UCLA
UCLA Electronic Theses and Dissertations

Title

Submodular Maximization for Geometric Robotic Perception

Permalink

https://escholarship.org/uc/item/6j63p18j

ISBN

9798173791160

Author

Thorne, David Paul

Publication Date

2026-11-10

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

University of California
Los Angeles

Submodular Maximization for Geometric Robotic Perception

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Aerospace Engineering

by

David Paul Thorne

2026

© Copyright by
David Paul Thorne
2026

Abstract of the Dissertation

Submodular Maximization for Geometric Robotic Perception

by

David Paul Thorne

Doctor of Philosophy in Aerospace Engineering

University of California, Los Angeles, 2026

Professor Brett T. Lopez, Chair

Modern geometric sensors collect an abundance of data used to generate detailed maps which inform decision-making algorithms in autonomous robots. However, processing high volumes of perception data online is not feasible given the limited computation and memory resources available to most robots. Instead, autonomy algorithms remain tractable by prioritizing smaller subsets of perception data via heuristic downsampling, which lacks generality and requires extensive, environment-specific tuning. Posing this prioritization instead as a combinatorial optimization problem quantifies the value of each observation, so size-constrained, informative subsets can be selected directly. This dissertation develops the theory and algorithms for solving such combinatorial problems online within provable suboptimality bounds. The central tool is submodular maximization, where set functions exhibiting diminishing returns admit polynomial-time algorithms with near-optimal guarantees. Three primary contributions formulate and optimize submodular functions to improve memory, computation, or task performance in LiDAR Simultaneous Localization And Mapping (SLAM), map distillation, and geometric change detection.

The first contribution addresses three such problems in LiDAR SLAM, optimizing keyframe selection and usage to reduce memory and computation demands. Two of the problems are solved as optimizations over a set of learned point cloud descriptors, which simplify otherwise

expensive point-wise overlap computations. The second contribution extends the use of descriptors to geometric map distillation, the problem of selecting informative submaps from all geometric data collected in a mapping session. A novel submodular function which quantifies map informativeness using descriptors is optimized with a dynamically reordered streaming algorithm to solve geometric map distillation online. The final contribution applies geometric map distillation to online geometric change detection. In this framework, representative submaps are selected for local scenes and used in a submap-to-submap change detection algorithm that is both accurate and computationally efficient. Each contribution is validated on field tests and open-source datasets, with an emphasis on long-duration missions through unstructured environments. Open-source implementations make the contributions widely available for testing and development.

The dissertation of David Paul Thorne is approved.

Anushri Dixit

Bahman Gharesifard

Robert T. M'Closkey

Sriram Narasimhan

Brett T. Lopez, Committee Chair

University of California, Los Angeles

2026

The Bravest—grope a little—
And sometimes hit a Tree
Directly in the Forehead—
But as they learn to see—

Either the Darkness alters—
Or something in the sight
Adjusts itself to Midnight—
And Life steps almost straight.

— Emily Dickinson

Table of Contents

1 Introduction **1**

1.1 Overview 1

1.2 Motivation and Research Statement 3

1.3 Related Work 4

1.3.1 Submodular Maximization 4

1.3.2 Online Robotic Mapping: Simultaneous Localization And Mapping (SLAM) 6

1.3.3 Geometric Change Detection 9

1.4 Contributions and Outline of Dissertation 10

1.4.1 Contribution 1: Submodular Optimization for Keyframe Selection and Usage in SLAM 10

1.4.2 Contribution 2: OptMap Geometric Map Distillation via Submodular Maximization 11

1.4.3 Contribution 3: Online Geometric Change Detection via Scene Decomposition 12

1.4.4 Dissertation Outline 12

2 Preliminaries **14**

2.1 Notation 14

2.2 Learned Point Cloud Descriptors 15

2.3 Submodular Maximization 17

2.3.1 Submodular Set Functions 17

2.3.2 The Greedy Algorithm 18

2.3.3 Streaming Submodular Maximization . . . 18
2.3.4 Incomplete Information . . . 20
2.3.5 Submodular Coverage and Clustering Functions . . . 21
2.3.6 Exemplar-Based Clustering . . . 22

3 Submodular Optimization for Keyframe Selection and Usage in SLAM 23

3.1 Overview . . . 24
3.1.1 Problem Formulation . . . 26
3.2 Keyframe Selection via Feature Descriptor . . . 27
3.3 Submap Generation via Submodular Maximization . . . 30
3.4 Map Summary via Streaming Submodular Optimization . . . 33
3.4.1 Algorithmic Implementation . . . 36
3.5 LiDAR Keyframing Results . . . 38
3.5.1 Online Keyframe Selection . . . 38
3.5.2 Submodular Submap Generation . . . 39
3.5.3 Map Summarization . . . 39
3.6 Conclusion . . . 40

4 OptMap: Geometric Map Distillation via Submodular Maximization . . 41

4.1 Overview . . . 42
4.1.1 Chapter-Specific Symbols . . . 45
4.2 Optimal Geometric Map Distillation . . . 46
4.2.1 Continuous Exemplar-Based Clustering . . . 47
4.3 Dynamically Reordered Streaming Submodular Maximization . . . 55
4.3.1 Streaming Submodular Dynamic Reordering . . . 56

4.3.2 Approximate Pairwise Information for CEBC . . . 59
4.4 Algorithmic Implementation . . . 64
4.5 OptMap Results . . . 64
4.5.1 Geometric Map Distillation Reward Function Evaluation . . . 65
4.5.2 Dynamic Reordering Effectiveness Ablation Study . . . 69
4.5.3 Real-Time Map Summarization Evaluation . . . 73
4.6 Discussion . . . 75

5 Online Geometric Change Detection via Scene Decomposition . . . 77
5.1 Overview . . . 77
5.1.1 Chapter-Specific Symbols . . . 80
5.2 Methods . . . 81
5.2.1 Scene Identification . . . 81
5.2.2 Submap Generation . . . 84
5.2.3 Change Detection using Submaps & Occlusion Filtering . . . 85
5.2.4 Change Manager . . . 89
5.3 Results . . . 90
5.3.1 Hardware Platform . . . 91
5.3.2 Submap Generation Cost: OptMap versus a Point-wise Pipeline . . . 93
5.3.3 Ablation Study . . . 95
5.3.4 LT-Mapper Parking Lot Dataset . . . 99
5.4 Conclusion . . . 100

6 Conclusion . . . 101
6.1 Summary of Contributions . . . 101

6.2 Limitations and Future Work 103
6.3 Concluding Remarks 105

References **106**

List of Figures

2.1 Global descriptor generation. The point cloud is projected into a range image (gray) and fed into the local features extraction module (orange) which consists of convolutional neural networks and a multi-head self-attention block. The NetVLAD module (blue) pools all the local features to generate the global descriptor. 15

3.1 (Left) Dense point cloud map of 2.3 km forest loop with smaller summary maps. The dense map was generated during online SLAM and uses 456 keyframes as opposed to the summary maps which use 300 and 75 keyframes and are built in under one second. (Top-right) Modified Clearpath Robotics Warthog robot with LiDAR sensor used for collecting the forest loop dataset. (Bottom-right) Unique keyframe selection, each keyframe (axis) is chosen to capture a unique part of the map. 25

3.2 Map summaries for 6.2 km Mout Water Dataset from Graces Quarters. Left figure shows the full keyframe map using 527 keyframes and 186 MB. Right shows the summary map using 300 keyframes and 110 MB. Insets show detailed comparison of the maps. 37

3.3 Submap generation comparison using 6.2 km Mout Water dataset using identical keyframe selection strategies. Top shows keyframes used per submap (DLIOM avg. 10.4 Ours avg. 2.3). Middle shows the minimum Hessian eigenvalue (DLIOM avg. 377 Ours avg. 407). Bottom shows per scan computation time (DLIOM avg. 67.3 ms Ours avg. 45.4 ms). 38

4.1 OptMap is a geometric map distillation algorithm which provides size-constrained, informative maps for downstream use within autonomy systems. 42

4.2 The OptMap pipeline consists of four stages: (i) LiDAR mapping (ii) descriptor set generation, (iii) dynamically reordered streaming submodular maximization, and (iv) map loading. Point clouds from LiDARs are used to generate descriptors which are then saved to computer storage. Redundant descriptors are then filtered to improve evaluation time. The dynamically reordered streaming submodular maximization uses an online approximate value of each element to address input order bias and select informative solutions. 43

4.3 Section of a full and reduced set generated by Algorithm 6. All elements in the full set (blue outlines) are no further than $\mathcal{E}$ apart, and all elements of the reduced set (orange interior) are no further than $2\mathcal{E}$ apart. The weight assigned to r_2 is the sum of weights from its source set $\hat{w}_2 = w_2 + w_3 + w_4$. The closest element of S to e_2 and r_2 does not have to be the same element of the solution S. 50

4.4 Sensor selection with a four sensor limit. Solution set in blue, other sensors in orange. From left to right: optimal solution, greedy solution which does not consider input order, streaming solution with arbitrary input order (indicated by numbers in sensor footprints), and the streaming solution with dynamic reordering (DR) which modifies the input order according to approximate element values. Sensors 3 and 4 selected in the streaming solution due to input order bias. Assumed that OPT is known *a priori* and only one solution is maintained by the streaming variants. 55

4.5 Mutual information with respect to (4.8) as percent overlap between spherical caps. Overlap percent is obtained as a function of the distance between descriptors using a dense Monte Carlo integration over the surface of $\mathbb{S}^n$, with a fourth-order polynomial function $\psi(x)$. 62

4.6 Normalized percent overlap of distilled maps for reward functions when maximized via the greedy algorithm, higher indicates better performance. Sculpture Garden is a custom dataset in a semi-forested environment, and Parkland Mount is from [1]. CEBC (Ours) is evaluated using reduced input sets constructed with admission thresholds of $\mathcal{E} = 0.05, 0.25$, and 0.5. 66

4.7 Descriptor distance between sequential descriptors in Sculpture Garden (top) and Newer College Dataset [1] Parkland Mount (bottom) datasets. Outliers is the number of sequential distances > 0.25 and total distance is the sum of all sequential distances. The Euclidean distance to the first descriptor is shown in red as a reference. 67

4.8 Average function evaluation time and input set size for Sculpture Garden dataset. Ours is CEBC. Evaluation time decreases with input set size. Multiple reduced set distance thresholds $\mathcal{E}$ are used to show that computation time approaches EBC as the reduced and full sets converge. 68

4.9 Percent of overlap from dense map to output map for ablation test on KITTI 05 [2], Courtyard, and Lab datasets shown in Fig. 4.10. Results are normalized by the highest scoring method for each dataset. KITTI 05 output maps made with fifty scans, Courtyard maps made with twelve scans, and Lab maps made with eight scans. 69

4.10 Dense maps used for Fig. 4.9. Each map is made from 250 scans from a 32-beam Ouster OS0, with the sensor trajectory indicated by the yellow path. Left is Courtyard which entails fifteen loops of an outdoor courtyard followed by a single mapping pass through tight corridors and an adjacent alley in the figure top. Right is Lab which contains multiple mapping passes capturing the interior of a large lab space and a single pass capturing the outdoor alley and nearby tight hallways. 69

4.11 Sample output maps using different methods from Fig. 4.9. Methods from left to right are OptMap with descriptor-based dynamic reordering, OptMap with pose-based dynamic reordering, OptMap with both descriptor and pose dynamic reordering, Sieve-Streaming, random selection, and the evenly spaced heuristic. Dynamic reordering methods capture the entire map where the other three methods only select scans from the loop seen at the bottom of each map. 70

4.12 The pose-based dynamic reordering is unable to accurately select the highest value elements in cases where nearby scans capture significantly different environments. These two maps from the Lab dataset (Fig. 4.10) were generated using the same pose-based approximation dynamic reordering method, and demonstrate the large standard deviation from the pose-based dynamic reordering as scans are arbitrarily selected from one side of the wall (yellow ovals) leading to occasional suboptimal results. 70

4.13 Output OptMap maps for Semantic KITTI 02 with 100, 250, and 500 scans. Color indicates z-coordinate of point. OptMap is capable of generating map summaries with flexible size and resolution for autonomous planning at various scales. 74

5.1 A car door is opened between the first and second mapping passes of the same session and detected in under one minute. Base points are colored by height (z-axis); change points are shown in blue in the right figure. 78

5.2 Scenes spatially decompose the trajectory rather than the geometric map into convex hulls with minimal overlap. Each labeled convex shape over the yellow trajectory curve represents a unique scene where change detection occurs separately. The scene outlined in dashed white represents the current scene which is still being built. 82

5.3 Before the change detection algorithm occurs, OptMap is called to generate $\mathcal{C}_j^{1:v-1}$ and $\mathcal{R}_j^v$. During change detection five sequential steps occur: 1) ground plane removal (segmentation), 2) nearest-neighbor based change detection, 3) noise filtering, 4) occlusion filtering, and 5) ground plane filter after changes are clustered into candidate objects. Detected changes are clustered and managed as objects with respective change clouds, object IDs, average coordinates, and a positive/negative flag. 87

5.4 Occlusion filter on a submap-to-submap change detection instance. Top three images show submap A, B, and the set of candidate change points that are in submap B but not A. Bottom-left shows range image projection and subtraction for submap A and candidate change points at each scan pose in submap A. If a point is confirmed by any range image subtraction step it is added to the set of confirmed changes shown in the bottom-right. 88

5.5 Left: Graces Quarters, MD testing site. Right: Modified Clearpath Warthog with Ouster OS1 128-beam LiDAR used for experiments. 91

5.6 Figure Eight dataset, 3.9 km over 25.4 minutes. Boxes shown in white are requested summary map areas bounded by $x_{min} = -20 - u$, $x_{max} = 20 + u$, $y_{min} = -20(1 + u)$, and $y_{max} = 20$ for $u = \{0, 10, 20\}$ with the summary map generated for $u = 20$ by OptMap shown in the blue subfigure. 92

5.7 Ground truth changes for the Short change detection dataset. Top-left shows an overview of the map and views of each of the four changes. View 1 shows a backpack ($\sim$ 20 cm) which was introduced for view 1a and removed for view 1b. View 2 shows a car with a door opened between observations, and view 3 shows a person who appeared between observations. 96

5.8 Change detection results from LT-Mapper Parking Lot dataset session 02 (Jan 25, 2021) to session 04 (Jan 26, 2021). Left figure shows the changes from CDSD, and right figure shows the changes detected by ELite [3]. Positive change points are colored blue, negative change points are colored red, and the background map is colored by z-axis. The inset on the top-left of both figures shows a detailed difference in detection outputs for either algorithm. . 96

List of Tables

3.1 End-to-end accuracy of the proposed method compared to open-source LiDAR SLAM algorithms. Tests (A)-(C) are from the UCLA campus (Royce Hall, Sculpture Garden, and Court of Sciences). 35

4.1 OptMap computation time results. Left column gives the dataset, where sequences from three standardized datasets were chosen alongside one custom dataset collected on a 128-beam Ouster OS1 LiDAR (Forest Loop pictured in Fig. 3.1). Distilled map size indicates how many scans were used to generate the map, optimization time is the time from function call to when the solution set is selected, and load time is the remaining time spent loading point clouds from storage to computer memory. Method (1) uses the initial solution bounds of Section 3.4.1 and method (2) does not. 73

5.1 Change-detection ablation (mean of 5 runs per variant), reported as the average across the single-session Short (4 changes) and Long (18 changes) datasets. A ground-truth object counts as detected when a change cluster of the same polarity falls within 1 m of it. GT detection percentages are cumulative: *Whole* counts objects with $> 75\%$ of the GT object's points, *Partial* counts those at $> 50\%$. Spurious detections are split into small and large at 125 points. *Dropped* is the mean number of change-detection instances discarded because of a full queue. **Bold** marks the best value, underline marks the second best value, and red marks the worst value in each column. Middle group: alternative detection frameworks. Bottom group: ablations of CDSD. 90

5.2 Sensitivity to the change detection distance gate τ. All other parameters at their default values. The $\tau = 0.30$ row is the default operating point. Results averaged over five runs on the Short and Long datasets. 91

5.3 Sensitivity to the maximum scene volume α. Scenes were restricted to 2D for these experiments, so α is reported as an area. All other parameters at their default values. The $\alpha = 120$ row is the default operating point. Results averaged over five runs on the Short and Long datasets. Submap size increases proportionally with α. 91

5.4 Map loading and processing time for generating summary submap for a subset of long-duration mapping mission. Load, Box filter, and Voxel filter (LBV) method implemented using open-source libraries, with the box filter and voxel filter times listed separately. OptMap performs streaming optimization over descriptors then map loading. 93

5.5 Corroboration of CDSD's online change detections by ELite [3] on the multi-session parking lot dataset [4]. Total agreement is the percentage of all CDSD change points that have an ELite change point of the same polarity within 0.5 m. The earlier session of each pair is used as the base map. 99

List of Symbols

Symbol	Meaning
P_i	i'th point cloud (scan), $P_i = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$, $\mathbf{p}_j \in \mathbb{R}^3$.
$\mathcal{P}$	Scans collected in a mapping session, $\mathcal{P} = \{P_1, \ldots, P_{\lvert\mathcal{P}\rvert}\}$.
$\mathfrak{P}$	Set of all possible point clouds.
$\mathbf{x}_i$	Pose of the i'th scan, $\mathbf{x}_i \in SE(3)$.
$\mathbf{0}$	Null element, $\mathbf{0} = [0, 0, \ldots, 0] \in \mathbb{R}^{n+1}$.
$\mathbb{S}^n$, $\mathcal{A}(\mathbb{S}^n)$	n-hypersphere and its surface area.
E	Ground / Input set.
S	Solution set, $S \subseteq E$.
k	Cardinality constraint.
OPT	Optimal value, $f(S^*) = OPT$.
$d(a, B)$	Euclidean distance to a set, $d(a, B) = \min_{b \in B} \lvert\lvert a - b \rvert\rvert_2$, $a, b \in \mathbb{R}^{n+1}$.
$f(a\lvert B)$	Marginal value, $f(a\lvert B) = f(B \cup a) - f(B)$.
$\phi(\cdot)$	Learned descriptor function, $\phi : \mathfrak{P} \to \mathbb{S}^n$.

Acknowledgments

Chapter 2 is a version of D. Thorne, N. Chan, Y. Ma, C. S. Robison, P. Osteen, and B. T. Lopez, "Submodular Optimization for Keyframe Selection & Usage in SLAM," *IEEE International Conference on Robotics and Automation (ICRA)*, 2025. N. Chan helped develop the code implementation for the map summarization contribution, Y. Ma developed the code for LiDAR descriptor generation, C. S. Robison and P. Osteen helped provide testing datasets for the contributions, and B. T. Lopez provided advising. Chapter 3 is a version of D. Thorne, N. Chan, C. S. Robison, P. Osteen, and B. T. Lopez, "OptMap: Geometric Map Distillation via Submodular Maximization," under review for *IEEE Transactions on Robotics (T-RO)*. N. Chan helped develop and optimize the featured streaming submodular algorithm, C. S. Robison and P. Osteen helped provide testing datasets, and B. T. Lopez provided advising. Chapter 4 is a version of D. Thorne, S. J. C. Chua, N. Joshi, A. Wong, C. S. Robison, P. Osteen, and B. T. Lopez, "Online Geometric Change Detection via Scene Decomposition," under review for *IEEE Robotics and Automation Letters (RA-L)*. S. J. C. Chua helped develop prototype algorithms and performed a literature review, N. Joshi helped develop the occlusion checking method, A. Wong helped develop map maintenance tools, C. S. Robison and P. Osteen helped provide testing datasets and algorithm validation, and B. T. Lopez provided advising.

First, I would like to thank the members of my committee, Brett Lopez, Anushri Dixit, Bahman Gharesifard, Robert T. M'Closkey, and Sriram Narasimhan. My development as a researcher has been honed by your feedback and our conversations over the years. I learned more in half-hour conversations with you than I could have reading papers and textbooks for a week, and I hope that I absorbed some amount of your collective wisdom and insight.

To my advisor Brett Lopez, your role in this process is impossible to quantify and greatly appreciated. I began graduate school with no research experience and very little knowledge in algorithms or robotics. Despite this, you welcomed me as one of the first members of your newly formed lab. You encouraged me to pursue a doctoral degree and then to follow

my own passion projects on multi-robot coordination. When that initial effort was rebuffed, you were not mad but instead turned my attention towards a new direction in LiDAR perception. Your criticism pushed me to become a better communicator and researcher, and your reassurance gave me the confidence to read harsh reviews and submit ambitious works.

To my labmates, colleagues, and mentors at UCLA, the most valuable part of the past five years has been simple conversations with each of you. To Kenny Chen, I did not know that listening to your presentations on Direct LiDAR Odometry and its successors would be so fruitful, but this dissertation stands on the shoulders of your work and guidance. To Alessandro Bullitta, David Hill, Hanbyeol Yoon, and Laqshya Taneja, your companionship while studying for the preliminary exam was crucial for my sanity, and I am excited to follow each of your careers. To my labmates Yanlong Ma, Helene Levy, and Samuel Gessow, we have each experienced so much together, and my research would not have been possible without our frequent conversations and support. To the rest of my current and former labmates, Aaron Sabu, Alex Zhen, Cat Nystrom, Ryan Nemiroff, Sarah Enayati, Bill Wu, James Tseng, Eden Zafran, McKay McFadden, Jacob Sayono, Matthew Guemmer, Aleksandra Dudek, Nathan Chan, Aiden Wong, Nakul Joshi, Krithi Koodi, Ryu Adams, Samuel Jia Cong Chua, and Stanley Wei, everyone I have worked with is brilliant with bright futures ahead, and I am incredibly lucky to have learned something from each of you.

To my former supervisors and colleagues at NASA JPL, my year spent as an academic part-time intern both fulfilled a lifelong dream and shaped my understanding of the challenges we face in extreme autonomous robotics. To my supervisors Evan Clark and Sawyer Brooks, you gave me multiple chances that I am not sure I deserved at the time, despite funding squeezes and technical challenges. Many talented engineers and scientists worked on IceNode and CADRE with me, many of whom cannot be named, but Xavier Zapien, Federico Rossi, Paul Glick, Jacob Kosberg, Jean-Pierre De La Croix, and Amir Rahmani in particular challenged me to deliver excellent work and grow as an engineer.

To the Army Research Laboratory (ARL) members Philip Osteen, Christa Robison, Cole Barbes, and Brian Kaukeinen, our collaboration was the most rewarding part of my doctoral

journey and my regular visits to Graces Quarters were an invaluable grounding exercise in deploying my algorithms on very real robots. Similarly, to members of the Autonomy & Intelligence Laboratory at Northeastern University including Michael Everett, Zihao Dong, and Leonard Jung, working alongside you to develop our research projects together has been exciting and extremely valuable to my journey through graduate school.

I must acknowledge the support of my parents Chris Thorne and Amy Mortensen, my brother Christopher Bass Thorne, and my grandmother Cherie Seiver. Each of you taught me the value of education from a very young age, and I seemingly took that advice to heart! The values I learned from you are more valuable than any research I could ever publish. Particularly for Mom and Dad, as a kid I took for granted how much you had to sacrifice for me. If you were ever on a difficult overnight shift or patrol in who-knows-where thinking that you were there for me, I hope you know that I have begun to understand and appreciate just what that means. Needless to say, I would not be here without your efforts.

Finally, to my wife and best friend Sarah Enayati, no amount of acknowledgment is enough for you. You deserve credit for this dissertation more than anyone else, and I could write a hundred more pages just to tell you how grateful I am for you.

Vita

2021	Received B.S. in Aerospace Engineering from UCLA
2021–2023	Part-time academic mechanical engineering intern NASA Jet Propulsion Laboratory
2024	Received M.S. in Aerospace Engineering from UCLA
2021–2026	Graduate Researcher Mechanical and Aerospace Engineering Department, UCLA

Publications

David Thorne, Samuel Jia Cong Chua, Nakul Joshi, Aiden Wong, Christa S. Robison, Philip Osteen, Brett T. Lopez, "Online Geometric Change Detection via Scene Decomposition," *arXiv preprint arXiv:2609.17302*, 2026. (Under Review for IEEE Robotics and Automation Letters, RA-L)

David Thorne, Nathan Chan, Christa S. Robison, Philip Osteen, Brett T. Lopez, "OptMap: Geometric Map Distillation via Submodular Maximization," *arXiv preprint arXiv:2512.07775*, 2025. (Under Review for IEEE Transactions on Robotics, T-RO)

Samuel G. Gessow, **David Thorne**, Brett T. Lopez, "Information Control Barrier Functions: Preventing Localization Failure in Mobile Systems Through Control," *IEEE Control Systems Letters*, 2025.

David Thorne, Nathan Chan, Yanlong Ma, Christa S. Robison, Philip Osteen, Brett T. Lopez, "Submodular Optimization for Keyframe Selection & Usage in SLAM," *IEEE International Conference on Robotics and Automation (ICRA)*, Atlanta GA, 2025.

Zihao Dong, Jeff Pflueger, Leonard Jung, **David Thorne**, Philip Osteen, Christa S. Robison, Brett T. Lopez, Michael Everett, "LiDAR Inertial Odometry and Mapping Using Learned Registration-Relevant Features," *IEEE International Conference on Robotics and Automation (ICRA)*, Atlanta GA, 2025.

David Thorne and Brett T. Lopez, "Multi-Robot Task Assignment and Path Finding for Time-Sensitive Missions with Online Task Generation," *arXiv preprint arXiv:2310.06153*, 2024.

CHAPTER 1

Introduction

1.1 Overview

LiDAR sensors can be used to build highly detailed maps which, when used to the full extent, enable precise situational awareness and optimized decision making in autonomous mobile robots. Modern sensors, however, collect upwards of three gigabytes of data per minute, meaning the stream of raw data must be managed for downstream autonomy algorithms to remain tractable. This front-end role is usually filled by Simultaneous Localization And Mapping (SLAM) algorithms which remain computationally efficient by using hand-tuned heuristics. These heuristics lack generality and require extensive, environment-specific tuning to balance computation and memory demands against algorithmic performance. Further, each downstream algorithm places *unique requirements* on the map it needs to optimize performance and processing time, which means no single global map is suitable. The goal of this dissertation is to develop the theory and algorithms for delivering application-specific submaps from modern robotic perception systems. This goal is achieved throughout by formulating and solving combinatorial optimization problems, where subsets of all collected perception data are selected to maximize computational performance, informativeness, or some other application-specific objective. This paradigm is demonstrated with applications to LiDAR SLAM, map distillation, and online geometric change detection.

Finding optimal solutions to combinatorial problems is NP-hard, but many relevant objectives can be formulated as submodular functions, i.e., combinatorial functions which ex-

hibit diminishing returns. The advantage of this type of formulation is that submodular functions can be optimized in polynomial time with provably tight suboptimality guarantees [5,6]. Two famous algorithms for optimizing submodular functions are the greedy algorithm [6] and streaming submodular algorithms [7, 8]. These algorithms have since been used in numerous robotics [9–14] and computational applications [15–19]. A theme throughout all submodular maximization is the ability to quantify mutual information between the optimization variables. For example, submodular algorithms for multi-robot exploration [14] model the amount of overlapping information collected by two agents exploring similar locations. Similarly, visual odometry algorithms [11] quantify the value added by a feature for pose estimation with respect to a previous set of tracked features. Metrics developed for valuing a single point cloud collected by depth sensors (e.g., LiDAR or RGB-D) also lend themselves to measuring the mutual information between separate clouds, including point-to-point overlap [20] and a point's contribution to constraining scan registration [21].

Because most LiDAR scans contain tens of thousands of points, quantifying mutual information with point-wise operations is prohibitively expensive for applications which require online optimization. This challenge was initially recognized in regard to the place recognition problem, where localizing a point cloud in a map with no prior was only possible given sufficient offline computation time [22]. Early attempts at finding scan similarity without point-wise operations yielded histogram point cloud descriptors [23,24]. More recently, neural networks have been trained to encode point cloud similarity directly, with one approach producing descriptors whose distance decreases monotonically with the overlap between the input clouds [25–27]. Newer point cloud descriptors generate feature vectors for patches or clusters of points [28,29], with each new generation of descriptors proving to be more robust and accurate. Recent improvements in point cloud descriptors mean they could be deployed as a general metric for approximating scan similarity, particularly within the combinatorial problems that this dissertation proposes.

This dissertation improves the efficiency and flexibility of geometric perception systems by formulating and solving submodular maximization problems, both where heuristics are currently used and where no formulation has been posed before. First, keyframe selection and usage — a key driver of memory and computation time demands for LiDAR SLAM — are formulated and solved as submodular maximization problems. Second, geometric map distillation is proposed as a general problem format for optimal subset selection in LiDAR mapping, and a novel submodular function and algorithm called OptMap solves it online. Finally, OptMap is used as a core component of an online geometric change detection algorithm called Change Detection via Scene Decomposition (CDSD). All contributions are demonstrated on large-scale field tests, and open-source implementations of OptMap and CDSD are released.

1.2 Motivation and Research Statement

As robots are increasingly deployed to large dynamic environments, perception systems are tasked with processing larger quantities of data to deliver accurate localization, mapping, object segmentation and tracking, place recognition, and more. While modern geometric sensors generate highly detailed representations of the environment which could accomplish each task individually, increasing computational burdens mean modern algorithms rely on wasteful and inaccurate heuristics to prune sensor data. The result is often wasted resources or task performance as heuristics discard meaningful or useful observations. This dissertation instead formulates and optimizes task-specific feature selection and general geometric map distillation as a submodular optimization problem with formal optimality bounds. Thus, this dissertation both shows that online subset optimization in geometric perception is computationally feasible and improves key performance metrics for a range of common applications across autonomous robots.

1.3 Related Work

1.3.1 Submodular Maximization

Submodular maximization is a powerful tool for combinatorial optimization problems which exhibit monotonicity and diminishing returns because it provides efficient solutions to NP-hard problems with tight suboptimality guarantees. The most important result in the field of submodular maximization is the proven performance of the greedy algorithm when maximizing a non-negative monotone submodular function with a cardinality constraint, which achieves a $1 - \frac{1}{e} \approx 0.63$ suboptimality bound. This bound is optimal for any algorithm that queries the oracle function (i.e., the submodular function) a polynomial number of times [5,6], and results have been extended to general matroid constraints [30]. Submodular maximization also admits constraints beyond cardinality, including knapsack, set coverage, partition, and general matroid constraints, where a matroid is an abstraction of the linear independence of the columns of a matrix [31]. The computation and memory complexity of the greedy algorithm has been improved [32–34], with extensions proposed for non-monotone functions [35,36].

Recent work in submodular theory includes the definition of approximate submodularity [37], higher-order submodularity [38], and submodular maximization with limited information [39]. Advances in submodular theory are frequently applied to computing and machine learning where common tasks include dataset summarization, image classification, and search result generation [15–17]. Additional algorithmic applications of higher-order submodularity and optimization with limited information include multiple distributed submodular maximization algorithms [10,14,40]. Distributed submodular algorithms often rely on the independence of 'distant' elements in distributed decision making, the central idea being that high marginal value elements which share little to no mutual information can be added at the same time.

1.3.1.1 Streaming Submodular Algorithms

Many applications of submodularity in computing such as search result generation and document summarization require parsing extremely large datasets which cannot be stored in computer memory. This is particularly challenging for greedy algorithms which make k passes over the dataset to build a solution of size k, requiring the dataset to be frequently exchanged between computer storage and memory. In response, streaming submodular algorithms were concurrently proposed by [7] and [41]. The former proposed Sieve-Streaming, which makes a single pass over a dataset while generating a solution with a $(\frac{1}{2} - \epsilon)$ suboptimality ratio, where ϵ is a granularity parameter.

This work was improved multiple times; [8] expanded Sieve-Streaming to p-matchoid constraints formed by the union of multiple matroids, and proposed a solution for non-monotone functions. Feldman et al. [42] proposed a stochastic subsampling approach to reduce memory usage and computational complexity with a suboptimality guarantee similar to Sieve-Streaming in expectation. Kazemi et al. [43] and [44] made further memory and computation improvements, the latter via post-processing functions which operate on an inflated solution. Finally, [45] identified structural bias in streaming submodular algorithms and proposed a constraint which enforces the solution must utilize a set number of elements from identified sets. In the original streaming submodular work [7], it was noted that Sieve-Streaming is more likely to select elements seen earlier, and the solution proposed involved simply randomizing the stream order.

Similar to streaming submodularity is the recent class of dynamic submodular algorithms which maintain a solution throughout continuous ground set insertions and deletions [46–48]. These algorithms are not immediately applicable to LiDAR mapping as they contain the same bias as streaming algorithms (and therefore cannot simultaneously maintain a solution with incoming scans), and streaming submodular algorithms present superior memory and

computation performance when the ground set is fully available.

1.3.1.2 Submodular Maximization in Robotics

Submodular maximization has numerous applications outside of mathematics and computing, including a significant place in robotic planning algorithms. Graph theory often intersects with combinatorial optimization where greedy algorithms motivated by submodularity have been used to great effect [18, 19, 49]. It is no surprise then that the numerous robotic planning problems posed as graph-based optimizations also have numerous solutions applying submodularity including pose graph SLAM [50–52], path planning [49, 53], and area search and coverage [54, 55]. Multi-robot coordination has also received significant interest including the previously mentioned distributed submodular algorithms [10, 14, 40], and viewpoint selection [9, 13]. Distributed multi-robot exploration algorithms assume each robot must select one action from a larger set of possible actions, which is posed as a partition constraint in submodular maximization. In [10] and [40], the authors use higher-order monotonicity and the independence of multiple action sets to develop efficient algorithms for deployment on robot teams. Submodularity is less frequently used in robotic perception applications, with notable exceptions coming from previously mentioned pose graph SLAM works [50–52] which use the Fisher Information Matrix (FIM) of the pose graph to quantify the expected value of loop closures. The most relevant work for this dissertation applied submodularity to visual odometry [11], which used a greedy approach to select visual features that are both informative and persistent across multiple frames.

1.3.2 Online Robotic Mapping: Simultaneous Localization And Mapping (SLAM)

Simultaneous Localization And Mapping (SLAM) is an enabling technology for autonomous robots operating in previously unknown and unmapped environments. Many high-performance

and open-source algorithms have been proposed [20, 56, 57], and several active research directions are captured by various surveys [58, 59]. LiDAR SLAM algorithms rely on point cloud registration algorithms such as GICP [60] to align incoming scans to a global map or local submap composed of scans with known poses from earlier in the trajectory. The ability to register scans against a globally aligned submap significantly reduces drift, but requires saving reference scans called keyframes which results in linear memory growth with distance traveled. Further, selecting submaps for registration requires a trade-off between alignment accuracy and computation time. Keyframe-based methods are conveniently posed as graph-based SLAM problems for accurate loop closure [61] using minimal memory and computation resources [50, 52].

LiDAR was the sensor of choice for localization and mapping in the DARPA Subterranean Challenge where numerous teams developed novel SLAM algorithms, mapping strategies, and integrations between perception and planning [62–65]. Several branches of LiDAR SLAM research were inspired by the DARPA Subterranean Challenge, including robustness to degeneracy [66, 67], map fusion and cooperative SLAM [68, 69], and long-duration map management through change detection [4, 70].

1.3.2.1 Geometric Maps for Mobile Robots

Key performance metrics for geometric maps include the level of detail and coverage area, the complexity of applying map updates (e.g., loop closures), and the level of flexibility or specificity a map has towards serving multiple functions [71]. For LiDAR mapping, direct geometric maps are generated via the union of point clouds as complete, dense maps which use all collected scans [57, 72] or as lightweight keyframe maps which contain a subset of selected scans [20, 56, 73]. Dense point clouds are saved as octrees or kd-trees for nearest neighbor searches with a sublinear complexity of $O(\log N)$ where N is the number of points in the clouds. Keyframe maps are easily maintained over long-duration missions as they

require less memory and can be updated via pose-graph optimization, but complete maps provide a higher degree of detail. 3D occupancy grids are the most commonly used map type for autonomous planning algorithms [74]. Their primary advantages are that they have minimal memory requirements, are easily searched, explicitly model free-space, and are easily generated from multiple fused sensors [75–78]. The downside is that they require computationally expensive ray tracing to generate, and can be slow to update in batches (e.g., after a loop closure). Hierarchical graph-based maps can be memory efficient and carry significant detail [79], but are often designed by experts for specific applications and therefore are not well-suited to multiple diverse algorithms. Lightweight hierarchical maps can be shared over bandwidth constrained networks for a minimalist form of cooperative mapping [80, 81], but these maps share even less scene context. A trend for recent online mapping algorithms is to reduce the memory burden of long-duration missions [82, 83]. This continued optimization means future maps generated from SLAM may become more sparse and thus less useful for downstream autonomy algorithms.

1.3.2.2 Learned Point Cloud Global Descriptors

LiDAR descriptors compress dense point clouds into compact vectors where the distance between vectors encodes the similarity between point clouds. While a few model-free LiDAR descriptors exist (e.g., [23, 24]), most work in this field has been towards developing neural networks to generate descriptors for global place recognition. Notable recent advances in learned point cloud descriptors include the use of transformers to generate yaw-invariant descriptors [25, 84, 85], and the use of 3D convolutions to enable descriptor generation on sparse point clouds [86]. Descriptors used in this dissertation are generated using [25], where each scan is converted to a range image then converted into a single scan-based descriptor. Feature-based point cloud descriptors have also emerged, which use neural networks to generate descriptor vectors for local clusters of points [28, 29]. These feature-based descriptors

have proven to be more robust for global place recognition and are capable of providing relative pose estimates between clouds, but they are more expensive to compute and compare than scan-based descriptors made by [25].

1.3.3 Geometric Change Detection

Change detection is the problem of identifying differences in a mapped environment over time such as opened doors or moved objects. Change detection for mobile robots originated with vision-based methods, where the primary challenges pertain to comparing images taken in differing lighting conditions and from different points of view. Recent visual methods use increasingly sophisticated approaches to overcome challenges with varying points of view and environmental conditions. Lin et al. [87] use cross-attention modules to compare multiple images of the same scene with varied fields of view, and more recently NeRFs and Gaussian splats have become popular for overcoming field of view differences [88]. These representations render the scene from a common viewpoint so that images captured along different trajectories can be compared directly. Despite this progress, purely visual change detection still relies on training robust models and recognizing features from different viewpoints and with variable lighting.

Geometric change detection on the other hand attempts to solve the problem using geometric maps collected by LiDAR or RGB-D sensors. Geometric change detection has a rich history in remote sensing [89], where aerial mapping is used to monitor environmental changes over time. Within robotics, geometric methods can be separated into online and offline approaches. Offline methods focus on multi-session environmental monitoring and static map building, where tasks such as dynamic object removal [90], multi-session map alignment [91, 92], and dense change detection can be performed after mapping [3, 4, 93, 94]. Online geometric change detection has historically received less attention, but has become more important as autonomous robots are increasingly deployed into dynamic environments.

Methods that only use geometric sensing primarily perform scan-to-map change detection, comparing single scans to prior maps with distance-based thresholding [95–97] or by converting LiDAR point clouds into range images and applying established vision-based change detection methods [98]. Because a single scan is sparse and observes the scene from one viewpoint, these comparisons must separate genuine change from differences in coverage and occlusion, which limits the density of changes that can be reported reliably.

Recent developments in visual foundation models have led to open-set image-text embeddings [99] and object detection and segmentation [100, 101]. This allows robots equipped with geometric and visual sensing modalities to build spatio-temporal maps with embedded semantic knowledge. Tracking semantic entities across sessions allows changes to be reported at the level of objects rather than individual points [102]. While promising, this approach relies on a predefined vocabulary to query the map, which the authors of [102] note leaves the open-set problem unresolved, and requires multiple mutually calibrated sensing modalities rather than a single sensor.

1.4 Contributions and Outline of Dissertation

This dissertation formulates multiple open problems in geometric perception as submodular maximization, and proposes several functions and algorithms to optimize these problems. To this end, three primary contributions are made which advance submodular theory and improve LiDAR SLAM, map distillation, and change detection.

1.4.1 Contribution 1: Submodular Optimization for Keyframe Selection and Usage in SLAM

LiDAR Simultaneous Localization And Mapping (SLAM) has shown remarkable progress in recent years with centimeter-level accuracy across kilometer-long trajectories. However,

the computation time and memory allocated towards LiDAR SLAM increases with time and as new environments are mapped, meaning long-duration SLAM tends to fail due to a lack of resources. The first contribution of this dissertation, presented in Chapter 3, identifies keyframes — which are scans saved to memory primarily to serve as maps for aligning future scans — as a primary driver of increases in computation time and memory allocation. The contribution therefore enables long-duration LiDAR SLAM by optimizing the processes by which keyframes are selected and used via submodular maximization. A streaming submodular approach to map summarization is also proposed, which generates size-constrained keyframe maps for efficient communication and downstream processes. In order to solve these optimization problems online, this contribution introduces learned point cloud descriptors as a tool for quantifying the value of sets of LiDAR scans.

1.4.2 Contribution 2: OptMap Geometric Map Distillation via Submodular Maximization

The second contribution, presented in Chapter 4, formulates the problem of geometric map distillation, or optimal subset selection given a collected dataset of geometric perception data. Geometric map distillation generalizes the keyframe selection problems of the previous contribution to the optimal selection of any subset of collected geometric perception data. This contribution also proposes an online algorithm for solving geometric map distillation with arbitrary position, time, and size constraints. That algorithm, called OptMap, uses a novel submodular function which extends the function used for the previous contribution to address input set bias and effectively reduce input set sizes. This submodular function is then optimized using a dynamically reordered streaming submodular algorithm, which uses any of three proposed pairwise information heuristic functions to mitigate streaming order bias via a pseudo-greedy optimization.

1.4.3 Contribution 3: Online Geometric Change Detection via Scene Decomposition

The final contribution, presented in Chapter 5, uses OptMap as a core component in an online geometric change detection framework. This framework, called Change Detection via Scene Decomposition (CDSD), identifies unique scenes in the mapped environment to decompose the normally expensive point-wise change detection process into a series of tractable problems defined over locally consistent scenes. In addition to scene identification and OptMap for loading informative submaps, this contribution proposes a change detection algorithm meant to handle comparisons with submaps that may have differing positions or fields of view. Finally, a change manager clusters detected change points into change objects, which simplify map maintenance and give planners a clean, noise-filtered view of detected changes. By solving change detection as a submap-to-submap comparison rather than a scan-to-map comparison, CDSD reaches a whole-object detection rate of 89.4% where scan-to-map detection reaches only 10.6%.

1.4.4 Dissertation Outline

The remainder of the dissertation is structured as follows:

- Chapter 2 presents mathematical preliminaries for learned point cloud descriptors, submodular theory, and provides an overview of the notation used throughout.
- Chapter 3 introduces work on submodular optimization for keyframe selection and usage in SLAM which uses submodular maximization to optimize keyframe selection both at sensor rate and in an *a posteriori* process, and to select keyframe submaps which best constrain scan registration.
- Chapter 4 presents OptMap, an online algorithm for solving geometric map distillation.

OptMap optimizes a novel submodular function over the set of scan descriptors with a dynamically reordered streaming algorithm to produce maximally informative maps for large mapping datasets in less than a second.

- Chapter 5 introduces Change Detection via Scene Decomposition (CDSD), a framework for online geometric change detection. CDSD utilizes OptMap and a submap-to-submap change detection algorithm to find geometric changes in mapped environments online and with an order of magnitude better accuracy than scan-to-map methods.

- Chapter 6 summarizes the dissertation contributions, discusses limitations, proposes directions for future work, and provides concluding remarks.

CHAPTER 2

Preliminaries

The first two contributions in this dissertation formulate a robotic perception task as a combinatorial optimization over LiDAR scans, proves the resulting objective is submodular, and solves it with a greedy or streaming algorithm using learned point cloud descriptors as a proxy for scan similarity. The third contribution applies these submodular algorithms towards a relevant and open problem in online geometric change detection. This section collects the notation, descriptor background, and submodular optimization theory common to all three so that the subsequent chapters can focus on their individual contributions.

2.1 Notation

A *point cloud* is a finite set of points $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$ where each $\mathbf{p}_j \in \mathbb{R}^3$, and $\mathfrak{P}$ denotes the set of all possible point clouds. A LiDAR sensor collects a sequence of point clouds, or *scans*, where the i'th scan is denoted P_i and is associated with a pose $\mathbf{x}_i \in SE(3)$ estimated by an odometry algorithm. The set of all scans collected during a mapping session is denoted $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$, where the subscript indicates the sequence number of the scan. Each scan is compressed into a *descriptor* $e_i = \phi(P_i)$ by the learned function $\phi : \mathfrak{P} \to \mathbb{S}^n$ described in Section 2.2, where descriptors are unit-length $(n+1)$-dimensional vectors lying on the n-hypersphere $\mathbb{S}^n$. It is assumed throughout that a scan, its pose, and its descriptor arrive together as a packet $\{i, P_i, \mathbf{x}_i, e_i\}$ at sensor rate where i is the sequence number.

Combinatorial optimizations in this dissertation select a *solution set* S from a *ground or*

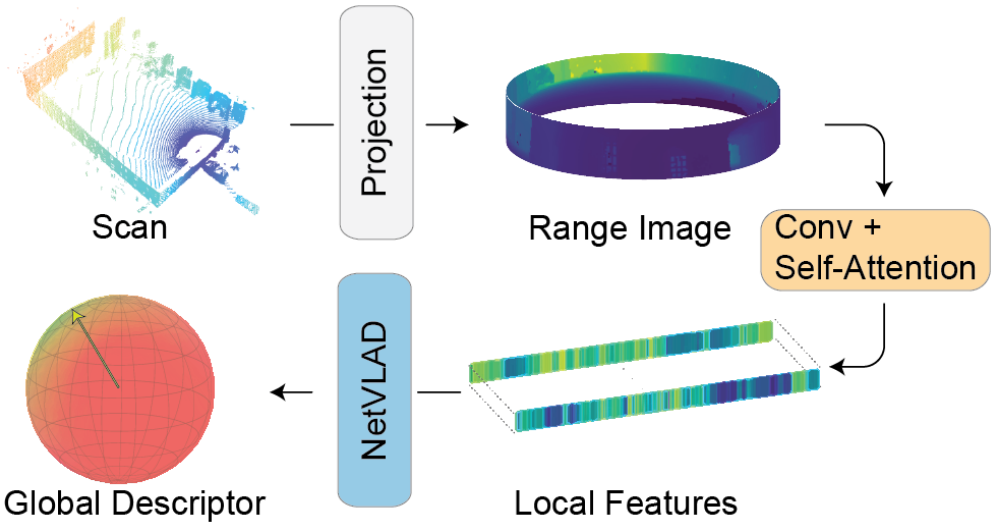


Figure 2.1: Global descriptor generation. The point cloud is projected into a range image (gray) and fed into the local features extraction module (orange) which consists of convolutional neural networks and a multi-head self-attention block. The NetVLAD module (blue) pools all the local features to generate the global descriptor.

input set E subject to a cardinality constraint $|S| \leq k$. Individual chapters specialize this notation where a distinguished subset carries its own meaning, e.g., the keyframe set $\mathcal{K}$, the reduced input set R, and the scene Z; each is defined at its point of use. The List of Symbols in the preliminary pages collects the symbols used throughout the dissertation.

2.2 Learned Point Cloud Descriptors

A natural metric for the mutual information of two point clouds P_i and P_j is *overlap*, the fraction of points in P_i that are sufficiently close to any point in P_j. Overlap is a useful measure of similarity, but computing it directly is expensive because individual scans contains hundreds of thousands of points and geometric maps contain millions of points. LiDAR descriptors instead compress dense point clouds into compact vectors where the distance between vectors encodes the similarity between the point clouds, allowing overlap to be estimated with simple vector operations [84].

Formally, this dissertation uses a discriminative and generalizable function $\phi : \mathfrak{P} \rightarrow \mathbb{S}^n$ mapping point clouds to global descriptor vectors such that, for any point clouds P_a, P_i, P_j and their corresponding descriptors $\phi(P_a)$, $\phi(P_i)$, $\phi(P_j)$,

$$\sigma(P_i, P_a) \geq \sigma(P_j, P_a) \implies ||\phi(P_i) - \phi(P_a)||_2 \leq ||\phi(P_j) - \phi(P_a)||_2, \tag{2.1}$$

where σ represents similarity between two point clouds. Descriptors are normalized to lie on the n-hypersphere such that $||\phi(P)||_2 = 1$, which bounds the distance between any two descriptors as $||\phi(P_i) - \phi(P_j)||_2 \leq 2$. The mutual information between two point clouds via descriptors is therefore quantified as

$$\mathcal{I}(P_i, P_j) = 2 - ||\phi(P_i) - \phi(P_j)||_2, \tag{2.2}$$

where the network is trained such that two point clouds with more overlap have less distance between their descriptors.

This dissertation adopts the neural network proposed by [25], with the overall framework presented in Fig. 2.1. Compared to point- and discretization-based methods [85,86,103], this network is compact and generalizable because it achieves fast execution without any semantic information. Instead of using the overlap between projected range images to supervise training, the similarity between two point clouds P_i and P_j is measured directly using the 3D Jaccard index [104]. Defined as $J(P_i, P_j) = |P_i \bigcap P_j|/|P_i \bigcup P_j|$, the Jaccard index is the equivalent of overlap in the 3D point cloud domain and provides a direct measure of point cloud similarity. Unless otherwise specified, the dimension of the output descriptor is 256.

2.3 Submodular Maximization

This dissertation relies on a range of topics across submodular theory. The following subsections provide readers with the necessary background knowledge to understand each of the contributions presented later.

2.3.1 Submodular Set Functions

Let X be a set of elements and 2^X be its power set.

Definition 1. A set function $f : 2^X \to \mathbb{R}_{\geq 0}$ is said to be *monotone non-decreasing* if $f(\emptyset) \geq 0$ and $f(B) \geq f(A)$ for all subsets $A \subseteq B \subseteq X$.

Definition 2. A set function $f : 2^X \to \mathbb{R}_{\geq 0}$ is *submodular* if for $A \subseteq B \subseteq X$ and element $x \in X \setminus B$

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B). \tag{2.3}$$

The quantity $f(A \cup \{x\}) - f(A)$ is the *marginal value* of x with respect to A, written throughout with the simplified notation $f(x|A)$. The intuitive way to understand Definition 2 in the context of submodular maximization is that the marginal value of any element must diminish as the solution grows in size, i.e., submodularity is the mathematically defined notion of diminishing returns. The problems considered in this dissertation take the form

$$\max_{S \subseteq E, |S| \leq k} f(S), \tag{2.4}$$

which is NP-hard in general but admits efficient approximation algorithms with tight sub-optimality guarantees when f is monotone non-decreasing and submodular.

Algorithm 1 Greedy Submodular Maximization

Given: E, k
Output: S
1: $S \leftarrow \emptyset$
2: **while** $|S| < k$ **do**
3: $\quad e = \arg\max_{e \in E \setminus S} f(e|S)$
4: $\quad S = S \cup \{e\}$
5: **end while**

2.3.2 The Greedy Algorithm

The seminal result in submodular optimization is the suboptimality bound of the simple greedy algorithm, given in Algorithm 1, for monotone non-decreasing submodular functions under a cardinality constraint $|S| \leq k$. The solution $S^{\#}$ is built over k iterations, each of which selects the element with the highest marginal value with respect to the partial solution. If S^* is the optimal solution to (2.4), then from [5, 6]

$$f(S^{\#}) \geq (1 - e^{-1}) f(S^*), \tag{2.5}$$

so the greedy solution is at worst a factor of 0.63 of the optimal. This bound is optimal for any algorithm that queries the oracle function a polynomial number of times, which justifies using greedy maximization to solve (2.4) to a known suboptimality bound. The cost of this guarantee is that Algorithm 1 requires k passes over the ground set, or $O(k|E|)$ function evaluations, and requires the entire ground set to be available in memory.

2.3.3 Streaming Submodular Maximization

Many applications of submodularity require parsing datasets too large to hold in memory, which is particularly challenging for the greedy algorithm because it must make k passes over the ground set to build a solution of size k. Streaming submodular algorithms instead maximize a submodular function using only a single pass over the ground set. If the optimal

Algorithm 2 Sieve-Streaming with Known Solution Bounds [7]

Given: $E, \underline{OPT}, \overline{OPT}, k$
Output: $\arg\max_{S_v} f(S_v)$

1: // Create OPT guesses and solution sets
2: $O = \{(1+\epsilon)^i \mid i \in \mathbb{Z},\ \underline{OPT} \leq (1+\epsilon)^i \leq \overline{OPT}\}$
3: **for** $v \in O$ **do**
4: $\quad S_v \leftarrow \emptyset$
5: **end for**
6: // Make single pass over the ground set
7: **for** $i \in \{1, 2, ..., |E|\}$ **do**
8: $\quad$ // Find marginal values on parallel threads
9: $\quad$ **for** $v \in O$ **do**
10: $\quad\quad$ **if** $|S_v| < k$ & $f(e_i|S_v) \geq \frac{v/2 - f(S_v)}{k - |S_v|}$ **then**
11: $\quad\quad\quad S_v = S_v \cup e_i$
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: **end for**

solution value OPT were known *a priori*, a solution could be built by comparing the marginal value of each element against OPT; for example, if each element added to a solution of final size k has a marginal value of at least OPT/k, then the produced solution would be optimal. Using OPT/k as a threshold is not practical in most cases, but the central contribution of [7] is a relaxed threshold which yields a provably near-optimal solution. The threshold proposed by [7], used on line 10 of Algorithm 2, is

$$T(k, S) = \frac{OPT/2 - f(S)}{k - |S|}. \tag{2.6}$$

In most cases OPT cannot be known *a priori*, so multiple solutions are maintained in parallel where each solution has a guess v between the *a priori* bounds $\overline{OPT} \geq v \geq \underline{OPT}$. When multiple solutions are updated throughout the stream, OPT in (2.6) is replaced with that solution's guess v.

Proposition 1. *(Sieve-Streaming [7]) Given a submodular function f, known optimal solution bounds, and threshold $T(k, S)$, Algorithm 2 satisfies the following properties*

- *It outputs a solution S such that $|S| \leq k$ and $f(S) \geq (\frac{1}{2} - \epsilon) \cdot OPT$.*
- *It does 1 pass over the ground set E, stores at most $O(\frac{k \log k}{\epsilon})$ elements and has $O(\frac{\log k}{\epsilon})$ update time per element.*

Proof. Proposition 1 is a sub-case of Proposition 5.2 from [7] with *a priori* OPT bounds. □

Note that the inner loop of Algorithm 2 evaluates the marginal value of each element with respect to all maintained solutions, a step which can be conducted in parallel to dramatically improve computation times. Also note that the suboptimality bound in Proposition 1 does not require the evaluation of all elements, i.e., if all solutions reach the cardinality constraint the algorithm may terminate early. The trade-off relative to the greedy algorithm is that elements must be accepted or rejected before the entire ground set has been observed, which introduces a sensitivity to the order in which elements arrive; this input order bias and a method for addressing it are treated in the OptMap chapter.

2.3.4 Incomplete Information

This section is largely inspired by the work of [39]. Evaluating the exact marginal value of an element can be prohibitively expensive, so it is sometimes useful to reason about a submodular function through partial knowledge of its values. This notion of incomplete information is stated formally as follows.

Definition 3. (κ-wise information [39]) Given a submodular function f over a set X, the κ-wise information set is defined as the set of tuples $\{(S, f(S)) | S \subseteq X, |S| \leq \kappa\}$. When $\kappa = 2$, this is referred to as *pairwise information*.

Knowing pairwise information for a given function f and set X is equivalent to having a look-up table for the value of f given any solution S such that $|S| \leq 2$. Using pairwise information, the estimated marginal value of element $x \in X$ with respect to any solution

$A \subseteq X$ can be defined as

$$\overline{f}(x|A) = \min_{a \in A} f(x|a). \tag{2.7}$$

The pairwise marginal value maintains submodularity and monotonicity of f, and [39] proposes an optimistic greedy algorithm which is shown to produce empirically near-optimal results for several submodular maximization problems. The OptMap chapter uses pairwise information as a computationally lightweight means for approximating the marginal value of elements with respect to a solution when such marginal values would typically require expensive computations.

2.3.5 Submodular Coverage and Clustering Functions

Two classes of submodular function recur throughout this dissertation for selecting informative subsets of descriptors, referred to as *coverage* and *clustering* using similar concepts from previous submodular summarization work [105]. Coverage functions minimize the mutual information between the elements of the solution set. An elementary coverage function over descriptors is formulated as an area coverage problem on $\mathbb{S}^n$. Consider each descriptor as the center of a hyperspherical cap with unit radius[1], where the cap with center e is notated $Cap(e) \subset \mathbb{S}^n$. An elementary coverage function is then the area of the union of the caps

$$f_{cov}(S) = \mathcal{A}(\cup_{e_i \in S} Cap(e_i)), \tag{2.8}$$

where $\mathcal{A}(\cdot)$ is the surface area of a subset of $\mathbb{S}^n$. Coverage encourages diverse solutions because high marginal value elements are those which are distant from other elements of the solution. The primary advantage of coverage functions is that the evaluation complexity only scales with the size of the solution set, which is often much smaller than the ground

[1] A hyperspherical cap is the set of points on the surface of a hypersphere which are all within a fixed distance from the center of the cap. Interested readers can find formal definitions of spherical caps and their areas in [106].

set. (2.8) can be proven monotone non-decreasing submodular in a similar manner to many other submodular area coverage functions (e.g., [54] Proposition 2).

Clustering functions instead maximize the mutual information between the solution and ground sets, and are defined in the following subsection.

2.3.6 Exemplar-Based Clustering

Two of the following chapters pose map summarization as a k-medoid problem, in which the goal is to minimize the sum of pairwise dissimilarities between the elements of the solution and the elements of the ground set. Using the Euclidean distance between descriptors, the k-medoid loss [107] is

$$L_{ebc}(S) = \frac{1}{|E|} \sum_{i=1}^{|E|} d(e_i, S), \tag{2.9}$$

which measures how well the solution S summarizes the ground set E. Minimizing (2.9) directly is intractable for even modest datasets, but it can be converted into a submodular function to be maximized by introducing an auxiliary *null element* $\mathbf{0}$, giving the exemplar-based clustering function

$$f_{ebc}(S) = L_{ebc}(\mathbf{0}) - L_{ebc}(S \cup \mathbf{0}). \tag{2.10}$$

Exemplar-based clustering is proven monotone non-decreasing and submodular in [107, 108] given an appropriate null element. Because descriptors lie on the unit hypersphere, the origin $\mathbf{0} = [0, 0, \ldots, 0]$ is a convenient choice as it is a uniform distance of 1 from all descriptors, so $L_{ebc}(\mathbf{0}) = 1$ and $f_{ebc}(\emptyset) = 0$. Clustering functions of this form identify solutions that are both *representative* and *diverse*, since an element must be distinct from the other solution elements and similar to many ground set elements in order to have a high marginal value.

CHAPTER 3

Submodular Optimization for Keyframe Selection and Usage in SLAM

This chapter presents the first contribution of the dissertation. LiDAR SLAM algorithms have proven to be remarkably accurate, and provide state estimates and maps which the rest of the autonomy stack relies on. However, the amount of computation time and memory required for LiDAR SLAM grows as new environments are mapped, which could lead to catastrophic failures in long-duration missions. This contribution minimizes the growing computation and memory burdens in long-duration LiDAR SLAM by optimizing how keyframes are selected and used, where keyframes are the saved scans used as the map for aligning incoming scans. By formulating this optimization as a submodular maximization, this contribution ensures that the solution is provably near-optimal and achieved in polynomial time. Importantly, this chapter is the first in this dissertation to utilize learned point cloud descriptors and formulate a keyframe selection problem which can occur both as scans are received and over an entire mapping session. Finally, a streaming submodular approach to map summarization generates size specified maps for efficient communication and downstream processes, serving as a preliminary version of the geometric map distillation algorithm developed in the following chapter. This chapter originally appeared as [83].

3.1 Overview

LiDAR is an increasingly popular sensing modality for Simultaneous Localization And Mapping (SLAM) because of its superior range measurement accuracy across different environments and conditions. LiDAR odometry estimation algorithms work by computing a rigid body transform that best aligns the most recent scan, i.e., point cloud, with previously registered point clouds. Given the density of LiDAR data, having a SLAM system that saves all incoming point clouds would be infeasible from both a memory and computation point of view. A more pragmatic approach is to save unique point clouds called keyframes that can be combined to form descriptive submaps for future scan alignment. Despite keyframes and submaps playing a central role in SLAM, their treatment has largely been left to heuristics which lack generality and require extensive, environment-specific tuning to achieve adequate performance. Another shortcoming of many LiDAR SLAM pipelines is their inability to create maps on-the-fly tailored for sharing with other agents or downstream processes that require specific information. This chapter addresses the issues of online keyframe selection and usage by leveraging the unique properties of submodular optimization to i) save unique keyframes according to a neural network which encodes point cloud similarity and ii) generate submaps via an optimization of the keyframes which best constrain scan alignment. A principled method for generating size-constrained summary maps for sharing or downstream processes is also proposed.

The key insight is that decisions regarding keyframe selection and usage in addition to map summarization can be formulated as combinatorial optimization problems. Submodularity, or the mathematically defined notion of diminishing returns, captures the trade-offs inherent in these problems and provides efficient algorithms for approximately solving NP-hard combinatorial problems with suboptimality guarantees. Submodular optimization has been used in a variety of robot applications such as single- and multi-agent coverage

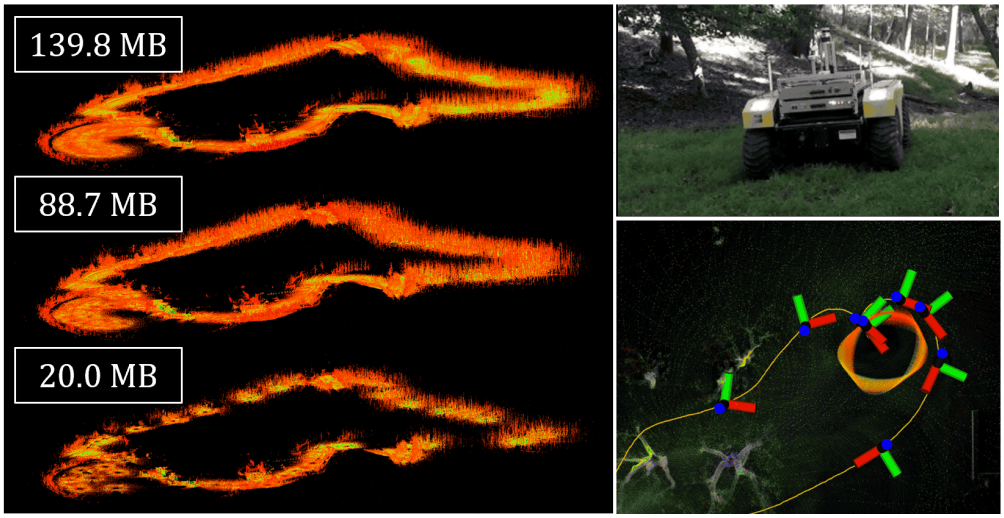


Figure 3.1: (Left) Dense point cloud map of 2.3 km forest loop with smaller summary maps. The dense map was generated during online SLAM and uses 456 keyframes as opposed to the summary maps which use 300 and 75 keyframes and are built in under one second. (Top-right) Modified Clearpath Robotics Warthog robot with LiDAR sensor used for collecting the forest loop dataset. (Bottom-right) Unique keyframe selection, each keyframe (axis) is chosen to capture a unique part of the map.

[10, 12, 49], but has been used sparingly in SLAM. The most relevant is that of [11] which used submodular optimization for selecting the most useful set of features to track over a window for visual-inertial odometry. Other SLAM works that utilize submodular optimization have primarily focused on sparsifying pose graphs [52, 109].

The contributions of this chapter are threefold. First, a keyframe selection strategy is proposed which saves point clouds with unique descriptors from a neural network, resulting in a sparser set of keyframes that use less memory than the baseline. Next, submodular optimization is leveraged to generate smaller submaps which better constrain scan alignment, leading to improvements in computation time when combined with sparse keyframe sets. The final contribution combines the LiDAR neural network and streaming submodular algorithms to generate size-constrained maps for efficient communication and fast operation in downstream processes. A key element to these methods is the efficient comparison between point clouds enabled by a neural network which encodes the similarity of LiDAR point clouds as descriptor vectors [25]. The results are demonstrated on long duration datasets includ-

ing a 2.3 km loop (pictured in Fig. 3.1) and a 6.2 km out-and-back at the Army Research Laboratory Graces Quarters test facility.

3.1.1 Problem Formulation

This chapter is concerned with formulating and efficiently solving combinatorial optimization problems of the form (2.4) that arise in SLAM. The definitions of monotonicity and submodularity given in Section 2.3.1, the greedy suboptimality bound of Section 2.3.2, and the streaming algorithm of Section 2.3.3 are used throughout this chapter and are not restated here. The one additional result required is a relaxation of submodularity, since functions that are monotone but not submodular can still enjoy suboptimality guarantees under greedy selection. The following two definitions originate in [37], but use the notation of [11].

Definition 4. The submodularity ratio of a non-negative set function $f : S \to \mathbb{R}$ with respect to a set A is

$$\gamma_A \triangleq \min_{L \subseteq A, S : |S| \leq \kappa, S \cap L = \emptyset} \frac{\sum_{s \in S} (f(L \cup \{s\}) - f(L))}{f(L \cup S) - f(L)}, \tag{3.1}$$

where κ is a strictly positive integer.

Definition 5. Let $f : S \to \mathbb{R}$ be a non-negative monotone set function and let S^* be its optimal solution subject to a cardinality constraint. The set $S^{\#}$ computed by the greedy algorithm is then given by

$$f(S^{\#}) \geq (1 - e^{-\gamma_{S^{\#}}}) f(S^*), \tag{3.2}$$

where $\gamma_{S^{\#}}$ is the submodularity ratio of the greedy solution.

Remark 1. Functions with submodularity ratios between 0-1 are henceforth referred to as approximately submodular due to the suboptimality guarantees they enjoy from Definition 5.

Chapter-specific notation. Following Section 2.1, E is the ground set of descriptors for all

collected point clouds $\mathcal{P}$. The set of keyframes is denoted $\mathcal{K}$, and the local submap $S \subseteq \mathcal{K}$. All sets have a defined ordering function where E and $\mathcal{K}$ are ordered chronologically and the ordering of S is defined by when an element is added. A matching upper case letter indicates a member of a set, e.g., $K \in \mathcal{K}$. The position of an element is given by a subscript index K_i, and the notation $S_{<i}$ denotes the subset of S such that only elements with index less than i are included, e.g., $S = \{K_0, K_1, K_2\},\ S_{<2} = \{K_0, K_1\}$.

This section shows how submodular optimization can be used to reduce the computational burden of SLAM systems without sacrificing localization accuracy. In the first section, point cloud descriptors are used to select a diverse and memory efficient set of keyframes. Next, submaps are generated by selecting keyframes which best constrain scan alignment using submodular optimization. Finally, the approach to quickly summarizing environments using streaming submodular optimization is described.

3.2 Keyframe Selection via Feature Descriptor

Selecting a keyframe from the stream of point clouds is a process that must balance localization performance—i.e., ensuring a candidate keyframe has sufficient overlap with saved keyframes for point cloud alignment—and memory usage, since any new keyframe must be kept in memory. One could formulate a combinatorial optimization problem to select the optimal set of keyframes $\mathcal{K}^*$ given localization and memory constraints. While an offline solution to this optimization is presented in Section 3.4, the online version is more nuanced since keyframe selection occurs before all scans are acquired. Despite this causality constraint, submodularity can still be utilized to obtain a suboptimality bound for the keyframing strategy.

The approach relies on the descriptor network discussed in Section 2.2 to efficiently determine the similarity between any two point clouds. The marginal value of any point

cloud is defined as the minimum Euclidean distance between the descriptor of a point cloud P and any previously selected keyframe $f(P|\mathcal{K}) = \min_{K \in \mathcal{K}} ||\phi(P) - \phi(K)||_2$. This marginal value inherently defines an objective function

$$f(\mathcal{K}) = \sum_{K_i \in \mathcal{K}} \min_{K_j \in \mathcal{K}_{<i}} ||\phi(K_i) - \phi(K_j)||_2. \tag{3.3}$$

Although (3.3) is not directly maximized, a suboptimality bound is still obtained by only selecting keyframes with high marginal value because f is a non-decreasing monotonic submodular set function, as proven in the following proposition.

Proposition 2. *The set function $f : \mathcal{K} \to \mathbb{R}$ in (3.3) is a non-decreasing monotone submodular function.*

Proof. Begin by defining the value of the empty set $f(\emptyset) = 0$. Monotonicity is proven because the marginal value is the Euclidean distance to the nearest keyframe which must be positive. Submodularity is proven by verifying the marginal values using two keyframe sets $\mathcal{K}_A \subseteq \mathcal{K}_B$ follow Definition 2 $f(K|\mathcal{K}_A) \geq f(K|\mathcal{K}_B) \implies \min_{K_A \in \mathcal{K}_A} ||\phi(K) - \phi(K_A)||_2 \geq \min_{K_B \in \mathcal{K}_B} ||\phi(K) - \phi(K_B)||_2$ which must be true since $\arg\min_{K_A \in \mathcal{K}_A} ||\phi(K) - \phi(K_A)||_2 \in \mathcal{K}_B$, but $\arg\min_{K_B \in \mathcal{K}_B} ||\phi(K) - \phi(K_B)||_2 \notin \mathcal{K}_A$. In other words, the minimizer w.r.t. $\mathcal{K}_A$ must be in $\mathcal{K}_B$ but the minimizer w.r.t. $\mathcal{K}_B$ may not belong to $\mathcal{K}_A$. □

Any keyframe set is additionally required to be sufficient for localization, formalized using degeneracy, a sensor- and environment-agnostic metric proposed in [104]. To enforce this constraint, a keyframe is selected whenever $\delta = \frac{m^2}{\lambda_{min}(\mathcal{H}_{tt})\sqrt{z}} \geq \beta$, where $\lambda_{min}(\mathcal{H}_{tt})$ is the minimum eigenvalue of the Hessian of the scan alignment cost function (discussed thoroughly in Section 3.3) and m and z are adaptive parameters defined in [104].

Algorithm 3 describes the keyframe selection strategy, where a keyframe is selected according to a feature distance condition with threshold α, and a degeneracy condition with

Algorithm 3 Online Keyframe Selection

Given: $\mathcal{K} \leftarrow Keyframes, P \leftarrow recentScan, \alpha, \beta$
Output: bool $saveKeyframe$, γ
1: $\Delta = \min_{K \in \mathcal{K}} ||\phi(P) - \phi(K)||_2$
2: **if** $\Delta \geq \alpha$ **then**
3: return $\{true, \ 0\}$
4: **end if**
5: $\delta \propto \frac{1}{\lambda_{min}(\mathcal{H}_{tt})}$
6: **if** $\delta \geq \beta$ **then**
7: return $\{true, \ \alpha - \Delta\}$
8: **end if**
9: return $\{false, \cdot\}$

threshold β. In addition to a boolean variable representing whether to save a keyframe or not, the algorithm returns the added suboptimality (with respect to (3.3)) γ such that the marginal value of the new keyframe is $\alpha - \gamma$. Line 1 finds the smallest distance to any keyframe in descriptor space. If the closest distance is greater than α, line 3 returns true and reports no additional suboptimality. Lines 5-8 compute degeneracy and save a keyframe if it exceeds the threshold β. If neither condition is met, Algorithm 3 returns false. By knowing the marginal value of each keyframe at the time of selection, a known suboptimality bound is maintained, given in the following proposition.

Proposition 3. *The keyframe set $\mathcal{K}$ generated by Algorithm 3 maintains a suboptimality bound relative to an optimal keyframe set $\mathcal{K}^*$ of $f(\mathcal{K}) \geq \frac{\alpha}{2} f(\mathcal{K}^*) - \sum_{K \in \mathcal{K}} \gamma_K$.*

Proof. The marginal value of each keyframe in an optimal set $\mathcal{K}^*$ is upper bounded by the maximum distance between any points on a unit hypersphere $f(K|\mathcal{K}) \leq 2 \rightarrow f(\mathcal{K}^*) \leq 2|\mathcal{K}|$. When keyframe K is added, its marginal value is $\alpha - \gamma_K$. Thus, the total value is found by taking a sum over all placed keyframes $f(\mathcal{K}) = \sum_{K \in \mathcal{K}} (\alpha - \gamma_K) = \alpha|\mathcal{K}| - \sum_{K \in \mathcal{K}} \gamma_K \geq \frac{\alpha}{2} f(\mathcal{K}^*) - \sum_{K \in \mathcal{K}} \gamma_K$. □

3.3 Submap Generation via Submodular Maximization

LiDAR odometry derives its accuracy from scan alignment algorithms which find the pose (translation $\mathbf{t} \in \mathbb{R}^3$ and rotation $\mathbf{q} \in \mathbb{S}^3$) that minimizes the nonconvex cost function

$$\sum_{c \in \mathcal{C}} \mathcal{E}(\mathbf{t}, \mathbf{q}, P, T), \tag{3.4}$$

with the sum of error residuals $\mathcal{E}$ over all sufficiently close, i.e., corresponding, points in $\mathcal{C}$ between the scan P and target scan T. Computing an accurate solution to (3.4) in real-time requires intelligently selecting the target scan, e.g., a submap $S \subseteq \mathcal{K}$. The plane-to-plane cost function from [60] is used.

Submap generation carries a trade-off between alignment accuracy and runtime, as previous work demonstrated the advantage of including corresponding points at the edge of the scan, but computation time scales with submap size [110], [57]. Poor scan alignment is primarily caused by self-similar environments, e.g., long featureless hallways, which introduce close local minima. The minimum eigenvalue associated with translation of the Hessian of (3.4) $\lambda_{min}(\mathcal{H}_{tt})$ has been used to quantify how well constrained (3.4) is [67, 111, 112]. The Hessian is assumed available in analytic form by differentiating (3.4) with respect to three translational variables to find a Jacobian J_t, which can be used to approximate the Hessian via $\mathcal{H}_{tt} \approx J_t^T J_t$. Thus, the objective is to find the submap $S \subseteq \mathcal{K}$ for a given scan P that solves

$$\begin{aligned} &\max_{S \subseteq \mathcal{K}} f(S) = \lambda_{min}(\mathcal{H}_{tt}(P, S)) \\ &\text{Subject to } |S| \leq k, \end{aligned} \tag{3.5}$$

which can be interpreted as finding the submap S that maximizes the minimum eigenvalue of $\mathcal{H}_{tt}$. To make solving (3.5) tractable, the following assumption is made.

Assumption 1. The marginal value of adding a keyframe to the submap for (3.5) is dom-

Algorithm 4 Submodular Submap Generation

Given: $\mathcal{K} \leftarrow Keyframes$, $P \leftarrow recentScan$
Output: $S \leftarrow Submap$

$S, UpperBounds \leftarrow \emptyset$
// Find max Hessian eigenvalue for keyframes
for $i \in \{0, 1, ..., |\mathcal{K}| - 1\}$ **do**
  $\mathcal{H}_{tt} = approxHess(P, K_i)$
  $UpperBounds[i] = \lambda_{max}(\mathcal{H}_{tt})$
end for
// Greedy optimization
$\mathcal{H}_{prev} = \mathbf{0}^{3\times3}$, $\Delta_{max} = \infty$
while $|S| \leq k$ **and** $\Delta_{max} \geq 1$ **do**
  $\Delta_{max} = 0$, $\partial\mathcal{H}_{tt} = zeros(|\mathcal{K}|)$
  for $i \in \{0, 1, ..., |\mathcal{K}| - 1\}$ **do**
    if $K_i \in S$ **or** $UpperBounds[i] \leq \Delta_{max}$ **then**
      continue
    end if
    $\partial\mathcal{H}_{tt}[i] = approxHess(P, K_i)$
    $\Delta_{max} = max(\Delta_{max}, \lambda_{min}(\mathcal{H}_{prev} + \partial\mathcal{H}_{tt}[i]))$
  end for
  $i_{best} = \arg\max_i \lambda_{min}(\mathcal{H}_{prev} + \partial\mathcal{H}_{tt}[i])$
  $\mathcal{H}_{prev} = (\mathcal{H}_{prev} + \partial\mathcal{H}_{tt}[i_{best}])$
  // Add to submap and remove points from scan
  $S = S + K_{i_{best}}$
  $P = P - K_{i_{best}}$
end while

inated by the additional value provided by new correspondences being made, i.e., points shared between the scan and new keyframe which did not exist in the previous submap.

Assumption 1 allows $\mathcal{H}_{tt}$ to be approximated as a sum of sub-Hessians. The sub-Hessian associated with keyframe K_i is denoted $\partial\mathcal{H}_{tt}[i]$ and approximated as $\partial\mathcal{H}_{tt}[i] = \mathcal{H}_{tt}(P - S_{<i}, K_i)$, where $S_{<i}$ is the submap including all keyframes with index less than i, and the subtraction of point clouds indicates removing the intersection of points ($A - B = A - (A \cap B)$). Because each Hessian is the outer product of the Jacobian with itself, the Hessian must be positive semi-definite $\mathcal{H}_{tt} = J_t^T J_t \succeq 0$, which intuitively means each keyframe cannot make (3.4) less constrained. Any keyframe which introduces enough new correspondences

such that $\partial\mathcal{H}_{tt}[i] \succ 0$ is called a *constructive keyframe*.

Proposition 4. *The set function $f : S \to \mathbb{R}$ in (3.5) is an increasing monotone approximately submodular function under Assumption 1 as long as k in the cardinality constraint $|S| \leq k$ is small enough so each keyframe is constructive.*

Proof. This proof is similar to the proof of approximate submodularity in Proposition 11 of [11]. (3.5) is shown to be monotone increasing via the marginal gain

$$\begin{aligned}
f(K_i|S) &= \lambda_{min}(\mathcal{H}_{tt}(P, S \cup K_i)) - \lambda_{min}(\mathcal{H}_{tt}(P, S)) \\
&= \lambda_{min}(\mathcal{H}_{tt}(P, S) + \partial\mathcal{H}_{tt}[i]) - \lambda_{min}(\mathcal{H}_{tt}(P, S)) \\
&= \min_{||\mathbf{u}||=1} \mathbf{u}^T[\mathcal{H}_{tt}(P, S) + \partial\mathcal{H}_{tt}[i]]\mathbf{u} - \min_{||\mathbf{v}||=1} \mathbf{v}^T\mathcal{H}_{tt}(P, S)\mathbf{v} \\
&\geq \bar{\mathbf{u}}^T[\mathcal{H}_{tt}(P, S) + \partial\mathcal{H}_{tt}[i]]\bar{\mathbf{u}} - \bar{\mathbf{u}}^T\mathcal{H}_{tt}(P, S)\bar{\mathbf{u}} \\
&= \bar{\mathbf{u}}^T\partial\mathcal{H}_{tt}[i]\bar{\mathbf{u}} > 0,
\end{aligned}$$

where $\bar{\mathbf{u}}$ is the minimizer of $\min_{||\mathbf{u}||=1} \mathbf{u}^T[\mathcal{H}_{tt}(P, S) + \partial\mathcal{H}_{tt}[i]]\mathbf{u}$, and the final line is true because the keyframe is constructive. The numerator of Definition 4 is the minimum marginal gain attainable, so if the function is monotone increasing, the submodularity ratio must be bounded away from 0 proving (3.5) is approximately submodular. □

Algorithm 4 shows the approach to submap generation, which begins by finding and saving maximum Hessian eigenvalues (lines 3-6). Note that keyframes at least twice the sensor range from the current pose are not considered, as they share no overlapping points. Each greedy iteration (lines 9-23) computes the highest marginal value keyframe, then adds it to the submap and removes its points from the scan such that the following iterations will compute appropriate sub-Hessians. The resulting submap enjoys a suboptimality bound from Definition 5 if it reaches the cardinality constraint or achieves the maximum attainable value of (3.5) if the marginal value approaches zero. Hessians are only approximated using a

percent ($\sim 25\%$) of randomized scan points as previous work has shown this has a minimal effect on the relative magnitude of ICP Hessian eigenvalues [113].

Computation is saved by eliminating the need to test most keyframes in each greedy iteration via an application of the classical Weyl inequalities, which state

$$\lambda_{i+j-1}(A+B) \leq \lambda_i(A) + \lambda_j(B) \leq \lambda_{i+j-n}(A+B), \tag{3.6}$$

where A and B are Hermitian matrices with ordered eigenvalues such that $\lambda_1 \geq ... \geq \lambda_n$. Weyl's $1st$ inequality ($i = n, j = 1$) upper bounds the minimum eigenvalue of the Hessian $\lambda_{min}(\mathcal{H}_{tt} + \partial\mathcal{H}_{tt}[i]) \leq \lambda_{min}(\mathcal{H}_{tt}) + \lambda_{max}(\partial\mathcal{H}_{tt}[i])$. Next, note that the maximum eigenvalues of the Hessians obtained on line 5 upper bound the sub-Hessian eigenvalues $\lambda_{max}(\mathcal{H}_{tt}(P, K)) \geq \lambda_{max}(\partial\mathcal{H}_{tt}[i])$. This is because the maximum eigenvalue is monotone non-decreasing (Weyl's $2nd$ inequality $i = 1, j = n$) and the Hessian obtained on line 5 is the sum of the sub-Hessian with its counterpart formed by points shared by the scan, keyframe, and previous submap. This combined with the minimum eigenvalue upper bound imply an upper bound on the marginal value of adding keyframe K_i of $\Delta\lambda_{min}(S \cup K) \leq \lambda_{max}(\mathcal{H}_{tt}(P, K))$. This means any keyframe with an upper bound smaller than the best marginal gain in the current iteration can be ignored.

3.4 Map Summary via Streaming Submodular Optimization

Online keyframe selection must sufficiently constrain scan alignment, but too many keyframes yield unnecessarily dense maps that are not well suited for sharing across a team nor for downstream processes such as trajectory generation. To address this issue, this dissertation proposes a method to generate summary keyframe sets after completing a SLAM session using streaming submodular optimization. This new feature can quickly generate a summary set of keyframes given a prescribed number of keyframes or memory limit, with computation

time that only scales quadratically with the number of scans collected and notably not with the prescribed number of keyframes. The approach is also applicable to offline selection of keyframes from, e.g., a large dataset with recorded registered point clouds.

Two classes of submodular function are available for this task, coverage and clustering, both defined over descriptors in Sections 2.3.5 and 2.3.6. Coverage functions such as (2.8) are attractive because their evaluation complexity scales only with the size of the solution set, which is far smaller than the set of all registered scans. However, coverage only rewards solutions whose elements are dissimilar from one another, so it has no mechanism for preferring keyframes that are representative of the scans they stand in for. A summary map selected this way can be maximally spread out in descriptor space while still omitting the regions where most of the session was actually spent. Clustering functions reward both properties, since an element must be distinct from the other selected keyframes *and* similar to many of the collected scans in order to have a high marginal value. This robustness is worth the added evaluation cost for map summarization, where the output is judged by how well it stands in for the full session, and exemplar-based clustering is therefore adopted here.

Evaluating either class directly over point clouds would be intractable. Overlap between two scans is the natural measure of their mutual information, but computing it requires a nearest-neighbor search over clouds containing tens of thousands of points. Learned descriptors reduce that comparison to the Euclidean distance between two fixed-length vectors, so the cost of evaluating a summarization objective becomes independent of how many points each scan contains.

The LiDAR map summarization task is formulated as the k-medoid problem described in Section 2.3.6, where the goal is to minimize the sum of pairwise dissimilarities between keyframes and elements of the ground set (all registered scans). Applying the k-medoid loss (2.9) to a keyframe set gives $L(\mathcal{K}) = \frac{1}{|E|}\sum_{e\in E} d(e, \phi(\mathcal{K}))$, which measures how well a keyframe set summarizes an environment. Minimizing this loss function is generally in-

Table 3.1: End-to-end accuracy of the proposed method compared to open-source LiDAR SLAM algorithms. Tests (A)-(C) are from the UCLA campus (Royce Hall, Sculpture Garden, and Court of Sciences).

	End-to-End Translational Error [m]			RAM Usage [MB]		
Algorithm	A (652.7 m)	B (530.8 m)	C (551.4 m)	A	B	C
FAST-LIO2 [57]	0.0380	0.1162	0.0992	327.9	387.1	**327.8**
DLIOM [104]	**0.0120**	0.0226	0.0301	1021.3	833.2	630.0
Ours	0.0156	**0.0149**	**0.0143**	**273.3**	**264.2**	335.1

tractable for even small datasets. At approximately 1×10^{-5} seconds per set evaluation, brute force would take over five years to find an optimal summary map of 10 keyframes from 100 total scans. Following (2.10), an auxiliary element (scan) is assumed with a feature vector equal to the zero vector ($\mathbf{0} \in \mathbb{R}^{256}$), and a streaming submodular algorithm instead maximizes

$$\max_{\mathcal{K} \subseteq E} f(\mathcal{K}) = L(\mathbf{0}) - L(\mathcal{K} \cup \mathbf{0}), \tag{3.7}$$

which is proven monotone submodular in [107].

The Sieve-Streaming algorithm of [7] is followed, given in Algorithm 2, which makes a single pass over all scans and adds a keyframe to a solution set whenever its marginal value exceeds a v-dependent threshold, returning the highest value solution set. The maximum value of any single scan is $\max_{e \in E} f(e|\mathbf{0}) = 1$ because each descriptor is mapped to a unit hypersphere and the auxiliary element is a distance of 1 from all scans, which supplies the *a priori* bounds required by Algorithm 2. The greedy maximization of (3.7) requires a pass over the dataset for each keyframe added (total of k) and the corresponding complexity is $O(k \cdot |E|^2)$. On the other hand, the streaming submodular method only requires a single pass, improving the runtime complexity to $O(\frac{1}{\log \epsilon}|E|^2)$ and eliminating the need to have the entire dataset loaded in RAM at any point.

Algorithm 5 Heuristic Solution for Initial Lower Bound

Given: E, k
Output: $\underline{OPT} \leftarrow f(S_h)$
1: $d_{tot} = 0$
2: **for** $i \in \{2, 3, ..., |E|\}$ **do**
3: $\quad d_{tot} = d_{tot} + ||e_i - e_{i-1}||_2$
4: **end for**
5: $S_h = \{e_1\}, i = 2$
6: **while** $|S_h| < k$ **do**
7: $\quad d = 0$
8: $\quad$ **while** $d \leq d_{tot}/k$ **do**
9: $\quad\quad d = d + ||e_i - e_{i-1}||_2$
10: $\quad\quad i = i + 1$
11: $\quad$ **end while**
12: $\quad S_h = S_h \cup e_{i-1}$
13: **end while**

3.4.1 Algorithmic Implementation

Two implementation details make streaming summarization practical, and both are reused by the map distillation algorithm of the following chapter. Neither is specific to the summarization objective (3.7); they concern how tightly the solution value can be bracketed before the stream begins, and how the ground set can be restricted before optimization starts.

Initial solution bounds. The time complexity of streaming submodular maximization algorithms can vary significantly depending on the existence or quality of *a priori* solution value bounds, because tight bounds allow for fewer maintained solutions which ultimately require fewer computations. The upper bound is immediate: the objective is maximized when the solution set contains the entire ground set, in which case the loss term vanishes and $\overline{OPT} = 1$. A tighter lower bound than 0 is obtained from a heuristic solution, described in Algorithm 5. The heuristic solution finds k elements with equal distance along the sequence of descriptors, where the equal spacing works best in mapping datasets which explore new environments at consistent rates. Lines 1-4 accumulate the total descriptor path length by summing the distance between all sequential ground set elements. The heuristic solution is

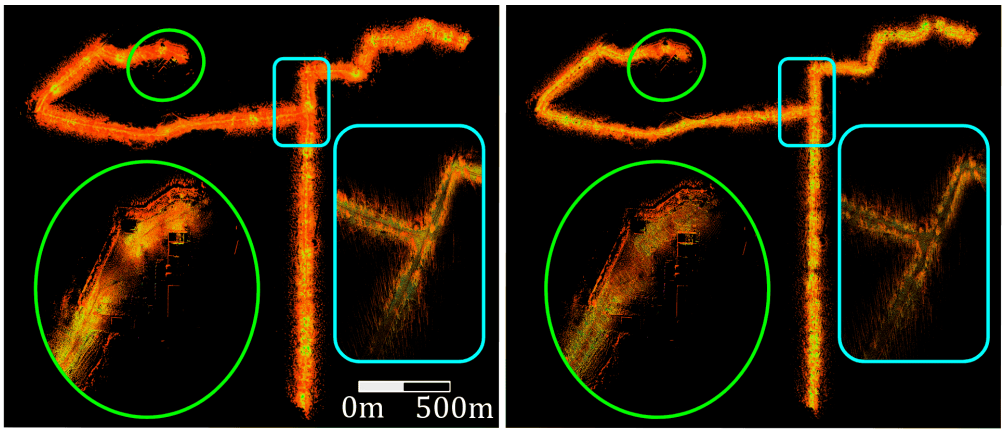


Figure 3.2: Map summaries for 6.2 km Mout Water Dataset from Graces Quarters. Left figure shows the full keyframe map using 527 keyframes and 186 MB. Right shows the summary map using 300 keyframes and 110 MB. Insets show detailed comparison of the maps.

thus built (lines 5-13) by selecting elements from the ground set at regular intervals of d_{tot}/k. The lower bound is obtained by evaluating the heuristic solution on the summarization objective, which requires one evaluation pass.

Position and time constraints. The ground set can be further constrained by applying position and time constraints before optimization. Position constraints are defined by a subset $A \subseteq \mathbb{R}^3$ where the boundaries of A can be defined using hyperplanes or other common shapes. Time constraints are defined with a set of initial and final timestamps (t_i, t_f). Each element of the ground set is a scan with an associated position $x(e_i)$ and time $t(e_i)$. The filtered ground set is then the subset with poses and timestamps inside the defined constraints, defined as $E_c := \{e_i \in E |\, x(e_i) \in A\,,\, t(e_i) \in [t_i, t_f]\}$. Multiple constraints can be applied, giving the ability to summarize specified portions of a mapping session rather than the session as a whole. Where a specific region is requested in later chapters, a convex hull position constraint is used, where a box is built by taking the convex hull of each vertex of the box.

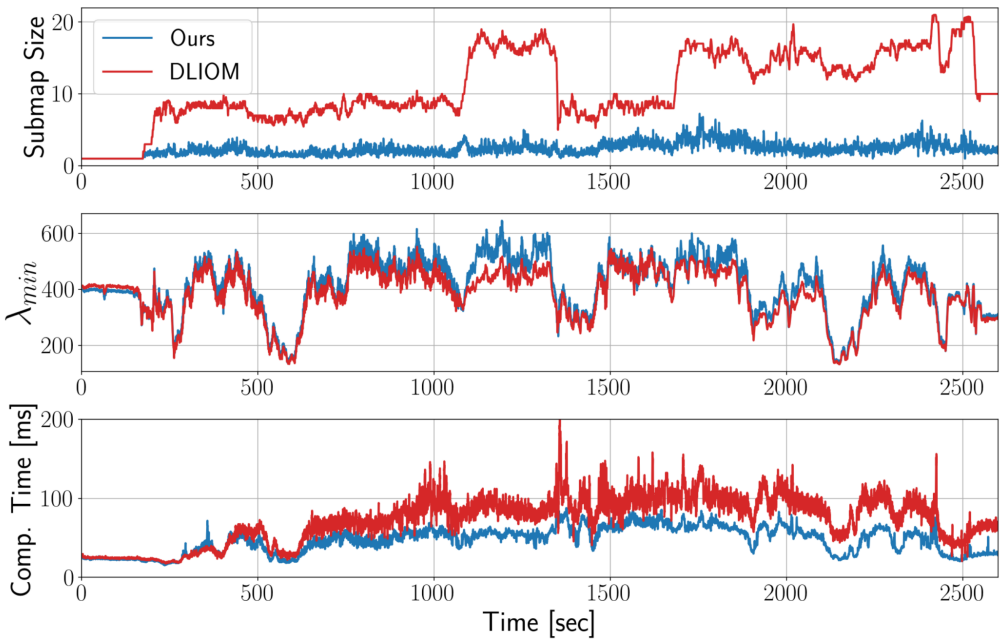


Figure 3.3: Submap generation comparison using 6.2 km Mout Water dataset using identical keyframe selection strategies. Top shows keyframes used per submap (DLIOM avg. 10.4 Ours avg. 2.3). Middle shows the minimum Hessian eigenvalue (DLIOM avg. 377 Ours avg. 407). Bottom shows per scan computation time (DLIOM avg. 67.3 ms Ours avg. 45.4 ms).

3.5 LiDAR Keyframing Results

A modified version of DLIOM [104] is used, where keyframes used for scan-to-submap alignment are the primary source of memory usage. The experiments use datasets collected at UCLA with a 10 Hz 32-beam Ouster OS0 and at the Army Research Laboratory Graces Quarters facility with a 20 Hz 128-beam Ouster OS1 mounted on a Clearpath Warthog. Graces Quarters datasets include a 2.3 km loop and 6.2 km out-and-back shown in Fig. 3.1 and Fig. 3.2. The neural network was trained on UCLA campus data for 30 epochs.

3.5.1 Online Keyframe Selection

The first experiment compares end-to-end accuracy and memory usage (measured by RSS) of the proposed approach against DLIOM [104] and a competitive open-source LIO algorithm [57]. Note that memory allocation is a complex process which RSS does not entirely capture

as running these algorithms on different hardware can affect RSS. All results in Table 3.1 were made using an i7-10875H 8-Core CPU with 16 GB of RAM. The proposed method achieves comparable end-to-end accuracy with DLIOM despite using 64% less memory on average. This is due to a reduction in the number of keyframes selected, at 80% fewer keyframes than DLIOM. This efficient approach to keyframe selection is shown in the bottom-right of Fig. 3.1 where eight keyframes each capture unique areas around a large sculpture.

3.5.2 Submodular Submap Generation

Submap generation uses fewer keyframes to improve computation time without affecting scan alignment. Fig. 3.3 shows a reduction from an average submap size of 10.4 keyframes using DLIOM to 2.3 using the proposed method over a 6.2 km, 43 minute trajectory using an identical set of keyframes (placed every 5 m). The proposed method achieves a slightly improved alignment constraint over the trajectory, improving from an average minimum Hessian eigenvalue of 377 to 407, while also decreasing the average computation time per scan from 67 to 45 ms. This improvement is due to the submap generation selecting a smaller set of keyframes which can include distant keyframes with nonzero marginal value that heuristics might not consider.

3.5.3 Map Summarization

Fig. 3.1 and Fig. 3.2 show the map summarization tool in use on datasets at Graces Quarters. The summary maps for the Forest Loop (2.3 km, 28.5 minutes) selected 75 and 300 keyframes and were generated in 0.67 and 0.69 seconds. The Mout Water (6.2 km, 43 minutes) summary map was generated using 300 keyframes in 1.65 seconds. The insets in Fig. 3.2 show a similar level of detail despite the summary map using 40% less memory. The streaming approach improved the computation time of the summary map generation, as the same forest loop maps took 1.94 and 7.06 seconds respectively using a greedy approach.

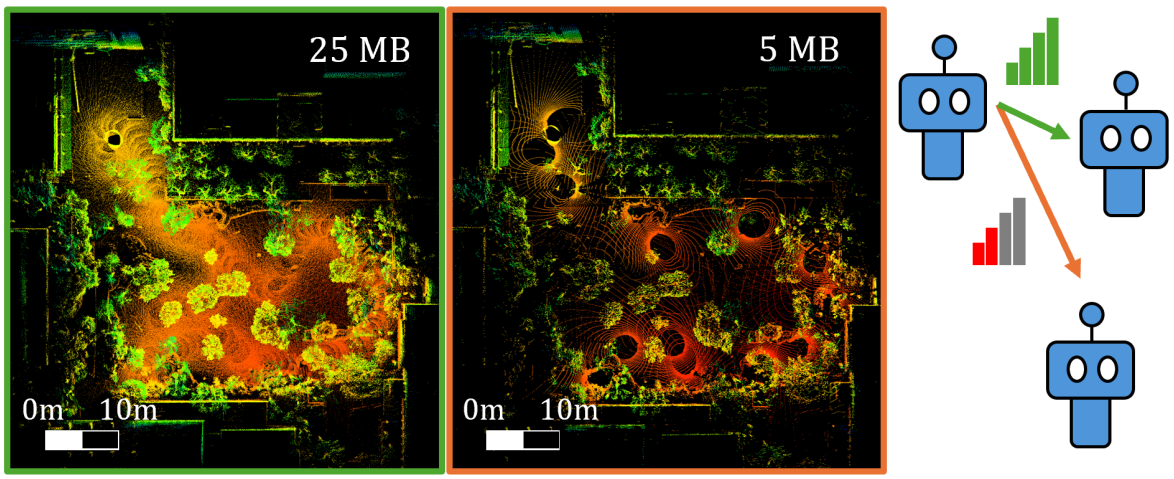
3.6 Conclusion

This chapter presented a novel online and offline solution for keyframe selection in addition to a new keyframe submap generation procedure. Online keyframe selection identifies unique keyframes by generalizing the common distance threshold to feature space where each descriptor encodes point cloud similarity. Similar localization accuracy is maintained while reducing the average submap size and improving per scan computation time by optimizing submap generation for scan alignment constraint. Finally, the streaming submodular approach to map summarization generates size specified maps for efficient communication and downstream processes.

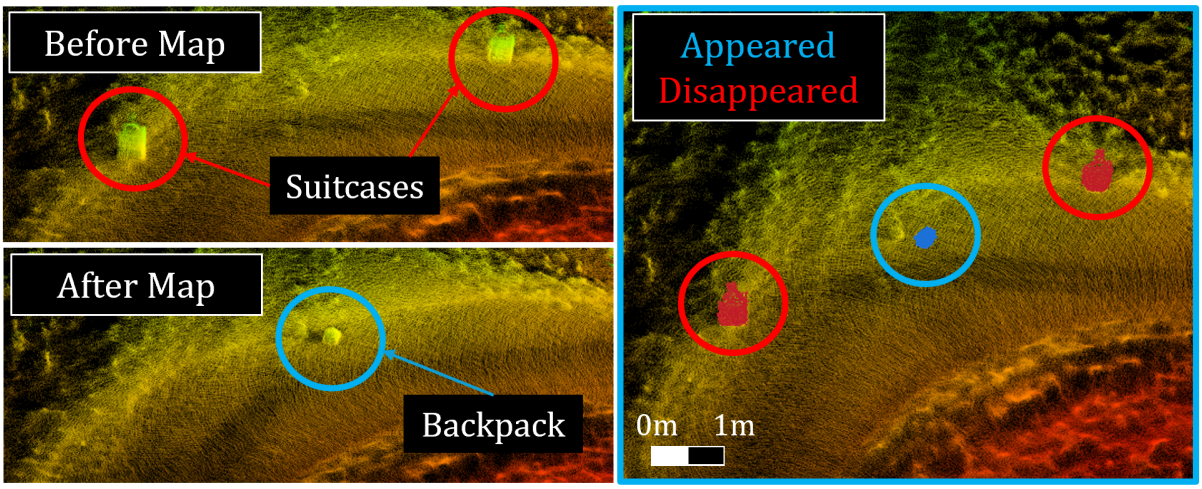
CHAPTER 4

OptMap: Geometric Map Distillation via Submodular Maximization

The previous chapter established the need for computationally efficient means of summarizing large LiDAR datasets, and proposed an initial solution by optimizing over a set of learned point cloud descriptors. This chapter, the second contribution of the dissertation, generalizes this summarization approach to propose a new class of problems in perception: *geometric map distillation*, which is posed as an optimal subset selection over the set of all collected geometric perception data. The proposed algorithm OptMap optimizes a novel submodular function via a dynamically reordered streaming submodular algorithm, solving the resulting NP-hard optimization in a single pass with near-optimality guarantees. The function is a novel variant of exemplar-based clustering called Continuous Exemplar-Based Clustering (CEBC), which addresses input set bias towards repeated and redundant scans by weighting each element according to its distance from the previous descriptor. The dynamically reordered streaming algorithm also addresses input order bias in streaming submodular algorithms via a pseudo-greedy approach which only relies on an approximation of scan-to-scan overlap, for which three variants are proposed and evaluated. This chapter originally appeared as [114].



(a) OptMap application maps for multi-robot map sharing. Maps can be generated to summarize a full session (Sculpture Garden) with a given size dictated by communication bandwidth constraints.



(b) OptMap application maps for change detection (left). Online change detection demands concise yet dense maps. OptMap can provide these dense maps given a location or region query from a user.

Figure 4.1: OptMap is a geometric map distillation algorithm which provides size-constrained, informative maps for downstream use within autonomy systems.

4.1 Overview

Modern autonomous systems use a modular software architecture with separate algorithms to perceive the environment, plan collision-free paths, estimate vehicle motion, and make high-level decisions. Many of these algorithms require a geometric map with *unique requirements* to optimize performance and processing time. For example, multi-robot teams with bandwidth constraints could share summary maps tailored to current network conditions

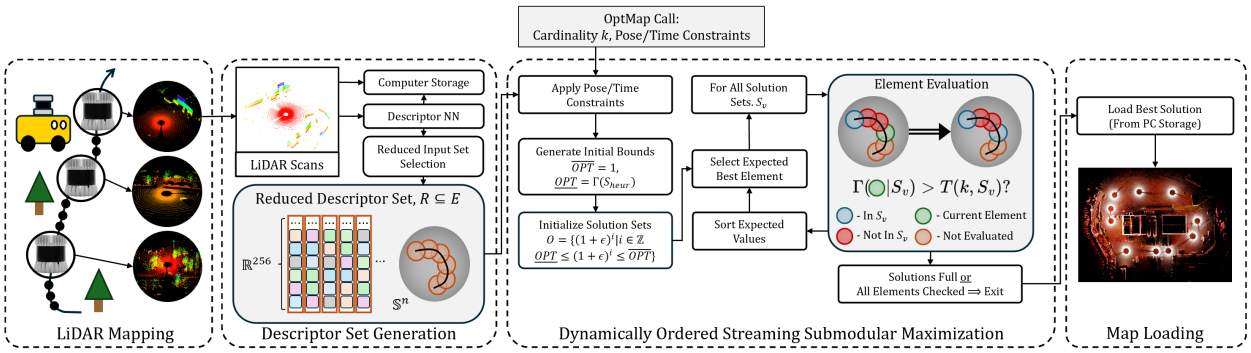


Figure 4.2: The OptMap pipeline consists of four stages: (i) LiDAR mapping (ii) descriptor set generation, (iii) dynamically reordered streaming submodular maximization, and (iv) map loading. Point clouds from LiDARs are used to generate descriptors which are then saved to computer storage. Redundant descriptors are then filtered to improve evaluation time. The dynamically reordered streaming submodular maximization uses an online approximate value of each element to address input order bias and select informative solutions.

(Fig. 4.1a), while detecting changes in previously mapped areas demands dense, concise maps of local scenes (Fig. 4.1b). This dissertation calls the problem of selecting informative application-specific maps from the complete set of collected perception data *geometric map distillation*. One approach to solving geometric map distillation is to maintain a single map with every unique geometric feature and use point operations, such as voxelization and nearest neighbor search, to provide application-specific maps. However, this is prohibitively expensive from a memory and computation perspective as modern LiDAR sensors collect $250,000$-point scans at 20 Hz, or about 50 MB every second. Achieving the examples in Fig. 4.1 online therefore requires both an informativeness metric that avoids point-wise operations entirely and an optimization algorithm efficient enough to apply it across the tens of thousands of scans collected in a long mapping session. To this end, this chapter proposes OptMap, an online geometric map distillation algorithm which quantifies informativeness indirectly via a set of learned point cloud descriptors, and solves the resulting NP-hard optimization in a single pass with near-optimality guarantees.

Geometric map distillation is best formulated as a combinatorial optimization problem (i.e., optimal subset selection [31]) that aims to maximize the informativeness of an output map given all available perception data. For point clouds, informativeness is readily quan-

tified using the amount of redundant data or overlapping regions, but optimizations that rely on direct overlap computations are expensive because geometric maps contain millions of points. Solving such problems with formal suboptimality bounds is thus only possible as an offline process. Previous research sought to simplify overlap calculations by learning to encode point cloud similarity via compact descriptor vectors [25], as described in Section 2.2. Although most descriptors are used for global place recognition [84–86], the preceding chapter used them to formulate submodular reward functions to summarize large LiDAR datasets [83]. By combining descriptors with submodularity, or the mathematically defined property of diminishing returns, [83] leveraged several efficient submodular maximization algorithms for map summarization with no direct overlap computations. The most famous such algorithm is the greedy algorithm (Section 2.3.2), which guarantees a $(1 - \frac{1}{e})$-suboptimality bound for NP-hard cardinality constrained submodular problems [5, 6]. A recent class of streaming submodular algorithms (Section 2.3.3) provides a weaker $(1/2 - \epsilon)$-suboptimality bound (where ϵ is a small granularity constant) with better complexity scaling pertinent when optimizing over large datasets [7, 8].

This chapter leverages learned descriptors and streaming submodular maximization to develop an algorithm for online geometric map distillation called OptMap (Fig. 4.2). This chapter proposes a novel submodular function which assigns weights to elements based on their similarity to prior scans. This reduces bias towards over-sampled locations and allows for computation improvements by recognizing redundant geometric data. This submodular function is optimized by a streaming submodular maximization algorithm that uses a novel dynamic reordering approach to improve solution quality and address multiple forms of input order bias. This is achieved via a pseudo-greedy optimization which only relies on an approximation of scan-to-scan overlap, for which three variants are proposed and evaluated. The results demonstrate OptMap's ability to generate informative maps subject to flexible position, time, and size constraints with minimal computational and memory requirements.

OptMap is intended to run alongside any ROS1 or ROS2 LiDAR odometry package, and open-source code (including a lightly modified open-source LiDAR odometry algorithm [20]) is released for easy development[1]. OptMap could be adapted to other perception sensors given similar descriptors. To summarize, the contributions are:

- **Continuous Exemplar-Based Clustering**. Novel submodular function which quantifies informativeness without oversampling bias and evaluates quickly over descriptors with no point-wise operations.
- **Dynamic Reordering**. Addresses sensitivity of streaming submodular algorithms to input order by approximating the marginal values of elements to ensure stream order prioritizes scans with high expected value.
- **Scan-to-Scan Overlap Approximations**. Three approximations of scan-to-scan overlap are proposed and tested for dynamic reordering.
- **Ablation Studies**. Studies demonstrate improvements due to each contribution.
- **Open-Source Code**. Released ROS1 and ROS2 packages. OptMap is meant to be run alongside any LiDAR odometry algorithm for quick testing and development.

4.1.1 Chapter-Specific Symbols

This chapter uses the dissertation-wide notation of Section 2.1 and the submodular optimization background of Sections 2.3.1 to 2.3.3. It additionally introduces a reduced input set $R \subseteq E$ with elements r_i selected using the reduced set admission threshold $\mathcal{E}$, the Continuous Exemplar-Based Clustering (CEBC) value $\Gamma(S)$ together with the approximate pairwise marginal value $g(\cdot)$ used to compute ordering scores, the element weights w_i, and the sorted and unsorted ordering score sets Θ_S and Θ_U. Each symbol is defined at its point of use,

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

and all are listed alongside the dissertation-wide notation in the List of Symbols in the preliminary pages.

4.2 Optimal Geometric Map Distillation

Geometric map distillation is a combinatorial optimization problem which can be formulated as

$$\max_{S \subseteq \mathcal{P}, |S| \leq k} h(S), \tag{4.1}$$

where $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$ is the set of LiDAR scans collected in a mapping session, as established in Section 2.1. The cardinality constraint $|S| \leq k$ indicates that the solution cannot have more than k scans. Solutions are constrained to be sets of scans in order to reduce the dimensionality of (4.1). Equation (4.1) is computationally challenging to solve online for any function that requires point-wise operations such as overlaps used in the previous chapter, so the easier to optimize objective of this section is defined as

$$\max_{S \subseteq E, |S| \leq k} f(S), \tag{4.2}$$

where $E := \{e_1, e_2, \ldots, e_{|E|}\} \subseteq \mathbb{S}^n$ is the set of descriptors for scans in $\mathcal{P}$. The remainder of this section will define a function for geometric map *informativeness*, i.e., $f(S)$ in (4.2), which generates representative and diverse solutions when optimized. "Representative" and "diverse" are borrowed from [105], where representative solutions are those that capture the most unique geometric information, and diverse solutions are those that capture as many different environments from the input set as possible. As established in Section 2.3.5 and argued in the previous chapter, clustering functions are better suited than coverage functions to this task because they reward both properties. The clustering function developed here addresses the two remaining shortcomings of exemplar-based clustering identified in the

previous chapter: its evaluation complexity scales with the size of the ground set, and it rewards all elements equally even when those elements are repeated or redundant.

4.2.1 Continuous Exemplar-Based Clustering

This section proposes a novel variant of exemplar-based clustering called Continuous Exemplar-Based Clustering (CEBC) which finds the unbiased clustering solution inspired by an integral over a continuous descriptor trajectory. The intuition for CEBC is that the set of collected descriptors is similar to a set of unequally spaced discrete measurements from a continuous trajectory in $\mathbb{S}^n$. A first-order approximation of a path integral over this trajectory would then be a weighted sum over the discrete descriptors, where each element is weighted by the path length traveled since the previous element. The CEBC loss term is then a weighted sum of distances to the solution set given as

$$L_{cebc}(S) = \frac{1}{d_{tot}} \sum_{i=1}^{|E|} w_i d(e_i, S), \tag{4.3}$$

where d_{tot} is a normalization term and w_i is the weight for element e_i. The weight for element e_i is then the distance to the previous element so that $w_i = ||e_i - e_{i-1}||_2$. Hence the normalizing term is $d_{tot} = \sum w_i$, which normalizes the loss so that it is independent of the length of the mapping session.

(4.3) can be transformed into a submodular function similarly to exemplar-based clustering using an appropriate null element. This results in the formal definition of CEBC

$$\Gamma(S) = L_{cebc}(\mathbf{0}) - L_{cebc}(S \cup \mathbf{0}), \tag{4.4}$$

where $\mathbf{0} = [0, 0, ..., 0] \in \mathbb{R}^{n+1}$ is the origin again. Similar to (2.10), because the origin is a unit distance from all points on $\mathbb{S}^n$, $L_{cebc}(\mathbf{0}) = 1$ and $\Gamma(\emptyset) = 0$. (4.4) is hence the proposed

optimization objective in (4.2).

Proposition 5. *Let $\mathbb{S}^n$ be the set of all points on an n-hypersphere and $2^{\mathbb{S}^n}$ be its power set. The set function $\Gamma : 2^{\mathbb{S}^n} \to \mathbb{R}$ from (4.4) is a monotone non-decreasing, submodular function.*

Proof. (4.4) is shown to be monotone non-decreasing first, then submodular. Recall that $d(a, B) = \min_{b \in B} ||a - b||_2$ and $L_{cebc}(\mathbf{0}) = 1$. (4.3) is monotone non-increasing because it is a sum of minimum distances from a point to a set. Hence (4.4) can be written as $\Gamma(S) = 1 - L_{cebc}(S \cup \mathbf{0})$ and $\Gamma(S)$ is monotone non-decreasing because it is the negative of a monotone non-increasing function.

To prove submodularity via Definition 2, it must be shown that $\Gamma(x|A) \geq \Gamma(x|B)$ for sets $A \subseteq B \subseteq \mathbb{S}^n$ and $x \in \mathbb{S}^n \setminus B$. First note that $d(x, \mathbf{0}) = 1 \ \forall x \in \mathbb{S}^n$. Plugging (4.3) into (4.4) and simplifying therefore yields

$$\begin{aligned}\Gamma(A) &= \frac{1}{d_{tot}} \sum_{i=1}^{|E|} w_i \cdot 1 - \frac{1}{d_{tot}} \sum_{i=1}^{|E|} w_i d(e_i, A \cup \mathbf{0}) \\ &= 1 - \frac{1}{d_{tot}} \sum_{i=1}^{|E|} w_i d(e_i, A \cup \mathbf{0}).\end{aligned}$$

Using $\Gamma(x|A) = \Gamma(A \cup \{x\}) - \Gamma(A)$ and canceling the leading constant terms, the marginal value of x with respect to A is then

$$\Gamma(x|A) = \frac{1}{d_{tot}} \sum_{i=1}^{|E|} w_i \Big(d(e_i, A \cup \mathbf{0}) - d(e_i, A \cup \mathbf{0} \cup \{x\}) \Big).$$

The summed term $d(e_i, A \cup \mathbf{0}) - d(e_i, A \cup \mathbf{0} \cup \{x\})$ can be simplified by considering two possible cases, either e_i is closer to the new element x or $A \cup \mathbf{0}$. In the former case, the term simplifies to $d(e_i, A \cup \mathbf{0}) - ||x - e_i||_2$, and in the latter case it simplifies to $d(e_i, A \cup \mathbf{0}) -$

$d(e_i, A \cup \mathbf{0}) = 0$. Combining both cases yields

$$\Gamma(x|A) = \frac{1}{d_{tot}} \sum_{i=1}^{|E|} w_i \Big(\max(0, d(e_i, A \cup \mathbf{0}) - ||x - e_i||_2) \Big).$$

Now $d(e_i, A \cup \mathbf{0}) - ||x - e_i||_2$ is monotone non-increasing with set A because $||x - e_i||_2$ is constant and $d(e_i, A \cup \mathbf{0})$ is monotone non-increasing. Therefore $\max(0, d(e_i, A \cup \mathbf{0}) - ||x - e_i||_2) \geq \max(0, d(e_i, B \cup \mathbf{0}) - ||x - e_i||_2)$, and because the inequality holds for each element of the sum $\Gamma(x|A) \geq \Gamma(x|B)$. □

A property of (4.4) is that sequentially identical elements (e.g., LiDAR scans taken from the same position) will have low weights and therefore reduce the bias towards over-represented scans. Because clustering function evaluation complexity scales with the size of the input set, this property can also be used to reduce computation time by carefully eliminating redundant scans. This is formalized by constructing a *reduced set* (denoted R where $R \subseteq E$) which omits such elements. Algorithm 6 describes the procedure for selecting this reduced input set. The goal of the remainder of this section is to show that the value of CEBC does not change much when evaluated using a full or reduced set, and that any solution composed of elements from the full set has a close solution composed of elements from the reduced set. Selecting the reduced set is informed by a non-binding assumption that sequential descriptors are close to each other, written as $||e_i - e_{i-1}||_2 \leq \sigma$ where σ is some constant of the environment, sensor rate, and descriptor generating neural network. The value of σ cannot be known, but a parameter $\mathcal{E}$ is set which should be greater than σ for the reduced set selection. The following proofs proceed under the assumption that $\mathcal{E}$ is greater than the maximum distance between sequential descriptors σ, then a follow up corollary shows that the derived bound remains tight despite a small number of outliers.

Algorithm 6 gives the method for selecting a reduced input set with individual elements $r_i \in R$. The elements of the reduced set are selected such that sequential elements are

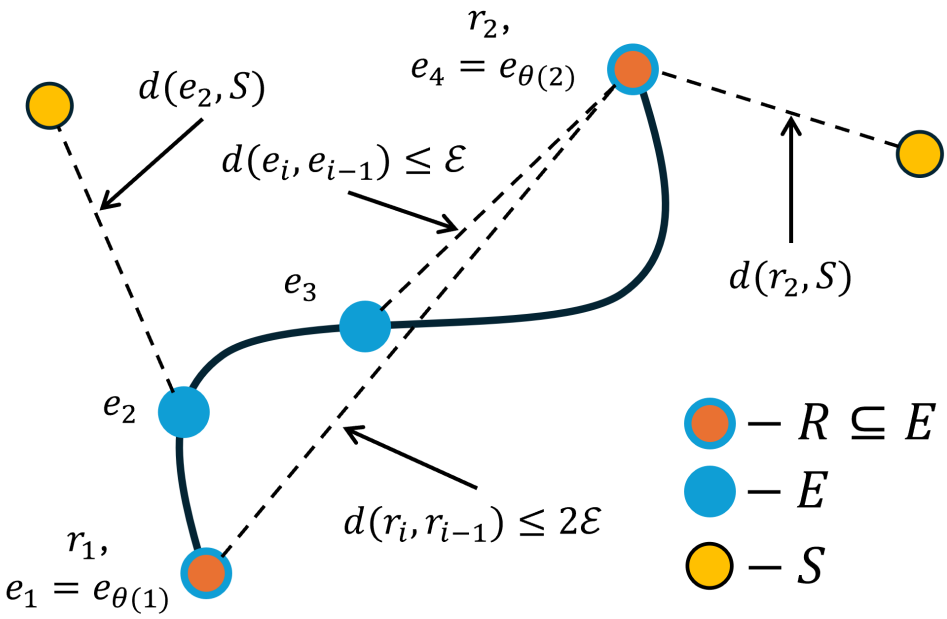


Figure 4.3: Section of a full and reduced set generated by Algorithm 6. All elements in the full set (blue outlines) are no further than $\mathcal{E}$ apart, and all elements of the reduced set (orange interior) are no further than $2\mathcal{E}$ apart. The weight assigned to r_2 is the sum of weights from its source set $\hat{w}_2 = w_2 + w_3 + w_4$. The closest element of S to e_2 and r_2 does not have to be the same element of the solution S.

separated by at least the *reduced set admission threshold* $\mathcal{E}$, a user-selected parameter which sets how much redundant geometric information is discarded. Larger values of $\mathcal{E}$ produce smaller reduced sets and faster function evaluations at the cost of a looser approximation. It is assumed that when building a reduced input set, an injective mapping function $\theta : \mathbb{Z} \to \mathbb{Z}$ is saved which translates indices from the reduced set to the corresponding index in the full set such that $r_i = e_{\theta(i)}$. It will thus be useful to formalize the relationship between reduced set elements and the full set elements which were omitted. The set of elements $\eta_i = \{e_j | j \in [\theta(i-1)+1, \theta(i-1)+2, ..., \theta(i)]\}$ is called the source set for r_i. Elements in R are assigned a weighting (weights for the reduced set are denoted as $\hat{w}$) for the purpose of evaluating (4.3) which is equal to the sum of weights in their source set, i.e., $\hat{w}_i = \sum_{j=\theta(i-1)+1}^{\theta(i)} w_j$. Intuitively, an element in the reduced set combines the value of all the elements of its source set. The following lemma proves that any reduced set element and its source set are approximately the same distance to any solution.

Algorithm 6 Reduced Input Set Selection

Given: $E, \mathcal{E}$
Output: R

$R = \{e_1\}, \hat{W} = \{0\}$, $w = 0$
for $i = \{2, 3, ..., |E|\}$ **do**
  $w = w + ||e_i - e_{i-1}||_2$
  if $w \geq \mathcal{E}$ **then**
    $R = R \cup \{e_i\}$
    $\hat{W} = \hat{W} \cup \{w\}$
    $w = 0$
  end if
end for
$R = R \cup \{e_{|E|}\}$
$\hat{W} = \hat{W} \cup \{w\}$

Lemma 1. *Consider an input set of elements E with no outliers such that $||e_i - e_{i-1}||_2 \leq \mathcal{E} \, \forall \, e_i, e_{i-1} \in E$. Also consider a corresponding reduced set R built using Algorithm 6 where each element $r_i \in R$ has the source set $\eta_i = \{e_j | j \in [\theta(i-1)+1, \theta(i-1)+2, ..., \theta(i)]\}$. Then $|d(r_i, S) - d(e_j, S)| \leq 2\mathcal{E} \, \forall \, e_j \in \eta_i$.*

Proof. First, note that the distance between r_i and its source set is bounded by $||r_i - e_j||_2 \leq 2\mathcal{E} \, \forall \, e_j \in \eta_i$. Each element of the source set is bounded by $||e_j - e_k||_2 \leq \mathcal{E} \, \forall \, e_j, e_k \in \eta_i$. Similarly, $||r_i - e_{\theta(i)-1}||_2 \leq \mathcal{E}$. So $||r_i - e_j||_2 \leq ||r_i - e_{\theta(i)-1}||_2 + ||e_{\theta(i)-1} - e_j||_2 \leq \mathcal{E} + \mathcal{E} = 2\mathcal{E}$.

Consider the elements in S closest to r_i and $e \in \eta_i$ and denote them as v and u respectively. Note that v and u do not need to be the same element as shown in Fig. 4.3. Then the distance from r_i to S can be related to e_j by $||r_i - v||_2 \leq ||r_i - u||_2 \leq ||r_i - e||_2 + ||e - u||_2 \leq 2\mathcal{E} + ||e - u||_2$. This simplifies to $||r_i - v||_2 \leq ||e - u||_2 + 2\mathcal{E}$, which can be manipulated into $|d(r_i, S) - d(e, S)| \leq 2\mathcal{E} \, \forall \, e \in \eta_i$. □

The following theorem proves that the value of a solution is arbitrarily similar for (4.4) when using E and R.

Theorem 1. *Consider an input set of elements E with no outliers such that $||e_i - e_{i-1}||_2 \leq$*

$\mathcal{E}\,\forall\, e_i, e_{i-1} \in E$. *Also consider a corresponding reduced set R built using Algorithm 6. Let the submodular function $\Gamma_E(S) = L_{cebc}(\mathbf{0}) - L_{cebc}(S \cup \mathbf{0})$ indicate the input set (either E or R) by the subscript. Then, for any solution S, $|\Gamma_E(S) - \Gamma_R(S)| \leq 2\mathcal{E}$.*

Proof. The proof follows from Lemma 1 as the value of each source set can be associated with the corresponding element of the reduced set. Using (4.3),

$$|\Gamma_E(S) - \Gamma_R(S)| = \frac{1}{d_{tot}}\Big|\sum_{i=1}^{|E|} w_i d(e_i, S) - \sum_{i=1}^{|R|} \hat{w}_i d(r_i, S)\Big|.$$

Using $\sum_{i=1}^{|E|}(\cdot) = \sum_{i=1}^{|R|}\sum_{j=\theta(i-1)+1}^{\theta(i)}(\cdot)$,

$$= \frac{1}{d_{tot}}\Big|\sum_{i=1}^{|R|}\sum_{j=\theta(i-1)+1}^{\theta(i)} w_j d(e_j, S) - \hat{w}_i d(r_i, S)\Big|$$

Factoring out $\hat{w}_i = \sum_{j=\theta(i-1)+1}^{\theta(i)} w_j$,

$$= \frac{1}{d_{tot}}\Big|\sum_{i=1}^{|R|}\sum_{j=\theta(i-1)+1}^{\theta(i)} w_j (d(e_j, S) - d(r_i, S))\Big|.$$

Finally, applying Lemma 1 yields the desired inequality

$$|\Gamma_E(S) - \Gamma_R(S)| \leq \frac{1}{d_{tot}}\Big|\sum_{i=1}^{|R|}\sum_{j=\theta(i-1)+1}^{\theta(i)} 2w_j\mathcal{E}\Big|$$
$$= \frac{2\mathcal{E}}{d_{tot}}\sum_{i=1}^{|R|}\hat{w}_i = 2\mathcal{E}.$$

□

Theorem 1 assumes no outliers (i.e., $||e_i - e_{i-1}||_2 \leq \mathcal{E}$). The following corollary demonstrates that the bound remains tight even in the presence of a small number of outliers.

Corollary 1. *Consider an input set E in which all but N sequential pairs satisfy $||e_i - e_{i-1}||_2 \leq \mathcal{E}$, and a corresponding reduced set R built using Algorithm 6. Then for any*

solution S, $|\Gamma_E(S) - \Gamma_R(S)| \leq 2\mathcal{E} + \frac{2N(2+\mathcal{E})}{d_{tot}}$.

Proof of Corollary 1. Without loss of generality, assume that the N outliers are the final N elements of the reduced set, where elements of the full input set may still exist between these outliers. Lemma 1 still applies to each of the first $|R| - N$ reduced set elements because its proof depends only on the sequential distances within that element's own source set. For an outlier r_i it does not apply, but because the maximum distance between points on the hypersphere is 2, $|d(r_i, S) - d(e_j, S)| \leq 2$ for any $e_j \in \eta_i$. The same maximum bounds an outlier's weight by $\hat{w}_i \leq 2 + \mathcal{E}$, since Algorithm 6 admits an element as soon as the accumulated distance reaches $\mathcal{E}$, so the accumulator is below $\mathcal{E}$ before the admitting step and that step is at most 2.

Starting from the third line of Theorem 1 and applying the triangle inequality to move the absolute value inside the sum over the reduced set,

$$|\Gamma_E(S) - \Gamma_R(S)| \leq \frac{1}{d_{tot}} \sum_{i=1}^{|R|} \Big| \sum_{j=\theta(i-1)+1}^{\theta(i)} w_j (d(e_j, S) - d(r_i, S)) \Big|.$$

Separating the sum into the $|R| - N$ inliers and the N outliers, then applying Lemma 1 to the former and $|d(r_i, S) - d(e_j, S)| \leq 2$ to the latter,

$$|\Gamma_E(S) - \Gamma_R(S)| \leq \frac{1}{d_{tot}} \sum_{i=1}^{|R|-N} \sum_{j=\theta(i-1)+1}^{\theta(i)} 2w_j\mathcal{E} \quad + \frac{1}{d_{tot}} \sum_{i=|R|-N+1}^{|R|} \sum_{j=\theta(i-1)+1}^{\theta(i)} 2w_j.$$

Using $\hat{w}_i = \sum_{j=\theta(i-1)+1}^{\theta(i)} w_j$ in both terms, the inlier sum is bounded by $\sum_{i=1}^{|R|-N} \hat{w}_i \leq \sum_{i=1}^{|R|} \hat{w}_i = d_{tot}$ and each of the N outlier weights by $\hat{w}_i \leq 2 + \mathcal{E}$, which gives

$$|\Gamma_E(S) - \Gamma_R(S)| \leq 2\mathcal{E} + \frac{2N(2+\mathcal{E})}{d_{tot}},$$

as desired. □

Theorem 1 and Corollary 1 establish that for any arbitrary solution, the value of (4.4) evaluated on the full or reduced set is similar. Next, any solution selected from E is shown to have a similar valued solution selected from R.

Theorem 2. *For any solution $S_E \subseteq E$ there exists a similar solution $S_R \subseteq R$ such that $\Gamma(S_R) \geq \Gamma(S_E) - \mathcal{E}$.*

Proof. By Algorithm 6, any element of the full set has a corresponding element of the reduced set no further than $\mathcal{E}$ away. This means the elements of solution S_E can each be shifted by up to $\mathcal{E}$ in order to find a corresponding solution S_R. Outliers do not need to be shifted because they are always included in both the full and reduced sets if they are further than $\mathcal{E}$ from the previous element. By optimally shifting each element of S_E to S_R, each element of E is now at most $\mathcal{E}$ further from a solution element, i.e., $d(e_i, S_R \cup \mathbf{0}) \leq d(e_i, S_E \cup \mathbf{0}) + \mathcal{E}$. This results in a tight bound for the shifted solution value which is shown by using (4.4) with the loss (4.3)

$$\Gamma(S_R) = L_{cebc}(\mathbf{0}) - L_{cebc}(S_R \cup \mathbf{0}) \geq 1 - \frac{1}{d_{tot}} \sum_{i=1}^{|E|} w_i (d(e_i, S_E \cup \mathbf{0}) + \mathcal{E}).$$

Separating the sum and using $\sum_{i=1}^{|E|} w_i = d_{tot}$,

$$\Gamma(S_R) \geq 1 - \frac{1}{d_{tot}} \sum_{i=1}^{|E|} w_i (d(e_i, S_E \cup \mathbf{0})) - \mathcal{E}.$$

Simplifying yields the inequality $\Gamma(S_R) \geq \Gamma(S_E) - \mathcal{E}$. □

Intuitively, Theorem 2 says the average distance from an input set to solution set cannot increase by more than $\mathcal{E}$ and thus any solution composed of elements from E has an arbitrarily similar solution composed of elements from R. To summarize, CEBC is well suited for geometric map distillation because it selects the solution which best represents the input

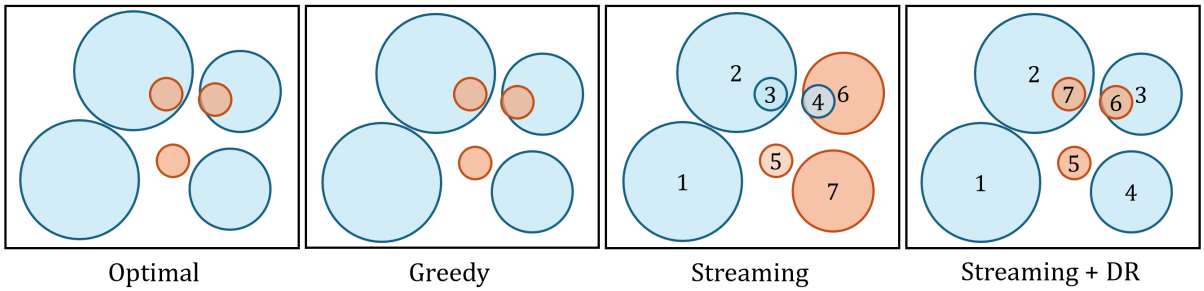


Figure 4.4: Sensor selection with a four sensor limit. Solution set in blue, other sensors in orange. From left to right: optimal solution, greedy solution which does not consider input order, streaming solution with arbitrary input order (indicated by numbers in sensor footprints), and the streaming solution with dynamic reordering (DR) which modifies the input order according to approximate element values. Sensors 3 and 4 selected in the streaming solution due to input order bias. Assumed that OPT is known *a priori* and only one solution is maintained by the streaming variants.

set, and avoids bias towards over-sampled positions. Importantly, CEBC is submodular, which means near-optimal solutions can be found in polynomial time.

4.3 Dynamically Reordered Streaming Submodular Maximization

This section presents a streaming submodular algorithm for geometric map distillation which efficiently addresses input order bias. Input order bias is defined for streaming submodular maximization as the increased likelihood of solutions containing elements solely due to their position in the stream of the input set. Bias is introduced to streaming algorithms because of the need to make decisions to include elements before evaluating the entire stream, i.e., using limited information, and the fact that streaming algorithms must use relaxed admission thresholds (such as (2.6)). Figure 4.4 shows an example of this bias, where suboptimal elements are included in the baseline streaming solution to a simple area coverage problem after (i) the streaming input threshold becomes negative after exceeding $OPT/2$, and (ii) less valuable elements of the input set are given arbitrary preference by being evaluated first.

This section therefore proposes a dynamically reordered streaming submodular algorithm to account for this input order bias. This is accomplished by using approximate marginal

values of unchecked elements to dynamically reorder the stream such that the next element always has high expected value. The dynamically reordered streaming submodular algorithm is presented first under the assumption that a marginal value approximation function for CEBC exists. The section concludes by defining three functions that approximate the marginal value of (4.4) using pairwise information.

4.3.1 Streaming Submodular Dynamic Reordering

This section proposes *dynamic reordering* as a solution to input order bias in streaming submodular algorithms. The goal of dynamic reordering is to evaluate elements with high expected value earlier in the stream where they are more likely to be included in the generated solution. This is achieved by maintaining an approximate marginal value for all unevaluated elements in the stream in the form of an *ordering score*. Each time an element is added to a solution, the ordering scores of all remaining unevaluated elements are discounted by an approximate pairwise marginal value function $g : \mathbb{R} \to [0, 1]$ which outputs the approximate degree of mutual value between any two elements. The input to this approximate pairwise marginal value is a single distance value between the elements, and Section 4.3.2 defines three possible approximations. If g were the true pairwise marginal value, and only a single solution set were maintained, the ordering score would be equal to the optimistic marginal value of Definition 3 given by (2.7), and the algorithm would reduce to a greedy optimization of (2.7) with (2.6) confirming elements. Because multiple solutions are maintained in parallel and the stream order must be the same for all solutions, ordering scores are averaged across all solutions. Deciding which element to present next at each step in the stream introduces additional computational burden, but this can be mitigated by limiting the set of elements which can be searched over at each step.

The dynamically reordered streaming submodular algorithm for optimizing CEBC (4.4) is given by Algorithm 7. Intuitively, Algorithm 7 closely mirrors Sieve-Streaming (Algorithm 2)

Algorithm 7 Dynamically Reordered (DR) Streaming Submodular Maximization

Given: $E, \underline{OPT}, \overline{OPT}, k, F$
Output: $\arg\max_{S_v} f(S_v)$

1: // Create OPT guesses and solution sets
2: $O = \{(1+\epsilon)^i | i \in \mathbb{Z}, \underline{OPT} \leq (1+\epsilon)^i \leq \overline{OPT}\}$
3: **for** $v \in O$ **do**
4: $\quad S_v = \emptyset$
5: **end for**
6: // Create partitions of indices and order scores
7: $\Theta_S \leftarrow ones(Fk)$
8: $\Theta_U \leftarrow ones(|E| - Fk)$
9: // Make single pass over the input set
10: **for** $a \in \{1, 2, ..., |E|\}$ **do**
11: $\quad i = \arg\max_j \Theta_S(e_j)$
12: $\quad \Theta_S.remove(e_i)$
13: $\quad \Theta_S.append(\Theta_U.first)$
14: $\quad \Theta_U.removeFirst()$
15: $\quad$ // Parallel evaluation step over all solutions
16: $\quad$ **for** $v \in O$ **do**
17: $\quad\quad ElementEvaluation(i, E, \Theta_S, \Theta_U, S_v)$
18: $\quad$ **end for**
19: **end for**

with additional steps for efficiently tracking and updating the score of CEBC and the ordering scores. Multiple solutions are initialized on lines 2-5 identically to Sieve-Streaming. Lines 7-8 include the first deviation, where a sorted and unsorted set are initialized. Ordering scores are kept in two sets, the sorted set Θ_S and unsorted set Θ_U. The sorted set contains at most Fk elements, where F is a parameter and k is the solution cardinality constraint. The ordering score for element e_i is denoted as $\Theta_S(e_i)$ or $\Theta_U(e_i)$ depending on which set it belongs to. Note that the input set order is randomized before optimization. The single evaluation pass occurs on lines 10-19, where lines 11-14 take the highest scored element from Θ_S for evaluation and move an element from the unsorted to the sorted set as a result. The purpose of the sorted set is to make the arg max search on line 11 cheaper, as it only needs to search a set of size Fk rather than a set of size $|E|$.

Algorithm 8 describes the steps for evaluating elements and updating ordering scores.

Algorithm 8 Element Evaluation & Ordering Score Update

Given: $i, E, \Theta_S, \Theta_U, S_v$

1: $\Theta_{S,n} = \Theta_S, \Theta_{U,n} = \Theta_U$
2: $\Gamma(e_i|S_v) = 0$
3: // Find marginal value of the element
4: **for** $j \in \{1, 2, ..., |E|\}$ **do**
5: $\quad d = ||e_i - e_j||_2$
6: $\quad$ **if** $d < d(e_j, S_v)$ **then**
7: $\quad\quad$ // Update ordering scores
8: $\quad\quad \Gamma(e_i|S_v) += \frac{w_j}{d_{tot}}(d(e_j, S_v) - d)$
9: $\quad\quad$ **if** $e_j \in \Theta_S$ **then**
10: $\quad\quad\quad \Theta_{S,n}(e_j) += \frac{1}{|O|}(g(d) - g(d(e_j, S_v)))$
11: $\quad\quad$ **else if** $e_j \in \Theta_U$ **then**
12: $\quad\quad\quad \Theta_{U,n}(e_j) += \frac{1}{|O|}(g(d) - g(d(e_j, S_v)))$
13: $\quad\quad$ **end if**
14: $\quad$ **end if**
15: **end for**
16: // Check streaming marginal value threshold
17: **if** $|S_v| < k \& \Gamma(e_i|S_v) \geq \frac{v/2 - \Gamma(S_v)}{k - |S_v|}$ **then**
18: $\quad S_v = S_v \cup e_i$
19: $\quad \Theta_S = \Theta_{S,n}, \Theta_U = \Theta_{U,n}$
20: **end if**

It begins by making a copy of the sorted and unsorted sets (line 1) which will be saved if the element is included in the solution. The marginal value $\Gamma(e_i|S_v)$ of the element is found incrementally during the evaluation pass (lines 3-15). CEBC measures the weighted average distance from each element of the input set to the solution set, so the marginal value of e_i is determined by finding the elements in the input set $e_j \in E$ such that $||e_i - e_j||_2 < d(e_j, S)$ (lines 5-6). For each of these elements, the ordering scores for all elements in the sorted and unsorted sets are updated according to the approximation function $g(\cdot)$ (lines 10, 12). The reason dynamic reordering is efficient when optimizing CEBC despite being a pseudo-greedy process is that evaluating CEBC already requires a loop over all the elements of the input set, so the ordering scores can be computed cheaply alongside this already required loop. $g(\cdot)$ is assumed to be monotone-increasing such that $d < d(e_j, S_v)$ means $g(d) < g(d(e_j, S_v))$. This means that lines 10 and 12 decrease the ordering score of e_j because the new element

e_i has a higher degree of mutual information with e_j.

4.3.2 Approximate Pairwise Information for CEBC

Central to the efficiency of dynamic reordering is the ability to approximate the relative marginal value of an element with respect to a given solution. This is done in Algorithm 8 by evaluating approximate pairwise marginal values via the function $g(\cdot)$. Recall from Section 2.3.4 that the pairwise marginal value is the approximate marginal value of an element using pairwise information, or the mutual information between just any two elements in a solution. In this subsection, three such approximations are derived. The first uses scan poses as a proxy for overlap, the second uses an area coverage relaxation of (4.4), and the third combines the previous two.

A position-based approximation uses the distance between scan poses as a proxy for the overlap between them. Given the position of all elements in the solution set as $X_S = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{|S|}]$ where $\mathbf{x}_i \in \mathbb{R}^3$, the pose-based approximate pairwise marginal value of an element with position $\mathbf{x}_e$ is

$$g_{pose}(d(\mathbf{x}_e, X_S)) = 1 - \max\left(0, \frac{-\log(d(\mathbf{x}_e, X_S))}{\alpha} + 0.1\right), \tag{4.5}$$

where $d(\mathbf{x}_e, X_S)$ is the distance from the element to the nearest solution pose and α is a tunable radius parameter. Note that if (4.5) is used for dynamic reordering, the inputs on lines 8 and 10 of Algorithm 8 should use poses rather than descriptors. The log function introduces a decay which is intended to mimic the overlapping area of 2D sensors with circular footprints. This heuristic favors spatially well-separated solutions, which is effective in environments where coverage correlates with position such as outdoor mapping. However, it can break down when small positional differences correspond to large changes in the environment (e.g., scans taken from either side of a wall or between building floors).

A descriptor-coverage approximation is formulated by converting (4.4) into an area coverage function (similar to (2.8)). For the reader's convenience, recall that $\mathbb{S}^n$ is the hypersphere on which all descriptors lie. The surface area of a subset of the hypersphere is denoted by the function $\mathcal{A} : \mathbb{S}^n \to \mathbb{R}$ where the surface area for any $A \subseteq \mathbb{S}^n$ is defined as $\int_A 1 d\boldsymbol{\sigma}$. Also recall that (2.8) is the surface area of a set of spherical caps centered at the solution elements (descriptors). It is convenient to imagine $\mathbb{S}^n$ as a plane or a familiar 2-sphere, as much of the same intuition holds despite the high dimensionality. To convert (4.4) into a coverage function, the average distance from every point on $\mathbb{S}^n$ to S is used instead of the average distance from every point in E to S. Converting the weighted sum over E from (4.3) into an integral over the surface of $\mathbb{S}^n$ yields the following coverage loss function

$$L_{cebc} \approx L_{cover}(S) = \frac{1}{\mathcal{A}(\mathbb{S}^n)} \int_{\mathbb{S}^n} d(x, S) d\boldsymbol{\sigma}, \tag{4.6}$$

where $\mathcal{A}(\mathbb{S}^n)$ is again the surface area of the hypersphere. Converting this loss term into a submodular function is similar to exemplar-based clustering and CEBC

$$f_{cover}(S) = L_{cover}(\mathbf{0}) - L_{cover}(S \cup \mathbf{0}), \tag{4.7}$$

using the origin $\mathbf{0} = [0, 0, ..., 0] \in \mathbb{R}^{n+1}$ as the null element.

Evaluating (4.7) is computationally expensive because it requires integrating over the surface of a high-dimensional hypersphere. Additionally, the fraction of the hypersphere covered by a spherical cap is vanishingly small [106], so (4.6) will equal 1 even for very large solutions. Both of these problems can be solved by separating the hypersphere into regions that are either close to or far from elements of a solution and integrating over these regions separately. The region near S is defined as the union of all unit radius spherical caps centered at its descriptors, denoted by $\mathcal{C}(S) := \{x \in \mathbb{S}^n | \ \exists e \in S, ||e - x||_2 \leq 1\}$. The complement of

$\mathcal{C}(S)$ is defined accordingly as $\mathcal{C}^-(S) := \{x \in \mathbb{S}^n | \ ||e - x||_2 > 1, \forall e \in S\}$. Because $\mathcal{C}$ and $\mathcal{C}^-$ are complements, $\mathcal{A}(\mathbb{S}^n) = \mathcal{A}(\mathcal{C}) + \mathcal{A}(\mathcal{C}^-)$.

An equivalent function for maximizing (4.7) that only integrates over $\mathcal{C}$ is

$$\Gamma_C(S) = \mathcal{A}(\mathcal{C}) - \int_{\mathcal{C}} d(x, S) d\boldsymbol{\sigma}. \tag{4.8}$$

The following theorem shows that (4.7) achieves maximum value with the optimal solution of (4.8).

Theorem 3. *The elements that maximize (4.7) are equivalent to the elements that maximize (4.8).*

Proof. Note that $\mathbf{0}$ is always unit distance from any surface element of the hypersphere, so evaluating (4.6) with only the null element is $L_{cover}(\mathbf{0}) = \frac{1}{\mathcal{A}(\mathbb{S}^n)} \int_{\mathbb{S}^n} 1 d\boldsymbol{\sigma} = 1$. By plugging in (4.6) and using $L_{cover}(\mathbf{0}) = 1$, (4.7) can be rewritten as

$$\begin{aligned} f_{cover}(S) &= 1 - \frac{1}{\mathcal{A}(\mathbb{S}^n)} \int_{\mathbb{S}^n} d(x, S \cup \mathbf{0}) d\boldsymbol{\sigma} \\ &= \frac{1}{\mathcal{A}(\mathbb{S}^n)} (\mathcal{A}(\mathbb{S}^n) - \int_{\mathbb{S}^n} d(x, S \cup \mathbf{0}) d\boldsymbol{\sigma}). \end{aligned}$$

Because $\mathcal{C}$ is the portion of $\mathbb{S}^n$ near to S, the integral can be evaluated as the sum of integrals over components $\mathcal{C}$ and $\mathcal{C}^-$ so that $\int_{\mathbb{S}^n} d(x, S \cup \mathbf{0}) d\boldsymbol{\sigma} = \int_{\mathcal{C}^-} 1 d\boldsymbol{\sigma} + \int_{\mathcal{C}} d(x, S) d\boldsymbol{\sigma}$,

$$f_{cover}(S) = \frac{1}{\mathcal{A}(\mathbb{S}^n)} \left(\mathcal{A}(\mathbb{S}^n) - \int_{\mathcal{C}^-} 1 d\boldsymbol{\sigma} - \int_{\mathcal{C}} d(x, S) d\boldsymbol{\sigma} \right).$$

Recalling $\int_{\mathcal{C}^-} 1 d\boldsymbol{\sigma} = \mathcal{A}(\mathcal{C}^-)$ and $\mathcal{A}(\mathcal{C}) = \mathcal{A}(\mathbb{S}^n) - \mathcal{A}(\mathcal{C}^-)$,

$$f_{cover}(S) = \frac{1}{\mathcal{A}(\mathbb{S}^n)} \left(\mathcal{A}(\mathcal{C}) - \int_{\mathcal{C}} d(x, S) d\boldsymbol{\sigma} \right),$$

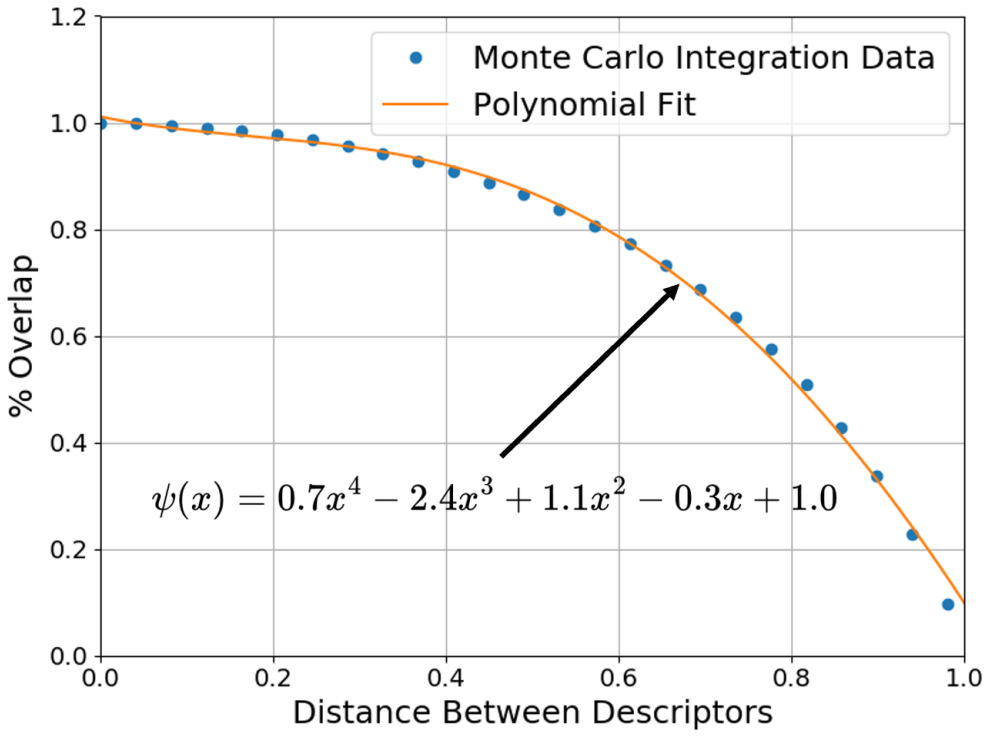


Figure 4.5: Mutual information with respect to (4.8) as percent overlap between spherical caps. Overlap percent is obtained as a function of the distance between descriptors using a dense Monte Carlo integration over the surface of $\mathbb{S}^n$, with a fourth-order polynomial function $\psi(x)$.

which, using (4.8), simplifies to

$$f_{cover}(S) = \frac{1}{\mathcal{A}(\mathbb{S}^n)} \Gamma_C(S).$$

Taking $\arg\max$ over S of both sides yields

$$\arg\max_S f_{cover}(S) = \arg\max_S \Gamma_C(S)$$

where the property that scaling a cost function by a constant does not change the optimal solution has been used. Hence, the elements that maximize (4.7) also maximize (4.8). □

Computing the marginal value of (4.8) remains difficult due to integrating over multiple spherical caps, but can be simplified using pairwise information. Note that any solution with

one element will have constant value as any two solutions could be related by an isometric rotation,

$$\Gamma_C(\{e\}) = \mathcal{A}(\mathcal{C}) - \int_{\mathcal{C}} ||x - e||_2 d\boldsymbol{\sigma} = c. \tag{4.9}$$

The goal is to approximate the degree of mutual value between two elements e_i and e_j as $g(||e_i - e_j||_2)$. Overlap between any two spherical caps can be computed offline for various distances between cap centers, and then approximated online using the polynomial given as $\psi(x)$ in Fig. 4.5. The mutual value between elements is then one minus this overlap. Spherical caps have minimal overlap if their centers are further than 1.1 apart, so the pairwise marginal value is

$$g_{desc}(d(e_i, e_j)) = \begin{cases} 1 - \psi(d(e_i, e_j)) & \text{if } d(e_i, e_j) < 1.1 \\ 1 & \text{otherwise,} \end{cases} \tag{4.10}$$

where $d(e_i, e_j)$ is used as shorthand for $||e_i - e_j||_2$. The single element score constant c is dropped because only the relative value of ordering scores is required. This approximation works well when the descriptor-generating neural network produces accurate descriptors for the mapped environment. It can break down when operating in very noisy environments (e.g., forests) or in environments that are underrepresented in the network training data.

A combined approximation leverages the complementary strengths of the pose- and descriptor-based approximations. Since only the relative value of ordering scores is required, the two can be summed to form a third approximation

$$g_{comb} = g_{pose} + g_{desc}. \tag{4.11}$$

Dynamic reordering is therefore a *pseudo-greedy* approach, where at each step the greedy choice is made with respect to one of the three pairwise information approximations de-

rived above, applied through (2.7), rather than the true marginal value of (4.4). Because these approximations only set the order of the stream and never the decision to admit an element, dynamic reordering retains both the suboptimality guarantee and the favorable computational requirements of the underlying streaming algorithm.

4.4 Algorithmic Implementation

OptMap inherits two implementation details from the previous chapter, described in Section 3.4.1, and applies both without modification. The *a priori* solution bounds use the same construction: the upper bound is $\overline{OPT} = 1$, since (4.4) is maximized when the solution set contains the entire input set and the second loss term vanishes, and the lower bound is obtained by evaluating the evenly spaced heuristic solution of Algorithm 5 on (4.4), which requires one evaluation pass. The only difference in this chapter is that the path length accumulated on lines 1-4 of Algorithm 5 is computed alongside the reduced set selection of Algorithm 6, so it costs nothing additional. Position and time constraints are likewise applied to the input set exactly as described in Section 3.4.1, and are what allow OptMap to distill a specified portion of a mapping session rather than the whole of it. An example is shown in Fig. 4.1b, where position and time constraints customize a distilled map for a downstream application.

4.5 OptMap Results

This section provides results that demonstrate OptMap's theoretical contributions, expected distilled map quality, and computation performance. The first experiment investigates possible map distillation reward functions with a further ablation study on the reduced set selection threshold. The second result demonstrates the value of dynamic reordering towards reducing input order bias. The final result exhibits the low computation requirements

of OptMap in large map distillation problems and highlights the benefits of using tight *a priori* solution bounds. Datasets include custom LiDAR mapping sessions collected on the UCLA campus using 32-beam Ouster OS0 and OS1 sensors (Sculpture Garden, Courtyard, and Lab), and three custom datasets from the Army Research Laboratory collected using a 128-beam Ouster OS1 sensor. Open-source mapping sessions from the Newer College [1], Oxford Spires [115], and Semantic KITTI [2] datasets are also used for evaluation.

Dense comparison maps used for Section 4.5.1 and Section 4.5.2 are generated by combining 250 scans then applying a 0.5 m voxel filter to ensure uniform point density. The reduced set selection threshold is $\mathcal{E} = 0.25$, sorted set size factor is $F = 10$, and distance radius for the pose-based CEBC approximation function is $\alpha = 15$ across all results unless otherwise noted. The descriptor neural network is trained on UCLA campus datasets not used in any results for 30 epochs. A modified variant of [20] with loop closures and custom OptMap interfaces provides LiDAR scans with labeled poses and motion correction. An odometry-only variant without loop closures is available open-source[2]. All results were generated by real-time data playback using an i7-1165G7 CPU and NVIDIA 2060 GPU for descriptor generation.

4.5.1 Geometric Map Distillation Reward Function Evaluation

The first experiment seeks to find the best function for quantifying informativeness in geometric map distillation and studies the effects of the reduced set selection threshold parameter $\mathcal{E}$. Three possible variants are considered: a direct implementation of descriptor coverage (2.8), Exemplar-Based Clustering (EBC) (2.10), and Continuous Exemplar-Based Clustering (CEBC, Ours) (4.4). CEBC was further split into three variants with reduced set thresholds of $\mathcal{E} = 0.05$, 0.25, and 0.5. Each function was optimized using the greedy algorithm to iso-

[2]Modified version of DLIO [20] for OptMap: https://github.com/vectr-ucla/direct_lidar_inertial_odometry/tree/feature/optmap

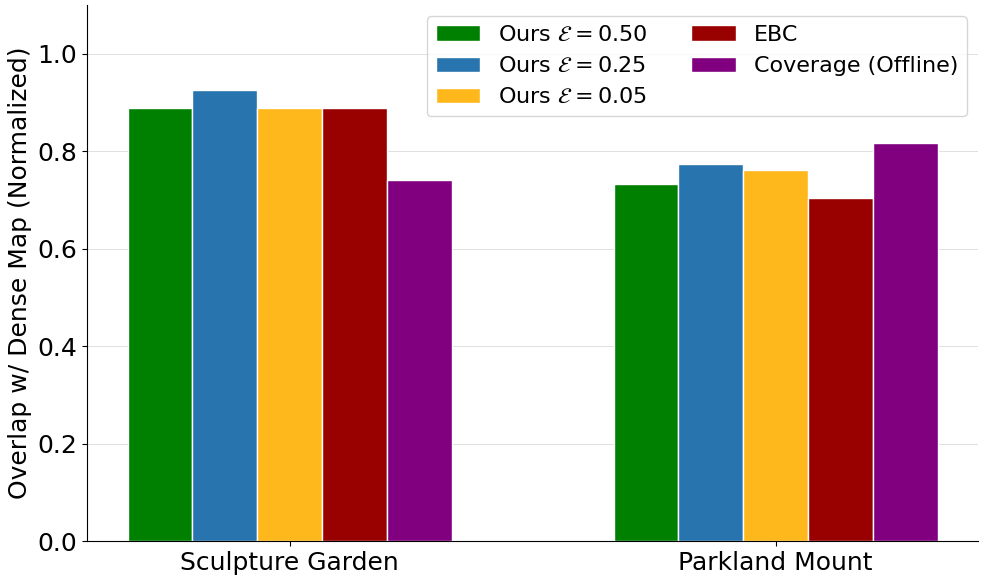


Figure 4.6: Normalized percent overlap of distilled maps for reward functions when maximized via the greedy algorithm, higher indicates better performance. Sculpture Garden is a custom dataset in a semi-forested environment, and Parkland Mount is from [1]. CEBC (Ours) is evaluated using reduced input sets constructed with admission thresholds of $\mathcal{E} = 0.05, 0.25$, and 0.5.

late results from any bias present in streaming submodular optimization. The datasets used were the semi-urban outdoor Sculpture Garden and Parkland Mount from [1]. To measure informativeness, the overlap between each function's output map and a dense comparison map is used. The number of scans used in output maps was increased until any output exceeded 50% overlap. The overlap percent for each output was then normalized with an optimal result generated by greedily selecting scans that overlap with the dense map.

Results from the first experiment are seen in Fig. 4.6 where CEBC performs near the optimal across both datasets and with multiple reduced set thresholds, EBC performs well on Sculpture Garden, and coverage performs well on Parkland Mount. Normalized overlap percentages for Sculpture Garden were 89%, 93%, 89%, 89%, and 74% for CEBC with $\mathcal{E} = 0.5$, 0.25, 0.05, EBC, and coverage respectively. Normalized overlap for Parkland Mount was 73%, 78%, 76%, 70%, and 82% for the five variants respectively. Increasing the reduced set threshold from $\mathcal{E} = 0.25$ to 0.5 reduced overlap, suggesting that higher thresholds discard useful information. However, CEBC with all reduced set thresholds equaled or exceeded

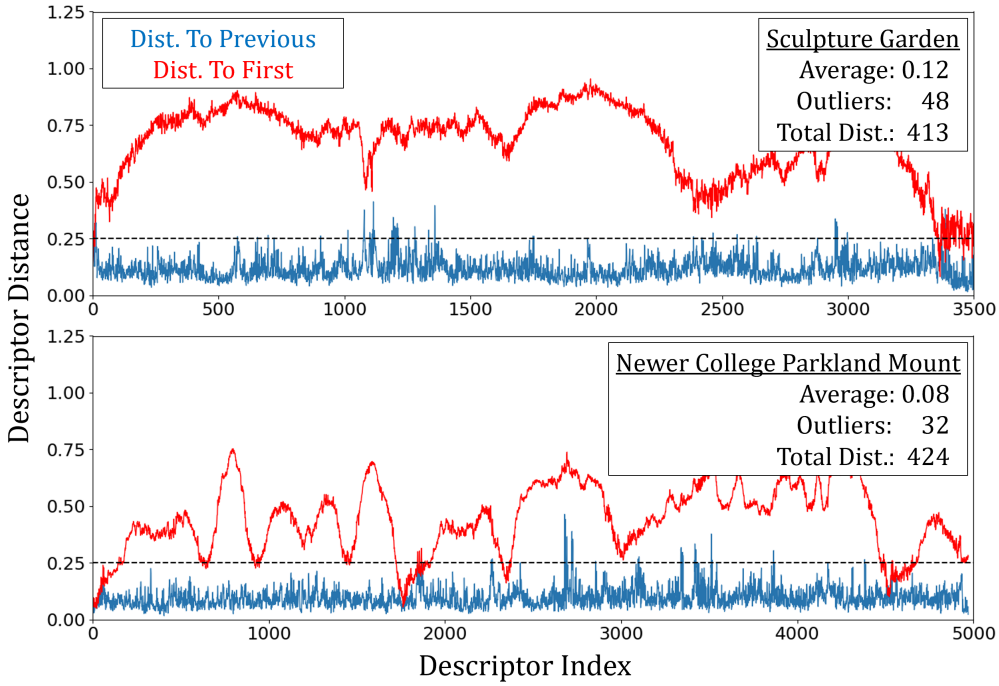


Figure 4.7: Descriptor distance between sequential descriptors in Sculpture Garden (top) and Newer College Dataset [1] Parkland Mount (bottom) datasets. Outliers is the number of sequential distances > 0.25 and total distance is the sum of all sequential distances. The Euclidean distance to the first descriptor is shown in red as a reference.

EBC, indicating that CEBC's sequential weighting helped quantify informativeness. CEBC also compares favorably with the coverage function which performed well on Parkland Mount but poorly on Sculpture Garden. This inconsistency arose because the coverage function only rewards self-dissimilar solutions, making it less robust to outlier descriptors. This resulted in lower overlap on the Sculpture Garden dataset which had more sequential descriptors further than 0.25 apart (48 vs. 32, Fig. 4.7).

Fig. 4.7 shows the sequential distance between scan descriptors for the two datasets used in this result. Using the default reduced set threshold $\mathcal{E} = 0.25$, a relatively low number of outliers are present for each dataset, 48 for Sculpture Garden and 32 for Parkland Mount. Given that Parkland Mount is a longer dataset than Sculpture Garden, the ratio of outliers is higher for Sculpture Garden, which is evident in the resulting function comparison. Note that the sequential distance between descriptors is a function of multiple variables including

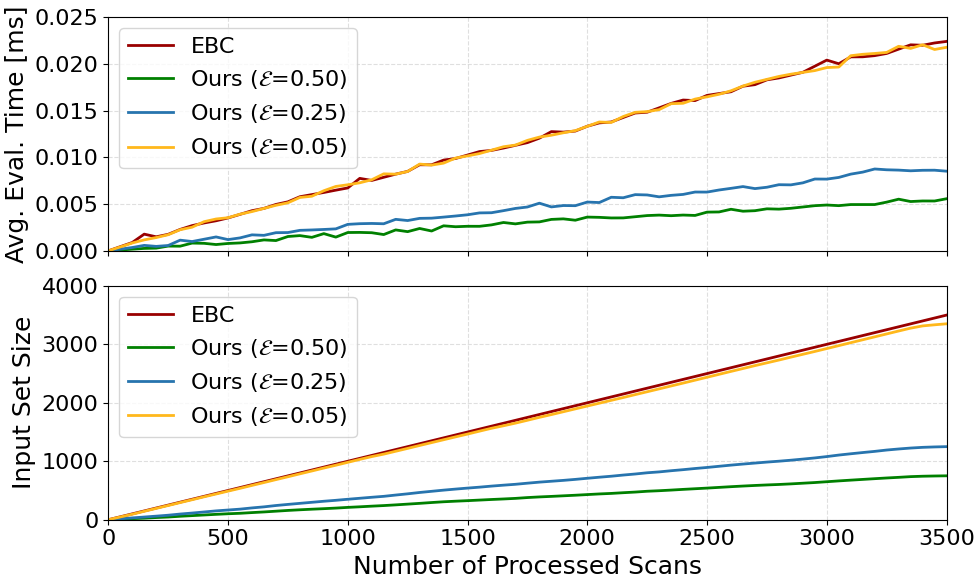


Figure 4.8: Average function evaluation time and input set size for Sculpture Garden dataset. Ours is CEBC. Evaluation time decreases with input set size. Multiple reduced set distance thresholds $\mathcal{E}$ are used to show that computation time approaches EBC as the reduced and full sets converge.

the environment, sensor rate, and sensor speed, but the tested datasets have no sudden changes that would indicate a true jump in sequential descriptor distance is warranted. Instead, these outliers are likely due to a lack of training data used for the descriptor neural network, indicating that more training could reduce the number of outliers and improve the theoretical bounds which rely on this number.

CEBC also uses a reduced input set to reduce function evaluation time as shown in Fig. 4.8. For CEBC with $\mathcal{E} = 0.5, 0.25, 0.05$ and EBC the final input set size for the Sculpture Garden was 753, 1256, 3376, and 3575 scans respectively. The final average evaluation time for each variant was 0.0057, 0.0087, 0.0219, and 0.0228 milliseconds showing a strong correlation between input set size and function evaluation time. Compared to EBC, CEBC with the best threshold from Fig. 4.6, $\mathcal{E} = 0.25$, saw a 62% reduction in average evaluation time and 65% reduction in input set size. Note that the greedy algorithm requires $O(kN)$ evaluations to find a solution, so a solution with 25 elements would require 2038 ms of optimization for EBC and 273 ms for CEBC with $\mathcal{E} = 0.25$.

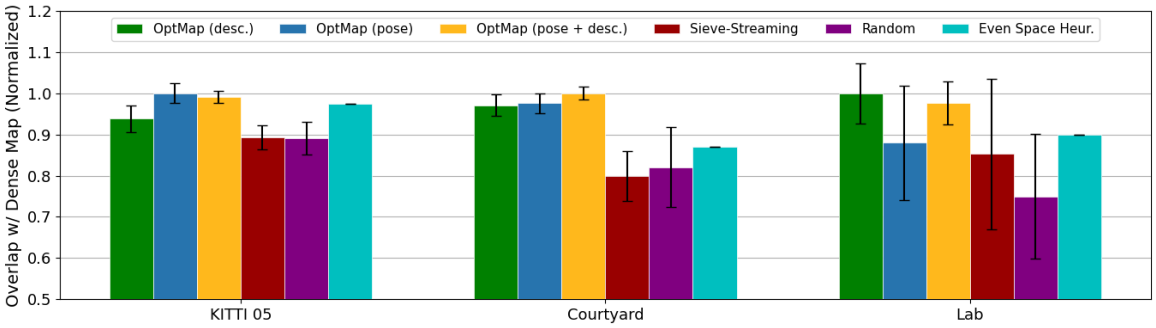


Figure 4.9: Percent of overlap from dense map to output map for ablation test on KITTI 05 [2], Courtyard, and Lab datasets shown in Fig. 4.10. Results are normalized by the highest scoring method for each dataset. KITTI 05 output maps made with fifty scans, Courtyard maps made with twelve scans, and Lab maps made with eight scans.

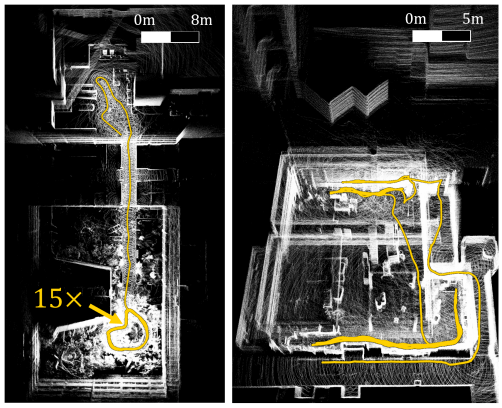


Figure 4.10: Dense maps used for Fig. 4.9. Each map is made from 250 scans from a 32-beam Ouster OS0, with the sensor trajectory indicated by the yellow path. Left is Courtyard which entails fifteen loops of an outdoor courtyard followed by a single mapping pass through tight corridors and an adjacent alley in the figure top. Right is Lab which contains multiple mapping passes capturing the interior of a large lab space and a single pass capturing the outdoor alley and nearby tight hallways.

4.5.2 Dynamic Reordering Effectiveness Ablation Study

The second experiment highlights the benefits of dynamic reordering and compares the relative strengths and weaknesses of multiple approximation functions used to generate ordering scores. The three approximations presented in Section 4.3.2 are evaluated: the pose approximation (4.5), the descriptor approximation (4.10), and their combination (4.11). An ablation study compares OptMap using these three approximation functions with the baseline Sieve-Streaming algorithm, random element selection, and the heuristic solution with

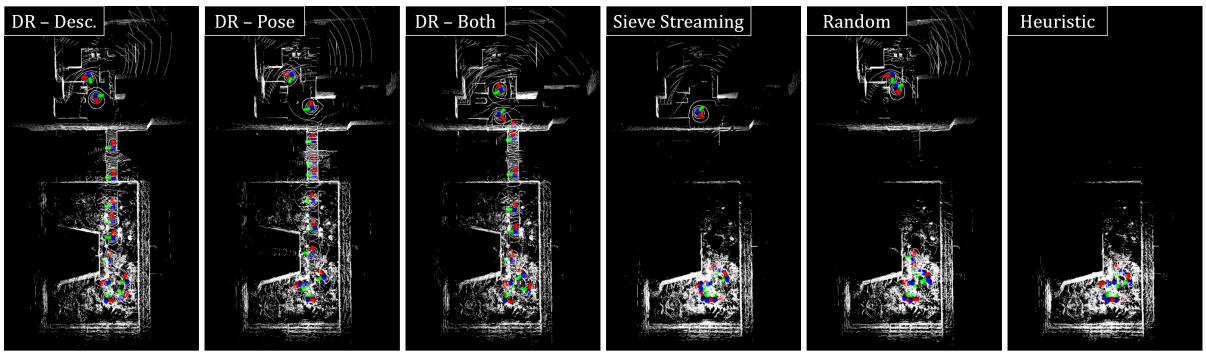


Figure 4.11: Sample output maps using different methods from Fig. 4.9. Methods from left to right are OptMap with descriptor-based dynamic reordering, OptMap with pose-based dynamic reordering, OptMap with both descriptor and pose dynamic reordering, Sieve-Streaming, random selection, and the evenly spaced heuristic. Dynamic reordering methods capture the entire map where the other three methods only select scans from the loop seen at the bottom of each map.

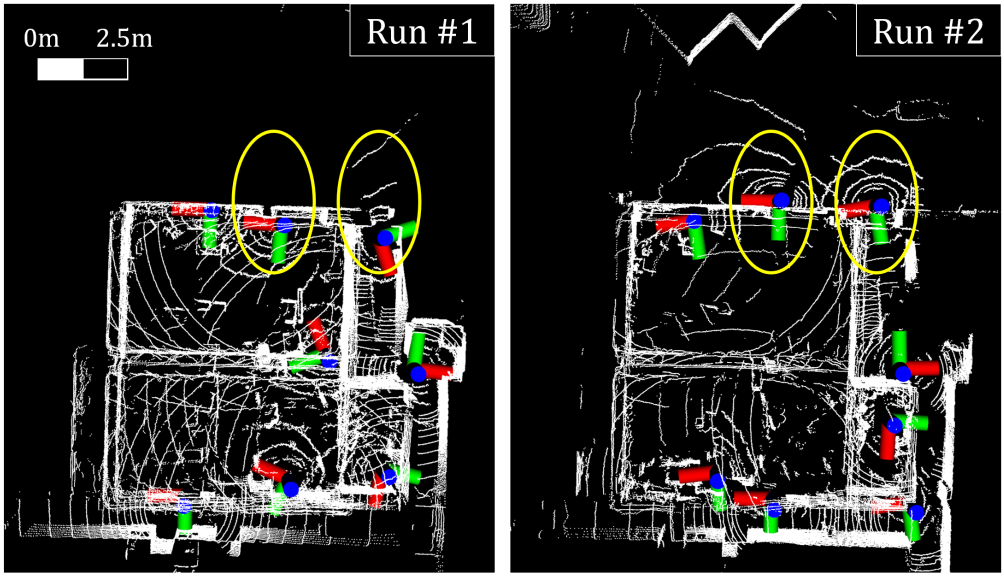


Figure 4.12: The pose-based dynamic reordering is unable to accurately select the highest value elements in cases where nearby scans capture significantly different environments. These two maps from the Lab dataset (Fig. 4.10) were generated using the same pose-based approximation dynamic reordering method, and demonstrate the large standard deviation from the pose-based dynamic reordering as scans are arbitrarily selected from one side of the wall (yellow ovals) leading to occasional suboptimal results.

even spacing described by Algorithm 5 of the previous chapter. Sieve-Streaming is OptMap with no dynamic reordering but has identical initial solution value bounds. All methods use input sets filtered by Algorithm 6.

Each optimization method generated ten distilled maps on KITTI 05 [2] and two custom datasets collected at UCLA with dense maps shown in Fig. 4.10. KITTI 05 is a medium

length dataset (2736 scans) collected on a fast moving car with no significant bias caused by remapping. Courtyard (Fig. 4.10 left) introduces a bias to the input set by repeating a small loop fifteen times before making a single mapping pass away from the loop. The bias in the input set is the result of the loop being significantly overrepresented, with the loop accounting for 86% of collected scans but only 31% of the unique distance traveled. Finally, Lab (Fig. 4.10 right) introduces input set bias by mapping the inside south and north walls of a lab space with four passes each. Scans in the Lab dataset were collected on either side of a thin wall and garage door to stress the pose approximation. The UCLA datasets are meant to show the value of dynamic reordering by biasing the input sets. Although this bias is uncommon in open-source datasets, it is not uncommon in practical applications where mapping sessions revisit the same location several times or remain stationary. The output map size for each dataset is hand-picked, with fifty scans used for KITTI 05, twelve scans used for Courtyard, and eight scans used for Lab.

The average overlap percent and its standard deviation for all three datasets are presented in Fig. 4.9. The results are normalized by the highest performing method for each dataset. All methods are expected to perform well on KITTI 05, but the fast moving car and relatively self-similar street environment stressed the descriptor-based approximation method. The descriptor, pose, and combined dynamic reordering methods have normalized average overlaps of 93.8%, 100%, and 99.0%, with normalized standard deviations of 3.2%, 2.4%, and 1.5% respectively. Sieve-Streaming, random, and the evenly spaced heuristic had normalized average overlaps of 89.3%, 89.1%, and 97.4% and standard deviations of 2.9% and 3.9% (the heuristic is deterministic and has no variance). Note that the 1/2-suboptimality bound is reached using only 1-3 scans on the UCLA datasets and 5 scans on KITTI 05, so Sieve-Streaming behaves similarly to random selection in each case.

Courtyard is meant to demonstrate the strengths of dynamic reordering, which are emphasized by the visual results presented in Fig. 4.11. The first three panels of Fig. 4.11 are

representative maps from the descriptor, pose, and combined dynamic reordering methods where multiple scans and sufficient coverage of the single pass are present for all three methods. The final three panels show representative maps from Sieve-Streaming, random, and the evenly spaced heuristic; each contains very few scans and minimal coverage from the single pass. The qualitative results match the overlap and standard deviations presented in Fig. 4.9. The average normalized overlaps are 97.1%, 97.6%, and 100% for the descriptor, pose, and combined approximation dynamic reordering, and the average normalized standard deviations are 2.5%, 2.4%, and 1.6%. In contrast, the average normalized overlaps for Sieve-Streaming, random, and the evenly spaced heuristic are 80.0%, 82.0%, and 87.0%. The average normalized standard deviations for Sieve-Streaming and random are 6.0% and 9.7%. Both Fig. 4.11 and Fig. 4.9 show that the value of dynamic reordering increases when an input set contains redundantly mapped areas. In the context of field operations, this is most apparent in providing summary maps when a robot has patrolled the same area many times, but then explores a new environment, i.e., a patrol followed by a reconnaissance task.

The Lab dataset highlights the strengths of the descriptor approximation for dynamic reordering, as it preserves scene context when scans from different environments are collected in close proximity. The average normalized overlap percents for the descriptor, pose, and combined methods are 100%, 88.0%, and 97.7% with standard deviations of 7.4%, 13.9%, and 5.3% respectively. The pose approximation had a significantly worse average and higher standard deviation because scans from either side of a thin wall will be scored similarly despite capturing significantly different environments, as shown in Fig. 4.12. This indicates that descriptors are best in structured environments where small pose differences can lead to large scene changes, and pose approximation is best in outdoor mapping where descriptors are less capable of learning unstructured environments. Sieve-Streaming and random had low average normalized overlap with 85.3% and 75.0% with particularly high standard deviation with 18.3% and 15.1%. The evenly spaced heuristic had a normalized overlap of 89.9%, which

Table 4.1: OptMap computation time results. Left column gives the dataset, where sequences from three standardized datasets were chosen alongside one custom dataset collected on a 128-beam Ouster OS1 LiDAR (Forest Loop pictured in Fig. 3.1). Distilled map size indicates how many scans were used to generate the map, optimization time is the time from function call to when the solution set is selected, and load time is the remaining time spent loading point clouds from storage to computer memory. Method (1) uses the initial solution bounds of Section 3.4.1 and method (2) does not.

Dataset (Num. Scans)	Map Size	(1) Opt. [ms]	(1) Load [ms]	**(1) Total**	(2) Opt. [ms]	(2) Load [ms]	**(2) Total**
Oxford Spires Obs. Quarter Seq. 1 [115] (2,894 scans)	25	2.2	65.2	**67.4**	14.0	68.0	**82.0**
	50	4.1	123.6	**127.7**	28.9	131.1	**160.0**
	100	8.4	243.9	**252.3**	57.6	252.2	**309.8**
Oxford Spires Blenheim Palace Seq. 3 [115] (6,970 scans)	25	3.8	66.8	**70.6**	23.0	67.7	**90.7**
	50	7.0	126.4	**133.4**	46.5	132.6	**179.1**
	100	15.6	252.6	**268.2**	93.1	256.4	**349.5**
Semantic KITTI 10 [2] (1,224 scans)	25	0.9	106.9	**107.8**	5.1	117.3	**122.4**
	50	6.5	210.5	**217.0**	11.2	227.7	**238.9**
	100	66.0	373.0	**439.0**	108.5	372.8	**481.3**
Semantic KITTI 02 [2] (4,663 scans)	100	26.9	405.7	**432.6**	67.1	411.5	**478.6**
	250	229.8	859.4	**1089.2**	642.5	660.7	**1303.2**
	500	329.6	1854.9	**2184.5**	2224.0	803.1	**3027.1**
Newer College Long Experiment [1] (26,559 scans)	100	80.5	363.6	**444.1**	493.8	354.4	**848.2**
	250	186.5	896.5	**1083.0**	1248.8	917.5	**2166.3**
	500	366.7	1919.7	**2286.4**	2584.4	1956.8	**4541.2**
Forest Loop (34,158 scans)	100	103.8	561.6	**665.4**	683.8	570.0	**1253.8**
	250	275.4	1389.9	**1665.3**	1694.4	1405.4	**3099.8**
	500	531.0	2672.3	**3203.3**	3353.4	2712.4	**6065.8**

matches the trend of the evenly spaced heuristic outperforming Sieve-Streaming, random, and either the descriptor or pose approximation *in datasets that stress the approximation*. Although no single approximation is best on every dataset, the combined approximation is the most robust: it is never more than 2.3% below the best method, compared with 6.2% for the descriptor approximation and 12.0% for the pose approximation, and it has the lowest standard deviation on all three datasets.

4.5.3 Real-Time Map Summarization Evaluation

The third experiment tests OptMap's return time on large LiDAR datasets with and without *a priori* bounds. By presenting the time to find and return actionable maps, this result also highlights OptMap's online map distillation capability in the most challenging scenarios

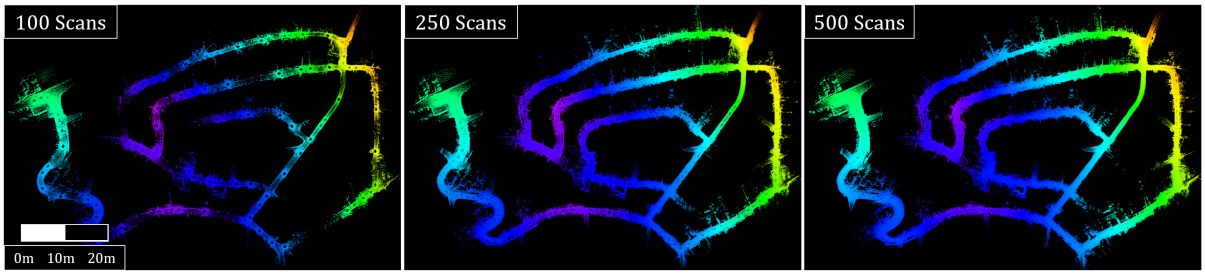


Figure 4.13: Output OptMap maps for Semantic KITTI 02 with 100, 250, and 500 scans. Color indicates z-coordinate of point. OptMap is capable of generating map summaries with flexible size and resolution for autonomous planning at various scales.

with large input set sizes. This experiment distills three small datasets using maps with 25, 50, and 100 scans: Oxford Spires Observatory Quarter Sequence 1 and Blenheim Palace Sequence 3 [115], and Semantic KITTI Sequence 10 [2]. Three larger datasets are distilled using 100, 250, and 500 scans: Semantic KITTI Sequence 02, the Newer College Dataset Long Experiment [1], and Forest Loop, a custom dataset collected in an unstructured forest environment. Load times are influenced by the number of points in each scan which is in turn set by the sensor. Oxford Spires used a 64-beam Hesai QT64, KITTI used a 64-beam Velodyne HDL-64E, Newer College used a 64-beam Ouster OS1, and Forest Loop used a 128-beam Ouster OS1. The two largest sessions are Newer College Dataset's Long Experiment [1] and Forest Loop, which contain 26,559 and 34,158 scans representing 44 and 28.5 minutes (or 18.6 and 34 GB) respectively. OptMap was run five times for each dataset with negligible variation ($<$ 20 ms total max) in the optimization and map load time.

The optimization, map loading, and total times for all six datasets are presented in Table 4.1. The third through fifth columns (labeled method 1) used *a priori* bounds and the sixth through eighth (labeled method 2) did not. The percent of total time saved by the *a priori* bounds is between 8%-50% with the larger datasets receiving larger benefits including between 46%-50% of time saved on the two largest datasets, Newer College Long Experiment and Forest Loop. The dramatic increase in time saved is due to the fact that tighter bounds reduce the number of solution sets that need to be maintained for larger input sets. The

total time increases with the distilled map size due to increases in both optimization and map loading times. Although descriptor generation is performed at sensor rate and not included in map return time, note that the average time to generate a descriptor for each of the six datasets was 8.16, 7.07, 11.62, 10.95, 6.57, and 8.47 ms in order with the table. Semantic KITTI is a challenging dataset for OptMap because it stresses the assumption that sequential descriptors are similar due to the fact that it is collected on a fast moving car.

Summarization is a key application for OptMap, with visual results presented in Fig. 4.13. Fig. 4.13 demonstrates generated summary maps with flexible size, where size-constrained summarization could be valuable for bandwidth-aware multi-robot map sharing or global place recognition. The presented maps show high-fidelity and extremely detailed maps which would not generally be available using voxelization or keyframing. Large dataset summarization examples show that a combination of evaluating informativeness over the set of descriptors rather than points and a single evaluation pass allows OptMap to return maps quickly for use in online autonomous planning and decision making.

4.6 Discussion

This chapter presented OptMap, an online geometric map distillation algorithm that can generate provably near-optimal maps from large LiDAR mapping datasets. This dissertation proposes multiple theoretical and algorithmic innovations which provide a means for quantifying informativeness in geometric maps, and allow for the efficient optimization of such maps. The novel submodular reward function Continuous Exemplar-Based Clustering (CEBC) quantifies map informativeness using point cloud descriptors to eliminate all point-wise operations from OptMap, while element weighting allows for the removal of redundant scans to mitigate over-represented elements in the input set and reduce the input set size. Dynamic reordering is a novel addition to streaming submodular maximization algorithms

which uses multiple approximation functions to address input order bias. Tight *a priori* solution bounds further reduce computation time. The results demonstrate the efficacy of CEBC, dynamic reordering, and tight *a priori* bounds across a range of LiDAR datasets.

Future work on OptMap can incorporate higher quality learned descriptors and other similarity metrics for improved distillation results across a wider variety of environments. Although the descriptors used here were shown to accurately describe a LiDAR scan, descriptors that better discriminate based on local features (i.e., groups of points from the same object) could enable more sophisticated submodular functions for selecting localization submaps or sharing relevant scans without knowing relative poses. Additional work in generalizing dynamic reordering could yield a novel generalized streaming submodular maximization algorithm that uses limited *a priori* information to address significant forms of input order bias in computationally efficient methods. Improved functionality could also be derived from general matroid constraints. For example, a knapsack constraint could enable input sets with multiple contributing sensor types, and set coverage constraints could allow for minimum size solutions which satisfy a defined coverage criterion. Such innovations can lead to a tighter, bidirectional integration between geometric mapping and the downstream autonomy algorithms or operators that rely on compact and informative geometric maps.

CHAPTER 5

Online Geometric Change Detection via Scene Decomposition

The final contribution utilizes the OptMap algorithm developed in the previous chapter to achieve accurate and online geometric change detection. Geometric change detection is the problem of finding differences in a mapped environment over time using depth sensors such as LiDAR or RGB-D. This chapter presents Change Detection via Scene Decomposition (CDSD), the first submap-based framework for online LiDAR change detection, which decomposes the mapped environment into locally consistent scenes and performs submap-to-submap change detection on these scenes rather than the entire map. CDSD's online detections are consistently corroborated by a state-of-the-art offline method, since decomposing the map splits the problem into many smaller subproblems which are easier to process. OptMap therefore serves as a key enabler in this new framework as it can quickly generate representative and size-constrained submaps for each scene. The framework thus only needs to load and operate on the specific set of scans which are of interest for a particular change detection instance. This chapter originally appeared as [116].

5.1 Overview

Geometric change detection identifies persistent changes in an environment, such as a fallen tree or opened door, by comparing point cloud maps acquired at different times. Detecting

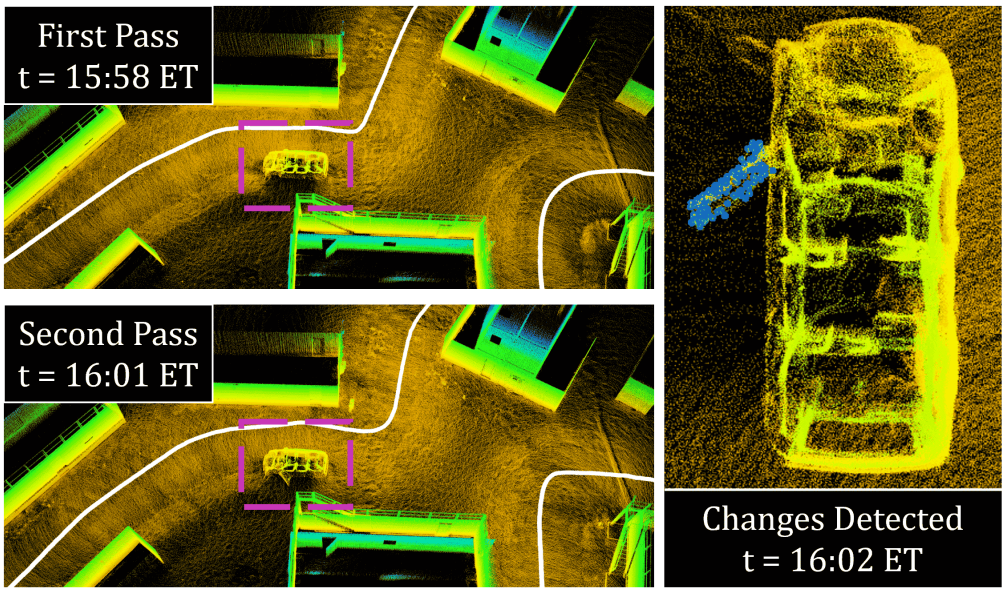


Figure 5.1: A car door is opened between the first and second mapping passes of the same session and detected in under one minute. Base points are colored by height (z-axis); change points are shown in blue in the right figure.

changes online (i.e., within a few seconds of observing the change) enables real-time situational awareness that can be used by autonomous and human agents operating in dynamic environments [63, 70]. Achieving accurate online change detection is inherently difficult though as it requires dense, precisely aligned point clouds to detect small objects. Recent LiDAR odometry and SLAM algorithms can provide centimeter accuracy [20, 117], enabling online scan-to-scan comparisons without the need for downstream refinement. However, individual scans concentrate points along a few scan lines that grow widely spaced with range, leaving them too sparse for reliable change detection. This sparsity can be addressed by aggregating multiple viewpoints in a submap, which is the union of a selected subset of the scans collected within a bounded region. Submaps recover the density for accurate change detection, but raise several open questions about how they should be constructed and compared online.

The first geometric change detection methods compare global maps collected via airborne LiDAR surveys to monitor structural changes such as new or demolished buildings [89]. More

recent approaches extend these ideas to ground-based mobile robotics, which use dynamic object removal and pose graph refinement to improve global map consistency for detecting small changes [3, 4]. While these methods achieve high accuracy, they are incompatible with real-time decision making because they require complete before and after global maps that are only available after a completed mapping session. Current online geometric change detection methods instead compare individual scans against a prebuilt map to significantly reduce computation time per change detection instance [96]. In addition to the limitation of using sparse individual scans, a prebuilt comparison map must either be produced offline from a previous session, or maintained online as a dense global map. Both prior maps waste computation time on non-overlapping regions and the latter grows unbounded in memory.

This chapter proposes Change Detection via Scene Decomposition (CDSD), a framework that 1) decomposes the mapped environment into locally consistent regions called *scenes*, 2) constructs representative cumulative and recent submaps for each scene immediately after it is mapped, and 3) detects changes between submaps despite possibly mismatched fields of view. The resulting framework is both memory and computationally efficient as it only requires loading and operating on a concise set of relevant scans at any given time. This chapter addresses several unique challenges introduced by submap-based geometric change detection, where the novel contributions are:

- **Scene Identification.** The mapped environment is spatially decomposed into unique and non-overlapping scenes by an online scene identification algorithm.
- **Submap Generation.** OptMap [114], the geometric map distillation algorithm developed in the preceding chapter, is used to generate representative, size-constrained submaps for each scene.
- **Change Detection.** A novel submap-to-submap change detection algorithm identifies candidate changes and verifies they are not artifacts of submap pose, field of view, or

occlusion differences.

- **Change Manager.** An efficient change management system that leverages a lightweight data structure to provide a convenient interface for mission/behavior planners.

The experimental results validate each of the four contributions through an ablation study, in which the complete framework detects 97.8% of ground-truth changes across multiple custom datasets. Removing any single component of the framework degrades detection performance or computation time. The overall framework performance is then quantified by comparison against a state-of-the-art offline geometric change detection algorithm [3], where CDSD reports change points that are consistently corroborated by the offline method. CDSD is released as open-source code[1] and validated with experimental results collected at the Army Research Laboratory facility in Graces Quarters, Maryland.

5.1.1 Chapter-Specific Symbols

This chapter uses the dissertation-wide notation of Section 2.1 and the descriptor background of Section 2.2, and builds directly on the OptMap algorithm of the preceding chapter. Recall that a point cloud is a finite set of points P_i with an associated pose $\mathbf{x}_i$, descriptor $e_i = \phi(P_i)$, and timestamp t_i, all indexed by the scan sequence number i and delivered together as a packet $\{i, P_i, \mathbf{x}_i, e_i\}$ at sensor rate. A *scene* is a bounded region denoted $Z \subset \mathbb{R}^3$ with volume $|Z|$, and the set of all scenes is $\mathcal{Z} = \{Z_1, Z_2, \dots\}$, where Z is used for scenes because S denotes the submodular solution set throughout this dissertation. The subset of descriptors collected within a scene is $D_Z = \{e_i \,|\, \mathbf{x}_i \in Z\}$. Each change detection instance is defined for a scene Z_j which has been visited v times, where $v \geq 2$, and uses two submaps. A *cumulative* submap $\mathcal{C}_j^{1:v-1}$ is the subset of scans collected during visits 1 to $v-1$, and a *recent* submap $\mathcal{R}_j^v$ is the subset of scans collected during visit v. Each symbol is defined at its point of use, and all

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

are listed alongside the dissertation-wide notation in the List of Symbols in the preliminary pages.

5.2 Methods

This chapter proposes a framework for fast and accurate geometric change detection. Each scan is assumed to have an accurate odometry estimate and a learned descriptor, as described in Sections 2.1 and 2.2. An online map merging algorithm such as [28] additionally allows the framework to handle multi-session change detection. At its core, the framework compares concise, dense submaps from unique locations to strike a balance between computation time and retaining sufficient geometric detail. The central insight is the spatial decomposition of the mapped environment into scenes, effectively splitting the large offline problem into several, much smaller subproblems with minimal effect on accuracy. Scene-based change detection requires addressing several subproblems, which are outlined here. In Section 5.2.1, an incremental algorithm is introduced to identify unique, non-overlapping scenes. Section 5.2.2 then describes how near-optimal cumulative and recent submaps $\mathcal{C}_j^{1:v-1}$ and $\mathcal{R}_j^v$ are selected for each scene. Next, Section 5.2.3 presents a submap-to-submap change detection algorithm with multiple filters which account for differences in submap positions and fields of view. Finally, in Section 5.2.4, change points are clustered into change objects, which simplify map maintenance and provide actionable data for planners.

5.2.1 Scene Identification

This subsection addresses the problem of partitioning the mapping session into scenes. As the fundamental unit of the framework, each scene is a bounded region of the sensor trajectory that, together with the timestamps of each visit, defines a change detection instance (Fig. 5.2). Ideally, scenes are selected such that running change detection independently on

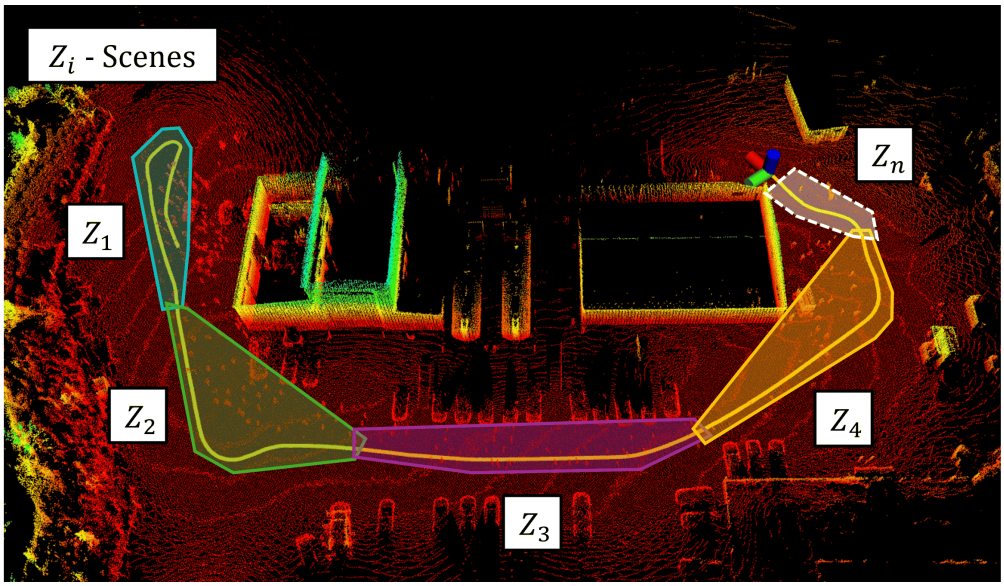


Figure 5.2: Scenes spatially decompose the trajectory rather than the geometric map into convex hulls with minimal overlap. Each labeled convex shape over the yellow trajectory curve represents a unique scene where change detection occurs separately. The scene outlined in dashed white represents the current scene which is still being built.

every scene reports the same changes as one comparison over the complete map at a fraction of the cost. Identifying suitable scene boundaries is therefore important because it impacts both computation time and detection accuracy.

Several factors that balance change detection accuracy and computation time must be considered when defining a good scene. First, every scan must belong to at least one scene, i.e., completeness. Second, scenes must be large enough to enable meaningful comparisons between $\mathcal{C}_j^{1:v-1}$ and $\mathcal{R}_j^v$. Scenes that are too small degenerate into scan-to-scan comparisons, which inherently struggle to distinguish real changes from differences in coverage and viewpoint. Third, to enable online computation times, scenes must be small enough for a fixed-size submap to sufficiently represent them. Fourth, all scans within a scene must be spatially cohesive by exhibiting sufficient pairwise overlap. This is to prevent wasteful cross-region comparisons during change detection instances. This chapter proposes a greedy heuristic that constructs scenes incrementally as scans are collected, satisfying the completeness and size constraints. Formulating optimal conditions for scenes and the online

Algorithm 9 Online Incremental Scene Identification

Given: Scan Packet $\{i, P_i, \mathbf{x}_i, e_i\}$

1: **if** $i = 1$ **then**
2: $Z_{cur} = \emptyset$
3: Return
4: **else if** $Z_{cur} = \emptyset$ and $\exists Z \in \mathcal{Z}$ such that $\mathbf{x}_i \in Z$ **then**
5: Return
6: **end if**
7: $Z_n = \text{conv}\big(Z_{cur} \cup \{\mathbf{x}_i\}\big)$
8: **if** $|Z_n| > \alpha$ or $\max_{e \in D_n} ||e_i - e||_2 > \beta$ **then**
9: $\mathcal{Z}.append(Z_{cur})$
10: `// Do not start new scene while in old scene`
11: **if** $\exists Z \in \mathcal{Z}$ such that $\mathbf{x}_i \in Z$ **then**
12: $Z_{cur} = \emptyset$
13: **else**
14: $Z_{cur} = \mathbf{x}_i$
15: **end if**
16: **else**
17: $Z_{cur} = Z_n$
18: **end if**

optimization of scenes is left to future work.

Algorithm 9 gives the procedure for constructing scenes incrementally. The algorithm receives a packet with each scan containing the sequence number i, point cloud P_i, pose $\mathbf{x}_i$, and descriptor e_i. The algorithm initializes an empty active scene the first time it is called (lines 1–3). When a packet is received and there is an active scene, the algorithm checks if the scene can expand to include the new scan or if the active scene should be closed. Line 7 shows the scene expansion step, where a potential new scene Z_n is found by taking the convex hull of the active scene and the new pose $\mathbf{x}_i$. Line 8 checks if this new scene violates the size constraint set by a parameter α or a scan pairwise overlap condition set by parameter β, where the shorthand notation D_n denotes the descriptor set in Z_n. As established in Section 2.2, the distance between descriptors is monotonically decreasing with the degree of overlap between scans, so the pairwise overlap condition ensures all scans in a scene have sufficient overlap. If either constraint is violated, the algorithm will attempt to start a new

scene. However, to avoid wasted change detection instances caused by overlapping scenes, a new scene is opened only if the current pose does not already fall within a previously established scene (lines 4 & 11).

Entry and exit timestamps are recorded for each visit to a scene. For a scene that has been visited v times, the exit timestamp of the $(v-1)$th visit is t_{out}^{v-1}, and the entry timestamp of the vth visit is t_{in}^{v}. The scene position bounds and these timestamps fully define a change detection instance.

5.2.2 Submap Generation

This subsection describes the process for selecting submaps that accurately represent a scene as defined by the physical and temporal constraints outlined in the previous subsection. The proposed approach uses OptMap, developed in the preceding chapter, to select size-constrained, maximally informative submaps for each scene. OptMap operates over scan descriptors rather than the point clouds, so submap selection reduces to computing vector norms, and the only points loaded are relevant to the current change detection instance. Using a prior global map instead requires a trade-off between local point density—which is required for accurate change detection—and memory allocation. Submap selection is defined first, followed by a summary of the relevant results of the preceding chapter, and finally the implementation details for this application.

First consider scene Z_j that has been visited v times and has timestamps t_{out}^{v-1} and t_{in}^{v}. The cumulative submap $\mathcal{C}_j^{1:v-1}$ will be a subset of scans in Z_j collected before t_{out}^{v-1} and the recent submap $\mathcal{R}_j^{v}$ will be a subset of scans in Z_j collected after t_{in}^{v}. These submaps will therefore be selected from the sets $D_Z^c = \{e_i \,|\, e_i \in D_Z, t_i \leq t_{out}^{v-1}\}$ and $D_Z^r = \{e_i \,|\, e_i \in D_Z, t_i \geq t_{in}^{v}\}$ respectively, where the space of all descriptor sets is $\mathcal{D}$.

Thus the objective is to select two cardinality-constrained submaps $\mathcal{C}_j^{1:v-1} \subseteq D_Z^c$ and

$\mathcal{R}_j^v \subseteq D_Z^r$ that each maximize $\Gamma(S) = L_{cebc}(\mathbf{0}) - L_{cebc}(S \cup \mathbf{0})$, which follows the null-element construction of (2.10) with $\mathbf{0}$ the vector of zeros, and where $L_{cebc} : \mathcal{D} \to \mathbb{R}$ is the clustering-based loss $L_{cebc}(S) = \frac{1}{w_{tot}} \sum_{e \in D_Z} w_e \, d(e, S)$ of (4.3). When selecting a cumulative submap $D_Z = D_Z^c$ and when selecting a recent submap $D_Z = D_Z^r$. L_{cebc} can be interpreted as a weighted clustering function where each descriptor e is assigned a weight w_e equal to the Euclidean distance between e and the preceding descriptor in the mapping session. The term w_{tot} is a normalization term equal to the sum of weights $w_{tot} = \sum_{e \in D_Z} w_e$ such that $L_{cebc} \in [0, 1]$. As shown in the preceding chapter, $\Gamma(S)$ is submodular, and the OptMap algorithm solves it within a theoretical $1/2 - \epsilon$ bound where ϵ is a granularity constant, per Proposition 1, and in practice most solutions are shown to significantly outperform the theoretical bound.

The change detection framework provides the spatial and temporal bounds for a scene as well as a cardinality constraint which is constant for a given mapping session. Notably, OptMap can gracefully handle scenes that have been visited three or more times, where $\mathcal{C}_j^{1:v-1}$ can be selected from two or more mapping passes. Alternative methods for submap generation struggle to select a representative subset given large potential discontinuities in the input set. The selected $\mathcal{C}_j^{1:v-1}$ and $\mathcal{R}_j^v$ are then used for the change detection algorithm presented in the following subsection.

5.2.3 Change Detection using Submaps & Occlusion Filtering

This section describes the change detection algorithm, which takes $\mathcal{C}_j^{1:v-1}$ and $\mathcal{R}_j^v$ from OptMap as inputs and returns a set of positive and negative change points. Change points are defined as points that can be confirmed to appear, disappear, or move between passes—a stricter criterion than methods that label any unmatched point as a change. This strict definition requires confirming that candidate change points have not simply been occluded due to differing fields of view between the submaps, motivating a novel occlusion filter. A ground

segmentation step is additionally included to make detecting objects near the ground easier, along with a novel ground filter that prevents ground segmentation errors from propagating into false change detections. The complete pipeline is shown in the change detection block of Fig. 5.3, and described in the following.

The first step is ground segmentation for $\mathcal{C}_j^{1:v-1}$ and $\mathcal{R}_j^v$ separately. The approach follows Patchwork++ [118], dividing the submap into a polar grid and fitting a ground plane Gaussian distribution estimate with a mean position and covariance for each bin. Points are labeled as non-ground if their Mahalanobis distance to the ground plane exceeds a threshold. The ground is assumed not to change between passes, so change detection is only performed on non-ground points, though this assumption is not binding and change detection could be applied to ground points separately in certain applications. Ground segmentation allows the change detector presented later in this section to find low objects close to the ground which would otherwise be difficult to find.

The change detector follows ground segmentation using a nearest neighbor distance threshold. A non-ground point is labeled as a *candidate change* if the average distance to the n nearest neighbors in the corresponding submap exceeds a threshold τ. As a rule of thumb, τ should be set greater than $d_{avg}/2$, where d_{avg} is the larger of the average nearest-neighbor distances in $\mathcal{C}_j^{1:v-1}$ and $\mathcal{R}_j^v$. This threshold effectively sets the minimum resolution of detectable change objects. Following nearest neighbor thresholding comes a simple noise filter which requires the average distance from a change point to its n nearest change points to be below a threshold ρ.

Candidate change points are then passed through an occlusion filter. This step is necessary because the sensor's trajectory will likely be different from the one taken when the scene was created, meaning the submaps might contain points from different parts of the environment due to differing fields of view. This intuition is shown in Fig. 5.4: submap B observes a large area that submap A cannot because a large wall occludes it. The approach

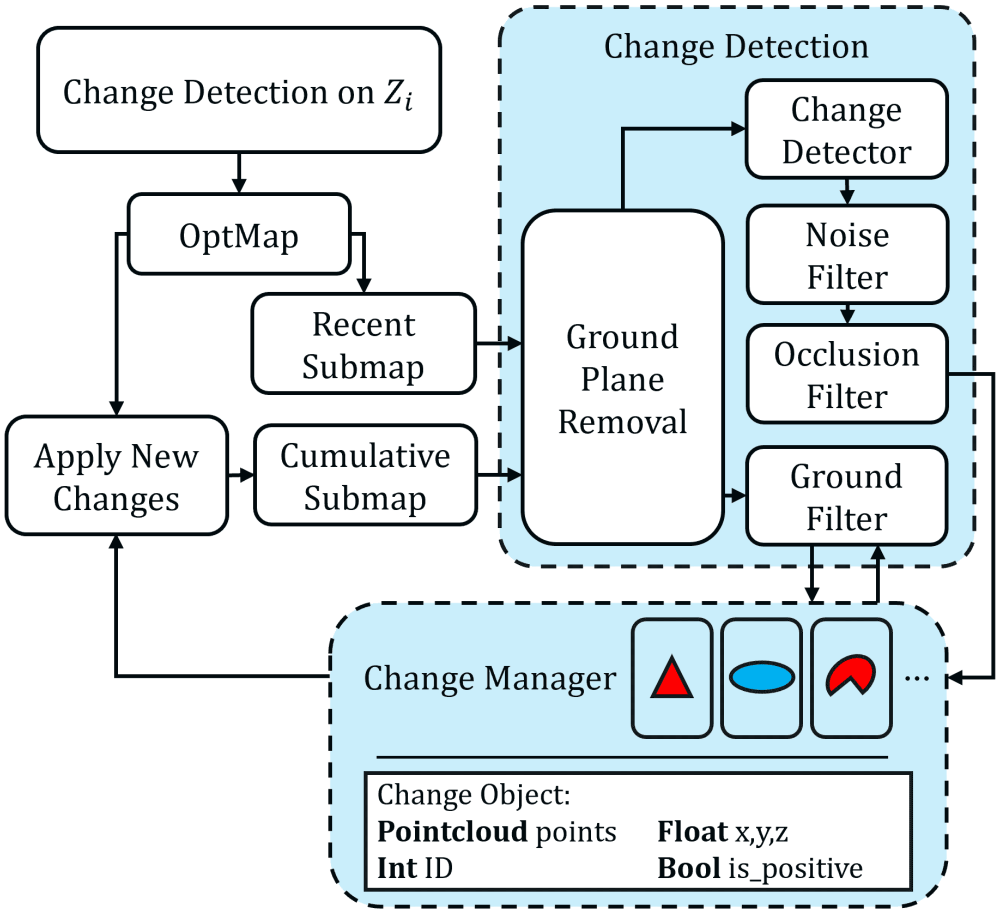


Figure 5.3: Before the change detection algorithm occurs, OptMap is called to generate $\mathcal{C}_j^{1:v-1}$ and $\mathcal{R}_j^v$. During change detection five sequential steps occur: 1) ground plane removal (segmentation), 2) nearest-neighbor based change detection, 3) noise filtering, 4) occlusion filtering, and 5) ground plane filter after changes are clustered into candidate objects. Detected changes are clustered and managed as objects with respective change clouds, object IDs, average coordinates, and a positive/negative flag.

to confirming a candidate change point is to check whether points were collected behind the candidate change from the perspective of the opposing submap. In other words, a positive change candidate from submap B must be confirmed by checking if submap A could have observed the change or not. In Fig. 5.4, the candidate changes were observed by submap B but not A, so the occlusion filter makes a set of range image pairs for each scan in submap A by projecting the candidate change points to the location of the scan and projecting submap A to the location of the scan. These paired range images are visible in the bottom-left of Fig. 5.4. Pixel-wise subtraction between paired range images yields a new range image

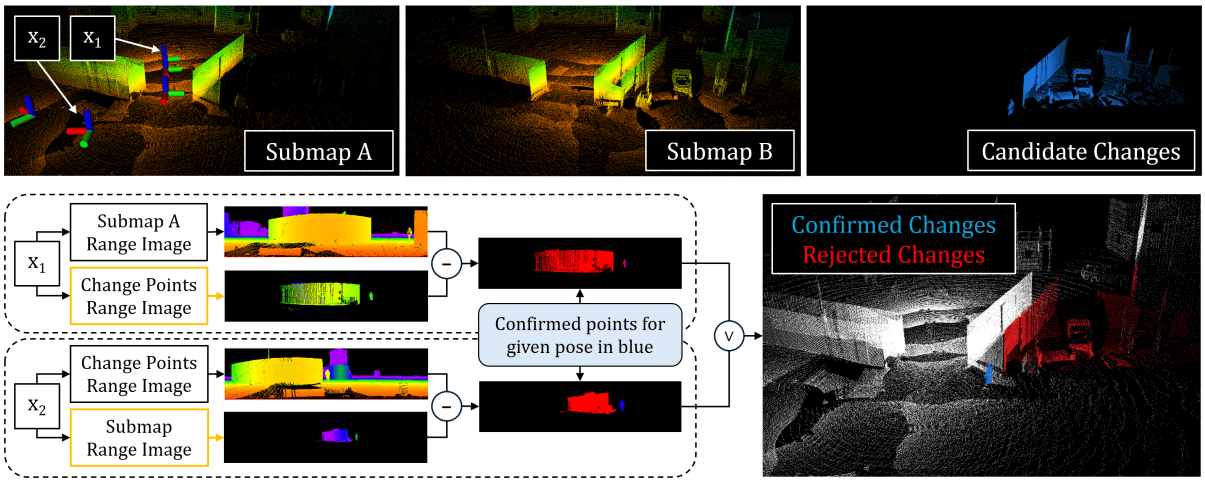


Figure 5.4: Occlusion filter on a submap-to-submap change detection instance. Top three images show submap A, B, and the set of candidate change points that are in submap B but not A. Bottom-left shows range image projection and subtraction for submap A and candidate change points at each scan pose in submap A. If a point is confirmed by any range image subtraction step it is added to the set of confirmed changes shown in the bottom-right.

where the pixel value for a confirmed point is positive if and only if a point in submap A was observed behind the candidate change point. This range image process simplifies what is normally performed as an expensive ray tracing step.

The final step is the ground filter, which is applied after the candidate change points have been clustered (see Section 5.2.4). If ground segmentation fails (e.g., due to reflections from standing water) in $\mathcal{C}_j^{1:v-1}$ or $\mathcal{R}_j^v$, the ground points from the failed segmentation could be incorrectly labeled as changes. The ground filter addresses this by treating each of the change objects as Gaussian distributions and comparing their estimated covariance and mean position to the nearest ground plane estimate. All positive change objects are compared to the ground estimate from $\mathcal{C}_j^{1:v-1}$, and negative change objects to the ground estimate from $\mathcal{R}_j^v$. For a change object to be rejected by this filter, it must satisfy two separate conditions motivated by the Bhattacharyya distance for measuring the similarity between Gaussian distributions ([119], Eq. 61). First, it must be close to the ground as measured by the signed distance from the object's mean position m_o to the planar estimate of the ground with mean position m_g and unit normal vector n_g: $B_1 = |(m_o - m_g) \cdot n_g|$. Second, it must have a similar

covariance as measured by the second term of the Bhattacharyya distance

$$B_2 = \frac{1}{2} \log \left(\frac{\det(\frac{\Sigma_g + \Sigma_o}{2})}{\sqrt{\det \Sigma_g \cdot \det \Sigma_o}} \right), \tag{5.1}$$

where Σ_g is the covariance of the ground and Σ_o is the covariance of the change object. Given parameters c_1 and c_2, a change object is rejected if both $B_1 < c_1$ and $B_2 < c_2$.

5.2.4 Change Manager

The change manager converts confirmed change points into higher-level abstractions via distance-threshold clustering. Abstracting change points into objects enables efficient online map maintenance and makes detected changes actionable for downstream planners, which can query object-level information about the environment. For map maintenance, observations of scenes visited more than twice can be consolidated into a single $\mathcal{C}_j^{1:v-1}$ by applying all previously detected changes. Each change object, shown by Fig. 5.3, contains the clustered points, a unique ID, a timestamp, the object's average coordinate, and a boolean indicating whether it is a positive or negative change. The change manager provides the following functions for downstream use: `update_map(cloud points, float time)` generates $\mathcal{C}_j^{1:v-1}$ by sequentially applying all negative change objects with timestamps less than the query time (only negative changes are needed since the input submap already contains all positive changes up to that time); and `publish_change_list(int n)` outputs metadata for all change objects with at least `n` points, giving planners a clean, noise-filtered view of detected changes.

Table 5.1: Change-detection ablation (mean of 5 runs per variant), reported as the average across the single-session Short (4 changes) and Long (18 changes) datasets. A ground-truth object counts as detected when a change cluster of the same polarity falls within 1 m of it. GT detection percentages are cumulative: *Whole* counts objects with >75% of the GT object's points, *Partial* counts those at >50%. Spurious detections are split into small and large at 125 points. *Dropped* is the mean number of change-detection instances discarded because of a full queue. **Bold** marks the best value, underline marks the second best value, and red marks the worst value in each column. Middle group: alternative detection frameworks. Bottom group: ablations of CDSD.

Method	GT detected (%)		Spurious		Compute (s)		Dropped ↓
	Whole (>75%)	Partial (>50%)	Small (<125 pts)	Large (≥125 pts)	Inst.↓	Sess.↓	
CDSD (Ours)	89.4	**97.8**	**2.1**	2.0	1.51	41.8	0
Scan-to-submap	29.4	53.6	3.4	2.2	**0.65**	68.9	26
Scan-to-map	10.6	16.1	6.2	34.1	2.32	100.9	103
w/o OptMap	52.2	79.7	3.4	**1.0**	1.20	**31.9**	0
w/o Ground Seg.	62.2	74.7	2.6	2.5	1.61	44.9	0
w/o Occlusion Filter	93.9	93.9	130.4	282.2	2.08	57.5	0
w/o Noise Filter	88.3	95.6	18.9	6.5	1.71	45.5	0
w/o Ground Filter	**95.6**	**97.8**	16.1	35.0	1.55	43.8	0

5.3 Results

The proposed change detection framework is validated with 1) a measurement of the cost of generating the submaps every change detection instance depends on, 2) an ablation study that demonstrates the value of each component of the framework, and 3) an experiment that compares detected changes on an open-source dataset to a state-of-the-art offline LiDAR change detection algorithm [3]. The results are generated with real-time sensor playback on a laptop i7-13620H CPU and NVIDIA 4060 GPU for generating descriptors. In addition to the custom datasets described in Section 5.3.1, the open-source dataset is the parking lot dataset originally presented in [4], collected with an Ouster OS1 64-beam LiDAR. Default parameters used for this testing are available alongside the open-source code.

Table 5.2: Sensitivity to the change detection distance gate τ. All other parameters at their default values. The $\tau = 0.30$ row is the default operating point. Results averaged over five runs on the Short and Long datasets.

τ (m)	GT whole (>75%)	Spurious Small (<125 pts)	Spurious Large (≥125 pts)
0.1	88.9	8.6	8.4
0.2	83.1	1.5	3.4
0.3	89.4	2.1	2.0
0.4	88.1	1.7	1.2
0.5	71.1	1.4	0.8

Table 5.3: Sensitivity to the maximum scene volume α. Scenes were restricted to 2D for these experiments, so α is reported as an area. All other parameters at their default values. The $\alpha = 120$ row is the default operating point. Results averaged over five runs on the Short and Long datasets. Submap size increases proportionally with α.

α (m^2)	GT whole (>75%)	Spurious (≥50 pts)	Compute (s/inst.)
30	50.0	4.4	0.667
60	34.7	4.1	0.861
120	89.4	4.8	1.510
240	88.9	5.5	2.935
480	97.2	6.8	5.431

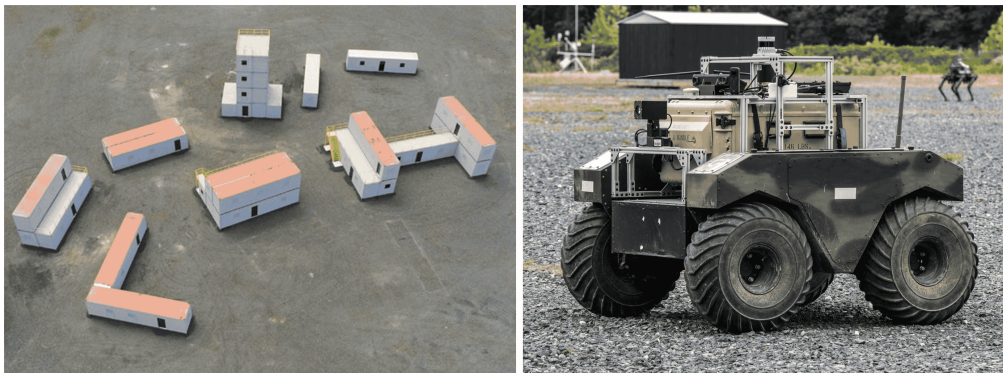


Figure 5.5: Left: Graces Quarters, MD testing site. Right: Modified Clearpath Warthog with Ouster OS1 128-beam LiDAR used for experiments.

5.3.1 Hardware Platform

Custom datasets were collected outdoors at the Army Research Laboratory facility in Graces Quarters, Maryland. The facility and robot equipped with an Ouster OS1 128-beam LiDAR

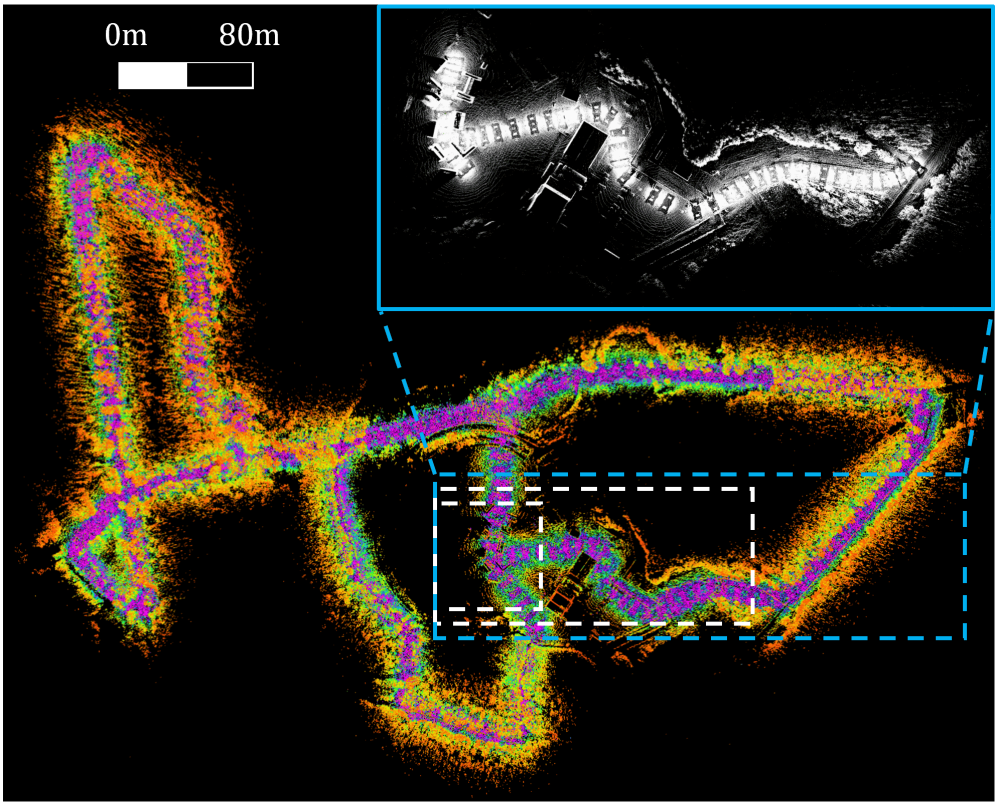


Figure 5.6: Figure Eight dataset, 3.9 km over 25.4 minutes. Boxes shown in white are requested summary map areas bounded by $x_{min} = -20-u$, $x_{max} = 20+u$, $y_{min} = -20(1+u)$, and $y_{max} = 20$ for $u = \{0, 10, 20\}$ with the summary map generated for $u = 20$ by OptMap shown in the blue subfigure.

are shown in Fig. 5.5. Two experiments were run with preplanned change objects moved during a single-session patrol mission, where the datasets collected are referred to as Short Patrol and Long Patrol or simply Short and Long. The Short Patrol contains four preplanned changes and the Long Patrol contains eighteen. The ground truth changes for the Short dataset are shown in Fig. 5.7. A third dataset, Figure Eight, was collected at the same facility with the same platform and sensor, and is used in Section 5.3.2 to measure the cost of generating a submap. It contains no preplanned changes and instead provides a long-duration mapping session, covering 3.9 km over 25.4 minutes.

Table 5.4: Map loading and processing time for generating summary submap for a subset of long-duration mapping mission. Load, Box filter, and Voxel filter (LBV) method implemented using open-source libraries, with the box filter and voxel filter times listed separately. OptMap performs streaming optimization over descriptors then map loading.

	OptMap			Load-Box-Voxel (LBV)			
Area [m^2]	Load [ms]	Process [ms]	Tot. [ms]	Load [ms]	Box [ms]	Voxel [ms]	Tot. [ms]
1,600	129	3	**133**	5,047	2,024	287	7,358
7,000	104	5	**109**	5,000	2,065	603	7,668
14,400	123	6	**129**	5,020	2,089	812	7,921
23,800	117	8	**126**	5,023	2,218	1,253	8,495
35,200	107	8	**115**	5,070	2,264	1,496	8,830
avg.	116	6	**122**	5,032	2,132	890	8,054

5.3.2 Submap Generation Cost: OptMap versus a Point-wise Pipeline

OptMap provides value to robotic autonomy stacks as a computationally and memory efficient map distillation algorithm. This result considers an illustrative example where an operator or downstream algorithm requests a summary map of a subset of a larger mapping session. Either a memory or communication bandwidth constraint limits the size of the requested summary, which is capped at 4,500,000 points (approx. 72 MB) for this scenario. The original mapping session is a 3.9 km, 25.4 minute traverse with approximately 30,450 scans visible in Fig. 5.6. The requested summary area is within a box defined by $x_{min} = -20 - u$, $x_{max} = 20 + u$, $y_{min} = -20(1 + u)$, and $y_{max} = 20$ where $u \in [0, 20]$ such that the final summary area encompasses approximately a quarter of the total mapped area.

OptMap is compared to a traditional approach consisting of Loading a dense map, applying a Box filter, and applying a Voxel filter (LBV). This alternative pipeline first loads a dense keyframe map of 3,536 scans, where keyframes are evenly spaced 1 m apart during mapping. The dense map is then passed through a box filter with the area constraints followed by a voxel filter with a leaf size of $l = \sqrt[3]{\frac{V}{4{,}500{,}000}}$ where V is the volume $V = (x_{max} - x_{min}) \cdot (y_{max} - y_{min}) \cdot (z_{max} - z_{min})$, and z_{min} and z_{max} are the observed floor and ceiling levels of the dataset. OptMap optimizes and loads a map with no more than 48

scans, which is derived by dividing the point budget by the average number of points per scan in the session.

Table 5.4 shows the resulting time to return summary maps for both methods, with the map load, box filter, and voxel filter times listed separately for LBV. LBV loads a dense global map at the beginning of each run, requiring 5.0 seconds on average. The box filter cost is nearly constant at roughly 2.1 s because it must traverse the full dense map regardless of the requested area, while the voxel filter grows with the size of the region it is given, taking 287 ms, 603 ms, 812 ms, 1,253 ms, and 1,496 ms for $u = \{0, 5, 10, 15, 20\}$ (where the table shows the summary map area). The increasing voxel filter time therefore drives the monotonic increase in processing time for LBV from 2,311 ms to 3,760 ms, highlighting the computational challenge of any point-wise operations on such large point clouds. OptMap on the other hand requires an average of 6 ms in optimization and 116 ms in map loading. OptMap therefore uses only 1.5% of the total return time of LBV on average. Even if a dense map of only the requested region were already in memory, eliminating both the map load and the box filter, voxelizing that region alone would still take 7.3 times longer on average than OptMap's complete optimization and map load. Additionally, LBV has a peak memory consumption associated with loading the full dense map with 321.7 M points, where OptMap memory usage is maximum when the final summary is loaded which always contained less than the 4.5 M point limit. Peak memory usage by LBV could be reduced by loading the map in chunks, but this would increase the processing time further. To summarize, OptMap completes this simple summarization task, which underlies a wide variety of robotics applications, in 122 ms on average while never holding more than the requested summary in memory, compared with 8,054 ms and a 321.7 M point dense map for LBV.

Even with optimizations around the LBV pipeline, conventional approaches to solving this simple problem would require loading dense point clouds and relying on heuristic down-sampling to achieve the required size in memory. Random subsampling, voxel grid filtering,

and saliency-based selection via principal component analysis are all defined over individual points. The cost of each therefore scales with the number of points in the map rather than the number of scans, which is prohibitive when a single mapping session contains hundreds of millions of points. OptMap makes no commitment before a mapping session to the size, resolution, or extent of the maps that will later be requested, as constraints are supplied at request time and the same descriptor set serves any of them. No point-wise operations are performed during OptMap's optimization, which is carried out entirely over descriptors, and OptMap returns a constrained summary map in less than a second even for mapping sessions as large as the one shown in Fig. 5.6. OptMap, developed in the previous chapter, is what enables CDSD. Decomposing the mapped environment into local scenes and comparing dense submaps of each scene immediately after it is remapped poses exactly the local summarization problem presented in this subsection, repeated twice per scene visit to generate the two submaps of a change detection instance. The size of those submaps is constrained to reduce the computational burden of searching for differences between them. LBV's average 3 seconds of box and voxel filtering alone means change detection could only be called once every 6 seconds, before considering the expensive change detection process that still needs to run after the submaps are generated.

5.3.3 Ablation Study

As noted by [4], no open-source point-wise ground truth change detection datasets are available in the form of raw sensor data, so the approach is validated with an ablation study on the Short and Long Patrol datasets. Instead of creating a point-wise ground truth for the custom datasets, the position of each known change object is hand labeled and the approximate number of points in the abstracted object is determined by running an offline variant of change detection. To classify a ground-truth change object as detected, a change cluster of the same polarity (positive or negative) must fall within 1 m of it and reach at least 50%

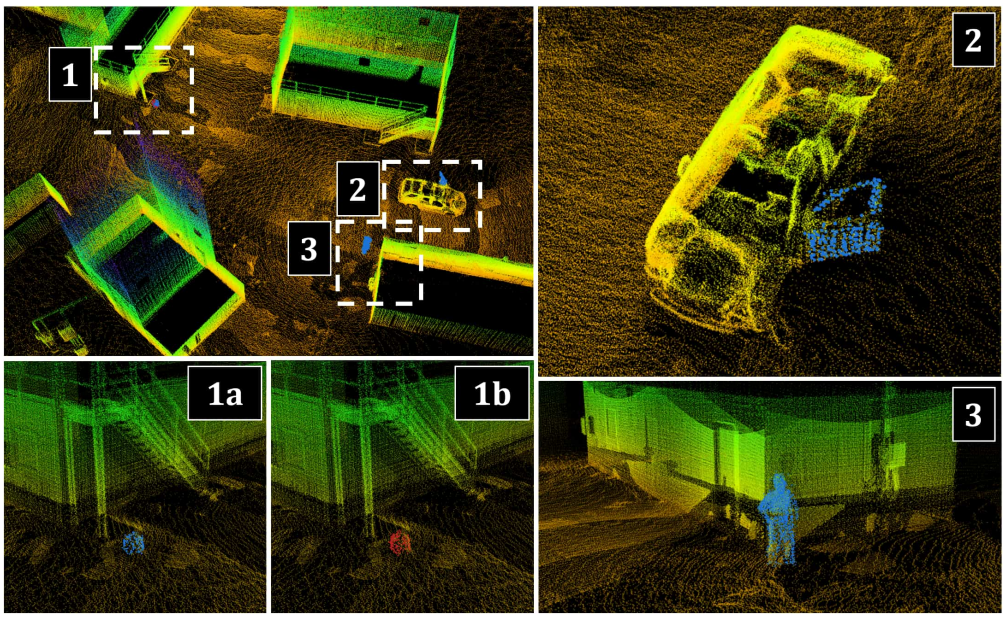


Figure 5.7: Ground truth changes for the Short change detection dataset. Top-left shows an overview of the map and views of each of the four changes. View 1 shows a backpack ($\sim$ 20 cm) which was introduced for view 1a and removed for view 1b. View 2 shows a car with a door opened between observations, and view 3 shows a person who appeared between observations.

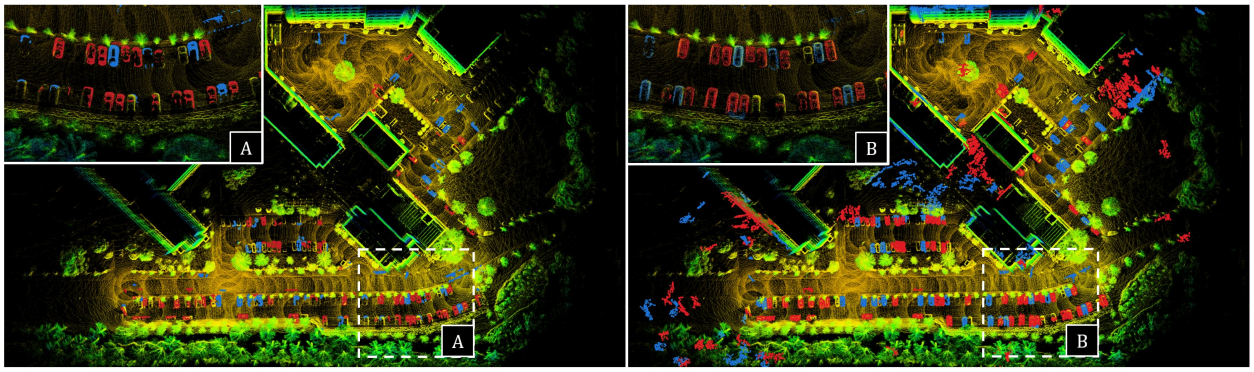


Figure 5.8: Change detection results from LT-Mapper Parking Lot dataset session 02 (Jan 25, 2021) to session 04 (Jan 26, 2021). Left figure shows the changes from CDSD, and right figure shows the changes detected by ELite [3]. Positive change points are colored blue, negative change points are colored red, and the background map is colored by z-axis. The inset on the top-left of both figures shows a detailed difference in detection outputs for either algorithm.

of the object's point count, with matches above 50% deemed partial and matches above 75% deemed complete.

Table 5.1 compares CDSD against two alternative online change detection strategies and an ablation comparing each component of the proposed framework. The two alternate

strategies—Scan-to-Submap (S2S) and Scan-to-Map (S2M)—are inspired by other online geometric change detection algorithms [95,96]. For both methods, a scan every 2.5 m traveled is compared to a prior map using the same set of filters that CDSD uses, where S2S uses scene-based OptMap submaps and S2M uses a dense keyframe prior map collected from the first pass of each dataset. The ablation variant without OptMap uses a submap with scans selected via an equal time stride heuristic, and the variant without ground removal also does not use the ground filter. Each method is evaluated based on the percent of known ground truth objects detected, number of spurious or incorrect changes detected, the per instance computation time, total computation time, and the number of dropped change detection instances. The cutoff for small and large spurious changes is set at 125 points to approximately match the smallest ground-truth change across both datasets.

S2M reaches only 10.6% whole and 16.1% partial detection and is not comparable in detection quality to CDSD's 89.4% whole detection rate. S2S recovers 29.4% of ground-truth objects whole and 53.6% partially. Per-instance computation time is 54% longer for S2M and 57% shorter for S2S, but on average S2M completed 43 change instances and S2S completed 106, requiring 141% and 65% more total computation time respectively overall despite each dropping many change instances. It is worth noting that S2S outperformed S2M across every category, indicating that scene-based submaps are a broadly useful addition to geometric change detection. The bottom group of Table 5.1 ablates the submap selection strategy and each change detection filter. Replacing OptMap's submodular scan selection with equal-time-spaced scans drops whole-object detection from 89.4% to 52.2%. This 37.2% drop is due to the fact that OptMap generates better cumulative submaps than evenly spaced heuristics, especially when the change occurs after a scene has been visited more than twice. Removing ground segmentation reduces complete detections to 62.2%, a decline exemplified by the backpack changes in Fig. 5.7, which are detected only 10% of the time without ground segmentation. Removing the occlusion filter increases whole-object detection rates to 93.9%,

at the cost of 130.4 small and 282.2 large spurious detections on average per session. Removing either the noise or ground-plane filter similarly increases spurious detections (18.9/6.5 and 16.1/35.0 small/large when removed, versus 2.1/2.0 for CDSD). Notably, per-instance computation time for CDSD is lower than every ablation except removing OptMap's submap selection (1.20 s versus 1.51 s). Averaged across all runs, the top four most computationally expensive steps take 1.05 s for nearest-neighbor search, 0.23 s for ground segmentation, 0.16 s for OptMap, and 0.09 s for object clustering. Aside from ground segmentation, each filter is computationally cheap to run and reduces the number of points that must be clustered. Removing the occlusion, noise, and ground filter thus leads to between 2.6% and 37.7% increases in computation time. Ground removal effectively offsets its required computation time by pruning points for the nearest-neighbor search which requires 0.34 s longer to complete without ground removal. In summary, each step is computationally efficient and meaningfully improves CDSD's detection performance.

Table 5.2 and Table 5.3 show sensitivity analysis for the minimum detection distance threshold τ and the scene size α. Both sweeps were averaged over five runs on the same two custom datasets used for Table 5.1. Table 5.2 shows that low detection thresholds all detect the ground truth changes, but the lowest thresholds also detect many spurious changes, with $\tau = 0.1\,\text{m}$ producing by far the most (8.6 small, 8.4 large). From $\tau = 0.2\,\text{m}$ upward, large spurious detections decrease monotonically (3.4, 2.0, 1.2, 0.8) while whole-object detection stays high until it falls off at $\tau = 0.5\,\text{m}$ (71.1%). The default value of $\tau = 0.3\,\text{m}$ catches almost all ground truth changes (89.4% whole) with few spurious changes detected (2.1 small, 2.0 large). Scenes were restricted to 2D for all experiments, so α is reported as an area. Table 5.3 shows that whole-object detection is poor for the smallest scenes (50.0% at $\alpha = 30\,\text{m}^2$ and 34.7% at $\alpha = 60\,\text{m}^2$) but improves substantially as the scene grows (89.4%, 88.9%, and 97.2% at $\alpha = 120$, 240, and $480\,\text{m}^2$ respectively), at the cost of per-instance compute time that grows from 0.667 s to 5.431 s across the sweep. Note that the number of

Table 5.5: Corroboration of CDSD's online change detections by ELite [3] on the multi-session parking lot dataset [4]. Total agreement is the percentage of all CDSD change points that have an ELite change point of the same polarity within 0.5 m. The earlier session of each pair is used as the base map.

Scenario	Sessions	Total Agreement (%)
Filling 1	02–03	89.6
Filling 2	04–06	89.1
Emptying 1	02–04	89.5
Emptying 2	03–04	89.4

scans in each submap changed proportionally to α. This matches the intuition that larger scenes lead to offline-like change detection performance in the limit, with $\alpha = 120\,\mathrm{m}^2$, the framework's default, offering a balance between accuracy and compute cost.

5.3.4 LT-Mapper Parking Lot Dataset

The agreement of the proposed method with ELite is quantified next against [3], a state-of-the-art open-source algorithm for offline LiDAR change detection, on the open-source multi-session LT-Mapper parking lot dataset [4]. The four scenarios listed in Table 5.5 come from four sessions, two daytime (03 and 06) and two nighttime (02 and 04). They are named Filling 1–2 and Emptying 1–2 after the dominant direction of change in each, positive and negative respectively. In a pre-processing step, highly dynamic points are removed using Removert [90] for both CDSD and ELite following [4], and ELite's baseline approach and parameters are otherwise used. PinNet [28] is used to align both sessions into a common coordinate frame so that CDSD can detect multi-session changes and, for a fair comparison, change detection is triggered only between sessions. Agreement with ELite is measured with one-way point overlaps within 0.5 m.

The comparison is summarized by Table 5.5, where 89.1% to 89.6% of CDSD's detected changes are corroborated by ELite. The agreement is remarkably consistent between sessions indicating that CDSD's detections are reliably true changes. The percent of ELite's detected changes in the session dominant polarity matched by CDSD is 54.2%, 63.5%, 50.5%, and

55.7% for Filling 1, Filling 2, Emptying 1, and Emptying 2 respectively. ELite matches are reported in the dominant polarity because ELite over-reports changes with respect to the change definition used here, due to occlusion and unmatched fields of view. CDSD labels only changes that can be confirmed via the occlusion filter, whereas ELite treats any unmatched point in either map as a change—even when the mismatch simply reflects one session having mapped a portion of the parking lot that the other did not. This is evident in Fig. 5.8, where the CDSD results on the left come almost entirely from moved cars, while ELite reports numerous changes caused by field of view and position differences seen near the edges of the parking lot. In short, CDSD reports a conservative set of changes at high precision, reflecting a definition of change shaped by the assumptions and information available online rather than the complete-session view of an offline method.

5.4 Conclusion

This chapter presented CDSD, the first submap-based framework for online LiDAR change detection which spatially decomposes the environment into scenes to enable dense, accurate comparisons online. Ablations confirmed that each of the four core components—scene identification, submap generation, occlusion-aware change detection, and change management—contributes to detection accuracy, and CDSD's online detections are consistently corroborated by a state-of-the-art offline method. Future work can build semantic and temporal understandings of dynamic environments by combining strong geometric change detection with recent advances in local point descriptors and visual language models.

CHAPTER 6

Conclusion

As autonomous robots mature, the ability to process and reason over data collected about the environment remains a bottleneck. Geometric sensors collect an abundance of data which could be used to generate informative maps, and downstream algorithms are increasingly data-hungry. However, mobile robots are still constrained by the limited computation and memory resources driven by size and cost sensitivity. Connecting data-rich perception with data-hungry decision-making algorithms will therefore continue to be an important branch of research and innovation. This dissertation presents several cases where this exact problem can be formulated and solved up to tight suboptimality bounds in real-time, primarily by leveraging submodular maximization.

6.1 Summary of Contributions

The contributions of this dissertation are summarized as follows.

- **Submodular Optimization for Keyframe Selection and Usage in SLAM.** Memory and computation demands increase over time and with newly explored environments in LiDAR SLAM. Optimizing how keyframes are selected and used in alignment submaps significantly reduces this growth for long-duration missions. Selecting fewer keyframes reduces the memory burden of saving point clouds and their normals, and reduces the computational burden of selecting submaps for aligning in-

coming scans. This is accomplished by comparing scans via learned point cloud descriptors to select keyframe sets which are of minimum size while ensuring they retain sufficient information for localizing incoming scans. Selecting submaps via an optimization of the alignment constraint also improves the computation time of iterative closest point algorithms. Both of these problems can be formulated and solved as submodular maximization problems, leading to polynomial time complexity and near-optimal results. The same selection problem is also solved *a posteriori* after completing a SLAM session, generating size specified maps for efficient communication and downstream processes.

- **OptMap: Geometric Map Distillation via Submodular Maximization.** Generalizing the keyframe selection problem from the previous contribution to selecting informative and size-constrained subsets of all geometric perception data collected in a mapping session yields a new problem class called geometric map distillation. This contribution proposes OptMap, which pairs a novel submodular reward function with a streaming submodular algorithm to solve geometric map distillation online. The submodular function addresses the problem of repeated or redundant scans biasing the input set while also reducing the size of the input set, which makes the optimization faster. The streaming submodular algorithm incorporates dynamic reordering, which addresses the problem of input order bias in previous streaming submodular algorithms. This dynamic reordering relies on an approximation of scan-to-scan overlap, for which three variants are proposed and evaluated.

- **Online Geometric Change Detection via Scene Decomposition.** Geometric change detection requires comparing tightly aligned, dense point clouds to find differences such as opened doors or moved objects. This process is prohibitively expensive when performed over an entire mapping session, but does not yield accurate results when performed over just a single scan. This contribution proposes Change Detection

via Scene Decomposition (CDSD), which achieves online, accurate change detection by decomposing the mapped environment into locally consistent scenes, generating size-constrained and informative submaps for each scene using OptMap, and then comparing the submaps with a multi-step submap-to-submap change detection algorithm.

6.2 Limitations and Future Work

The proposed contributions of this dissertation use submodular maximization and learned point cloud descriptors to better connect data-rich perception sensors with data-hungry autonomy algorithms. However, the proposed methods could be improved through both better implementation and further theoretical development. Four avenues for future work include better integrating the intuitions used for previously deployed heuristics, using more recent and robust learned descriptors, developing new submodular functions for other applications, and incorporating multiple sensor sources for semantic or combined information.

The methods in this dissertation set aside the intuition built up around these problems in order to take a more principled approach, but that intuition could still be put to use. Many of the previous heuristics used in perception systems arise from the fact that there are often many similarly valuable subsets of perception data which could be used to solve a problem. Those heuristics worked for a time for a reason: with so many good enough subsets available, a well-tuned heuristic was often able to recover one of them. The contributions of this dissertation do not rely on that luck, and instead make well-informed decisions with provable guarantees on how far from optimal they can be. This same abundance of near-optimal solutions is why the streaming algorithm used in OptMap is so successful when optimizing geometric map distillation. Its pseudo-greedy optimization over the proposed pairwise approximation functions only needs to find one of the many solutions which are theoretically near-optimal. Instead of ignoring these developed intuitions, submodular max-

imization could be used to address many of the problems presented in this dissertation with a prior informed by these heuristics.

Online subset optimization over perception data is made feasible in this dissertation only by learned point cloud descriptors, and OptMap in particular is reliant on the descriptors it is given. The state of the art in point cloud descriptors has developed significantly since the individual contributions of this dissertation were first written, most notably with the emergence of feature-based descriptors. Although they are more computationally expensive to generate and compare, feature-based descriptors have shown a significant improvement in place recognition capability and have demonstrated the ability to reliably obtain approximate scan-to-scan alignment with none of the point-wise operations required by ICP algorithms. Replacing the scan-based descriptors used throughout this dissertation with feature-based descriptors could therefore show better performance for constrained large map summarization.

The submodular functions proposed in this dissertation for keyframe selection and geometric map distillation are generic summarization objectives, and OptMap in particular optimizes informativeness without regard for what the resulting map will be used for. Although the applications of summarization are wide-reaching, many applications like submap selection for scan alignment would benefit from dedicated functions which avoid point-wise operations. Such functions must remain computationally efficient to optimize, which could be achieved by considering point clouds as a set of features characterized via principal component analysis, feature-based learned descriptors, or data from other sensors.

The previous point leads into the last direction for future work, which is integrating multi-sensor data. Every contribution in this dissertation reasons over LiDAR alone, but LiDAR is not the best sensor for every task, and cameras in particular can supply rich semantic information almost for free given the maturity of visual foundation models. An emerging field of study is the semantic-rich geometric map, which attaches semantics retrieved from

other sensing modalities (e.g., cameras, radars) to the geometric map. An example would be to perform object segmentation and identification on visual data, then attach the generated embeddings to voxels in the geometric map. These so-called spatio-temporal maps [102] lead to natural change detection and object-level searches which could be further optimized using the contributions of this dissertation.

6.3 Concluding Remarks

Robotics is still really, *really* hard. Regular advances provide the newest sensor or algorithm with incredible performance, but they are demonstrated in controlled environments or with unrealistic computation times. Such results drive further speculation about a coming revolution in robotics that will have unparalleled societal impact. The limiting factor for deploying autonomous robots is not a shortage of sophisticated solutions—those are plentiful. Instead, these solutions are bottlenecked by the ability to *efficiently* perceive and reason over the environment, which is what keeps them from working in the real world. Without a reliable state estimate, no controller can prevent catastrophic failure. Without a well-aligned and sufficiently dense map, no planner can avoid crashes.

This dissertation attempts to address some of these gaps by first naming and second solving the problem of optimal subset selection in geometric perception data. The contributions are a result of years of difficult development, testing, re-development, and writing, and still are not the solution to this problem. They are, however, a step towards enabling safe, beneficial autonomous robots for all humankind. I hope my work in the field will one day lead to a world where robots assume dangerous and menial work so that my peers, friends, and all others that live in our world have the time and freedom to pursue what makes them happy in life rather than merely what allows them to survive.

References

[1] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. 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)*, 2020, pp. 4353–4360.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[3] H. Gil, D. Lee, G. Kim, and A. Kim, "Ephemerality meets lidar-based lifelong mapping," *arXiv preprint arXiv:2502.13452*, 2025.

[4] G. Kim and A. Kim, "Lt-mapper: A modular framework for lidar-based lifelong mapping," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7995–8002.

[5] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, pp. 265–294, 1978.

[6] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Mathematics of operations research*, vol. 3, no. 3, pp. 177–188, 1978.

[7] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause, "Streaming submodular maximization: Massive data summarization on the fly," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 671–680.

[8] C. Chekuri, S. Gupta, and K. Quanrud, "Streaming algorithms for submodular function maximization," in *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I 42*. Springer, 2015, pp. 318–330.

[9] K. Suresh, A. Rauniyar, M. Corah, and S. Scherer, "Greedy perspectives: Multi-drone view planning for collaborative perception in cluttered environments," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 10 990–10 997.

[10] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Autonomous Robots*, vol. 43, pp. 485–501, 2019.

[11] L. Carlone and S. Karaman, "Attention and anticipation in fast visual-inertial navigation," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 1–20, 2018.

[12] G. Shi, I. E. Rabban, L. Zhou, and P. Tokekar, "Communication-aware multi-robot coordination with submodular maximization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8955–8961.

[13] M. Shamaiah, S. Banerjee, and H. Vikalo, "Greedy sensor selection: Leveraging submodularity," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 2572–2577.

[14] M. Corah, C. O'Meadhra, K. Goel, and N. Michael, "Communication-efficient planning and mapping for multi-robot exploration in large environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1715–1721, 2019.

[15] A. Dasgupta, R. Kumar, and S. Ravi, "Summarization through submodularity and dispersion," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013, pp. 1014–1022.

[16] D. Dueck and B. J. Frey, "Non-metric affinity propagation for unsupervised image categorization," in *2007 IEEE 11th international conference on computer vision*. IEEE, 2007, pp. 1–8.

[17] K. El-Arini and C. Guestrin, "Beyond keyword search: discovering relevant scientific literature," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 439–447.

[18] A. Frank and É. Tardos, "An application of submodular flows," *Linear algebra and its applications*, vol. 114, pp. 329–348, 1989.

[19] H. Lin, J. Bilmes, and S. Xie, "Graph-based submodular selection for extractive summarization," in *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, 2009, pp. 381–386.

[20] K. Chen, R. Nemiroff, and B. T. Lopez, "Direct lidar-inertial odometry: Lightweight lio with continuous-time motion correction," in *2023 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 3983–3989.

[21] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 809–816.

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

[23] K. P. Cop, P. V. Borges, and R. Dubé, "Delight: An efficient descriptor for global localisation using lidar intensities," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3653–3660.

[24] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4802–4809.

[25] J. Ma, J. Zhang, J. Xu, R. Ai, W. Gu, and X. Chen, "Overlaptransformer: An efficient and yaw-angle-invariant transformer network for lidar-based place recognition," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6958–6965, 2022.

[26] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[27] M. A. Uy and G. H. Lee, "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4470–4479.

[28] Y. Ma, N. S. Joshi, C. S. Robison, P. R. Osteen, and B. T. Lopez, "Pinnet: Keypoint-aware learned local descriptors with geometric embedding for loop closure in lidar slam," *arXiv preprint arXiv:2606.28637*, 2026.

[29] Z. Zhou, C. Zhao, D. Adolfsson, S. Su, Y. Gao, T. Duckett, and L. Sun, "Ndt-transformer: Large-scale 3d point cloud localisation using the normal distribution transform representation," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 5654–5660.

[30] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions—ii," *Mathematical Programming Studies*, vol. 8, pp. 73–87, 1978.

[31] S. Fujishige, *Submodular functions and optimization*. Elsevier, 2005.

[32] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization Techniques: Proceedings of the 8th IFIP Conference on Optimization Techniques Würzburg, September 5–9, 1977*. Springer, 2005, pp. 234–243.

[33] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause, "Lazier than lazy greedy," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[34] B. Mirzasoleiman, A. Badanidiyuru, and A. Karbasi, "Fast constrained submodular maximization: Personalized data summarization," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1358–1367.

[35] U. Feige, V. S. Mirrokni, and J. Vondrák, "Maximizing non-monotone submodular functions," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 1133–1153, 2011.

[36] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar, "Constrained non-monotone submodular maximization: Offline and secretary algorithms," in *International Workshop on Internet and Network Economics*. Springer, 2010, pp. 246–257.

[37] A. Das and D. Kempe, "Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011, pp. 1057–1064.

[38] S. Foldes and P. L. Hammer, "Submodularity, supermodularity, and higher-order monotonicities of pseudo-boolean functions," *Mathematics of Operations Research*, vol. 30, no. 2, pp. 453–461, 2005.

[39] A. Downie, B. Gharesifard, and S. L. Smith, "Submodular maximization with limited function access," *IEEE Transactions on Automatic Control*, vol. 68, no. 9, pp. 5522–5535, 2022.

[40] B. Gharesifard and S. L. Smith, "Distributed submodular maximization with limited information," *IEEE transactions on control of network systems*, vol. 5, no. 4, pp. 1635–1645, 2017.

[41] A. Chakrabarti and S. Kale, "Submodular maximization meets streaming: matchings, matroids, and more," *Mathematical Programming*, vol. 154, pp. 225–247, 2015.

[42] M. Feldman, A. Karbasi, and E. Kazemi, "Do less, get more: Streaming submodular maximization with subsampling," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[43] E. Kazemi, M. Mitrovic, M. Zadimoghaddam, S. Lattanzi, and A. Karbasi, "Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3311–3320.

[44] N. Alaluf, A. Ene, M. Feldman, H. L. Nguyen, and A. Suh, "An optimal streaming algorithm for submodular maximization with a cardinality constraint," *Mathematics of Operations Research*, vol. 47, no. 4, pp. 2667–2690, 2022.

[45] M. El Halabi, S. Mitrović, A. Norouzi-Fard, J. Tardos, and J. M. Tarnawski, "Fairness in streaming submodular maximization: Algorithms and hardness," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13 609–13 622, 2020.

[46] K. Banihashem, L. Biabani, S. Goudarzi, M. Hajiaghayi, P. Jabbarzade, and M. Monemizadeh, "Dynamic constrained submodular optimization with polylogarithmic update time," in *International Conference on Machine Learning*. PMLR, 2023, pp. 1660–1691.

[47] ——, "Dynamic algorithms for matroid submodular maximization," in *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2024, pp. 3485–3533.

[48] M. Monemizadeh, "Dynamic submodular maximization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9806–9817, 2020.

[49] H. Zhang and Y. Vorobeychik, "Submodular optimization with routing constraints," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[50] Y. Chen, L. Zhao, Y. Zhang, S. Huang, and G. Dissanayake, "Anchor selection for slam based on graph topology and submodular optimization," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 329–350, 2021.

[51] H. Kretzschmar, C. Stachniss, and G. Grisetti, "Efficient information-theoretic graph pruning for graph-based slam with laser range finders," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 865–871.

[52] K. Khosoussi, M. Giamou, G. S. Sukhatme, S. Huang, G. Dissanayake, and J. P. How, "Reliable graphs for slam," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 260–298, 2019.

[53] R. Bai, S. Yuan, H. Guo, P. Yin, W.-Y. Yau, and L. Xie, "Multi-robot active graph exploration with reduced pose-slam uncertainty via submodular optimization," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 10 229–10 236.

[54] M. Ramesh, F. Imeson, B. Fidan, and S. L. Smith, "Approximate environment decompositions for robot coverage planning using submodular set cover," *arXiv preprint arXiv:2409.03120*, 2024.

[55] Y.-S. Li and K.-S. Tseng, "Computation-aware multi-object search in 3d space using submodular tree," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5956–5962.

[56] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[57] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[58] K. Ebadi, L. Bernreiter, H. Biggie, G. Catt, Y. Chang, A. Chatterjee, C. E. Denniston, S.-P. Deschênes, K. Harlow, S. Khattak *et al.*, "Present and future of slam in extreme environments: The darpa subt challenge," *IEEE Transactions on Robotics*, 2023.

[59] D. Lee, M. Jung, W. Yang, and A. Kim, "Lidar odometry survey: recent advancements and remaining challenges," *Intelligent Service Robotics*, vol. 17, no. 2, pp. 95–118, 2024.

[60] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[61] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," *Georgia Institute of Technology, Tech. Rep*, vol. 2, p. 4, 2012.

[62] A. Agha, K. Otsu, B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund *et al.*, "Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge," *arXiv preprint arXiv:2103.11470*, 2021.

[63] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart *et al.*, "Cerberus in the darpa subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[64] J. Bayer, P. Čížek, and J. Faigl, "Autonomous multi-robot exploration with ground vehicles in darpa subterranean challenge finals." *Field Robotics*, vol. 3, no. 1, pp. 266–300, 2023.

[65] N. Kottege, J. Williams, B. Tidd, F. Talbot, R. Steindl, M. Cox, D. Frousheger, T. Hines, A. Pitt, B. Tam *et al.*, "Heterogeneous robot teams with unified perception and autonomy: How team csiro data61 tied for the top score at the darpa subterranean challenge," *IEEE Transactions on Field Robotics*, 2024.

[66] T. Tuna, J. Nubert, Y. Nava, S. Khattak, and M. Hutter, "X-icp: Localizability-aware lidar registration for robust localization in extreme environments," *IEEE Transactions on Robotics*, 2023.

[67] F. Han, H. Zheng, W. Huang, R. Xiong, Y. Wang, and Y. Jiao, "Dams-lio: A degeneration-aware and modular sensor-fusion lidar-inertial odometry," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2745–2751.

[68] Y. Huang, T. Shan, F. Chen, and B. Englot, "Disco-slam: Distributed scan context-enabled multi-robot lidar slam with two-stage global-local graph optimization," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1150–1157, 2021.

[69] Y. Chang, K. Ebadi, C. E. Denniston, M. F. Ginting, A. Rosinol, A. Reinke, M. Palieri, J. Shi, A. Chatterjee, B. Morrell *et al.*, "Lamp 2.0: A robust multi-robot slam system for operation in challenging large-scale underground environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9175–9182, 2022.

[70] A. D. Krawciw, J. Sehn, and T. D. Barfoot, "Change of scenery: Unsupervised lidar change detection for mobile robots," in *2024 21st Conference on Robots and Vision (CRV)*, 2024.

[71] L. Carlone, A. Kim, T. Barfoot, D. Cremers, and F. Dellaert, Eds., *SLAM Handbook. From Localization and Mapping to Spatial Intelligence*. Cambridge University Press, 2025.

[72] Y. Cai, W. Xu, and F. Zhang, "ikd-tree: An incremental kd tree for robotic applications," *arXiv preprint arXiv:2102.10808*, 2021.

[73] J. Zhang, S. Singh *et al.*, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[74] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[75] Y. Yue, P. N. Senarathne, C. Yang, J. Zhang, M. Wen, and D. Wang, "Hierarchical probabilistic fusion framework for matching and merging of 3-d occupancy maps," *IEEE Sensors Journal*, vol. 18, no. 21, pp. 8933–8949, 2018.

[76] Y. Yue, C. Yang, Y. Wang, P. C. N. Senarathne, J. Zhang, M. Wen, and D. Wang, "A multilevel fusion system for multirobot 3-d mapping using heterogeneous sensors," *IEEE Systems Journal*, vol. 14, no. 1, pp. 1341–1352, 2019.

[77] S. Garg, N. Sünderhauf, F. Dayoub, D. Morrison, A. Cosgun, G. Carneiro, Q. Wu, T.-J. Chin, I. Reid, S. Gould *et al.*, "Semantics for robotic mapping, perception and interaction: A survey," *Foundations and Trends® in Robotics*, vol. 8, no. 1–2, pp. 1–224, 2020.

[78] Y. Cai, F. Kong, Y. Ren, F. Zhu, J. Lin, and F. Zhang, "Occupancy grid mapping without ray-casting for high-resolution lidar sensors," *IEEE Transactions on Robotics*, 2023.

[79] S.-K. Kim, A. Bouman, G. Salhotra, D. D. Fan, K. Otsu, J. Burdick, and A.-a. Aghamohammadi, "Plgrim: Hierarchical value learning for large-scale exploration in unknown environments," in *Proceedings of the international conference on automated planning and scheduling*, vol. 31, 2021, pp. 652–662.

[80] J. Bayer and J. Faigl, "Decentralized topological mapping for multi-robot autonomous exploration under low-bandwidth communication," in *2021 European Conference on Mobile Robots (ECMR)*. IEEE, 2021, pp. 1–7.

[81] Z. Zhang, J. Yu, J. Tang, Y. Xu, and Y. Wang, "Mr-topomap: Multi-robot exploration based on topological map in communication restricted environment," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 794–10 801, 2022.

[82] Z. Dong, J. Pflueger, L. Jung, D. Thorne, P. R. Osteen, C. S. Robison, B. T. Lopez, and M. Everett, "Lidar inertial odometry and mapping using learned registration-relevant features," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 359–366.

[83] D. Thorne, N. Chan, Y. Ma, C. S. Robison, P. R. Osteen, and B. T. Lopez, "Submodular optimization for keyframe selection & usage in slam," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 5033–5039.

[84] X. Chen, T. Läbe, A. Milioto, T. Röhling, J. Behley, and C. Stachniss, "Overlapnet: A siamese network for computing lidar scan similarity with applications to loop closing and localization," *Autonomous Robots*, pp. 1–21, 2022.

[85] J. Arce, N. Vödisch, D. Cattaneo, W. Burgard, and A. Valada, "Padloc: Lidar-based deep loop closure detection and registration using panoptic attention," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1319–1326, 2023.

[86] J. Komorowski, "Minkloc3d: Point cloud based large-scale place recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1790–1799.

[87] C.-J. Lin, S. Garg, T.-J. Chin, and F. Dayoub, "Robust scene change detection using visual foundation models and cross-attention mechanisms," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 8337–8343.

[88] E. Martinson and P. Lauren, "Meaningful change detection in indoor environments using clip models and nerf-based image synthesis," in *2024 21st Conference on Robots and Vision (CRV)*, 2024, pp. 603–610.

[89] D. Lague, N. Brodu, and J. Leroux, "Accurate 3d comparison of complex topography with terrestrial laser scanner: Application to the rangitikei canyon (nz)," *ISPRS journal of photogrammetry and remote sensing*, vol. 82, pp. 10–26, 2013.

[90] G. Kim and A. Kim, "Remove, then revert: Static point cloud map construction using multiresolution range images," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 758–10 765.

[91] D. C. Herraez, M. Zeller, D. Wang, J. Behley, M. Heidingsfeld, and C. Stachniss, "Multi-session mapping and long-term localization for autonomous vehicles using radar," *IEEE Robotics and Automation Letters*, vol. 11, no. 3, pp. 2522–2529, 2026.

[92] W. Ali, Y. Cai, P. Jensfelt, and T.-M. Nguyen, "Probper-lilo: Probabilistic persistency modeling for life-long mapping," *IEEE Robotics and Automation Letters*, vol. 11, no. 3, pp. 2530–2537, 2026.

[93] J. Rowell, L. Zhang, and M. Fallon, "LiSTA: Geometric object-based change detection in cluttered environments," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[94] J. Lee, W. Chung, and J. Kim, "Lidar-based long-term mapping in snow-covered environments," *Sensors*, vol. 25, no. 21, p. 6805, 2025.

[95] J. Qian, V. Chatrath, J. Yang, J. Servos, A. P. Schoellig, and S. L. Waslander, "POCD: Probabilistic object-level change detection and volumetric mapping in semi-static scenes," in *Robotics: Science and Systems (RSS)*, 2022.

[96] L. Wellhausen, R. Dubé, A. Gawel, R. Siegwart, and C. Cadena, "Reliable real-time change detection and mapping for 3d lidars," in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, 2017, pp. 81–87.

[97] S. Jang, A. J. Lee, I. M. A. Nahrendra, and H. Myung, "Chamelion: Reliable change detection for long-term lidar mapping in transient environments," *IEEE Robotics and Automation Letters*, 2026.

[98] A. Krawciw, S. Lilge, and T. D. Barfoot, "Lasersam: Zero-shot change detection using visual segmentation of spinning lidar," *arXiv preprint arXiv:2402.10321*, 2024.

[99] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[100] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *European conference on computer vision*. Springer, 2024, pp. 38–55.

[101] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson *et al.*, "Sam 2: Segment anything in images and videos," in *International Conference on Learning Representations*, 2025.

[102] S. Zhao, G. Chen, H. Zhu, Z. Li, C. Yao, N. Zantout, S. Kim, W. Wang, J. Zhang, and S. Scherer, "Supermap: A spatio-temporal slam system for visual-language navigation," in *Proceedings of Robotics: Science and Systems (RSS)*, 2026.

[103] K. Vidanapathirana, M. Ramezani, P. Moghadam, S. Sridharan, and C. Fookes, "Logg3d-net: Locally guided global descriptor learning for 3d place recognition," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2215–2221.

[104] K. Chen, R. Nemiroff, and B. T. Lopez, "Direct lidar-inertial odometry and mapping: Perceptive and connective slam," *arXiv preprint arXiv:2305.01843*, 2023.

[105] H. Lin and J. Bilmes, "A class of submodular functions for document summarization," in *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, 2011, pp. 510–520.

[106] S. Li, "Concise formulas for the area and volume of a hyperspherical cap," *Asian Journal of Mathematics & Statistics*, vol. 4, no. 1, pp. 66–70, 2010.

[107] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[108] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization: Identifying representative elements in massive data," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[109] K. J. Doherty, D. M. Rosen, and J. J. Leonard, "Spectral measurement sparsification for pose-graph slam," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 01–08.

[110] K. Chen, B. T. Lopez, A.-a. Agha-mohammadi, and A. Mehta, "Direct lidar odometry: Fast localization with dense point clouds," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2000–2007, 2022.

[111] K. Ebadi, M. Palieri, S. Wood, C. Padgett, and A.-a. Agha-mohammadi, "Dare-slam: Degeneracy-aware and resilient loop closing in perceptually-degraded environments," *Journal of Intelligent & Robotic Systems*, vol. 102, pp. 1–25, 2021.

[112] A. Tagliabue, J. Tordesillas, X. Cai, A. Santamaria-Navarro, J. P. How, L. Carlone, and A.-a. Agha-mohammadi, "Lion: Lidar-inertial observability-aware navigator for vision-denied environments," in *Experimental robotics: The 17th international symposium*. Springer, 2021, pp. 380–390.

[113] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically stable sampling for the icp algorithm," in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.* IEEE, 2003, pp. 260–267.

[114] D. Thorne, N. Chan, C. S. Robison, P. R. Osteen, and B. T. Lopez, "Optmap: Geometric map distillation via submodular maximization," *arXiv preprint arXiv:2512.07775*, 2025.

[115] Y. Tao, M. Á. Muñoz-Bañón, L. Zhang, J. Wang, L. F. T. Fu, and M. Fallon, "The oxford spires dataset: Benchmarking large-scale lidar-visual localisation, reconstruction and radiance field methods," *International Journal of Robotics Research*, 2025.

[116] D. Thorne, S. J. C. Chua, N. Joshi, A. Wong, C. S. Robison, P. R. Osteen, and B. T. Lopez, "Online geometric change detection via scene decomposition," 2026.

[117] C. Campos *et al.*, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE transactions on robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[118] S. Lee, H. Lim, and H. Myung, "Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3d point cloud," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 276–13 283.

[119] T. Kailath, "The divergence and bhattacharyya distance measures in signal selection," *IEEE transactions on communication technology*, vol. 15, no. 1, pp. 52–60, 1967.