The satisfactory performance was attributed to using the elapsed *sensor* time to compute the compensated measurement time since a sensor's clock is generally more accurate than that of the processing computer.

4.5.2 Submap Multithreading

Fast and consistent computation time is essential for ensuring that incoming LiDAR scans are not dropped, especially on resource-constrained platforms. To this end, DLIOM offloads work not immediately relevant to the current scan to a separate thread which minimally interferes with its parent thread as it handles further incoming scans. Thus, the main point cloud processing thread has lower, more consistent computation times. The secondary thread builds the local submap kdtree used for scan-matching and builds data structures

corresponding to each keyframe which are needed by the submap. Speed of the submap building process is additionally increased by saving in-memory the computed kdtrees for each keyframe in order to quickly compute the Jaccard index of each keyframe, making that process negligibly different than an implicit nearest neighbor keyframe search. This thread can finish at any time without affecting DLIOM's ability to accept new LiDAR scans. Additionally, it periodically checks if the main thread is running so it can pause itself and free up resources that may be needed by the highly-parallelized scan-matching algorithm. Crucially, the submap changes at a much slower rate than the LiDAR sensor rate, and there is no strict deadline for when the new submap must be built. Therefore, the effect of this thread—which is to occasionally delay a new submap by a few scan iterations—has negligible impact on performance.

4.5.3 Velocity-Consistent Loop Closures

Although our odometry module constructs a submap of relevant keyframes, these keyframes are pulled directly from the globally optimized map. Therefore, we must be careful with how the state and keyframes get updated upon loop closure detection. In particular, the estimated position within the map will jump instantaneously, but a *continuous* trajectory would be beneficial for control and require fewer updates in the odometry module. We therefore allow the mapping module to perform this instantaneous update and maintain the robot pose in the *map* frame, but establish an offset from the *odometry* frame so that the robot pose and latest keyframe pose never jump. After an update, keyframes which have shifted must have their point clouds transformed in the odometry module; this is executed in a background thread, with submap keyframes being prioritized.

4.6 Experimental Results

In this section, we first provide an analysis of each proposed contribution to convince the reader that our core innovations are reasonable for improving the accuracy and resiliency of localization and mapping. Then, to validate our methods and system as a whole, DLIOM's accuracy and efficiency was compared against several current state-of-the-art and open-source systems. These include three LO algorithms, namely DLO [CLA22a], CT-ICP [DDJ21], and KISS-ICP [VGM23a], and three LIO algorithms, namely LIO-SAM [SEM20], FAST-LIO2 [XCH22], and DLIO [CNL23b]. We use the entirety of two public benchmark datasets, in addition to a self-collected dataset around a university campus, to compare the algorithms. These benchmark datasets include the Newer College dataset [RWC20] and the extension to Newer College Extension dataset [ZCF21]. Note that some well-known algorithms (e.g., Wildcat [RKC22] and X-ICP [TNN22]) could not be thoroughly compared against due to closed-source implementation and/or custom unreleased datasets; however, Wildcat [RKC22] was briefly evaluated against using the MulRan DCC03 dataset [KPC20] as they provide numerical results of this dataset in their manuscript. Finally, we demonstrate the usage of DLIOM in a fully closed-loop flight through several aggressive autonomous maneuvers in a lab setting using our custom aerial platform.

4.6.1 Analysis of Components

4.6.1.1 Slip-Resistant Keyframing

To showcase the resiliency of our slip-resistant keyframing strategy, which continually monitors scan-matching optimality and places an environmental keyframe during the onset of slippage, we use the Newer College Extension - Stairs dataset [ZCF21] and compare against LIO-SAM [SEM20] and FAST-LIO2 [XCH22]. Staircases are notoriously difficult for SLAM algorithms—especially those which are LiDAR-centric—due to the sensors' limited field-of-view in the Z direction. Because of this, tracking can be challenging as there are less data

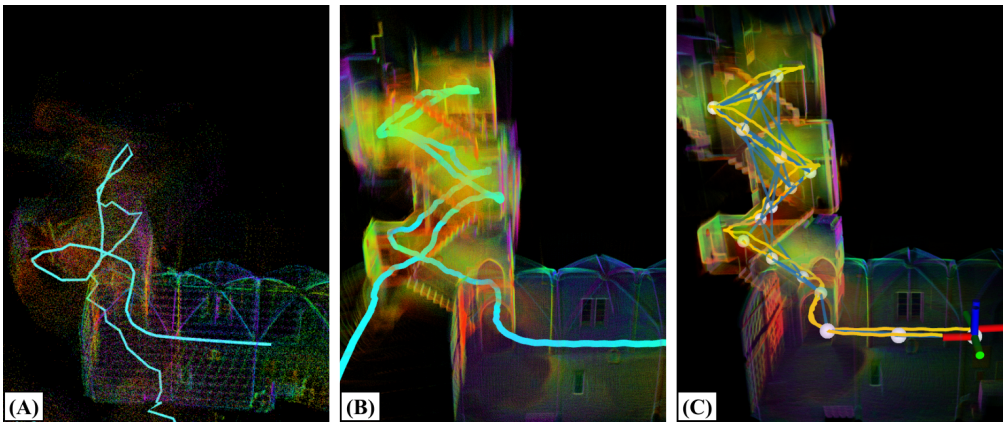

Figure 4.8: **Slip-Resistant Localization.** Comparison of maps and trajectories generated by (A) LIO-SAM [SEM20], (B) FAST-LIO2 [XCH22], and (C) our method, using the Newer College Extension - Stairs dataset [ZCF21]. For (A), we observed slippage right after entering the stairwell, while for (B), tracking was shaking during ascension (e.g., blurry map), with it slipping after descension at the bottom. For (C), our keyframe placement (white nodes) allowed our algorithm to track sufficiently both during ascension and descension, constructing a clear map and accurate trajectory.

points to associate with during ascension. This can be observed in Fig. 4.8. For LIO-SAM, the algorithm slipped right at the entrance of the stairwell, most likely due to a lack in sufficient features for feature extraction that the algorithm relies on. For FAST-LIO2, tracking was much better (as the algorithm also performs no feature extraction), but localization was jittery near the apex (e.g., blurry map at the top), and the algorithm completely slipped during descension. However, by detecting the onset of slippage, DLIOM can actively place new keyframes to continually anchor itself through space and therefore allow for tracking in challenging scenarios. This can be further seen in Fig. 4.1(D), in which our method was able to track through eight flights of stairs, while all other algorithms failed.

4.6.1.2 Jaccard Submapping

The efficacy of our submapping strategy, which directly computes each environmental keyframe's relevancy (i.e., overlap) through a computed 3D Jaccard index and subsequently extracts those which are most useful for scan-matching, is compared against a naive, implicit method.

Table 4.1: Comparison of Submapping Strategies

Submapping Strategy	Absolute Trajectory Error [m]		
	Max	Mean	RMSE
NN + Convex [CLA22a]	0.5898	0.0923 ± 0.444	0.1006
Jaccard Index	**0.2613**	**0.0604 ± 0.0361**	**0.0546**

More specifically, the naive approach extracts keyframes which are spatially nearby and those which construct the convex hull of keyframes. First proposed in [CLA22a], this strategy implicitly assumes that these keyframes are the best for globally aligning the current scan. However, as seen in Table 4.1, which compares trajectory error between the naive method ("NN + Convex") and our Jaccard method ("Jaccard Index") using the Newer College Extension - Cloister dataset (Fig. 4.4), this may not extract the most relevant submap for scan-to-map alignment and can be detrimental to accuracy and computational complexity. Heuristically extracting such keyframes risks using point clouds which are not used for scan-to-map, and therefore adds unnecessary operations during kdtree or covariance structure building. In contrast, an explicit extraction of the most useful keyframes provides scan-to-map a more practical set of keyframes to align with, ultimately helping with data association for GICP and reducing computational waste on keyframes which may not be used at all for registration. This applies for all scenarios, such as ascension of a staircase, where nearby keyframes may have zero overlap but would be extracted using the nearest-neighbor method.

4.6.1.3 Connective Mapping

Finally, we verify the effectiveness of adding connective factors between keyframes in our factor graph mapper. These factors provide additional constraints between overlapping nodes to better locally constrain each keyframe relative to one another, which can help with mapping accuracy after loop closure. We compared overall cloud-to-cloud distance to ground truth between a map generated with connectivity factors between keyframes, and a map generated

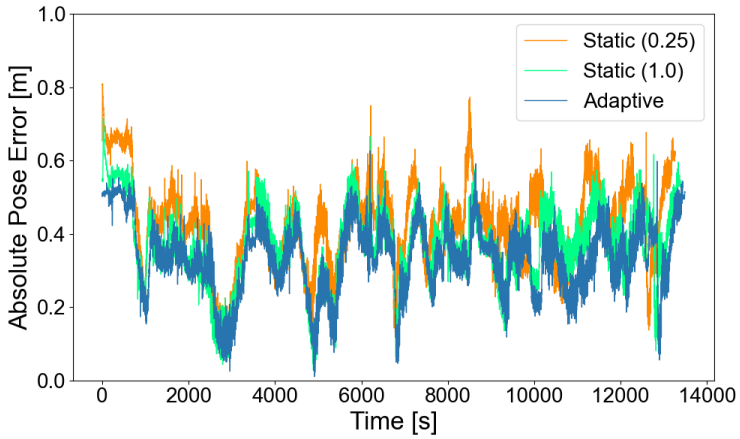

Figure 4.9: **Adaptive Scan-Matching.** A comparison of absolute pose error on the Newer College - Short Experiment dataset using adaptive and static scan-matching correspondence thresholds. We observed, on average, a lower trajectory error using our adaptive scaling technique as compared to static search thresholds which other methods typically use; this allows for more consistent localization in both small and large environments.

only with sequential ("odometry"-like) factors. The Newer College Extension - Maths (H) dataset was used in this experiment, and ground truth was provided by a high-grade Leica BLK360 laser scanner. Cloud-to-cloud error was computed using the CloudCompare application [GRM05] after manual registration. All clouds were voxelized with a leaf size of 0.1m to provide a fair comparison, and a maximum threshold of 1m was set for computing the average error to filter non-overlaping regions. Without connective factors, DLIOM's output map had a mean cloud-to-cloud distance to ground truth of 0.3285 ± 0.2289; however, with connective factors, this cloud-to-cloud distance reduced down to 0.2982 ± 0.2214. These connective factors between overlapping nodes can create a more accurate map after graph optimization by providing additional constraints for loop closures.

4.6.1.4 Adaptive Scan-Matching

Next, we compared our adaptive scan-matching technique, which scales the GICP maximum correspondence distance according to scan sparsity, against two statically-set thresholds. Specifically, we compared against a static correspondence distance of 0.25, which is typically

optimal for smaller environments (since points are closer together), and to the trajectory from a static correspondence distance of 1.0, which is more reasonable for larger, outdoor environments. We used the Newer College - Short Experiment dataset for this comparison, as it features three different sections of varying sizes ("Quad", "Mid-Section", and "Parkland"). The results are shown in Fig. 4.9, in which we observed our adaptive thresholding scheme to perform the best, followed by a static threshold of 1.0, and finally a threshold of 0.25 which performed the worst amongst the three. This is reasonable because the majority of the Short Experiment dataset is in medium and large ("Quad" and "Parkland") scenes, with about 20% of the trajectory in the smaller "Mid-Section." Because of this, a larger threshold would, on average, perform better than a smaller threshold (RMSE of 0.3810 ± 0.1063 versus 0.4140 ± 0.1167), but not as well as one that adapts to provide the best of both worlds (0.3571 ± 0.0971).

4.6.2 Benchmark Results

We compare the accuracy and efficiency of DLIOM against six state-of-the-art algorithms using public and self-collected datasets. Aside from extrinsics, default parameters at the time of writing for each algorithm were used in all experiments unless otherwise noted. Specifically, loop-closures were kept enabled for LIO-SAM and online extrinsics estimation disabled for FAST-LIO2 to provide the best results of each algorithm. For FAST-LIO2, we reduced the default crop otherwise it would fail in smaller environments. Deskewing was enabled for KISS-ICP, and for CT-ICP, voxelization was increased and data playback speed was slowed down to 25% otherwise the algorithm would fail due to significant frame drops. Loop closures were disabled in DLIOM to provide a more fair assessment. Trajectories were compared against the ground truth using evo [Gru17] in TUM [SEE12a] format and aligned with the Umeyama algorithm [Ume91] for all public benchmark datasets. Algorithms which did not produce meaningful results are indicated accordingly in the tables, and trajectory lengths for each dataset are indicated in italics to give a reader a sense of duration. All tests

were conducted on a 16-core Intel i7-11800H CPU.

4.6.2.1 Newer College Dataset

Trajectory accuracy and average per-scan time of all algorithms were also compared using the original Newer College benchmark dataset [RWC20]. For these tests, we used data from the Ouster OS1-64 (10Hz) in addition to its internal IMU (100Hz) to ensure accurate time synchronization between sensors. For certain Newer College datasets, the first 100 poses were excluded from computing FAST-LIO2's RMSE due to slippage at the start in order to provide a fair comparison. Short, Long, and Parkland experiments were routes recorded at a standard walking pace around several different sections, while Quad and Dynamic featured rapid linear and angular movements. The results are shown in Table 4.2, in which we observed our method to produce the lowest trajectory RMSE as compared to all other algorithms. Fig. 3.6 illustrates DLIOM's low trajectory error compared to ground truth for the Newer College - Long Experiment dataset even after over three kilometers of travel.

4.6.2.2 Newer College Extension Dataset

Additionally, we compared all algorithms using the latest extension to the Newer College benchmark dataset [ZCF21], which features three collections of data. Collections 1 and 3 contains three datasets each with progressively increasing difficulty, from "Easy" (E), which had slow paced movement, to "Medium" (M), which featured slightly more aggressive turn-rates and motions, and finally to "Hard" (H), which contained highly aggressive motions, rotations, and locations in both small and large environments. Collection 2 contains three datasets, each of which are highly different than the other to create a diverse set of environments. This includes traversing up and down a staircase, walking around a cloister with limited visibility, and a large-scale park with multiple loops. In this benchmark dataset, we used data from the Ouster OS0-128 (10Hz) in addition to the Alphasense Core IMU (200Hz),

Table 4.2: Comparison with Newer College Dataset

Algorithm	Type	Absolute Trajectory Error (RMSE) [m]					Average Comp. [ms]
		Short	Long	Quad	Dynamic	Parkland	
		1609.40m	*3063.42m*	*479.04m*	*97.20m*	*695.68m*	
DLO [CLA22a]	LO	0.4633	0.4125	0.1059	0.1954	0.1846	48.10
CT-ICP [DDJ21]	LO	0.5552	0.5761	0.0981	0.1426	0.1802	412.27
KISS-ICP [VGM23b]	LO	0.6675	1.5311	0.1040	Failed	0.2027	167.38
LIO-SAM [SEM20]	LIO	0.3957	0.4092	0.0950	0.0973	0.1761	179.33
FAST-LIO2 [XCH22]	LIO	0.3775	0.3324	0.0879	0.0771	0.1483	42.86
DLIO (None)	LIO	0.4299	0.3988	0.1117	0.1959	0.1821	34.88
DLIO (Discrete)	LIO	0.3803	0.3629	0.0943	0.0798	0.1537	**34.61**
DLIO (Continuous) [CNL23b]	LIO	0.3606	0.3268	0.0837	0.0612	0.1196	35.74
DLIOM	LIO	**0.3571**	**0.3252**	**0.0821**	**0.0609**	**0.1181**	36.21

Table 4.3: Comparison with Newer College Extension Dataset

Algorithm	Type	Absolute Trajectory Error (RMSE) [m]									Avg Comp. [ms]
		Quad-E	Quad-M	Quad-H	Stairs	Cloister	Park	Maths-E	Maths-M	Maths-H	
		246.67m	*260.36m*	*234.81m*	*57.04m*	*428.79m*	*2396.20m*	*263.62m*	*304.28m*	*320.56m*	
DLO [CLA22a]	LO	0.0866	0.1141	0.1490	0.1605	0.1608	0.8108	0.1658	0.7730	1.0864	31.34
CT-ICP [DDJ21]	LO	0.0974	0.1820	Failed	Failed	0.3558	0.7935	0.0970	0.1362	Failed	213.41
KISS-ICP [VGM23b]	LO	0.0960	0.2016	0.3663	Failed	0.7398	0.9631	0.0718	0.1203	0.2789	62.38
LIO-SAM [SEM20]	LIO	0.0714	0.0718	0.0909	Failed	0.0811	0.8371	0.0784	0.1168	0.0932	51.57
FAST-LIO2 [XCH22]	LIO	0.0491	0.0608	0.0670	Failed	0.0594	0.2678	0.0872	0.1024	0.0646	28.04
DLIO [CNL23b]	LIO	0.0388	0.0610	0.0631	0.1559	0.0855	0.2866	0.0786	0.0983	0.0863	**25.99**
DLIOM	LIO	**0.0350**	**0.0571**	**0.0615**	**0.0686**	**0.0546**	**0.2608**	**0.0609**	**0.0904**	**0.0609**	27.89

Table 4.4: Comparison with MulRan DCC03

Algorithm	Relative Trajectory Error (RPE)	
	Translation [%]	Rotation [°/m]
LIO-SAM [SEM20]	**2.4**	0.009
FAST-LIO2 [XCH22]	6.8	0.030
Wildcat [RKC22]	2.9	0.010
DLIOM	**2.4**	**0.007**

since this particular extension had well-synchronized data.

The results are shown in Table 4.3 for all tested algorithms. Particularly of note are the two Hard (H) datasets, in addition to the Stairs dataset. For Quad (H) and Maths (H), both of which had highly aggressive and unpredictable movements, CT-ICP failed (even when reducing playback speed down to 10%), while both DLO and KISS-ICP had significantly higher trajectory errors. This demonstrates the strength and need for fusing inertial measurement units for point cloud motion correction. For Stairs, most algorithms failed to produce meaningful results due to the difficult ascension and the limited vertical field-of-view from the LiDAR sensor. Of those which could track sufficiently, both DLO and DLIO had significantly high errors; however, by detecting and placing a new keyframe right at the onset of slippage, DLIOM is able to achieve a low RMSE of just 0.0686m. This is further illustrated in Fig. 4.8, in which keyframes (white nodes) are placed at locations with high scene change (e.g., through the door, in-between stairs).

4.6.2.3 MulRan Dataset

To compare against Wildcat [RKC22], we use the MulRan DCC03 [KPC20] dataset. This is shown in Table 4.4 (results for LIO-SAM, FAST-LIO2, and Wildcat retrieved from [RKC22]). While the details of specifically how the relative trajectory error (RPE) was computed are unclear, we assume that the translational metric was the average RPE with respect to

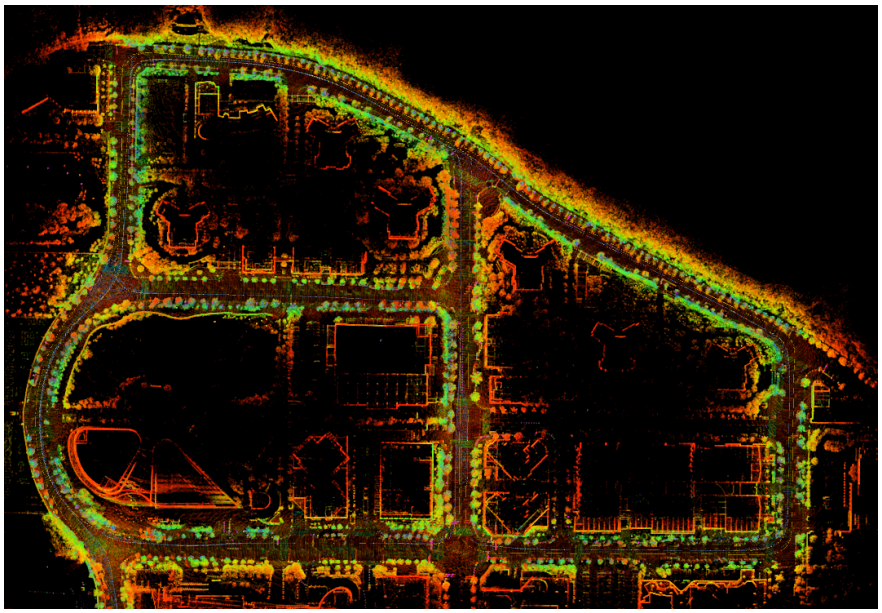

Figure 4.10: **MulRan DCC03.** Top-down view of the map of the MulRan DCC03 dataset, generated by DLIOM. This specific dataset featured approximately 5421.82 meters of travel from driving around three different loops in Korea.

the point distance error ratio using evo [Gru17], and the rotational metric was using evo's "rot_part" option. A fair comparison of absolute trajectory error could not be conducted, as numerical values were not provided by the authors, but DLIOM's ATE was on average 2.36m and 2.4°. Top-down map is shown in Fig 4.10.

4.6.2.4 UCLA Campus Dataset

We additionally showcase our method's accuracy using four large-scale datasets at UCLA for additional comparison (Fig. 3.7). These datasets were gathered by hand-carrying our aerial platform (Fig. 4.1) over 2261.37m of total trajectory. Our sensor suite included an Ouster OS1 (10Hz, 32 channels recorded with a 512 horizontal resolution) and a 6-axis InvenSense MPU-6050 IMU located approximately 0.1m below it. We note here that this IMU can be purchased for approximately $10, demonstrating that LIO algorithms need not require high-grade IMU sensors that previous works have used. Note that a comparison

Table 4.5: Comparison with UCLA Campus Dataset

Algorithm	Type	Royce Hall (A)		Court of Sciences (B)		Bruin Plaza (C)		Sculpture Garden (D)	
		652.66m		*526.58m*		*551.38m*		*530.75m*	
		Error [m]	Comp [ms]	Error [m]	Comp [ms]	Error [m]	Comp [ms]	Error [m]	Comp [ms]
DLO [CLA22a]	LO	0.0216	20.40	1.2932	20.77	0.0375	21.18	0.0178	21.62
CT-ICP [DDJ21]	LO	0.0387	351.85	0.0699	342.76	0.0966	334.15	0.0253	370.19
KISS-ICP [VGM23b]	LO	0.0689	50.28	0.4007	31.84	0.2412	35.31	0.0987	62.96
LIO-SAM [SEM20]	LIO	0.0216	33.21	0.0692	29.14	0.0936	39.04	0.0249	48.94
FAST-LIO2 [XCH22]	LIO	0.0454	15.39	0.0353	12.25	0.0363	14.84	0.0229	15.01
DLIO [CNL23b]	LIO	0.0105	**10.45**	0.0233	**8.37**	0.0301	**8.66**	0.0082	**10.96**
DLIOM	LIO	**0.0097**	12.91	**0.0228**	14.18	**0.0235**	15.37	**0.0078**	13.78

of absolute trajectory error was not possible due to the absence of ground truth, so as is common practice, we compute end-to-end translational error as a proxy metric (Table 4.5). In these experiments, our method outperformed all others across the board in end-to-end translational error. However, similar to the trends found in the Newer College datasets, our average per-scan computational time has slightly increased due to the new algorithmic additions since DLIO. Regardless however, our resulting maps can capture fine detail in the environment which ultimately provides more intricate information cues for autonomous mobile robots such as terrain traversability.

4.7 Discussion

This chapter presents Direct LiDAR-Inertial Odometry and Mapping (DLIOM), a reliable SLAM algorithm with an extreme focus on operational reliability and accuracy to yield real-time state estimates and environmental maps across a diverse set of domains. DLIOM mitigates several common failure points in typical LiDAR-based SLAM solutions through an architectural restructuring and several algorithmic innovations. Rather than using a single sensor fusion framework (e.g., probabilistic filter or graph optimization) to produce both localization and map as is typical in other algorithms, we separate these two processes into separate threads and tackle them independently. Leveraging a nonlinear geometric observer guarantees the convergence of IMU propagation towards LiDAR scan-matching and reliably initializes velocity and sensor biases, which is required for our fast coarse-to-fine motion correction technique. On the other hand, a factor graph, with nodes at keyframe locations determined by our odometry thread, continually optimizes for a best-fit map using connective factors between overlapping keyframes, which provide extra relative constraints to the optimization problem.

Fast and reliable localization is achieved hierarchically in the front-end's scan-matching, keyframing and submapping processes. An adaptive scan-matching method automatically

tunes the maximum distance between corresponding planes for GICP by computing a novel point cloud sparsity metric, resulting in more consistent registration in differently sized environments. Slip-resistant keyframing ensures a sufficient number of data correspondences between the scan and the submap by detecting abrupt scene changes using a new sensor-agnostic degeneracy metric. Finally, our submap is explicitly generated by computing the 3D Jaccard index between the current scan and each environmental keyframe to ensure maximal overlap in the submap for data correspondence searching. These ideas collectively enable a highly reliable LiDAR SLAM system that is not only agnostic to the operating environment, but is also fast and online for real-time usage on computationally-constrained platforms.

CHAPTER 5

Conclusion

Inspired by a real-world need for fast and consistent localization and mapping for computationally-limited robotic platforms, this dissertation presents four novel algorithms which attempt to address known deficiencies in current state-of-the-art systems. These algorithms seek to be simultaneously (1) fast and efficient with low computational complexity and high estimation accuracy through innovative algorithmic design, and (2) reliable and domain-agnostic with long-term operational reliability, enabled by their ability to generalize across a diverse set of operating environments without the need for manual intervention. These four core algorithms are summarized in the next section and all aim to help advance our society's engineering capabilities for autonomous mobile robots.

5.1 Summary of Contributions

- **Fast Localization with Dense Point Clouds.** A novel frontend localization solution that enables the direct use of dense point cloud scans without significant preprocessing. This is made possible through a custom speed-oriented pipeline with three core contributions to efficiently handle the large amounts of data from LiDAR sensors. This was the first of its kind, is open-source, and has attracted a significant amount of attention by the community.

- **Parallelizable Continuous-Time Motion Correction.** A fast and reliable LiDAR-inertial odometry algorithm that delivers accurate localization and detailed 3D map-

ping, made possible through a novel coarse-to-fine technique for constructing continuous-time trajectories for accurate and parallelizable point-wise motion correction. Up until this point, motion correction algorithms were either highly inefficient or highly inaccurate, but DLIO demonstrated the possibility for fast, accurate processing with minimal overhead through a condense archtecture which cut away several unnecessary processes.

- **Perceptive and Connective SLAM.** A new, reliable LiDAR SLAM algorithm that prioritizes computational efficiency, operational reliability, and real-world efficacy, which offers valuable insights into modern LiDAR SLAM systems, addressing common algorithmic failure points where current state-of-the-art algorithms struggle to achieve long-term operational reliability in the unstructured real world. This is made possible through several core innovations which are strategically located throughout both the front-end and back-end subsystems to target different scales in the data processing pipeline to create an architecture perceptive to the environment to comprehensively increase localization resiliency and mapping accuracy.

5.2 Limitations & Future Work

Limitations in the work presented in this dissertation expose several future directions for improving the proposed systems. For instance, improving the scalability of such algorthms would enable large-scale and long-term SLAM. We have shown that the time complexity of direct algorithms can be improved dramatically, but memory usage is still somewhat high. Improving on the space complexity through, for example, point cloud compression techniques would reduce the overall memory usage of such algorithms. Dynamic objects and detecting changes in the map would allow the robot to build an internal map of only static objects, which would improve map generalization and ultimately localization quality. In addition, the presented algorithms are all purely *geometric*. That is, the output pose and map representation are strictly in Euclidean space and solely provide metric measurements to

the system (i.e., XYZ pose, point cloud maps containing XYZ points, etc.). Conversely, there has been significant recent interest in *semantic* methods in both 2D and 3D domains, such as semantic segmentation and semantic SLAM. One could easily imagine pushing the work presented in this document even further to create fast and reliable *semantic* robot perception. Current methods for semantic perception involve the requirement of properly labeled images with ground truth semantic segmentation, which is tedious and often time-consuming, for machine learning algorithms. However, the intensity channel from LiDAR returns are often some function of distance and material (to represent the reflectivity of the object a point collided with). If, perhaps, this relationship could be explicitly formulated, the intensity would provide a notion of "semantics" which would not require any sort of training scheme. Such a process could better constrain SLAM algorithms by providing more information about the outside world, and how the intensity of a certain material changes as a function of distance for data association, to create 3D semantic LiDAR SLAM. Additionally, research on 3D semantic segmentation would pave the way for more intelligent path planning by enabling agents to truly understand their outside world. While there are several groups who are currently studying this problem (mostly in the 2D domain), there still remains several open questions towards this pursuit. What does it even mean for a system to "understand" its environment? How can we best use this understanding for navigation and task completion? Is there a proper measurement for optimal understanding of the environment? Such open questions lie at the intersection of engineering, neuroscience, and philosophy, and could very well never be answered.

Another potential future direction is to augment the pipeline with an additional exteroceptive sensor, such as a radar, to further help constrain localization on both local and global scales. Using a frequency-modulated continuous-wave (FMCW) radar, where a continuous-wave radio energy of known frequency is transmitted by the device the received, one can detect not only individual objects but also each object's movement speed by analyzing the difference in transmitted and received frequency; this is possible from the Doppler

phenomenon and is known as "Doppler velocity." This can be useful for SLAM is several different ways. A notable deficiency in state estimation using LO/LIO methods is its inability to directly measure linear velocity. While angular velocity can be directly measured via an IMU's gyroscope (and therefore can be integrated to get angular position), linear velocity (which is required for point cloud motion correction, optimization prior construction, etc) is estimated from integrating the linear acceleration from an accelerometer. It is important to know that integration of such measurements in general suffer from accumulated error. However, double-integrating linear acceleration to retrieve position is especially dangerous, since any measurement errors, however small, are accumulated over time, and due to integration error, a constant error in acceleration results in linear error in velocity and quadratic error growth in position. Thus, a direct measurement of linear velocity will decrease overall system sensitivity to measurement noise. In general, multimodal sensor fusion is still an open question, and further research into how best to combine various sensing modalities to maximize each sensor's utility (rather than, for example, throwing everything into a single pose graph) is crucial for the future of autonomy.

5.3 Concluding Remarks

Robotics is really, *really* hard. However, truly autonomous robots would be unparalleled in terms of societal impact and would provide unimaginable value to our human race. Fully self-driving vehicles would drastically improve road safety, reduce traffic congestion, and improve the carbon footprint of transportation as a whole. Healthcare robotics would help reduce the workload of overworked healthcare professionals by assisting in medication delivery and surgeries—improving overall patient satisfaction and outcomes. In agriculture, tasks such as planting, harvesting, and monitoring crops could be performed by autonomous robots to increase crop yield and improve the efficiency of farming operations. Robots could also be used to better monitor the environment by collecting data on air and water quality or

wildlife populations, helping researchers better understand the impact of human activities and develop strategies to mitigate it. Search and rescue robots could locate and help extract survivors in disaster-stricken areas, entering areas too dangerous for human rescuers and potentially saving countless lives. Finally, exploratory robots could be used in space or deep underwater exploration to collect data, conduct scientific experiments, search for life on other planets, and help scientists better understand the universe—potentially unlocking the meaning to our human existence. The potential for autonomous robots is limited only by our imagination and our ability to develop the corresponding technologies.

This dissertation may only be a small piece in the vast knowledge required to achieve truly autonomous robots, but if the contents of this document help with the advancement of our robotic capabilities even a little, then I will have succeeded in my goals as a graduate student and hope to continue doing so for the remainder of my time in this universe. As we continue to push the boundaries of our robotic capabilities, the need for passionate, curious scientists and engineers will grow. I encourage anyone with an interest in robotics to join this journey with me.

APPENDIX A

Unsupervised Monocular Depth Learning with Integrated Intrinsics and Spatio-Temporal Constraints

Monocular depth inference has gained tremendous attention from researchers in recent years and remains as a promising replacement for expensive time-of-flight sensors, but issues with scale acquisition and implementation overhead still plague these systems. To this end, this work presents an unsupervised learning framework that is able to predict at-scale depth maps and egomotion, in addition to camera intrinsics, from a sequence of monocular images via a single network. Our method incorporates both spatial and temporal geometric constraints to resolve depth and pose scale factors, which are enforced within the supervisory reconstruction loss functions at training time. Only unlabeled stereo sequences are required for training the weights of our single-network architecture, which reduces overall implementation overhead as compared to previous methods. Our results demonstrate strong performance when compared to the current state-of-the-art on multiple sequences of the KITTI driving dataset and can provide faster training times with its reduced network complexity.

A.1 Overview

Modern robotic agents take advantage of accurate, real-time range measurements to build a spatial understanding of their surrounding environments for collision avoidance, state estimation, and other navigational tasks. Such measurements are commonly retrieved via active sensors (e.g., LiDAR) which resolve distance by measuring the time-of-flight of a reflected

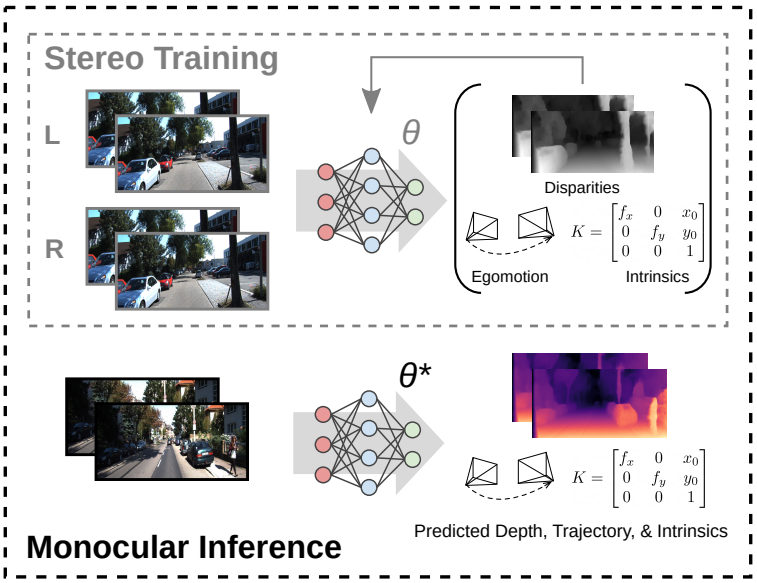

Figure A.1: **System Overview.** Our system regresses depth, pose and camera intrinsics from a sequence of monocular images. During training, we use two pairs of unlabeled stereo images and consider losses in both spatial and temporal directions for our network weights. During inference, only monocular images are required as input, and our system outputs accurately scaled depth maps and egomotion in addition to the camera's intrinsics.

light signal; however, these sensors are often costly [RB19], difficult to calibrate and maintain [Kat03, ML10], and can be unwieldy for platforms with a weight budget [LH17]. *Passive* sensors, on the other hand, have seen a tremendous surge of interest in recent literature to predict scene depth from input imagery using multi-view stereo [SCD06, SZF16, FCS10], structure-from-motion [HZ03, NLD11, SF16, DST00], or more recently, purely monocular systems [GLJ19, GMF19, BLW19, ZBS17, YS18, ZLH18], due to their smaller form factor and increasing potential to rival the performance of explicit active sensors with the advent of machine learning.

In particular, monocular depth inference is attractive since RGB cameras are ubiquitous in modern times and requires the least number of sensors, but this setup suffers from a fundamental issue of scale acquisition. More specifically, in a purely monocular system, depth can only be estimated up to an ambiguous scale and requires additional geometric information to resolve the units of the depth map. Such cameras typically capture

frames by projecting 3D scene information onto a 2D image plane, and abstracting higher dimensional depth information from a lower dimension is a fundamentally ill-posed problem. To resolve the scale factors of these depth maps, a variety of learning-based approaches have been proposed with differing techniques to constrain the problem geometrically [PAT18, GAB17, GMF19, GBC16, SCN06, LSL15, SSN08, TTL17, SSC19, GR20]. Temporal constraints, for example, are commonly employed [GLJ19, SSC19, FGW18, XW18] and is defined as the geometric constraint between two consecutive monocular frames, aiming to minimize the photometric consistency loss after warping one frame to the next. Spatial constraints [GAB17, GMF19, PAG19], on the other hand, extract scene geometry not through a forward-backward reconstruction loss (i.e., temporally) but rather in left-right pairs of stereo images with a predefined baseline. Most works choose to design their systems around either one or the other, and while a few systems have integrated both constraints before in a multi-network framework [ZGW18, LWL18, MDM18], none have taken advantage of both spatial and temporal constraints in a single network to resolve these scale factors.

To this end, we developed an unsupervised, single-network monocular depth inference approach that considers both spatial and temporal geometric constraints to resolve the scale of a predicted depth map. These "spatio-temporal" constraints are enforced within the reconstruction loss functions of our network during training (Fig. A.1), which aim to minimize the photometric difference between a warped frame and the actual next frame (forward-backward) while simultaneously maximizing the disparity consistency between a pair of stereo frames (left-right). Unlike previous approaches, we consider camera intrinsics as an additional unknown parameter to be inferred and demonstrate accurate inference of both depth and camera parameters through a sequence of purely monocular frames; this is all performed in a single end-to-end network to minimize implementation overhead.

Our spatio-temporal network is inspired by [ZGW18, MDM18, LWL18] that uses an effective combination of losses to simultaneously regress depth and egomotion in a single network. Additionally, to provide freedom from manual calibration, the network is

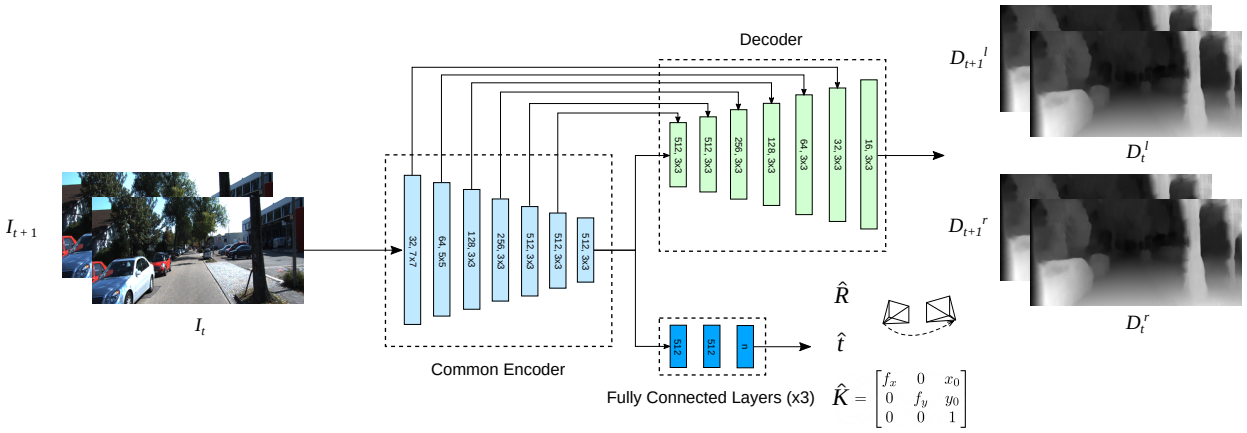

Figure A.2: **Architecture Overview.** Our system uses a common convolutional-based encoder between the different outputs, which compresses the input images into a latent space representation. This representation is then sent through either a trained decoder to retrieve left-right stereo image disparities, or through different groups of fully connected layers to estimate egomotion ($n = 3$) or camera intrinsics ($n = 4$). In the common encoder, each block uses a series of two convolutional layers, the first with stride 2 and the second stride 1 (zero padding), and with input dimensions and kernel sizes as specified. The transposed convolutional blocks in the decoder are similarly structured, with pooling indices received from the corresponding encoder's feature maps.

also capable of learning camera intrinsics which can be useful when a video source is unknown [SSC19, GLJ19]. To predict depth, we use a photoconsistency loss between stereo image pairs, a left-right consistency loss between image disparity maps [GAB17], and a disparity map smoothing function [HKJ13]. To estimate egomotion and camera intrinsics, we leverage a unique loss function that accounts for the photometric difference between temporally adjacent images. By combining these losses, we show that we can obtain scaled visual odometry information and accurate camera parameters. Furthermore, by observing the similarities between the architecture of the depth network encoder and the pose network's convolutional layers, we can effectively eliminate architecture redundancy by merging them via a common encoder (Fig. A.2) and can provide faster training times without significant loss in performance.

Our main contributions are as follows: (1) we develop an unsupervised, single-network architecture for monocular depth inference which takes advantage of the geometric con-

straints found in both spatial and temporal directions; (2) a novel loss function that integrates unknown camera intrinsics directly into the single-network training procedure; and (3) extensive performance and run-time analyses of our proposed architecture to verify our methods. These efforts were in support of NASA's Jet Propulsion Laboratory's Networked Belief-aware Perceptual Autonomy (NeBula) framework [AOM21] as part of Team CoSTAR in the DARPA SubT Challenge.

A.2 Related Work

Depth estimation using monocular images and deep learning began with supervised methods over large datasets and ground truth labeling [EPF14, EF, LSL15]. Although these methods produced accurate results, acquiring ground truth data for supervised training requires expensive 3D sensors, multiple scene views, and inertial feedback to obtain even sparse depth maps [GBC16]. Later work sought to address a lack of available high-quality labeled data by posing monocular depth estimation as a stereo image correspondence problem, where the second image in a binocular pair served as a supervisory signal [GBC16, GAB17, PAT18]. This approach trained a convolutional neural network (CNN) to learn epipolar geometry constraints by generating disparity images subject to a stereo image reconstruction loss. Once trained, networks were able to infer depth using only a single monocular color image as input. While this work achieved results comparable to supervised methods in some cases, occlusion and texture-copy artifacts that arose with stereo supervision motivated learning approaches using a temporal sequence of images as an alternative [GAB17, GMF19]. CNNs trained using monocular video regressed depth using the camera egomotion to warp a source image to its temporally adjacent target. To address the additional problem of camera pose, [ZBS17, BLW19, YS18, ZLH18, GMF19, GLJ19, GAP20, YWW18] trained a separate pose network.

The learning of visual odometry (VO) and depth maps has useful application in visual

simultaneous localization and mapping (SLAM). Visual SLAM leverages 3D vision to navigate an unknown area by determining camera pose relative to a constructed global map of an environment. To build and localize within a map, VO in a SLAM pipeline must solve at metric scale. Geometric approaches to monocular SLAM using first principled solutions, such as structure from motion (SfM) [KV91], resolved scaling issues using external information [CWW, GR20]. Building on such methods, work in data-driven monocular VO obtained scale using sources such as GPS sensor fusion [PL17] or training supervision [CWW, GAP20]. Unsupervised approaches using a camera alone remain attractive however, due to the reduction of manual effort associated with fewer sensors. Promising research in this area combined visual constraints (e.g., monocular depth [ZBS17, BLW19, YS18, ZLH18, GMF19, GLJ19], stereo depth [ZGW18, MDM18, LWL18, GR20], or optical flow [ZLH18, LY19]) to achieve scale consistent outcomes.

Network architecture for visual odometry and dense depth map estimation separate depth and pose networks into two CNNs, one with convolutional and fully connected layers and the other an encoder-decoder structure [BKC17], respectively. In the case where only monocular images are used in training, the self-supervision inherent in estimation is less constrained, having only pose generated from temporal constraints to determine depth, and vice versa [LY19]. The work of [VR17, ZBS17] for example, suffered from scaling ambiguity issues [ZGW18]. Training using binocular video, on the other hand, made use of independent constraints from spatial and temporal image pairs that offered an enriched set of sampled images for network training. This "spatio-temporal" approach allowed for the regression of depth from spatial cues generated by epipolar constraints, which were then passed to the pose network to independently estimate VO using temporal constraints [ZGW18, MDM18, LWL18, GMF19, LY19].

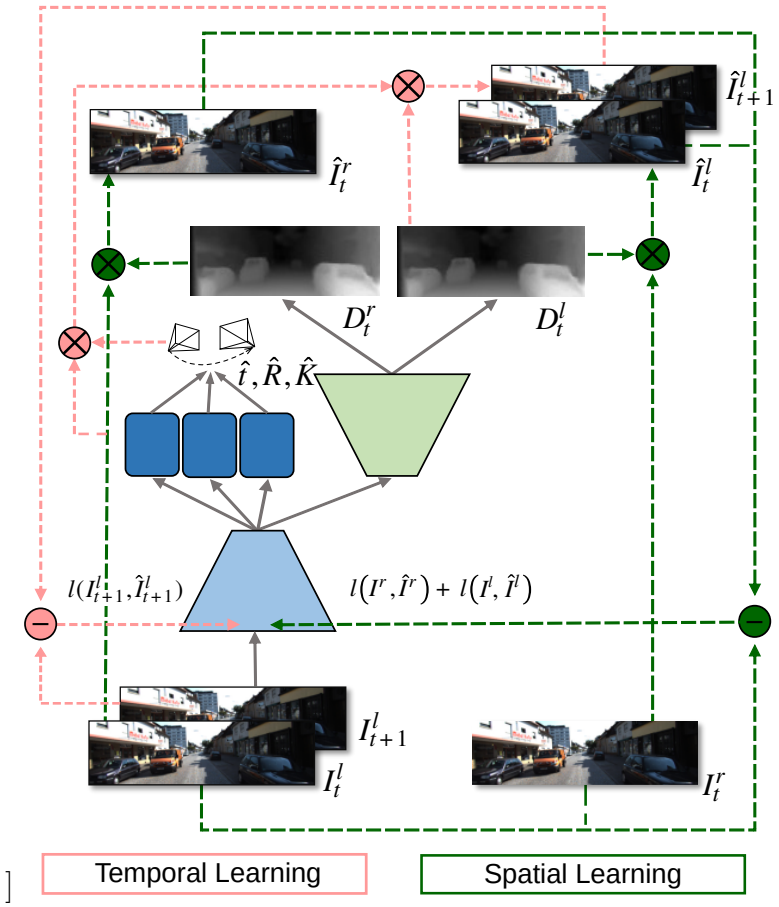

Figure A.3: **Training Diagram.** Our single-network system runs a timed sequence of left images through the common encoder (light blue trapezoid) to generate outputs that are fed to the fully connected (FC) layers (blue rectangles) and the decoder (green trapezoid). Outputs from the FC layers and the decoder are the camera pose and intrinsics, and disparity maps, respectively. The disparities are used to find left-right reprojected images (green dashed lines), while the disparities, camera pose and intrinsics determine the temporal reprojections (pink dashed lines). All input and output images are framed in black for clarity.

A.3 System Description

Our framework is inspired by [ZGW18, MDM18, LWL18], but rather than requiring two separate networks for depth and pose estimation, we use a common encoder for both tasks in a novel single-network architecture (Fig. A.3). That is, given two temporally adjacent input images at times t and t', our network first convolves these inputs through a series of convolutional blocks in a common encoder, and then predicts either disparities through a de-

coder, or camera pose and intrinsics through fully connected layers. In the decoder network, the encoder's latent representation of the input images is re-upsampled using transposed convolutions with pooling indices from the encoder to fuse low-level features, as inspired by [GAB17, ZGW18]. We use rectified linear units (ReLU) [NH10] as activation functions in all layers of this decoder except for the prediction layer, which uses a sigmoid function instead. The decoder predicts left-to-right and right-to-left disparities D at both timesteps, which are then either used to reconstruct the right stereo images for a spatially-constrained geometric loss during training via bilinear sampling, or used to construct the depth during inference. In the fully connected layers, translation $\hat{t}_{t' \to t}$, rotation $\hat{R}_{t' \to t}$, and camera intrinsics $\hat{K}$ are predicted independently in three separate and decoupled groups of fully connected layers for better performance [LWL18]. These outputs are then either taken at face value during inference as the predicted egomotion and camera parameters, or used as inputs (along with the estimated depth map) to warp the current frame to the next for our temporal reconstruction loss as described previously.

A.3.1 Notation

A color image, I, is composed of pixels with coordinates $p_{ij} \in \mathbb{R}^2$, where $I_{ij} = I(p_{ij})$. In temporal training, we denote images at time t as I^t, and images temporally adjacent as source frame $I^{t'}$. A pixel at time t' is transformed to its corresponding pixel at time t using homogeneous transformation matrix $T_{t' \to t} \in \mathbb{SE}(3)$ and camera intrinsics matrix $K \in \mathbb{R}^{3 \times 3}$, where pixels in homogeneous coordinates, $\tilde{p} = (p, 1)^T$, are denoted p for simplicity.

Rectified stereo image pairs are given by I^r, I^l, where the superscripts for time have been dropped for convenience, and superscripts l, r correspond to the left and right images respectively. D^l represents the disparity map that warps I^r to the corresponding I^l, and we define per pixel disparity as $d^l_{ij} = D^l(p_{ij})$. Thus $I^l_{ij} = I^r_{i+d^l,j}$, and $d^r_{i+d^l,j} = D^r(p_{i+d^l,j})$ is the disparity that does the reverse operation. Depth per pixel z is then determined by the relation, $z = Bf_x/d$, where f_x is the x-component focal length and B is the horizontal

baseline between stereo cameras.

A.3.2 Preliminaries

We can obtain the projected pixel coordinates and depth map using equation,

$$z^t p^t = K R_{t' \to t} K^{-1} z^{t'} p^{t'} + K t_{t' \to t} \,, \tag{A.1}$$

where the intrinsics matrix, K, is written explicitly as:

$$K = \begin{bmatrix} F & X_0 \\ 0 & 1 \end{bmatrix} \,,\; F = diag(f_x,\; f_y),\; X_0 = [x_0,\; y_0]^T \,, \tag{A.2}$$

and R and t are the rotation matrix and translation vector arguments of transformation matrix T [GLJ19]. Note that in this work we assume no lens distortion and a zero skew coefficient in the camera, and that stereo cameras have equal intrinsic parameters. Equation (A.1) constitutes the temporal reconstruction loss at training used to determine the camera egomotion, R and t, and the camera intrinsics K in a single network.

A.4 Single-Network Architecture

A.4.1 Optimization Objective

Our loss function is made up of a novel temporal reconstruction term and four spatial reconstruction terms [GAB17], [GR20]. Error regression for the following losses allows the network to correctly predict a target image temporally and spatially during training in order to infer depth, pose, and camera intrinsics from a monocular image sequence at test time. The temporal reconstruction term of the loss function is implicitly defined where $l_{te}(I^{l,t}, I^{l,t'}) \to \hat{I}^{l,t}$, and the spatial reconstruction terms are composed of a photoconsistency

loss, l_p, a left-right consistency loss l_{lr}, and a disparity smoothness loss l_r,

$$\begin{aligned} l\left(f(I^l;\theta), I^l, I^r\right) = \lambda_p\big(l_p(I^l, \hat{I}^l) + l_p(I^r, \hat{I}^r)\big) \\ + \lambda_{te} l_p(I^{l,t}, \hat{I}^{l,t}) + \lambda_{lr} l_{lr}\left(D^l, D^r\right) \\ + \lambda_r\left(l_r(D^l, I^l) + l_r(D^r, I^r)\right) . \end{aligned} \tag{A.3}$$

The argument I in the loss function is the original image and $\hat{I}$ is the reprojected image, and individual losses are weighted by λ labeled with corresponding subscripts.

A.4.2 Spatio-Temporal Reconstruction Loss

The *photoconsistency loss* compares image appearance using the structural simularity index measure (SSIM) and an absolute error between generated and sampled images [WBS04, GAB17, MDM18]:

$$l_p(I, \hat{I}) = \frac{1}{N}\sum_{i,j} \alpha \frac{1 - SSIM\left(I_{ij}, \hat{I}_{ij}\right)}{2} + (1-\alpha)\left|I_{ij} - \hat{I}_{ij}\right| . \tag{A.4}$$

The loss is composed of three terms in total (two spatial losses and a temporal loss). N in this equation is the number of image pixels and the weight α is set to 0.85.

For reprojected images, we assume equal camera intrinsics produce right and left stereo images. The focal length $\hat{f}_x$ from instrinsics matrix $\hat{K}$ in (A.2) is co-predicted via the learned disparity and penalized using spatial reconstruction losses. For stereo image inputs, predicted disparity maps are used to generate the left view from a right image, and vice versa. Depth values calculated from the disparity maps are then input to the temporal reconstruction loss to generate the left target image from temporally adjacent source images, i.e. to generate the temporal image arguments for (A.4), we put (A.1) in the form where for

pixels $P = \{p_i, i = 1 \dots N\}$,

$$\sum_{i,j} |I_{ij}^{l,t} - \hat{I}_{ij}^{l,t}| \rightarrow \sum_{p \in P} \left| z^t p^t - \left[\hat{K} \hat{R}_{t' \rightarrow t} \hat{K}^{-1} \frac{b \hat{f}_x}{d^l} p^{t'} + \hat{K} \hat{t}_{t' \rightarrow t} \right] \right|, \tag{A.5}$$

is the absolute error between the left image and the reprojected image, and the structure similarity measure is generated by the same mappings between I_{ij} and pixel p_i.

We distinguish this loss function from previous works in the following ways: depth estimation in the above temporal relation is derived using spatial losses from within the same network, and the temporal loss infers both egomotion and camera intrinsics. This goes beyond work that used solely temporal constraints and a separate depth network [GLJ19], [SSC19] or spatio-temporal work that used predetermined intrinsics and a separate depth network [GMF19], [LWL18, MDM18, ZGW18], [LY19].

A.4.3 Spatial Reconstruction Loss

The *left-right disparity consistency loss* is used to obtain consistency between disparity maps [GAB17]. During training, the network predicts disparity maps D^l and D^r using only left image sequences as input and then penalizes the difference between the left-view disparity map and the warped right view, as well as the right-view and the warped left view,

$$l_{lr}(D^l, D^r) = \frac{1}{N} \sum_{i,j} \left| d_{ij}^l - d_{i+d^l,j}^r \right| + \left| d_{ij}^r - d_{i+d^r,j}^l \right|. \tag{A.6}$$

The *disparity smoothness loss* penalizes depth discontinuities that occur at image gradients ∂I [HKJ13]. To obtain locally smooth disparities, an exponential weighting function is used on disparity gradients ∂d:

$$l_r(D, I) = \frac{1}{N} \sum_{i,j} \left| \partial_x d_{ij} \right| e^{-|\partial_x I_{ij}|} + \left| \partial_y d_{ij} \right| e^{-|\partial_y I_{ij}|}. \tag{A.7}$$

A.4.4 Learning Camera Intrinsics

For predicted parameters $\hat{K},\ \hat{R},\ \hat{t}$ in (A.1), penalizing differences via training loss ensures $\hat{K}\hat{t}$ and $\hat{K}^{-1}\hat{R}\hat{K}$ converge to the correct values. To determine parameters individually, the translational relation fails because it is under-determined since there exists incorrect values of $\hat{K}$ and $\hat{t}$ such that $\hat{K}\hat{t} = Kt$. The rotational relationship, $\hat{K}\hat{R}\hat{K}^{-1} = KRK^{-1}$, however, does uniquely determine $\hat{K},\ \hat{R}$ such that they are equal to $K,\ R$, and therefore provides sufficient supervisory signal to estimate these values accurately.

Proof: From the above relation we obtain $\hat{R} = \hat{K}^{-1}KRK^{-1}\hat{K}$, and we constrain $\hat{R}$ to be $SO(3)$, i.e. $\hat{R}^T = \hat{R}^{-1}$ and $det(\hat{R}) = 1$. Substituting $\hat{R}$ into the relationship $\hat{R}\hat{R}^T = I$, we find that $AR = RA$ where $A = K^{-1}\hat{K}\hat{K}^TK^{-T}$. The value $det(\hat{K}^{-1}KRK^{-1}\hat{K})$ is equal to 1, therefore the determinant of A is also equal to 1. Moreover, the characteristic equation of A shows A always has an eigenvalue of 1 [GLJ19]. Thus the eigenvalue of A is equal to 1 with an algebraic multiplicity of 3, implying A is the identity matrix, or the eigenvalues are unique. If we assume A has 3 distinct eigenvalues, because $A \in \mathbb{R}^{3\times3}$ and $A = A^T$, we may choose the eigenvectors of A such that they are real. But because $AR = RA$, for every eigenvector, v of A, Rv is also an eigenvector. For an eigenvalue with algebraic multiplicity 1, the corresponding eigenspace is $dim(1)$, thus $R\text{v} = \mu\text{v}$ for some scalar μ, implying each eigenvector of A is also an eigenvector of R. If R is $SO(3)$, however, it has complex eigenvectors in general, which contradicts this assertion. Therefore A must be the identity matrix, and $\hat{K}\hat{K}^T = KK^T$. Referring to K from (A.2), we observe,

$$KK^T = \begin{bmatrix} FF + X_0X_0^T & X_0 \\ X_0^T & 1 \end{bmatrix}, \tag{A.8}$$

which implies $\hat{X}_0 = X_0$ and $\hat{F} = F$, or $\hat{K} = K$. ■

It is clear from above that for $R = I$, the relation $AR = RA$ holds trivially, and $\hat{K}$ cannot be uniquely determined. Thus the tolerances with which F in (A.2) can be determined (in

units of pixels) with respect to the amount of camera rotation that occurs is quantified as,

$$\delta f_x < \frac{2f_x^2}{w^2 r_y}; \quad \delta f_y < \frac{2f_y^2}{h^2 r_x}, \tag{A.9}$$

where r_x and r_y are the x and y-axis rotation angles (in radians) between adjacent frames, and w and h are the width and height of the image, respectively. For a complete proof on the relation between the strength of supervision on K and the closeness of R to I, see [GLJ19].

A.5 Experimental Results

In this section, we evaluate our proposed framework using the KITTI driving dataset [MG15]. Network architecture was implemented using the TensorFlow framework [AAB15] and models were trained on a single NVIDIA GeForce RTX 2070 Super GPU with 8GB of memory using a batch size of 4. Adam optimizer [KB14] was used to train the network parameters, with exponential decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.99$ and learning rate α initially set to 0.001 but gradually decreased throughout training. Standard data augmentation techniques were used to increase the size of the dataset during training.

We compare our method against the current state-of-the-art using conventional metrics (i.e., Abs Rel, Sq Rel, RMSE, RMSE log, and Accuracy for depth, and Absolute Trajectory Error (ATE) for egomotion) as per [ZSZ20]. Table A.1 provides a quantitative overview of how our work fits in current literature, and Fig. A.4 accompanies this table with a visual, qualitative comparison. Tables II and III evaluate odometry and camera intrinsics estimation performance, and Fig. A.5 shows a comparison of trajectory on the KITTI Odometry dataset. We additionally evaluate our framework's run-time performance to show the benefits of our reduced network size and overhead while maintaining comparable performance with current methods which can be seen in Fig. A.6.

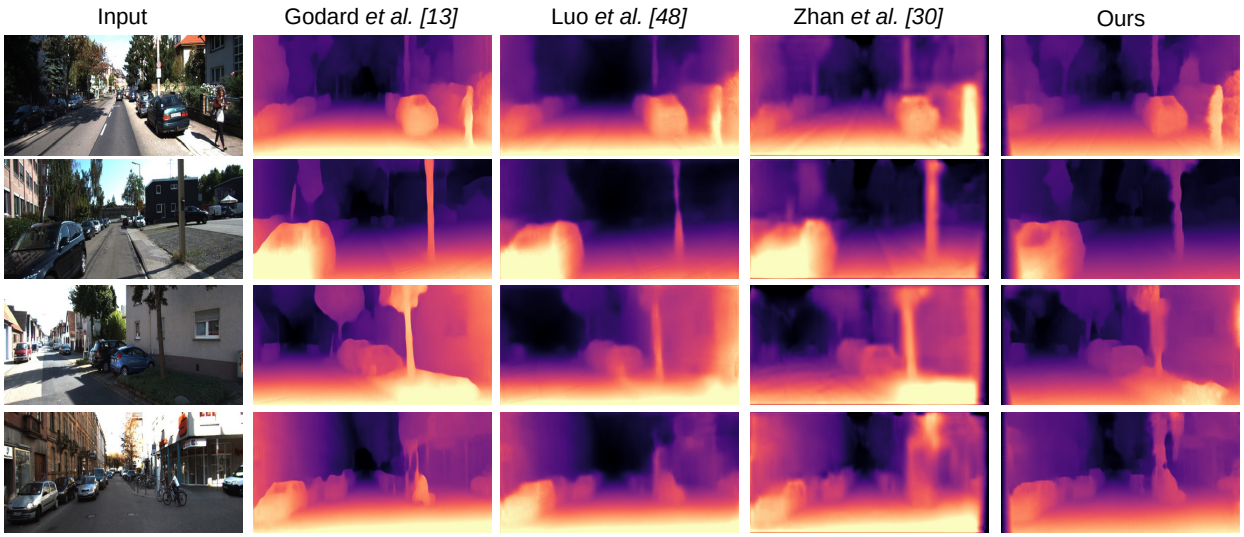

Figure A.4: **Qualitative Comparison of Depths.** Visual comparison of regressed depth maps between our method and various state-of-the-art methods ([GMF19, LY19, ZGW18]) on four images from the KITTI Eigen split (images of other methods were retrieved from [GMF19]). Even with a reduced size and complexity, our network can accurately regress depth maps given a single monocular image.

A.5.1 Performance of Depth Estimation

To evaluate the performance of our network's depth inference, we use a standard Eigen split [EPF14] on the KITTI dataset [MG15] as per convention and compare against several state-of-the-art methods with a depth cap of $80m$. We train, validate, and test our network using these splits and compare our depth estimation accuracy against multiple other works across several metrics, as shown in Table A.1 Ground truth data for the testing set is calculated via projecting the Velodyne LiDAR data onto the image plane.

Several important observations can be extracted from Table A.1. First, and most notably, our method is the only one which provides a combined architecture for both pose and depth regression (as indicated in the "Comb." column). Whereas all previous methods split egomotion and depth estimation tasks into separate pipelines, we reduce network redundancy (and hence, number of parameters needed for training) by sharing the input latent representation for both tasks via a common encoder. A second observation is that only two

Table A.1: Comparison of Monocular Depth Estimation.

Cropped regions from [GAB17] were used for performance evaluation all methods. In the column labeled "Type", "D" indicates supervised training with ground truth, "T" indicates temporal training only, "S" indicates spatial training only, and "ST" indicates a spatio-temporal training approach. Column "A.C." denotes whether additional components (such as pre-/post-processing methods) were used in addition to neural networks, and column "Comb." denotes whether pose and depth networks were combined. Column "Int." denotes whether that method can simultaneously regress camera intrinsics {Yes (Y), No (N)} (a dash "-" indicates no applicability). We evaluate using the Eigen split [EPF14] on the KITTI dataset [MG15] and cap depth to $80m$ as per standard practice [GAB17]. Results from other methods were taken from their corresponding papers. For error metrics, lower is better; for accuracy, higher is better.

					Error Metrics				Accuracy Metrics		
Method	Type	A.C.	Comb.	Int.	Abs Rel	Sq Rel	RMSE	RMSE_{log}	<1.25	$<1.25^2$	$<1.25^3$
T.S. Mean	D	-	-	-	0.361	4.826	8.102	0.377	0.638	0.804	0.894
[ZLH18]	T	**Y**	N	N	0.150	1.124	5.507	0.223	0.806	0.933	0.973
[YS18]	T	**Y**	N	N	0.149	1.060	5.567	0.226	0.796	0.935	0.975
[SSC19]	T	N	N	**Y**	0.135	1.070	5.230	0.210	0.841	0.948	0.980
[GLJ19]	T	**Y**	N	**Y**	0.128	0.959	5.230	-	-	-	-
[GAP20]	T	**Y**	N	N	**0.111**	**0.785**	**4.601**	**0.189**	**0.878**	**0.960**	**0.982**
[GBC16]	S	N	-	N	0.177	1.169	5.285	0.282	0.727	0.896	0.958
[GAB17]	S	N	-	N	0.148	1.344	5.927	0.247	0.803	0.922	0.964
[PAT18]	S	N	-	N	0.163	1.399	6.253	0.262	0.759	0.911	0.961
[PAG19]	S	N	N	N	0.116	0.935	5.158	0.210	0.842	0.945	0.977
[LY19]	ST	**Y**	N	N	0.127	0.936	5.008	0.209	0.841	0.946	0.979
[GMF19]	ST	**Y**	N	N	0.127	1.031	5.266	0.221	0.836	0.943	0.974
[LWL18]	ST	N	N	N	0.183	1.730	6.570	0.268	-	-	-
[MDM18]	ST	N	N	N	0.139	1.174	5.590	0.239	0.812	0.930	0.968
[ZGW18]	ST	N	N	N	0.144	1.391	5.869	0.241	0.803	0.928	0.969
Ours	ST	N	**Y**	**Y**	0.141	1.227	5.629	0.239	0.809	0.927	0.962

Table A.2: Regressed Camera Intrinsics Compared to Ground Truth.
Note that ground truth values have been adjusted to match the scaling and cropping done for training. All values are in units of pixels.

Camera Parameter	Learned	Ground Truth
Horizontal Focal Length (f_x)	298.4 ± 2.3	295.8
Vertical Focal Length (f_y)	483.1 ± 3.6	489.2
Horizontal Principal Point (x_0)	254.8 ± 2.4	252.7
Vertical Principal Point (y_0)	127.8 ± 1.7	124.9

other works before ours are also capable of estimating camera intrinsics during the inference phase ([SSC19, GLJ19], as indicated in the "Int." column). From these observations, our proposed method, to the best of our knowledge, is the first to demonstrate a simultaneous regression of pose, depth, and camera parameters in a reduced single-network design that uses both spatial and temporal geometric constraints.

However, this reduction in network complexity may come with trade-offs. First, through a quantitative lens, Table A.1 shows that our method is not the lowest in depth estimation error or highest in accuracy. With a reduced network complexity, a possible explanation lies in our encoder's shared network parameters that must balance both depth and pose/intrinsics pathways. However, even then, our error and accuracy is still strongly comparable to many state-of-the-art methods, many of which have additional components to compensate for occlusions, motion, etc. Compared to the current best with the lowest error and highest accuracy [GAP20], we are on average ∼75.9% as error-free and ∼95.6% as accurate. This can also be seen qualitatively in Fig. A.4. Our method lacks slightly in sharpness compared to [GMF19] but can provide finer edges than [LY19] and [ZGW18]. Depending on one's setup, the benefits of faster training through a more compact network could outweigh such trade-offs.

A.5.2 Learned Camera Intrinsics

To evaluate our system's ability to recover camera intrinsics (i.e., f_x, f_y, x_0, y_0) through the supervisory signal provided by the rotational component of (A.1), we follow a similar procedure as [GLJ19] and trained separate models on several different video sequences until convergence of these parameters for multiple independent results. During training, parameters were randomly initialized to begin with and empirically had no convergence issues throughout our experiments. We used ten video sequences of the "2011_09_28" subdataset chosen to have the same ground truth calibration done that day, and Table A.2 shows the resulting mean and standard deviation of those ten tests. All experiments were done on the left stereo color camera ("image_02") of the vehicle setup. For all four variables, we observe that, on average across all ten tests, our method can learn the parameters accurately and within a reasonable bound.

A.5.3 Egomotion

We carried out our pose estimation performance evaluation using four sequences from the KITTI Odometry dataset [GLU12] and compared against several state-of-the-art methods, including UnDEMoN [MDM18], SfMLearner [ZBS17], and VISO-M [GZS11]. For a quantitative comparison, we adopt the absolute trajectory root-mean-square error (ATE) for both translational (t_{ate}) and rotational (r_{ate}) components per standard practice [SEE12b]. We note that we used the same model that was trained for depth estimation to output our egomotion estimation, and that these four test sequences were not part of our training set. Sequence 07 is shown in Fig. A.5.

From Table II, we observe that for both translational and rotational errors in all four sequences, our method outperformed SfMLearner [ZBS17] and VISO-M [GZS11] and is comparable with UnDEMoN's [MDM18] performance. In contrast to these methods, our system co-predicts egomotion and camera intrinsics (alongside disparity) in a single network such

Table A.3: Comparison of Odometry Estimation

This comparison includes [MDM18, ZBS17, GZS11] and uses absolute trajectory error for translation (t_{ate}) and rotational (r_{ate}) movement. Comparison was done on four sequences of the KITTI dataset.

Seq.	Ours		UnDEMoN [MDM18]		SfMLearner [ZBS17]		VISO-M [GZS11]	
	t_{ate}	r_{ate}	t_{ate}	r_{ate}	t_{ate}	r_{ate}	t_{ate}	r_{ate}
00	0.0712	0.0014	**0.0644**	0.0013	0.7366	0.0040	0.1747	**0.0009**
04	**0.0962**	0.0016	0.0974	**0.0008**	1.5521	0.0027	0.2184	0.0009
05	**0.0689**	**0.0009**	0.0696	**0.0009**	0.7260	0.0036	0.3787	0.0013
07	0.0753	0.0013	**0.0742**	**0.0011**	0.5255	0.0036	0.4803	0.0018

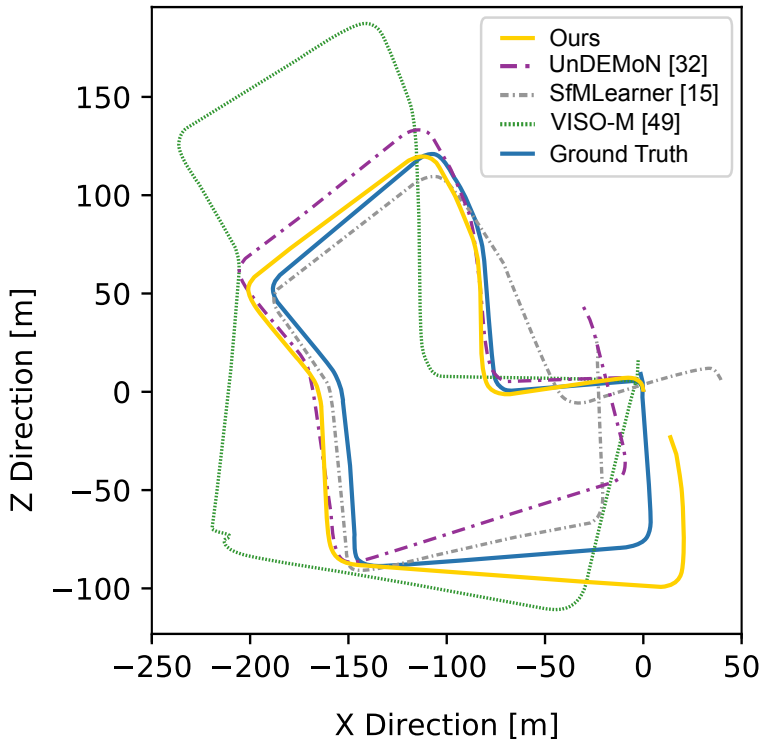

Figure A.5: **Trajectory Comparison.** Visual comparison of estimated egomotion between our method and several others ([MDM18, ZBS17, GZS11]) on Sequence 07 of the KITTI Odometry dataset. Corresponding t_{ate} and r_{ate} metrics can be found in Table A.3.

that the loss functions for these free parameters are tied together. This may explain the slight loss in accuracy, especially when compared to [MDM18], but the upside is that our method is a reduction in computational complexity as there are fewer weights in our architecture to optimize over.

A.5.4 Run-Time Evaluation

Our single-network architecture design via a common encoder decreases overall network complexity (and therewith the number of network parameters) which can decrease the necessary time to optimize weights. Previously, when using a 7-layer CNN instead of our common encoder for pose and intrinsics regression, network size was ∼30 million in trainable parameters; however, after replacing those layers with a common encoder, the number of trainable

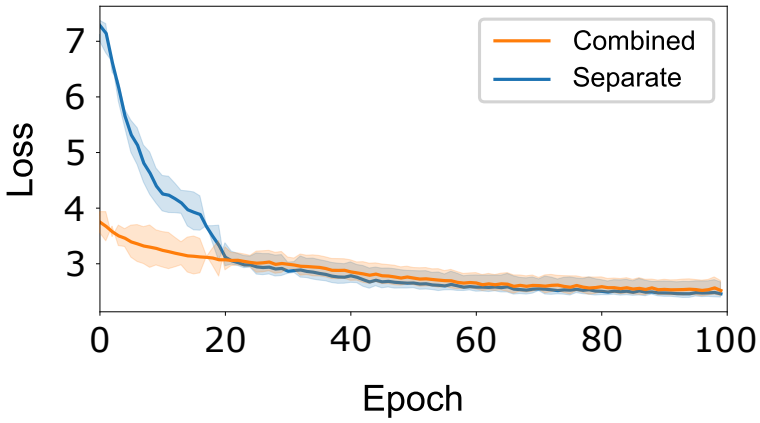

Figure A.6: **Loss Comparison.** Average training loss of the first 100 epochs for both architecture variants. The solid lines represent the mean across ten different runs, and the shaded areas represent one standard deviation (∼70% confidence). The "Separate" architecture used a 7-layer CNN for pose and intrinsics, while "Combined" used a common encoder.

parameters reduced to ∼27.6 million. This is around an 8% savings.

Fig. A.6 shows the effects of this decrease through a run-time comparison. In this figure, the mean (solid) and standard deviation (shaded) loss for each epoch is calculated across ten independent runs for each model. We observed a smaller initial average loss in the combined network, where in this case initialization of weights from the 7-layer CNN that may contribute to a higher loss before being optimal is not necessary. Over time though, we observed that these losses cross paths (roughly at 20 epochs), which is likely caused by the additional 2.4 million parameters for its function approximation. Thus, for those looking to maximize accuracy, a separated network may be better; however, for others who need reasonable results very quickly, a combined network can provide that in just a few epochs.

A.6 Discussion

In this work we have presented an unsupervised, single-network monocular depth inference approach for joint prediction of environmental depth, egomotion, and camera intrinsics. Through training our neural network to learn spatial and temporal constraints between

stereo and temporally-adjacent pairs, we are able to resolve solutions at metric scale using only monocular video at test time. We distinguish our work from other monocular inference approaches by creating a single, fully differentiable architecture for depth prediction and visual odometry. To further reduce human effort and manual intervention, we also take advantage of intrinsics observability in the system by learning the camera parameters embedded within the temporal reconstruction loss. We verify the success of our system using the KITTI dataset, where our results show we are able to achieve performance comparable to the state-of-the-art in monocular vision while solving for intrinsics and decreasing overhead and overall training complexity.

In future work we plan to quantify network robustness to initialization error during the training of camera parameters. We are also interested in comparing depth and odometry results between predetermined and learned intrinsics, to analyze their effects on prediction outcomes. To improve performance, we expect that the expansion of training to other datasets will provide a more diverse collection of scenes for evaluation, and learned intrinsics will allow for pooling of datasets as another avenue for training. Additionally addressing occlusion and moving objects will ensure added support for higher complexity scenes captured within these datasets.

REFERENCES

[AAB15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, and et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.", 2015.

[AOM21] Ali Agha, Kyohei Otsu, Benjamin Morrell, David D Fan, Rohan Thakker, Angel Santamaria-Navarro, Sung-Kyun Kim, et al. "Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge." *arXiv preprint arXiv:2103.11470*, 2021.

[BD06] T. Bailey and H. Durrant-Whyte. "Simultaneous localization and mapping (SLAM): part II." *IEEE Robotics and Automation Magazine*, **13**(3):108–117, 2006.

[BGA20] Amanda Bouman, Muhammad Fadhil Ginting, Nikhilesh Alatur, Matteo Palieri, David D Fan, Thomas Touma, Torkom Pailevanian, Sung-Kyun Kim, Kyohei Otsu, Joel Burdick, et al. "Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion." In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[Bha10] Nitin Bhatia. "Survey of Nearest Neighbor Techniques." *International Journal of Computer Science and Information Security*, 2010.

[BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." *IEEE Transactions on PAMI*, **39**(12):2481–2495, 2017.

[BLW19] Jia-Wang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. "Unsupervised Scale-consistent Depth and Ego-motion Learning from Monocular Video." *arXiv:1908.10553 [cs]*, October 2019.

[BM92] Paul J Besl and Neil D McKay. "Method for registration of 3-D shapes." In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pp. 586–606, 1992.

[BMT07] Grant Baldwin, Robert Mahony, Jochen Trumpf, Tarek Hamel, and Thibault Cheviron. "Complementary filter design on the Special Euclidean group SE (3)." In *2007 European Control Conference (ECC)*. IEEE, 2007.

[BR14] Jose Luis Blanco and Pranjal Kumar Rai. "nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees." https://github.com/jlblancoc/nanoflann, 2014.

[CCC16] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age." *IEEE Transactions on Robotics*, 2016.

[CCM22] Tsang-Kai Chang, Kenny Chen, and Ankur Mehta. "Resilient and Consistent Multirobot Cooperative Localization With Covariance Intersection." *IEEE Transactions on Robotics*, **38**(1):197–208, 2022.

[CLA22a] Kenny Chen, Brett T. Lopez, Ali-akbar Agha-mohammadi, and Ankur Mehta. "Direct LiDAR Odometry: Fast Localization With Dense Point Clouds." *IEEE Robotics and Automation Letters*, **7**(2):2000–2007, 2022.

[CLA22b] Kenny Chen, Brett T. Lopez, Ali-akbar Agha-mohammadi, and Ankur Mehta. "Direct LiDAR Odometry: Fast Localization With Dense Point Clouds." *IEEE Robotics and Automation Letters*, **7**(2):2000–2007, 2022.

[CM92] Yang Chen and Gérard Medioni. "Object modelling by registration of multiple range images." *Image and Vision Computing*, 1992.

[CNL23a] Kenny Chen, Ryan Nemiroff, and Brett T Lopez. "Direct LiDAR-Inertial Odometry and Mapping: Robust Localization and Connective Mapping." *arXiv*, 2023.

[CNL23b] Kenny Chen, Ryan Nemiroff, and Brett T Lopez. "Direct LiDAR-Inertial Odometry: Lightweight LIO with Continuous-Time Motion Correction." *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[CPL21] Kenny Chen, Alexandra Pogue, Brett T. Lopez, Ali-Akbar Agha-Mohammadi, and Ankur Mehta. "Unsupervised Monocular Depth Learning with Integrated Intrinsics and Spatio-Temporal Constraints." *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2451–2458, 2021.

[CWW] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. "VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem.".

[DB06] H. Durrant-Whyte and T. Bailey. "Simultaneous localization and mapping: part I." *IEEE Robotics & Automation Magazine*, **13**(2):99–110, 2006.

[DB18] David Droeschel and Sven Behnke. "Efficient Continuous-Time SLAM for 3D Lidar-Based Online Mapping." In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5000–5007, 2018.

[DBK21] Simon-Pierre Deschênes, Dominic Baril, Vladimír Kubelka, Philippe Giguere, and François Pomerleau. "Lidar Scan Registration Robust to Extreme Motions." In *Conference on Robots and Vision*, 2021.

[DDJ21] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. "Ct-icp: Real-time elastic lidar odometry with loop closure." *arXiv:2109.12979*, 2021.

[DDJ22] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. "CT-ICP: Real-time elastic LiDAR odometry with loop closure." In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 5580–5586. IEEE, 2022.

[DST00] Frank Dellaert, Steven M Seitz, Charles E Thorpe, and Sebastian Thrun. "Structure from motion without correspondence." In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 2, pp. 557–564. IEEE, 2000.

[ECP20] Kamak Ebadi, Yun Chang, Matteo Palieri, Alex Stephens, Alex Hatteland, Eric Heiden, Abhishek Thakur, Nobuhiro Funabiki, Benjamin Morrell, Sally Wood, et al. "LAMP: Large-scale autonomous mapping and positioning for exploration of perceptually-degraded subterranean environments." In *IEEE International Conference on Robotics and Automation*, 2020.

[EF] David Eigen and Rob Fergus. "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture.".

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network." In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pp. 2366–2374. Curran Associates, Inc., 2014.

[EPW21] Kamak Ebadi, Matteo Palieri, Sally Wood, Curtis Padgett, and Ali-akbar Aghamohammadi. "DARE-SLAM: Degeneracy-aware and resilient loop closing in perceptually-degraded environments." *Journal of Intelligent & Robotic Systems*, **102**:1–25, 2021.

[FCD16] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. "On-manifold preintegration for real-time visual–inertial odometry." *IEEE Transactions on Robotics*, 2016.

[FCS10] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. "Towards internet-scale multi-view stereo." In *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010.

[FGW18] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. "Deep ordinal regression network for monocular depth estimation." In *Proceedings of the IEEE CVPR*, pp. 2002–2011, 2018.

[FHW23] Fuzhang Han, Han Zheng, Wenjun Huang, Rong Xiong, Yue Wang, and Yanmei Jiao. "DAMS-LIO: A Degeneration-Aware and Modular Sensor-Fusion LiDAR-inertial Odometry.", 2023.

[GAB17] C. Godard, O. M. Aodha, and G. J. Brostow. "Unsupervised Monocular Depth Estimation with Left-Right Consistency." In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[GAP20] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. "3d packing for self-supervised monocular depth estimation." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2485–2494, 2020.

[GBC16] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. "Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue." *European Conference on Computer Vision*, 2016.

[GLJ19] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova. "Depth From Videos in the Wild: Unsupervised Monocular Depth Learning From Unknown Cameras." In *IEEE/CVF ICCV*, 2019.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite." In *2012 IEEE Conference on CVPR*, pp. 3354–3361. IEEE, 2012.

[GMF19] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. "Digging Into Self-Supervised Monocular Depth Estimation." *International Conference on Computer Vision*, August 2019.

[GR20] W. N. Greene and N. Roy. "Metrically-Scaled Monocular SLAM using Learned Scale Factors." In *2020 IEEE ICRA*, pp. 43–50, 2020.

[GRM05] Daniel Girardeau-Montaut, Michel Roux, Raphaël Marc, and Guillaume Thibault. "Change detection on point cloud data acquired with a ground laser scanner." *ISPRS*, 2005.

[Gru17] Michael Grupp. "evo: Python package for the evaluation of odometry and SLAM." https://github.com/MichaelGrupp/evo, 2017.

[GZS11] Andreas Geiger, Julius Ziegler, and Christoph Stiller. "Stereoscan: Dense 3d reconstruction in real-time." In *2011 IEEE intelligent vehicles symposium*, pp. 963–968. Ieee, 2011.

[HKJ13] P. Heise, S. Klose, B. Jensen, and A. Knoll. "PM-Huber: PatchMatch with Huber Regularization for Stereo Matching." In *2013 IEEE International Conference on Computer Vision*, pp. 2360–2367, 2013.

[HKR16] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. "Real-time loop closure in 2D LIDAR SLAM." In *IEEE International Conference on Robotics and Automation*, 2016.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[Jac12] Paul Jaccard. "The Distribution of the Flora in the Alpine Zone." *New Phytologist*, **11**(2):37–50, 1912.

[Kat03] Rolf Katzenbeisser. "About the calibration of lidar sensors." In *ISPRS Workshop*, pp. 1–6, 2003.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[KCK21] Giseop Kim, Sunwook Choi, and Ayoung Kim. "Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments." *IEEE Transactions on Robotics*, 2021. Accepted. To appear.

[KK18] Giseop Kim and Ayoung Kim. "Scan Context: Egocentric Spatial Descriptor for Place Recognition within 3D Point Cloud Map." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Oct. 2018.

[KPC20] Giseop Kim, Yeong Sang Park, Younghun Cho, Jinyong Jeong, and Ayoung Kim. "MulRan: Multimodal Range Dataset for Urban Place Recognition." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Paris, May 2020.

[KV91] Jan J Koenderink and Andrea J Van Doorn. "Affine structure from motion." *JOSA A*, **8**(2):377–385, 1991.

[KYO21a] Kenji Koide, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno. "Adaptive Hyperparameter Tuning for Black-box LiDAR Odometry." In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7708–7714, 2021.

[KYO21b] Kenji Koide, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno. "Automatic Hyper-Parameter Tuning for Black-box LiDAR Odometry." In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5069–5074, 2021.

[KYO21c] Kenji Koide, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno. "Voxelized gicp for fast and accurate 3d point cloud registration." In *IEEE International Conference on Robotics and Automation*, 2021.

[LH17] Brett T Lopez and Jonathan P How. "Aggressive collision avoidance with limited field-of-view sensing." In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[LKK23] Hyungtae Lim, Daebeom Kim, Beomsoo Kim, and Hyun Myung. "AdaLIO: Robust Adaptive LiDAR-Inertial Odometry in Degenerate Indoor Environments.", 2023.

[Lop23] Brett T. Lopez. "A contracting hierarchical observer for pose-inertial fusion." *arXiv:2303.02777*, 2023.

[LSL15] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. "Learning depth from single monocular images using deep convolutional neural fields." *IEEE Transactions on PAMI*, **38**(10):2024–2039, 2015.

[LWL18] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. "UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning." *2018 IEEE International Conference on Robotics and Automation (ICRA)*, February 2018.

[LY19] Chenxu Luo, Zhenheng Yang, et al. "Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding." *IEEE Transactions on PAMI*, **42**:2624–2641, 2019.

[LZ21] Zheng Liu and Fu Zhang. "Balm: Bundle adjustment for lidar mapping." *IEEE Robotics and Automation Letters*, **6**(2):3184–3191, 2021.

[MBG22] Matteo Marchi, Jonathan Bunton, Bahman Gharesifard, and Paulo Tabuada. "LiDAR Point Cloud Registration with Formal Guarantees." In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 3462–3467, 2022.

[MDM18] V Madhu Babu, Kaushik Das, Anima Majumdar, and Swagat Kumar. "UnDEMoN: Unsupervised Deep Network for Depth and Ego-Motion Estimation." In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1082–1088, October 2018. ISSN: 2153-0866.

[MG15] Moritz Menze and Andreas Geiger. "Object Scene Flow for Autonomous Vehicles." In *CVPR*, 2015.

[ML10] Naveed Muhammad and Simon Lacroix. "Calibration of a rotating multi-beam lidar." In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5648–5653. IEEE, 2010.

[NCL23] Ryan Nemiroff, Kenny Chen, and Brett T. Lopez. "Joint On-Manifold Gravity and Accelerometer Intrinsics Estimation.", 2023.

[Nel] Erik Nelson. "B(erkeley) L(ocalization) A(nd) M(apping).".

[NH10] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines." In *ICML*, 2010.

[NLD11] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. "DTAM: Dense tracking and mapping in real-time." In *2011 ICCV*. IEEE, 2011.

[NYC21] Thien-Minh Nguyen, Shenghai Yuan, Muqing Cao, Lyu Yang, Thien Hoang Nguyen, and Lihua Xie. "MILIOM: Tightly coupled multi-input lidar-inertia odometry and mapping." *IEEE Robotics and Automation Letters*, **6**(3):5573–5580, 2021.

[PAG19] Sudeep Pillai, Rareş Ambruş, and Adrien Gaidon. "Superdepth: Self-supervised, super-resolved monocular depth estimation." In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[PAT18] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. "Towards real-time unsupervised monocular depth estimation on CPU." *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, July 2018.

[PL17] Sudeep Pillai and John J. Leonard. "Towards Visual Ego-motion Learning in Robots.", 2017.

[PMK18] Chanoh Park, Peyman Moghadam, Soohwan Kim, Alberto Elfes, Clinton Fookes, and Sridha Sridharan. "Elastic LiDAR Fusion: Dense Map-Centric Continuous-Time SLAM." In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1206–1213, 2018.

[PMT20] Matteo Palieri, Benjamin Morrell, Abhishek Thakur, Kamak Ebadi, Jeremy Nash, Arghya Chatterjee, Christoforos Kanellakis, Luca Carlone, Cataldo Guaragnella, and Ali-akbar Agha-Mohammadi. "Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time." *IEEE Robotics and Automation Letters*, **6**(2), 2020.

[PMW22] Chanoh Park, Peyman Moghadam, Jason L. Williams, Soohwan Kim, Sridha Sridharan, and Clinton Fookes. "Elasticity Meets Continuous-Time: Map-Centric Dense 3D LiDAR SLAM." *IEEE Transactions on Robotics*, **38**(2):978–997, 2022.

[PXH21] Yue Pan, Pengchuan Xiao, Yujie He, Zhenlei Shao, and Zesong Li. "MULLS: Versatile LiDAR SLAM via multi-metric linear least square." In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11633–11640, 2021.

[RB19] Santiago Royo and Maria Ballesta-Garcia. "An overview of lidar imaging systems for autonomous vehicles." *Applied Sciences*, **9**, 2019.

[RC11] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)." In *IEEE International Conference on Robotics and Automation*, 2011.

[RKC22] Milad Ramezani, Kasra Khosoussi, Gavin Catt, Peyman Moghadam, Jason Williams, Paulo Borges, Fred Pauling, and Navinda Kottege. "Wildcat: Online Continuous-Time 3D Lidar-Inertial SLAM." *arXiv:2205.12595*, 2022.

[RPM22] Andrzej Reinke, Matteo Palieri, Benjamin Morrell, Yun Chang, Kamak Ebadi, Luca Carlone, and Ali-Akbar Agha-Mohammadi. "LOCUS 2.0: Robust and Computationally Efficient Lidar Odometry for Real-Time 3D Mapping." *IEEE Robotics and Automation Letters*, pp. 1–8, 2022.

[RSS20] Tobias Renzler, Michael Stolz, Markus Schratter, and Daniel Watzenig. "Increased accuracy for fast moving LiDARS: Correction of distorted point clouds." In *IEEE International Instrumentation and Measurement Technology Conference*, 2020.

[RWC20] Milad Ramezani, Yiduo Wang, Marco Camurri, David Wisth, Matias Mattamala, and Maurice Fallon. "The Newer College Dataset: Handheld LiDAR, Inertial and Vision with Ground Truth." In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4353–4360, 2020.

[SC86] Randall C Smith and Peter Cheeseman. "On the representation and estimation of spatial uncertainty." *The international journal of Robotics Research*, **5**(4):56–68, 1986.

[SCD06] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. "A comparison and evaluation of multi-view stereo reconstruction algorithms." In *IEEE CVPR*, volume 1, pp. 519–528, 2006.

[SCN06] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. "Learning depth from single monocular images." In *NeurIPS*, 2006.

[SCT20] Alexander Schperberg, Kenny Chen, Stephanie Tsuei, Michael Jewett, Joshua Hooks, Stefano Soatto, Ankur Mehta, and Dennis Hong. "Risk-Averse MPC via Visual-Inertial Input and Recurrent Networks for Online Collision Avoidance." In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5730–5737, 2020.

[SE18] Tixiao Shan and Brendan Englot. "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain." In *International Conference on Intelligent Robots and Systems*, 2018.

[SEE12a] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. "A Benchmark for the Evaluation of RGB-D SLAM Systems." In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[SEE12b] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. "A benchmark for the evaluation of RGB-D SLAM systems." In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580. IEEE, 2012.

[SEM20] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping." In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5135–5142, 2020.

[SER21a] Tixiao Shan, Brendan Englot, Carlo Ratti, and Rus Daniela. "LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping." In *IEEE International Conference on Robotics and Automation*, 2021.

[SER21b] Tixiao Shan, Brendan Englot, Carlo Ratti, and Daniela Rus. "Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping." In *IEEE International Conference on Robotics and Automation*, pp. 5692–5698, 2021.

[SF16] Johannes L Schonberger and Jan-Michael Frahm. "Structure-from-motion revisited." In *Proceedings of the IEEE CVPR*, pp. 4104–4113, 2016.

[SHT09] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. "Generalized-ICP." In *Robotics: Science and Systems (RSS)*, 2009.

[SSC19] Cordelia Schmid, Cristian Sminchisescu, and Yuhua Chen. "Self-supervised Learning with Geometric Constraints in Monocular Video - Connecting Flow, Depth, and Camera." In *ICCV*, 2019.

[SSN08] Ashutosh Saxena, Min Sun, and Andrew Y Ng. "Make3d: Learning 3d scene structure from a single still image." *IEEE Transactions on PAMI*, **31**(5):824–840, 2008.

[SZF16] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. "Pixelwise view selection for unstructured multi-view stereo." In *European Conference on Computer Vision*, pp. 501–518. Springer, 2016.

[TNN22] Turcan Tuna, Julian Nubert, Yoshua Nava, Shehryar Khattak, and Marco Hutter. "X-ICP: Localizability-Aware LiDAR Registration for Robust Localization in Extreme Environments." *arXiv preprint arXiv:2211.16335*, 2022.

[TTC20] Andrea Tagliabue, Jesus Tordesillas, Xiaoyi Cai, Angel Santamaria-Navarro, Jonathan P How, Luca Carlone, and Ali-akbar Agha-mohammadi. "LION: Lidar-Inertial observability-aware navigator for Vision-Denied environments." In *International Symposium on Experimental Robotics*, 2020.

[TTL17] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction." *arXiv:1704.03489 [cs]*, April 2017.

[Ume91] S. Umeyama. "Least-squares estimation of transformation parameters between two point patterns." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**(4):376–380, 1991.

[VGM23a] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. "KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way." *IEEE Robotics and Automation Letters (RA-L)*, **8**(2):1029–1036, 2023.

[VGM23b] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. "KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way." *IEEE Robotics and Automation Letters (RA-L)*, **8**(2):1–8, 2023.

[VR17] Sudheendra Vijayanarasimhan, Susanna Ricco, et al. "SfM-Net: Learning of Structure and Motion from Video.", 2017.

[VSO08] José Fernandes Vasconcelos, Carlos Silvestre, and P Oliveira. "A nonlinear GPS/IMU based observer for rigid body attitude and position estimation." In *2008 47th IEEE Conference on Decision and Control*, pp. 1255–1260. IEEE, 2008.

[WBS04] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. "Image Quality Assessment: From Error Visibility to Structural Similarity." *IEEE Transactions on Image Processing*, **13**(4), April 2004.

[WZS22] Zhong Wang, Lin Zhang, Ying Shen, and Yicong Zhou. "D-LIOM: Tightly-coupled Direct LiDAR-Inertial Odometry and Mapping." *IEEE Transactions on Multimedia*, 2022.

[XCH21] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. "FAST-LIO2: Fast Direct LiDAR-inertial Odometry." *arXiv preprint arXiv:2107.06829*, 2021.

[XCH22] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. "Fast-lio2: Fast direct lidar-inertial odometry." *IEEE Transactions on Robotics*, 2022.

[XW18] Dan Xu, Wei Wang, et al. "Structured attention guided convolutional neural fields for monocular depth estimation." In *Proceedings of the IEEE CVPR*, 2018.

[YCL19] Haoyang Ye, Yuying Chen, and Ming Liu. "Tightly coupled 3d lidar inertial odometry and mapping." In *International Conference on Robotics and Automation*, pp. 3144–3150, 2019.

[YS18] Zhichao Yin and Jianping Shi. "GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose." *arXiv:1803.02276 [cs]*, March 2018.

[YWW18] Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. "Lego: Learning edge with geometry all at once by watching videos." In *Proceedings of the IEEE CVPR*, pp. 225–234, 2018.

[YYC23] Pengyu Yin, Shenghai Yuan, Haozhi Cao, Xingyu Ji, Shuyang Zhang, and Lihua Xie. "Segregator: Global Point Cloud Registration with Semantic and Geometric Cues." *arXiv preprint arXiv:2301.07425*, 2023.

[ZBS17] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. "Unsupervised Learning of Depth and Ego-Motion from Video." In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6612–6619, Honolulu, HI, July 2017. IEEE.

[ZCF21] Lintong Zhang, Marco Camurri, and Maurice Fallon. "Multi-Camera LiDAR Inertial Extension to the Newer College Dataset." *arXiv:2112.08854*, 2021.

[ZGW18] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. "Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction." *Conference on Computer Vision and Pattern Recognition*, April 2018.

[ZKS16] Ji Zhang, Michael Kaess, and Sanjiv Singh. "On degeneracy of optimization-based state estimation problems." In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 809–816, 2016.

[ZLH18] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. "DF-Net: Unsupervised Joint Learning of Depth and Flow using Cross-Task Consistency.", 2018.

[ZS14] Ji Zhang and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time." In *Robotics: Science and Systems*, volume 2, pp. 1–9, 2014.

[ZSZ20] Chaoqiang Zhao, Qiyu Sun, Chongzhen Zhang, Yang Tang, and Feng Qian. "Monocular Depth Estimation Based On Deep Learning: An Overview." *Science China Technological Sciences*, **63**(9):1612–1627, September 2020. arXiv: 2003.06620.