

ADAPTIVE DOWNSAMPLING AND REFINEMENT FOR FAST ITERATIVE
CLOSEST POINT

by

NEHA BHATT

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTERS OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

2026

Oakland University
Rochester, Michigan

Thesis Advisory Committee:

Dr. Jun Chen, Ph.D., Chair
Dr. Jared Strader, Ph.D.
Dr. Khalid Mirza, Ph.D.

© by NEHA BHATT, 2026
All rights reserved

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my thesis advisor, Dr. Jun Chen, Associate Professor at Oakland University, for his guidance, encouragement, and thoughtful feedback throughout this work. His direction and support were instrumental in shaping the technical approach and maintaining academic rigor from initial concept to final manuscript.

I would also like to thank the faculty and staff at Oakland University for providing the resources and learning environment that supported my graduate studies and this research.

Finally, I am grateful to my family and friends for their continued encouragement and support throughout my time at Oakland University.

NEHA BHATT

ABSTRACT

ADAPTIVE DOWNSAMPLING AND REFINEMENT FOR FAST ITERATIVE CLOSEST POINT

by

Neha Bhatt

Adviser: Dr. Jun Chen, Ph.D.

Three-dimensional point cloud registration is a foundational operation in numerous engineering and computing domains, including autonomous vehicle navigation, robotic manipulation, simultaneous localization and mapping (SLAM), and 3D scene reconstruction. The goal of rigid registration is to estimate an optimal rotation and translation that aligns a source point cloud to a target point cloud. The Iterative Closest Point (ICP) algorithm has long served as the standard method for this task, alternating between correspondence establishment through nearest-neighbor queries and transformation optimization via singular value decomposition. Despite its widespread adoption, ICP is constrained by two fundamental limitations that restrict its applicability to dense, real-world datasets. First, ICP exhibits a linear convergence rate, requiring a large number of iterations to reach an accurate solution, particularly when the initial misalignment between source and target is significant. Second, the per-iteration cost is dominated by KD-tree nearest-neighbor queries computed over every point in the source cloud, a cost that scales linearly with source cloud size and becomes a critical runtime bottleneck on the dense point clouds produced by modern LiDAR sensors.

This thesis proposes a modified ICP method that achieves fast convergence while maintaining robust alignment. An adaptive threshold-based downsampling technique is introduced that selectively reduces the number of source points processed during early

iterations, decreasing the computational cost of each iteration. The threshold is initialized proportional to the minimum inter-point spacing and halved at each iteration, producing a coarse-to-fine alignment progression. A dual stabilization mechanism incorporating Anderson Acceleration ensures monotonic energy decrease by evaluating all accepted updates against the full-cloud registration energy. This is followed by a full-cloud refinement stage designed to recover alignment accuracy lost during the downsampling phase.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	iii
ABSTRACT	iv
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF ABBREVIATIONS	xii
CHAPTER ONE	
INTRODUCTION	1
1.1. Motivations	1
1.2. Problem Formulation	2
1.2.1. Iterative Closest Point	2
1.3. Contributions	3
1.3.1. Outline	4
CHAPTER TWO	
BACKGROUND	6
2.1. Background	6
2.1.1. Point Cloud Registration and Iterative Closest Point	6
2.1.2. Variants of ICP	7
2.1.3. Acceleration Strategies for ICP	8
2.1.4. Motivation for Threshold-based downsampling technique	10
2.1.5. KD-tree	10
2.1.6. Nearest-Neighbor Search	11

TABLE OF CONTENTS—Continued

2.1.7.	Singular Value Decomposition	11
2.1.8.	Anderson Acceleration	12
2.1.9.	Baseline ICP	13
CHAPTER THREE		
METHODOLOGY		14
3.1.	Approach	16
3.1.1.	Minimum Inter-Point Spacing and Threshold Initialization	17
3.1.2.	Threshold-Based Downsampling	17
3.1.3.	Correspondence Search and Transformation Estimation	18
3.1.4.	Dual-Tier Acceptance: Anderson Acceleration with Energy-Based Fallback Criteria	19
3.1.5.	Full-Cloud Refinement Phase	20
3.1.6.	Complete Algorithm	21
3.1.7.	Parameter Selection: κ and $k_{\max}$	21
3.1.8.	Design Rationale and Alternatives Considered	21
3.1.9.	Summary	25
3.2.	Experimental Setup	25
3.2.1.	Hardware Platform	25
3.2.2.	Static Benchmark Datasets	25
3.2.3.	LiDAR Data Collection	26

TABLE OF CONTENTS—Continued

3.2.4. Manual Outlier Removal	27
CHAPTER FOUR	
EXPERIMENTS AND RESULTS	29
4.1. Results	29
4.1.1. Static Dataset Results	29
4.1.2. Runtime and Registration Accuracy	30
4.1.3. Sensitivity Analysis	31
4.1.4. LiDAR Dataset Results	34
4.1.5. Runtime and Accuracy Comparison	35
4.1.6. Parameter Discussions	37
CHAPTER FIVE	
CONCLUSIONS AND FUTURE WORK	40
5.1. Conclusions	40
5.1.1. Summary of Key Findings	41
5.1.2. Limitations	42
5.1.3. Practical Implications	42
5.2. Future Work	43
5.2.1. Extension to Point-to-Plane Method	43
5.2.2. Curvature-Aware and Geometry-Preserving Downsampling	44
5.2.3. Learned Adaptive Threshold Selection	44
5.2.4. Integrating Within a Full SLAM Pipeline	45

TABLE OF CONTENTS—Continued

5.2.5.	GPU Implementation	45
5.2.6.	Extension to Non-Rigid and Multi-Modal Registration	46
REFERENCES		47

LIST OF TABLES

Table 4.1	Comparison of Algorithm 3.2 versus baseline Fast ICP and Robust ICP [1] on static datasets.	32
Table 4.2	Sensitivity with respect to κ , on the Bunny Dataset with $k_{\max} = 8$.	33
Table 4.3	Sensitivity with respect to $k_{\max}$, on the Bunny dataset, with $\kappa = 16$.	33
Table 4.4	Sensitivity with respect to κ on the Buddha dataset, with $k_{\max} = 8$.	33
Table 4.5	Sensitivity with respect to $k_{\max}$ on the Buddha dataset, with $\kappa = 16$.	33
Table 4.6	Target sub-map and source scan timestamps for each LiDAR segment.	35
Table 4.7	Comparison of Algorithm 3.2 versus baseline Fast ICP and Robust ICP [1] on LiDAR Datasets.	37

LIST OF FIGURES

Figure 3.1	Ford Mustang Mach-E for on-road LiDAR data collection.	27
Figure 4.1	Illustration of the coarse-to-fine behavior of Algorithm 3.2 using the Stanford Bunny. (a) Original source and target clouds; (b)-(f) Adaptive downsampling phase; (g) Refinement phase.	30
Figure 4.2	Registration results on static datasets. <i>Top</i> : Initial overlap of target (green) and source (blue). <i>Bottom</i> : Final alignment by Algorithm 3.2.	34
Figure 4.3	LiDAR experiments on Technology Drive, Auburn Hills, MI, USA. <i>Top</i> : Drive route map and aggregated LiDAR point cloud. <i>Middle</i> : Initial overlap of target (green) and source (blue) for West, Middle, and East segments. <i>Bottom</i> : Final alignment achieved by Algorithm 3.2.	36

LIST OF ABBREVIATIONS

ICP	Iterative Closest Point
SVD	Singular Value Decomposition
SLAM	Simultaneous Localization and Mapping
NNS	Nearest-neighbor Search
RMSE	Root Mean Square Error
EPFL	École Polytechnique Fédérale de Lausanne

CHAPTER ONE

INTRODUCTION

1.1 Motivations

3-D point cloud registration is a foundational operation in many modern engineering systems. Autonomous vehicles rely on registration to align successive LiDAR scans for localization and mapping. The ICP algorithm has remained a standard method for rigid point cloud registration because of its reliable convergence behavior. However, ICP is constrained by a linear convergence rate that requires many iterations to come to an optimal solution, and a per-iteration nearest-neighbor search that scales linearly with source cloud size and that in turn increases runtime on dense datasets, making standard ICP unsuitable for time-sensitive applications such as real-time LiDAR odometry. During early iterations, the transformation estimate is far from optimal and the correspondences established at this stage are approximate queries. A well-chosen subset is sufficient to estimate a meaningful transformation update, and the distance threshold is progressively reduced to allow additional source points to participate in subsequent iterations.

Multi-resolution ICP methods have previously been developed to improve registration efficiency by aligning point clouds at different levels of spatial resolution [2]. These approaches typically begin with a coarse representation of the point clouds and progressively introduce finer geometric detail to improve alignment accuracy. [3] investigated efficient ICP variants, including point-selection and correspondence strategies, demonstrating ways to reduce computational complexity during registration. The proposed method follows a related coarse-to-fine principle, as both approaches aim to reduce computational effort during early registration and incorporate additional geometric information during later stages. However, conventional multi-resolution methods commonly employ multiple representations of the point clouds at different resolution

levels, whereas the proposed method reconstructs a filtered subset at each coarse iteration using distance-threshold comparisons. The threshold is initialized using the minimum inter-point spacing of the source cloud and reduced at successive iterations, generally allowing more points to participate in correspondence estimation. The motivation of this thesis is therefore to develop a registration method that adapts the source cloud size based on a progressively decreasing distance threshold, reducing the per-iteration query cost while aiming to maintain final registration accuracy. Furthermore, candidate transformations are evaluated against the full-cloud registration energy before acceptance, followed by a full-cloud refinement phase.

1.2 Problem Formulation

1.2.1 Iterative Closest Point

Given two point clouds $P = \{\mathbf{p}_i\}_{i=1}^M$ and $Q = \{\mathbf{q}_j\}_{j=1}^N$, the goal of ICP is to estimate a rigid transformation $(\mathbf{R}, \mathbf{t})$ that minimizes:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^M \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_{c(i)} \right\|^2, \quad (1.1)$$

where $\mathbf{R} \in SO(3)$ is a rotation matrix, $\mathbf{t} \in \mathbb{R}^3$ is a translation vector, and

$$c(i) = \arg \min_{j \in \{1, \dots, N\}} \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_j \right\| \quad (1.2)$$

assigns each source point to its nearest neighbor in Q . In this thesis, the term registration energy refers to the ICP objective function $E(\mathbf{R}, \mathbf{t})$, defined as the sum of squared Euclidean distances between transformed source points and their corresponding nearest-neighbor points in the target cloud. A lower registration energy indicates a smaller total squared correspondence error and therefore a better alignment to this objective. Standard ICP solves this by fixing the correspondences and updating $(\mathbf{R}, \mathbf{t})$ via singular value decomposition (SVD). While this approach is guaranteed to decrease the objective monotonically, convergence to the globally optimal solution is not guaranteed and

depends heavily on the quality of the initial transformation estimate. Moreover, the per-iteration nearest-neighbor search over all P source points is the dominant computational cost, particularly when P is large.

This thesis addresses the following problem: given a dense source cloud and a target cloud, compute a rigid transformation $(\mathbf{R}, \mathbf{t})$ that maintains alignment accuracy comparable to standard ICP while substantially reducing total computation time, especially on dense point clouds.

1.3 Contributions

This thesis is based on the author’s previously published research on adaptive downsampling and refinement for fast Iterative Closest Point [4]. The thesis builds upon this work by presenting the algorithmic formulation, implementation details, experimental evaluation, and discussion within a unified framework. This thesis makes the following contributions toward fast rigid point cloud registration:

- **Adaptive threshold-based downsampling strategy:** A source cloud subsampling method is proposed in which the distance threshold used to filter source points is initialized proportional to the minimum inter-point spacing $d_{\min}$ and is halved at each iteration. This produces a coarse-to-fine alignment progression by selecting source points through sequential distance comparisons with the most recently accepted reference point. The resulting subset reduces KD-tree nearest-neighbor search query volume during early iterations.
- **Two-phase optimization framework (adaptive downsampling phase 1 and a refinement phase 2 on full source cloud):** In the coarse phase of adaptive downsampling based on a distance threshold derived from minimum inter-point spacing, transformation estimates are computed on filtered subsets while the registration energy is evaluated on the full source cloud to ensure monotonic energy

decrease at each accepted iteration. A subsequent full-cloud refinement phase 2 recovers fine-grained alignment accuracy using the complete source cloud.

- **Comprehensive experimental evaluation:** Extensive experiments are conducted on both static and dynamic datasets, including Stanford Mesh models, the EPFL statue dataset, and real-world LiDAR sequences containing sensor noise and motion distortion. The proposed method achieves speedups of 13–58% on four of five static benchmark datasets and 44–64% on all three LiDAR datasets, while maintaining root mean square error (RMSE) comparable to the point-to-point Fast ICP baseline.

1.3.1 Outline

This thesis proposes a modified ICP method that achieves fast convergence while maintaining robust alignment. Specifically, we introduce an adaptive threshold-based downsampling technique that selectively reduces the number of source points processed during early iterations, decreasing the computational cost for each iteration. This is followed by a full-cloud refinement stage designed to recover alignment accuracy and mitigate any degradation introduced by the downsampling phase. The remainder of this manuscript is organized as follows. Chapter 2 reviews the background required for this work, including the point cloud registration and standard ICP algorithm, variants of ICP, KD-tree nearest-neighbor search, Anderson Acceleration, and existing ICP variants and downsampling strategies. Chapter 3 presents the proposed algorithm in detail, including the adaptive threshold-based downsampling procedure, the dual stabilization mechanism, and the full-cloud refinement phase. Chapter 4 describes the experimental setup and presents results on static benchmark datasets, including Stanford Mesh models and the EPFL statue dataset, experiments on real-world LiDAR sequences, parameter discussion, sensitivity analysis, runtime, and registration accuracy. Finally, Chapter 5 summarizes the key conclusions of this study and outlines directions for future work, including extensions

to non-rigid registration, comparison with broader benchmarks and integration in SLAM pipelines.

CHAPTER TWO

BACKGROUND

2.1 Background

2.1.1 Point Cloud Registration and Iterative Closest Point

Rigid point cloud registration [5–7] is the problem of estimating a rotation and translation that best align a source point cloud with a target point cloud, and it underlies a wide range of applications in robotics, 3D reconstruction, and autonomous navigation, including simultaneous localization and mapping (SLAM). The Iterative Closest Point (ICP) algorithm has long served as the cornerstone method for addressing this problem, alternating between correspondence establishment through closest point queries using KD-tree [8] and transformation optimization that minimizes the sum of squared distances between corresponding points.

Given two point clouds $P = \{\mathbf{p}_i\}_{i=1}^M$ and $Q = \{\mathbf{q}_j\}_{j=1}^N$, the goal of ICP is to estimate a rigid transformation $(\mathbf{R}, \mathbf{t})$ that minimizes:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^M \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_{c(i)} \right\|^2, \quad (2.1)$$

where $\mathbf{R} \in SO(3)$ is a rotation matrix, $\mathbf{t} \in \mathbb{R}^3$ is a translation vector, and

$$c(i) = \arg \min_{j \in \{1, \dots, N\}} \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_j \right\| \quad (2.2)$$

assigns each source point to its nearest neighbor in Q . The ICP algorithm, introduced by [5], remains the foundational method for solving this problem. ICP proceeds by alternating between two steps until convergence: (i) a correspondence step, in which each source point is matched to its nearest neighbor in the target cloud, typically accelerated using a KD-tree built over the target, and (ii) an alignment step, in which a closed-form rigid transformation minimizing the sum-of-squares error over the current

correspondences is computed, most commonly via SVD [9] of the cross-covariance matrix of the corresponding point pairs. These two steps are repeated, with the transformation applied cumulatively to the source cloud, until the registration energy converges or a maximum number of iterations is reached. While ICP helps in aligning the two point clouds, it has been constrained by certain limitations such as slow convergence due to its linear convergence rate [10], and its per-iteration cost is dominated by KD-tree nearest-neighbor queries on every source point [3, 11], which scales with the source cloud size and becomes a runtime bottleneck on dense point clouds.

2.1.2 Variants of ICP

A large body of work has extended the original ICP formulation to improve its accuracy, robustness to noise and outliers, and convergence behavior [3]. Point-to-plane variants replace the point-to-point error metric with a distance measured along the target surface normal, which improves convergence on smooth surfaces [12]. Symmetric formulations that consider both source and target surface information have also been proposed to improve stability [13]. To handle noisy or partially overlapping point clouds, robust cost functions and outlier-aware formulations have been introduced in place of the standard least-squares objective, reducing the influence of incorrect correspondences on the estimated transformation [14, 15]. Other techniques based on graphs and geometric transformations such as robust estimation with outliers [16], geometric features for robust superpoint matching [17], multi-resolution schemes [13], and graph-based correspondence formulations [18] have also been explored in the journey of point cloud registration. While these variants primarily target accuracy and robustness, they generally retain the per-iteration correspondence search over the full point set as the dominant computational cost, motivating a separate line of work focused explicitly on acceleration.

2.1.3 Acceleration Strategies for ICP

Because the nearest-neighbor search is the dominant cost per iteration, several strategies have been proposed to either reduce the number of iterations required or reduce the cost of each iteration. Acceleration techniques have also received considerable attention. Anderson-accelerated ICP applies fixed-point iteration theory to speed up convergence. Anderson Acceleration [19, 20] is a quasi-Newton technique originally developed for accelerating fixed-point iterations, and was first applied to the ICP problem by [21], who showed that treating each ICP update as a fixed-point iteration and extrapolating using a history of previous iterates can substantially reduce the number of iterations needed for convergence. Building on this idea, [1] proposed an Anderson-accelerated ICP formulation defined over the Lie algebra of $SE(3)$, coupled with an energy-based stabilization scheme that falls back to the SVD update whenever the accelerated step fails to reduce the registration energy, guaranteeing monotonic convergence. This energy-based acceleration criterion is important because accelerated updates are not guaranteed to decrease the registration error at every step, and thus the fallback criterion makes sure there is always decrease in registration energy. Despite these advances, each iteration of [1] still computes correspondences over the entire source cloud, so its practical convergence behavior remains sensitive to the quality of the initial correspondences and to how aggressively the acceleration is applied.

There is another technique of downsampling and voxelization, that operates on a reduced representation of the point cloud rather than reducing the iteration count. Voxel grid filtering and uniform or random subsampling are commonly used as a preprocessing step to reduce point density prior to registration [3]. The Voxelized Generalized ICP (VGICP) algorithm [22] extends the Generalized ICP framework by representing local neighborhoods with voxel statistics, avoiding explicit nearest-neighbor search while

retaining registration accuracy. Normal distribution Transform (NDT)-based methods [23] take a related approach, replacing explicit point correspondences with a probabilistic voxel-based representation of the target distribution. In the LiDAR odometry and SLAM domain, KISS-ICP [24] combines point-to-point ICP with an adaptively chosen correspondence threshold and voxel-based downsampling, and has become a widely used baseline for LiDAR odometry due to its simplicity and robustness. More broadly, foundational LiDAR odometry systems such as LOAM [25] and its ground-optimized variant [26], along with more recent elastic and tightly-coupled LiDAR-inertial systems [16], [27] have advanced state-of-the-art robustness. The contribution of this thesis is complementary to these SLAM-level systems; rather than addressing a full SLAM pipeline, it focuses on improving the ICP registration primitive itself, which serves as a building block within many of these methods. A common limitation of these downsampling-based approaches, however, is that the reduction in point density is typically fixed — determined once, in advance, by a chosen voxel resolution or sampling ratio — rather than adapted over the course of the registration process itself. A fixed resolution that is coarse enough to yield a meaningful speedup may discard points that are informative for alignment, while a resolution that is fine enough to preserve alignment accuracy may not sufficiently reduce the cost of early iterations, when the transformation is still far from optimal and precise correspondences are least needed.

Learning-based registration approaches such as Deep Closest Point [28] have also emerged as an alternative to classical optimization-ICP, using learned feature embedding to establish correspondences directly. These methods can be effective in the presence of partial overlap or large initial misalignment, but they require training data representative of the target domain and are generally considered complementary to, rather than a replacement for, classical ICP-based pipelines, which remain attractive for their lack of training requirements and their well-understood convergence properties.

2.1.4 Motivation for Threshold-based downsampling technique

Across this body of work, two main approaches have been used to accelerate ICP: reducing the number of iterations required for convergence, primarily through fixed-point acceleration techniques such as Anderson Acceleration and other methods [1, 18–20] and reducing the cost per iteration, primarily through fixed-resolution downsampling or voxelization of the point cloud [6], [3], [22], [23], [24]. What remains comparatively unexplored is a strategy that varies the working point-cloud density adaptively and progressively increasing over the course of registration itself; operating on a reduced subset of the source cloud selected through sequential distance-based filtering during early iterations, when the transformation is coarse and precise correspondences offer little benefit, and progressively densifying the working set as the alignment improves, without ever relaxing the underlying optimization objective, which must still be evaluated over the complete source cloud to avoid degrading final accuracy. This gap motivates the adaptive threshold-based downsampling and dual-phase refinement framework developed in this thesis, which is designed to be combined with existing convergence-acceleration techniques such as Anderson Acceleration rather than replace them, targeting the per-iteration correspondence-search cost that remains the dominant bottleneck in ICP registration. Although consecutive accepted points satisfy the distance threshold, minimum separation and uniform spatial coverage between every pair of retained points are not guaranteed.

2.1.5 KD-tree

A KD-tree is a binary tree that recursively partitions a point set by alternating a splitting dimension at each level of recursion. Given a set of points $Q = \{q_j\}_{j=1}^N \subset \mathbb{R}^3$ the tree is built as follows: at the root, the points are sorted along one coordinate axis (say, x),

and the median point along that axis is chosen as the splitting point,

$$q_{\text{split}} = \text{median}_j (q_j^{(x)}), \quad (2.3)$$

where $q_j^{(x)}$ denotes the x -coordinate of point q_j . All remaining points with a smaller x -coordinate than q_{split} are placed in the left subtree, and all points with a larger x -coordinate are placed in the right subtree. This process is then repeated recursively within each subtree, cycling through the y and z axes at successive depths, until each leaf node contains a single point (or a small fixed number of points, in practice).

2.1.6 Nearest-Neighbor Search

Nearest-Neighbor Search (NNS), as a form of proximity search, is the optimization problem of finding the point in a given set that is closest (or most similar) to a given point. After transforming the source point cloud, ICP must establish a correspondence between each source point and a point in the target point cloud. This correspondence is typically determined using nearest-neighbor search, which identifies the target point that has the minimum Euclidean distance to a given source point. The resulting correspondence set is then used by the subsequent alignment step to estimate the rigid transformation between the two point clouds.

2.1.7 Singular Value Decomposition

Once correspondences have been established via the KD-tree search and nearest-neighbor queries of Sections 2.1.5 and 2.1.6, the alignment step of ICP must solve for the rigid transformation $(\mathbf{R}, \mathbf{t})$ that minimizes the sum-of-squares energy of Eq. (2.1) for a fixed set of correspondences. This subsection derives the closed-form solution to this subproblem through SVD, following the classical result of [9].

2.1.7.0.1 Problem setup. For a fixed correspondence assignment, write p_i for each source point and $q_i \equiv q_{c(i)}$ for its assigned target point, so that the objective to be

minimized is

$$\min_{\mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3} \sum_{i=1}^M \left\| \mathbf{R} \mathbf{p}_i + \mathbf{t} - \mathbf{q}_{c(i)} \right\|^2. \quad (2.4)$$

2.1.8 Anderson Acceleration

A full ICP iteration correspondence search followed by SVD-based alignment can be viewed abstractly as a single application of a fixed-point map G to the current transformation estimate,

$$\mathbf{T}_{k+1} = G(\mathbf{T}_k), \quad (2.5)$$

where G encapsulates both steps of the ICP loop. Classical ICP simply iterates Eq. (2.5) directly, using only the most recent estimate $\mathbf{T}_k$ to compute the next one. This is a valid but often slow strategy. The direct fixed-point iteration converges only linearly, since it discards all information from earlier iterations once a new estimate is produced. Anderson Acceleration [19, 20] is a general technique for accelerating the convergence of fixed-point iterations of the form in Eq. (2.5), by using a short history of m previous iterates rather than only the most recent one. Define the residual of the fixed-point map at iteration k as

$$F_k = G(\mathbf{T}_k) - \mathbf{T}_k, \quad (2.6)$$

Anderson Acceleration forms the next iterate as a weighted combination of the m most recent function evaluations,

$$\mathbf{T}_{k+1} = \sum_{j=0}^m \theta_j^{(k)} G(\mathbf{T}_{k-j+1}), \quad (2.7)$$

where the coefficients $\theta_j(k)$ are chosen, at every iteration, to solve the linear least-square problem,

$$\theta_1^{(k)}, \dots, \theta_m^{(k)} = \arg \min_{\theta_1, \dots, \theta_m} \left\| F_k - \sum_{j=1}^m \theta_j (F_{k-j+1} - F_{k-j}) \right\|^2. \quad (2.8)$$

Anderson Acceleration of this form is known to behave as a quasi-Newton method for finding a root of the residual function F , and can improve the convergence rate of fixed-point iterations. Thus, Anderson Acceleration leverages an m -step history of past transformation estimates to compute an accelerated step, effectively bypassing slow linear convergence phases and dramatically reducing iteration overhead.

2.1.9 Baseline ICP

Sections 2.1.5–2.1.8 introduced the four components underlying a complete baseline ICP procedure: KD-tree correspondence, Nearest Neighbor Search for every source point to its nearest target point, SVD-based rigid alignment, and Anderson Acceleration with an energy-based fallback mechanism. In practice, these four components are combined into a single alternating loop - at every iteration, correspondences for the entire source cloud are first established via KD-tree nearest neighbor search; a candidate transformation is then computed in closed form via the SVD-based solution Eq. (2.4); this candidate is optionally refined using Anderson Acceleration Eq. (2.7), with the accelerated update accepted only if it reduces the registration energy of Eq. (2.1), and the unaccelerated SVD solution retained otherwise. This loop repeats until the change in registration energy falls below a tolerance or a maximum number of iterations is reached.

CHAPTER THREE

METHODOLOGY

This experiment is motivated by a practical bottleneck in deploying ICP-based registration in static point-cloud datasets and real-time robotics and SLAM pipelines. Classical ICP, introduced in Chapter 2, is attractive precisely because it requires no training data and offers a well-understood, monotonically convergent optimization procedure. Given an initial pose estimate, it reliably converges to an accurate rigid-body alignment between a source and target point cloud. This reliability, however, comes at a cost that becomes increasingly difficult to ignore as point cloud density grows. As established in Section 2.1.1, the per-iteration correspondence search - the step responsible for matching every source point to its nearest neighbor in the target cloud - scales directly with the number of points in the source cloud. On sparse, small point sets this cost is negligible, but on the dense point clouds increasingly common in modern 3D sensing, LiDAR scans, dense photogrammetric reconstructions, and mesh models - this per-iteration cost becomes a dominant factor in total registration time.

Existing acceleration strategies, reviewed in Chapter 2, address this cost from largely separate directions, each of which resolves only part of the problem. Convergence-acceleration techniques such as Anderson Acceleration (Section 2.1.8) treat the ICP update as a fixed-point iteration and extrapolate using a history of prior estimates, reducing the number of iterations required to reach convergence; however, they leave the cost of each individual iteration entirely unchanged, since every accepted step still requires a full correspondence search over the complete source cloud. Fixed-resolution downsampling and voxelization methods take the complementary approach, reducing the cost of each iteration by operating on a smaller or coarser representation of the point cloud; however, as discussed in Section 2.1.4, these methods fix the working resolution

once, in advance, and apply it uniformly across the entire registration run, regardless of how close the current transformation estimate already is to convergence. Neither direction, on its own, addresses the observation that early iterations - where the transformation is still far from optimal, have comparatively little to gain from a precise, full-resolution correspondence search, while later iterations, where fine-grained accuracy matters most, do.

This chapter presents an adaptive, two-phase registration framework designed to close this gap directly, by varying the density of the working point subset progressively within a single registration run rather than fixing it in advance. The framework begins with a reduced subset of the source cloud selected using a relatively large distance threshold during early iterations, when precise correspondences offer little benefit, and progressively densifies this working subset as the estimated transformation improves, while validating every candidate update against the full-cloud registration objective to ensure that each accepted transformation reduces the full-cloud registration energy. This design is intended to be complementary to, rather than a replacement for, existing convergence-acceleration techniques. For example, Anderson Acceleration is incorporated directly into the framework, so that the resulting method inherits the iteration-count benefits of prior acceleration work while also addressing the per-iteration correspondence search cost benefits from sparser subset formation of point clouds.

The remainder of this chapter is organized as follows. Section 3.1 presents the proposed framework in detail, its two-phase architecture, the adaptive threshold-based downsampling procedure and its initialization, the dual-tier acceptance criterion combining Anderson Acceleration and energy-based fallback, the full-cloud refinement phase, and the complete algorithm. Section 3.2 describes the hardware and datasets used to evaluate this framework.

3.1 Approach

Chapter 2 establishes that classical ICP is bottle necked by two coupled costs; one is linear convergence rate, and a per-iteration correspondence search whose cost scales with the size of the source cloud. Anderson Acceleration addresses the first cost but leaves the second untouched, while existing downsampling and voxelization methods address the second cost but fix the working resolution in advance rather than adapting it within a single registration run. This chapter presents the proposed framework, in which the adaptive downsampling phase 1 reduces the number of points involved in correspondence search during early iterations, when precise correspondences are least valuable, while progressively increasing that density as the estimated transformation improves, and while guaranteeing that every accepted update still reduces the true, full-cloud registration energy of Eq (2.1).

The proposed method consists of two sequential phases operating on the full source cloud P and target cloud Q .

1. **Phase 1: Adaptive Threshold Based Downsampling.** An initial threshold τ_0 is computed from the source cloud's own minimum inter-point spacing and halved at every iteration. At each iteration, a filtered subset $X \subseteq P$ is constructed at the current threshold, a candidate transformation is estimated on X along with Anderson Acceleration, and the candidate is accepted only if it reduces the full-cloud energy, with a dual-tier fallback to a classical SVD update.
2. **Phase 2: Full Cloud Refinement.** Once Phase 1 terminates, the algorithm performs additional iterations directly on the already transformed complete source cloud generated by Phase 1, without downsampling or acceleration, to recover any fine-grained accuracy that the coarse phase may have sacrificed.

The coarse phase supplies the computational savings of downsampling, the energy-based acceptance criterion preserves the correctness guarantee of classical ICP, and the refinement phase absorbs whatever accuracy the adaptive sampling might have not resolved on its own.

3.1.1 Minimum Inter-Point Spacing and Threshold Initialization

The coarse phase is parameterized by a distance threshold τ that controls how densely the source cloud is sampled at each iteration. This threshold is defined relative to the source cloud's own point density rather than an absolute unit. The initial threshold is then set as

$$\tau_0 = \kappa d_{\min} \quad (3.1)$$

where $d_{\min}$ is the minimum distance between two points in the source cloud and κ is a tunable parameter chosen so that τ_0 is large relative to the spatial extent of the cloud, ensuring that the very first coarse iteration operates on a small, reduced subset of P . At every subsequent coarse iteration the threshold is halved,

$$\tau_{k+1} = \frac{\tau_k}{2}, \quad (3.2)$$

so the distance criterion becomes less restrictive as the phase progresses, generally allowing more points to be selected, producing the coarse-to-fine progression during the coarse phase of the method. Because κ directly sets how sparse the very first iteration is, it is the primary parameter governing the trade-off between early-phase speed and the risk of retaining too few points to constrain the registration problem, a trade-off characterized empirically in Chapter 4.

3.1.2 Threshold-Based Downsampling

At each coarse iteration, the current threshold τ is used to construct a filtered subset $X \subseteq P$ via a spatial filtering algorithm. The filter uses a distance threshold τ to

Algorithm 3.1: Threshold-Based Downsampling

Input: Source cloud P , Threshold τ

Output: Filtered subset X

1. $X \leftarrow \{p_0\}, p_{\text{ref}} \leftarrow p_0$
 2. **for** $i = 1$ **to** N **do**
 3. **if** $\|p_i - p_{\text{ref}}\| \geq \tau$ **then**
 4. $X \leftarrow X \cup \{p_i\}, p_{\text{ref}} \leftarrow p_i$
 5. **end**
 6. **end**
 7. **return** X
-

determine whether a candidate point is retained based on its distance from the current reference point. Candidate points are considered in sequence, and a point is added to X only if its distance from the most recently accepted reference point is at least τ . Rather than checking every point already retained in X , the filter maintains a reference point p_{ref} , initialized to the first point of P . The candidate points are then considered once in sequence, and a point p_i is added to X , and simultaneously becomes the new reference point, only if the distance from the current reference point is at least τ . This guarantees that consecutive accepted points in the processing sequence are separated by at least τ . However, two nonconsecutive retained points may be closer than τ .

As τ shrinks according to Eq. (3.2), the distance-based selection criterion becomes less restrictive, allowing more points to be retained in X . Once $\tau \leq d_{\min}$, the separation constraint can no longer exclude any point of P , and X effectively recovers the full source cloud, which is one of the conditions used to terminate Phase 1.

3.1.3 Correspondence Search and Transformation Estimation

Given the filtered subset X at iteration k , correspondences are established by querying the KD-tree built over the target cloud Q , as described in Sections 2.1.5–2.1.6, assigning each point in X to its nearest neighbor in Q , rather than full source cloud P . Because as magnitude of X is less than P during early iterations, its correspondence step is

substantially cheaper than full-cloud query it replaces. A candidate transformation is then obtained by solving the SVD alignment of Section 2.1.7. The point cloud registration objective Eq (2.1) is solved using X as the source and Q as the target, yielding a candidate $\mathbf{T} = (\mathbf{R}, \mathbf{t})$ that minimizes the registration energy over the filtered subset alone. Restricting correspondence search and transformation estimation to X is what produces the computational savings of the coarse phase; however, it also means that transformation is optimal only with respect to the filtered subset, not the full-cloud P . The next step therefore validates this candidate against the true optimization objective before accepting it.

3.1.4 Dual-Tier Acceptance: Anderson Acceleration with Energy-Based Fallback Criteria

Before further refinement, the SVD-derived candidate from Section 3.1.3 is stored as a backup, $\mathbf{T}_{\text{backup}} \leftarrow \mathbf{T}$. In accordance with the approach introduced in [1], Anderson Acceleration Eq (2.6) is then applied to $\mathbf{T}$ using the history of transformation estimates, producing an accelerated candidate $\mathbf{T}_{\text{acc}}$. Because extrapolated updates are not guaranteed to reduce the true registration energy, the accelerated candidate is validated by recomputing correspondences and the registration energy over the entire source cloud P and not the filtered subset X , following the acceptance rule stated below

$$\mathbf{T} \leftarrow \begin{cases} \mathbf{T}_{\text{acc}}, & E_{\text{new}} < E_{\text{prev}} \\ \mathbf{T}_{\text{backup}}, & \text{otherwise,} \end{cases} \quad (3.3)$$

where E_{new} and E_{prev} denote the full-cloud energy of Eq. (2.1) evaluated at the current candidate and previous transformations, respectively. If the accelerated candidate is rejected, the method falls back to $\mathbf{T}_{\text{backup}}$ and re-evaluates the full-cloud energy a second time, E_{revert}

$$\text{accept } \mathbf{T}_{\text{backup}} \iff E_{\text{revert}} < E_{\text{prev}}. \quad (3.4)$$

This second check is what makes the fallback dual tier criterion. The method first attempts the faster-converging accelerated update, and only if that fails, it falls back to the guaranteed-descent SVD-update; only if both fail to improve on E_{prev} does the coarse phase terminate. This construction guarantees that every transformation accepted during Phase 1 strictly reduces the full-cloud registration energy even though the transformation is estimated on the reduced subset X which is what allows the coarse phase to use downsampling aggressively without ever compromising the original optimization objective of Eq (2.1).

3.1.5 Full-Cloud Refinement Phase

Phase 1 terminates under one of two conditions: either the threshold has decayed below the point spacing, $\tau \leq d_{\text{min}}$, meaning the filtered subset has effectively converged to the full-cloud; or neither tier of Eq (3.3)-(3.4) improved the energy, meaning the coarse phase reached a local plateau before exhausting the threshold schedule. Phase 2 is designed to behave differently depending on which of these two conditions triggered termination.

- If Phase 1 exhausted the full threshold cascade ($\tau_k \leq d_{\text{min}}$), the transformation is already close to a full-cloud optimum, and only a small number of full-cloud iterations are typically needed before the energy-based stopping criterion below is satisfied.
- If Phase 1 terminated early via the energy-plateau condition, the transformation may still be a meaningfully suboptimal full-cloud alignment, and Phase 2 is allotted up to k_{max} full-cloud iterations to recover additional accuracy.

In both cases, Phase 2 iteration recomputes correspondences and re-estimates the transformation using the complete source cloud P without downsampling or Anderson

Acceleration application. The transformation is accepted only if it continues to reduce the energy of the full source cloud.

3.1.6 Complete Algorithm

Algorithm 3.2 integrates the components described above into a single two-phase procedure.

3.1.7 Parameter Selection: κ and $k_{\max}$

The framework introduces two tunable parameters. κ (Eq. (3.1)) controls how sparse the very first coarse iteration is, larger values of κ increase the initial speedup potential but risk retaining too few points to adequately constrain the rigid-body estimate, which can destabilize the earliest iterations if pushed too far. $k_{\max}$ bounds the number of full-cloud refinement iterations available to Phase 2, trading off worst-case refinement runtime against the ability to recover from an unfavorable Phase 1 termination. Because both parameters interact with the geometry of the specific dataset being registered, including point density, curvature complexity, and initial misalignment all affect how quickly Phase 1 converges. Chapter 4 presents an empirical sensitivity analysis across both parameters to characterize this trade-off and to justify the values throughout the experimental evaluation.

3.1.8 Design Rationale and Alternatives Considered

Several design choices were considered for Algorithm 3.2 and this subsection explains why each was preferred over other alternatives.

3.1.8.0.1 Why full-cloud energy validation rather than subset-only validation. An alternate design would validate each candidate transformation only against the energy of the filtered subset X used to estimate it, which would be cheaper per iteration since it avoids the full-cloud correspondences recomputation in Eq. (3.3). This alternative was

Algorithm 3.2: Fast ICP with Adaptive Downsampling and Energy-Based Reversion

Input: Source cloud P , Target cloud Q , κ , $k_{\max}$

Output: Transformation $\mathbf{T}$

1. **Initialization:**
 2. Compute $d_{\min}$; $\tau \leftarrow \kappa d_{\min}$; Build KD-tree on Q
 3. Compute $\mathbf{T} \leftarrow I$; Initial energy E_{prev} on P using Eq 2.1
 4. % Phase 1: Adaptive threshold-based downsampling
 5. **while** $\tau \geq d_{\min}$ **do**
 6. **1. Correspondences on filtered subset**
 7. $X \leftarrow \text{FILTER}(P, \tau)$ via Algorithm 3.1
 8. Compute nearest neighbors for X in Q via KD-tree
 9. $\mathbf{T} \leftarrow$ minimizing (2.1) with X as source and Q as target
 10. **2. Transformation update**
 11. $\mathbf{T}_{\text{backup}} \leftarrow \mathbf{T}$
 12. $\mathbf{T} \leftarrow \text{AndersonAccel}(\mathbf{T})$
 13. **3. Evaluate energy on full cloud**
 14. Recompute nearest neighbors for P using $\mathbf{T}$
 15. Compute full-cloud registration energy E_{new} on P using (2.1)
 16. **4. Energy-based acceptance**
 17. **if** $E_{\text{new}} < E_{\text{prev}}$ **then**
 18. | $P \leftarrow \mathbf{T}P$, $E_{\text{prev}} \leftarrow E_{\text{new}}$
 19. **else**
 20. | $\mathbf{T} \leftarrow \mathbf{T}_{\text{backup}}$
 21. | Recompute nearest neighbors for P using $\mathbf{T}_{\text{backup}}$
 22. | Compute E_{revert} on P using (2.1)
 23. | **if** $E_{\text{revert}} < E_{\text{prev}}$ **then**
 24. | | $P \leftarrow \mathbf{T}P$, $E_{\text{prev}} \leftarrow E_{\text{revert}}$
 25. | **else**
 26. | | **break**
 27. | **end**
 28. **end**
 29. $\tau \leftarrow \tau/2$
 30. **end**
-

```

1. % Phase 2: Refinement
2.  $k \leftarrow 0$ 
3. while  $\tau < d_{\min}$  or  $k < k_{\max}$  do
4.    $\mathbf{T}_{\text{backup}} \leftarrow \mathbf{T}$ 
5.   Recompute nearest neighbors for  $P$  using  $\mathbf{T}$ 
6.    $\mathbf{T} \leftarrow$  minimizing (2.1)
7.   Compute full-cloud energy  $E$  on  $P$  using (2.1)
8.   if  $E > E_{\text{prev}}$  then
9.     break
10.  end
11.   $E_{\text{prev}} \leftarrow E, k \leftarrow k + 1$ 
12. end
13. return  $\mathbf{T}$ 

```

rejected because a transformation that reduces error on a sparse, non-representative subset of the cloud is not guaranteed to reduce error on the full cloud. Validating against the full cloud energy of Eq (2.1) is what allows the coarse phase to guarantee that the true registration objective is not a proxy defined over a subset, but decreases monotonically at every accepted iteration, directly preserving the correctness guarantee of classical ICP.

3.1.8.0.2 Why a halving schedule rather than a different decay rate. The threshold schedule of Eq. (3.2) decays geometrically with ratio 0.5. A slower decay ratio would produce many more coarse phase iterations, each only marginally denser than the last; because each accepted iteration still requires full cloud energy validation (Section 3.1.4), a slower schedule would multiply the number of these full cloud validation passes without proportional gain in alignment quality per step. A faster decay ratio would instead jump between working set densities so abruptly that a subset accepted at one threshold level could differ drastically in size and distribution from the next, risking instability in the correspondence structure between consecutive iterations. Halving is adopted as a reasonable, moderate choice within this trade-off guaranteeing that the iterations

described in (Section 3.1.1) produces a gradual densification of the working subset from one iteration to the next.

3.1.8.0.3 Why a dual-tier fallback rather than Anderson Acceleration or SVD

alone. Using Anderson Acceleration unconditionally would risk occasional instability or stagnation (Section 2.1.7), since accelerated updates are not guaranteed to reduce the true registration energy. Using only the accelerated SVD update, on the other hand, would go without the faster practical convergence that acceleration typically provides, and would not address the number of iterations cost that motivated incorporating Anderson Acceleration into the framework at all. The dual tier structure of Eqs. (3.3)-(3.4) is designed to obtain the benefit of acceleration whenever it helps, while retaining the unaccelerated update as a guaranteed-descent fallback whenever it does not gain any benefit.

3.1.8.0.4 Why a two-phase structure rather than adaptive downsampling through

out. An alternative design would continue adapting the threshold indefinitely rather than transitioning to a fixed full-cloud refinement phase once τ reached $d_{\min}$. Once $\tau_k \leq d_{\min}$, the filter of Algorithm 3.1 can no longer exclude any point, so continuing to adapt the threshold below this point would produce no further change in the working subset, there is no computational benefit left to extract from downsampling once the full cloud is already being used. Separating the pipeline into an adaptive Phase 1 followed by a fixed, guaranteed-descent Phase 2 makes this transition explicit, and ensures that whatever accuracy the coarse phase could not resolve - whether due to reaching the threshold floor or due to early termination via the energy-plateau condition is recovered by a Phase 2 operating unambiguously on the complete cloud.

3.1.9 Summary

This chapter presented a two-phase ICP framework in which an adaptive, progressively refined threshold governs how much of the source cloud is used for correspondence search at each iteration, while a dual tier, energy-validated acceptance criterion guarantees that every accepted transformation strictly reduces the true, full-cloud registration objective. By coupling this coarse-to-fine sampling schedule with Anderson Acceleration and a guaranteed-descent full-cloud refinement phase, the method is designed to inherit the iteration-count benefits of accelerated ICP while directly addressing the per-iteration correspondence search cost overhead characteristic of prior approaches. Section 3.1.8 justified the specific design choices against different alternatives. Section 3.2 below describes the experimental conditions and metrics used to evaluate this framework empirically against baseline Fast and Robust ICP [1] across static mesh models and real-world LiDAR datasets.

3.2 Experimental Setup

3.2.1 Hardware Platform

All static and dynamic dataset experiments were conducted on a PC with a 4-core CPU at 1.60 GHz, 8 GB RAM. The same hardware was used for all baseline methods and the proposed method.

3.2.2 Static Benchmark Datasets

Static evaluation uses five benchmark meshes drawn from two widely used 3D scanning repositories, the Stanford 3D Repository and the EPFL statue dataset. These datasets were selected to span a range of point cloud sizes and surface geometries, allowing registration performance to be assessed as a function of geometric complexity rather than on a single representative shape.

3.2.3 LiDAR Data Collection

Dynamic, real-world evaluation uses data collected from a Ford Mustang Mach-E electric vehicle (Figure 3.1) equipped with a roof-mounted LiDAR sensor and a tightly integrated GPS/IMU navigation unit. LiDAR data were acquired using an Ouster OS1-64 sensor [29], a 64-beam spinning LiDAR providing a full 360° horizontal field of view at 10 Hz, generating dense 3D point clouds. Data was collected across urban driving segments containing both structured geometry (buildings, curbs) and unstructured geometry (roadside vegetation, uneven terrain), representative of real-world autonomous driving conditions. Three sub-maps, East, Middle, and West segments of the drive route - were constructed by aggregating global frames at fixed timestamps, and each local scan was registered against its temporally localized sub-map rather than the full global map, reducing computational load and improving convergence stability. The initial pose for each scan was seeded from the preceding timestamp rather than the ground truth pose, to emulate the misalignment present in real-time SLAM operation.



Figure 3.1: Ford Mustang Mach-E for on-road LiDAR data collection.

3.2.4 Manual Outlier Removal

Before performing point cloud registration, the collected LiDAR data were visually inspected using RViz and CloudCompare to identify outliers and points outside the region of interest. These included points located below the expected ground surface, points appearing at unusually high elevations, and distant points outside the relevant building or environmental boundaries. After identifying these unwanted points through visualization, the corresponding point cloud coordinates were examined and manually removed using Microsoft Excel. This preprocessing step was performed to retain the relevant geometric features of the environment and exclude points that were not useful for the registration experiments. Although this manual approach allowed visual verification of

unwanted points, it required additional effort and may be less practical for large-scale datasets. In future work, the procedure could be automated using coordinate-based spatial filtering by defining minimum and maximum limits for the x, y, and z coordinates. For example, a lower z-coordinate threshold could remove points below the expected ground surface, while an upper z-coordinate threshold could eliminate points at unrealistic elevations above the buildings. Additional statistical outlier removal techniques could also be investigated to identify isolated measurements automatically. Additionally, the Random Sample Consensus (RANSAC) algorithm could be explored for automatic ground-plane estimation. RANSAC can identify a dominant planar surface in a point cloud despite the presence of outliers. Once the ground plane is estimated, points located significantly below it could be identified and removed using a specified distance threshold. This approach could be more adaptable than fixed coordinate thresholds when the ground elevation or sensor orientation varies. Such automation could reduce manual intervention, improve preprocessing consistency, and increase the efficiency of the overall registration workflow.

CHAPTER FOUR

EXPERIMENTS AND RESULTS

This chapter evaluates the adaptive downsampling and dual-phase refinement framework presented in Chapter 3 against the baseline Fast and Robust ICP methods of [1], using the hardware and datasets described in Sections 3.2. Evaluation proceeds in two parts, static benchmark datasets drawn from the Stanford 3D Scanning Repository and the EPFL statue collection, followed by real-world LiDAR sequences. For each dataset, runtime, registration accuracy along with mean and standard deviation are reported side by side, so that any computational benefit of the proposed method can be assessed jointly with its effect on alignment quality, directly addressing the trade-off motivating the design choices of Chapter 3. Two parameters used for different datasets, $\kappa = 16$, $\kappa = 1000$ for static and dynamic dataset respectively while $k_{\max} = 8$ is irrespective of the point clouds. The sensitivity of these two parameters is examined in Sensitivity Analysis in Section 4.1.3.

4.1 Results

4.1.1 Static Dataset Results

Following the parameter roles established in Section 3.1.1, static dataset experiments use $\kappa = 16$ and $k_{\max} = 8$ throughout. Five static datasets with different density of point clouds are taken from Stanford mesh models [30] and EPFL statues datasets [31]. All static results are collected on the hardware platform described in Section 3.2, with runtime and registration accuracy across repeated trials.

4.1.1.1 Qualitative Coarse-to-fine Behavior Figure 4.1 illustrates the coarse-to-fine progression of Algorithm 3.2 on the Stanford Bunny, whose source and target clouds contain 14,806 and 15,446 points respectively. As can be seen, during the adaptive

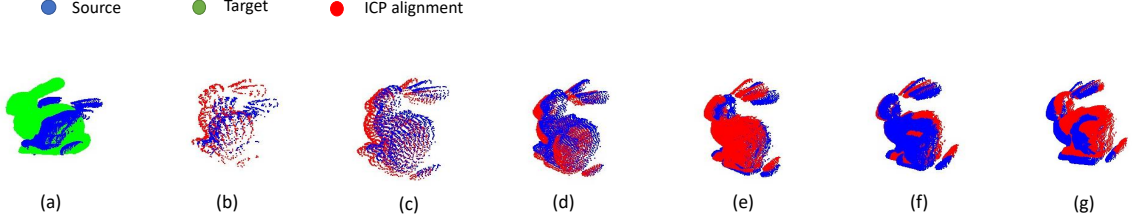


Figure 4.1: Illustration of the coarse-to-fine behavior of Algorithm 3.2 using the Stanford Bunny. (a) Original source and target clouds; (b)-(f) Adaptive downsampling phase; (g) Refinement phase.

downsampling phase as the threshold decreases, the point cloud becomes denser indicating an increase in the number of points in the source point cloud. Particularly, the source cloud X increases from only 1,338 points at Iteration 0 ($\tau = 16d_{\min}$) to 14,805 points by Iteration 5 ($\tau = d_{\min}$). At each iteration, the transformation estimated on the filtered subset is applied to the entire source cloud before the next round of downsampling, so the source progressively reorients toward the target while the working subset becomes denser. As shown in Fig. 4.1, the source cloud aligns aggressively with the target within only five coarse-phase iterations, leaving the refinement phase to resolve fine-scale residuals. This qualitative behavior mirrors the coarse-to-fine mechanism described in Section 3.1.2, where early iterations operate on small, reduced subset, and the working set densifies with every iteration.

4.1.2 Runtime and Registration Accuracy

Table 4.1 compares the proposed method against baseline Fast and Robust ICP across five static datasets. On four of the five datasets, the proposed method achieves a substantial runtime reduction with a marginal difference in RMSE. The evaluation reports runtime (seconds) and registration error (RMSE in meters) across all tested datasets. On the Bunny dataset, the proposed Algorithm 3.2 achieves a 34% speedup with a virtually identical RMSE. On the Buddha dataset, a 58% speedup is observed with a negligible RMSE difference. On the Dragon and William Tell datasets, Algorithm 3.2 is 13% and

34% faster, respectively, with a slight degradation in accuracy. This trade-off arises because the adaptive downsampling phase operates on spatially filtered subsets that prioritize computational efficiency over geometric completeness during early iterations. Importantly, despite the numerical increase in RMSE, the final alignments produced by the proposed method are visually indistinguishable from those of Fast ICP, as illustrated in Fig. 4.2, suggesting that the observed RMSE differences correspond to sub-millimeter residuals that do not affect practical registration quality. Monkey dataset is the only case where Algorithm 3.2 performs poorer in both speed and accuracy. Specifically, our method is slower than Fast ICP; however, the alignment quality remains accurate in practice. We attribute this to the complex geometry of the Monkey mesh, where the spatially filtered subset produces transformations that fail the energy check, causing early termination of the adaptive downsampling phase before τ reaches $d_{\min}$. The refinement phase must then compensate from a less favorable initialization, requiring additional iterations. This exception is discussed in detail in Section 4.1.6, alongside the geometric properties of the Monkey mesh that give rise to it.

4.1.3 Sensitivity Analysis

We characterize the sensitivity of Algorithm 3.2 to its two tunable parameters, κ and $k_{\max}$, using the Bunny and Buddha datasets. The results are shown in Tables 4.2, 4.3, 4.4, and 4.5. For Bunny, RMSE is robust to κ in the range $[8, 64]$, varying by less than 0.3%. At $\kappa = 128$, however, RMSE degrades sharply to 4.68×10^{-2} m, approximately twice the baseline error. Such a degradation originates from excessive sparsification during the initial stage of Phase 1, where the adaptive filter retains only 19 source points. This sparse and non-representative subset weakens the geometric constraints available for rigid registration, producing unstable correspondence structure and a poor intermediate transformation estimate. On the other hand, both runtime and RMSE are invariant to $k_{\max}$

Table 4.1: Comparison of Algorithm 3.2 versus baseline Fast ICP and Robust ICP [1] on static datasets.

Dataset	Method	Ave. Time $\pm$ STD (s)	RMSE (m)
Bunny	Fast ICP	0.417 ± 0.002	2.277×10^{-2}
	Robust ICP	0.569 ± 0.0158	2.277×10^{-2}
	Ours	0.275 ± 0.002	2.273×10^{-2}
Buddha	Fast ICP	8.923 ± 0.069	1.233×10^{-2}
	Robust ICP	10.974 ± 0.0476	1.233×10^{-2}
	Ours	3.777 ± 0.028	1.236×10^{-2}
Dragon	Fast ICP	6.271 ± 0.316	5.75×10^{-9}
	Robust ICP	6.732 ± 0.629	5.67×10^{-9}
	Ours	5.467 ± 0.170	5.67×10^{-6}
Monkey	Fast ICP	3.577 ± 0.081	1.48×10^{-8}
	Robust ICP	8.765 ± 2.336	1.44×10^{-8}
	Ours	6.814 ± 0.049	6.59×10^{-5}
William Tell	Fast ICP	2.895 ± 0.272	2.29×10^{-7}
	Robust ICP	5.747 ± 1.706	2.29×10^{-7}
	Ours	1.916 ± 0.039	2.204×10^{-3}

across the tested range, with all values agreeing around 2.27×10^{-2} m. This invariance is structural rather than empirical. On Bunny, Phase 1 exhausts its full threshold cascade and exits with $\tau < d_{\min}$, and so Phase 2 always terminates via its energy-plateau break regardless of $k_{\max}$. Note that the small variation of runtime is attributable to CPU variation rather than algorithmic work.

For the Buddha dataset, runtime varies significantly with respect to κ . Larger κ requires more Phase 1 iterations before the cascade either exhausts or triggers the energy-plateau break, hence increasing total runtime without improving RMSE. For $k_{\max}$, the behavior reveals a pattern of diminishing-returns. At $k_{\max} = 4$, the algorithm yields an RMSE of 1.24×10^{-2} m in 2.93 ± 0.01 s. At $k_{\max} = 17$, the algorithm reaches

Table 4.2: Sensitivity with respect to κ , on the Bunny Dataset with $k_{\max} = 8$.

κ	Time (s)	RMSE (m)
8	0.268 ± 0.008	2.27×10^{-2}
16	0.267 ± 0.003	2.27×10^{-2}
32	0.258 ± 0.001	2.28×10^{-2}
64	0.292 ± 0.001	2.27×10^{-2}
128	0.267 ± 0.001	4.68×10^{-2}

Table 4.3: Sensitivity with respect to $k_{\max}$, on the Bunny dataset, with $\kappa = 16$.

$k_{\max}$	Time (s)	RMSE (m)
4	0.268 ± 0.003	2.27×10^{-2}
16	0.267 ± 0.002	2.27×10^{-2}
32	0.268 ± 0.004	2.27×10^{-2}
64	0.289 ± 0.012	2.27×10^{-2}

Table 4.4: Sensitivity with respect to κ on the Buddha dataset, with $k_{\max} = 8$.

κ	Time (s)	RMSE (m)
4	2.90 ± 0.01	1.23×10^{-2}
16	3.82 ± 0.01	1.24×10^{-2}
64	3.95 ± 0.25	1.24×10^{-2}
80	4.01 ± 0.32	1.24×10^{-2}

Table 4.5: Sensitivity with respect to $k_{\max}$ on the Buddha dataset, with $\kappa = 16$.

$k_{\max}$	Time (s)	RMSE (m)
4	2.93 ± 0.01	1.24×10^{-2}
8	3.82 ± 0.01	1.23×10^{-2}
17	5.89 ± 0.03	1.23×10^{-2}
32	6.83 ± 0.04	1.23×10^{-2}

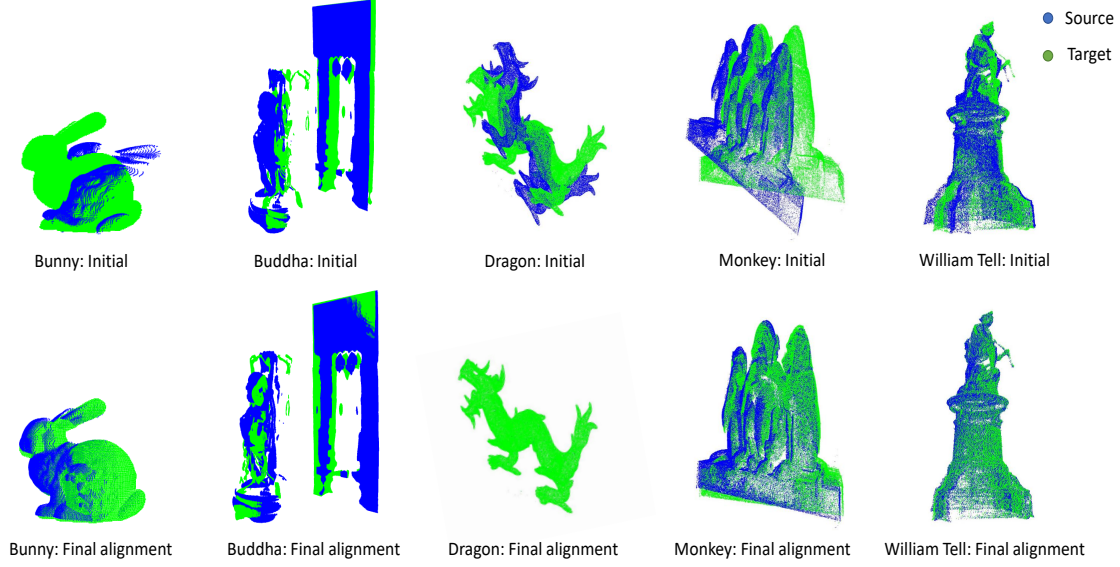


Figure 4.2: Registration results on static datasets. *Top*: Initial overlap of target (green) and source (blue). *Bottom*: Final alignment by Algorithm 3.2.

near-converged RMSE of 1.23×10^{-2} m in 5.89 ± 0.03 s. At $k_{\max} \geq 32$, Phase 2 actually exits via the energy-plateau break before reaching $k_{\max}$ and converges to 1.23×10^{-2} m in approximately 6.8 s. Beyond $k_{\max} \approx 8$, additional iterations continue to decrease the energy by progressively smaller amounts, but the alignment is already near-perfect and the extra iterations consume runtime without producing visible accuracy improvement. The 1% RMSE reduction from $k_{\max} = 4$ to $k_{\max} \geq 32$ comes at a $2.3\times$ runtime cost, illustrating that an excessive refinement budget on already-converged datasets adds runtime without practical benefit.

4.1.4 LiDAR Dataset Results

Real-world experiments were conducted using a Ford Mustang Mach-E electric vehicle equipped with a roof-mounted LiDAR sensor and a tightly integrated GPS/IMU navigation unit as shown in Fig 3.1.

Table 4.6: Target sub-map and source scan timestamps for each LiDAR segment.

Segment	Target Sub-map (Global Frames)	Source Scan (t_l)
East	0 s + 5 s	3 s
Middle	12 s + 16 s	15 s
West	36 s + 40 s	38 s

Table 4.6 details the temporal construction of each sub-map; for the East segment, the target sub-map aggregates global frames captured at 0 s and 5 s, against which a source scan captured at $t_1 = 3$ s, while its initial pose (GPS and heading) is taken from the preceding timestamp $t_1 - 1$. The Middle and West segments follow the same pattern at progressively later points in the drive route.

4.1.5 Runtime and Accuracy Comparison

Table 4.7 summarizes runtime and RMSE across the three LiDAR segments.

The three LiDAR segments show a consistent pattern of runtime improvement, though the magnitude varies segment to segment in a way that plausibly tracks the point-density difference in all three segments. On the East segment, the proposed method is approximately 50% faster than baseline Fast ICP, with virtually identical RMSE, confirming the speedup is achieved without compromising accuracy. The largest improvement is observed on the Middle segment, with a 64% reduction in runtime and a nearly identical RMSE to the baseline. On the West segment, the proposed method is 44% faster, with a marginally lower RMSE than the baseline, the only case across all evaluated datasets, static or dynamic, where the proposed method achieves both a runtime and an accuracy improvement simultaneously. That this occurs specifically on the lowest-density segment is consistent with the design rationale of Section 3.1.8: with fewer points overall, the adaptive filter’s early coarse subsets constitute a comparatively larger fraction of the full cloud, potentially allowing the coarse phase to reach a well-constrained estimate in

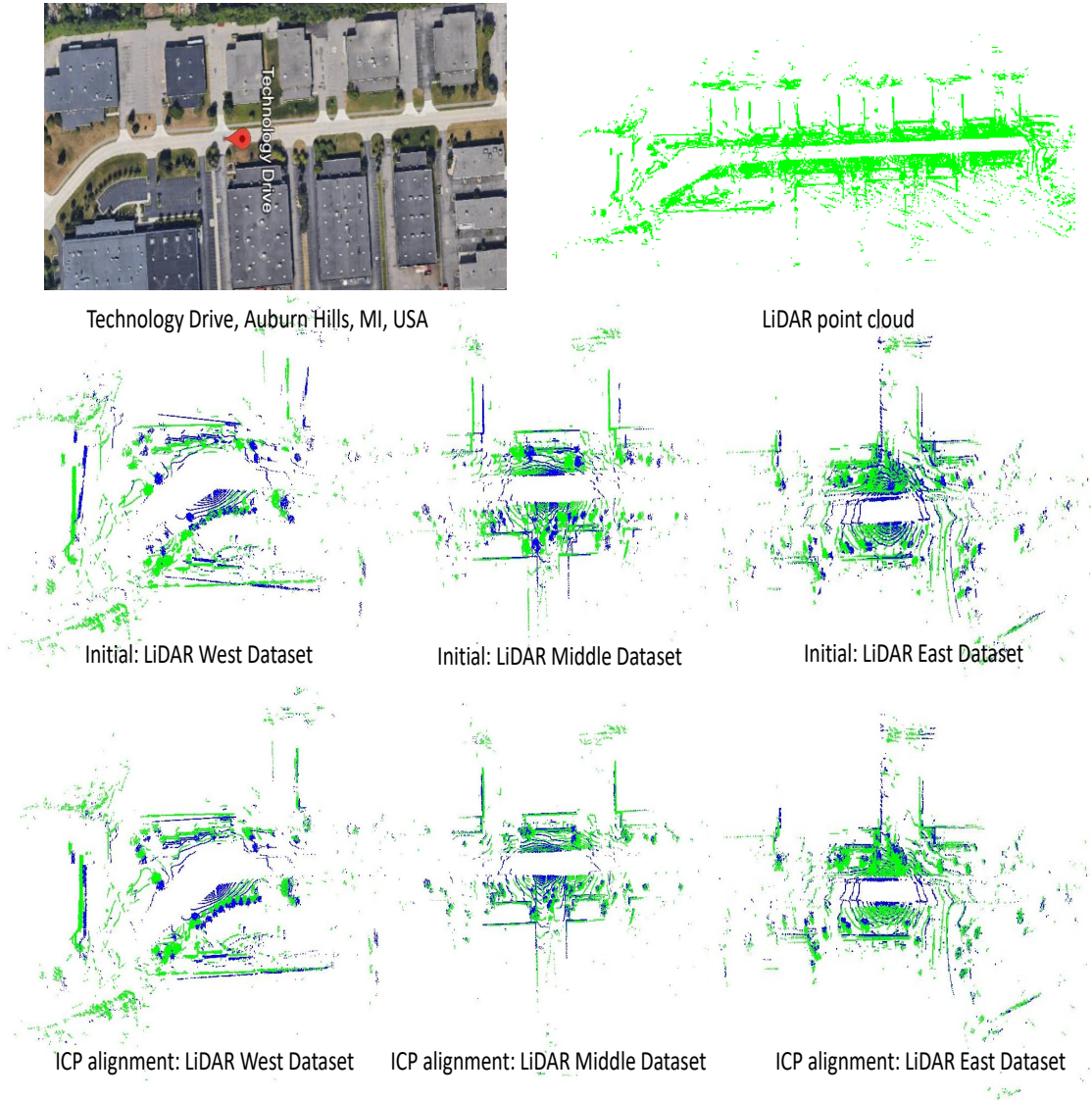


Figure 4.3: LiDAR experiments on Technology Drive, Auburn Hills, MI, USA. *Top*: Drive route map and aggregated LiDAR point cloud. *Middle*: Initial overlap of target (green) and source (blue) for West, Middle, and East segments. *Bottom*: Final alignment achieved by Algorithm 3.2.

fewer iterations. Across all three LiDAR sequences, the proposed approach consistently achieves speedups ranging from 44–64% while maintaining registration accuracy comparable to or, on West, marginally better than baseline Fast ICP, extending the runtime benefits observed on static mesh benchmarks (Section 4.1.1.1) to real-world, noisy sensor data.

Table 4.7: Comparison of Algorithm 3.2 versus baseline Fast ICP and Robust ICP [1] on LiDAR Datasets.

LiDAR set	Method	Ave. Time $\pm$ STD (s)	RMSE (m)
East	Fast ICP	0.371 ± 0.002	2.92×10^{-3}
	Robust ICP	0.634 ± 0.006	2.92×10^{-3}
	Ours	0.157 ± 0.002	2.92×10^{-3}
Middle	Fast ICP	0.730 ± 0.003	2.66×10^{-3}
	Robust ICP	0.925 ± 0.011	2.66×10^{-3}
	Ours	0.266 ± 0.002	2.65×10^{-3}
West	Fast ICP	0.543 ± 0.002	5.71×10^{-3}
	Robust ICP	0.698 ± 0.001	5.71×10^{-3}
	Ours	0.304 ± 0.002	5.60×10^{-3}

4.1.6 Parameter Discussions

Across the eight dataset/segment combinations evaluated in this chapter; five static benchmarks and three LiDAR drive segments, the proposed method achieves a runtime reduction on seven of eight, ranging from 13% to 64%, while maintaining RMSE within a particularly negligible margin of the baseline in all but one case. This section examines the two findings that depart from this overall pattern, the Monkey dataset exception in

Table 4.1, and the broader question of why RMSE increases at all on datasets where proposed method is faster.

4.1.6.0.1 The Monkey dataset exception. Unlike the other four static datasets, the proposed method is slower than baseline Fast ICP on Monkey. The Monkey mesh is notably larger than the other benchmarks and comprises three separate figures in distinct poses, each containing concentrated high-curvature regions - a concave eye socket, hands covering eyes, mouth, and ears. These structures increase nearest-neighbor ambiguity once the cloud has been spatially filtered (Section 3.1.2), a sparse subset drawn from a high-curvature region is more likely to produce a transformation estimate that fails the full-cloud energy check of Eq. (3.3), since a small number of points sampled from a sharply curved surface constrain the rigid-body estimate far less reliably than the same number of points drawn from a smooth low-curvature region. Consistent with this, Phase 1 terminates early on Monkey via the energy plateau branch, well before the threshold cascade reaches $d_{\min}$, so the refining phase of Section 3.1.5 must recover accuracy from a less favorable initialization than on the other datasets, requiring additional full-cloud iterations that erode the runtime advantage observed elsewhere. Notably, this is exactly the failure mode anticipated by the design rationale of Section 3.1.8, where the dual-tier acceptance criterion correctly detects and rejects the unreliable coarse phase estimate on this geometry rather than silently accepting them, so the method degrades gracefully to a slower, but still accurate, result rather than producing a poor alignment quickly.

4.1.6.0.2 Interpreting the RMSE difference. On Dragon, Monkey and William Tell, the proposed method’s RMSE is numerically higher than the baseline’s, in some cases by several orders of magnitude in relative terms (e.g., 5.67×10^{-6} vs. 5.67×10^{-9} m on Dragon). Taken at face value, this might appear to be a meaningful accuracy regression;

however, Figure 4.2 shows that the final alignments produced by the proposed method are visually indistinguishable from baseline Fast ICP across every dataset in Table 4.1. This is because the baseline RMSE values of these three datasets are already extremely small in absolute terms, meaning the datasets were already registered to sub-millimeter precision by the baseline; the proposed method’s comparatively higher but still small in absolute terms; RMSE reflects the trade-off inherent to the design rationale of Section 3.1.8, where the adaptive downsampling phase prioritizes computational efficiency over the exhaustive geometric completeness during early iterations, and on datasets where the baseline already achieves near-perfect alignment, this trade-off is visible in the RMSE column even though it has no visible effect on the resulting alignment where the final alignment produced by the proposed method are visually indistinguishable from baseline Fast ICP across every dataset as seen in Figure 4.2.

4.1.6.0.3 Summary of findings. Taken together, the static and LiDAR results in this chapter support the central claim motivating Chapter 3 that adaptively varying working point-cloud density within a single registration run, rather than fixing it in advance or leaving it untouched, recovers most of the computational savings available from downsampling while the full-cloud energy validation of Eq. (3.3)–(3.4) prevents that downsampling from degrading final alignment quality in any practically meaningful way. For example, with the Monkey dataset serving as an informative counterexample illustrating precisely the geometric conditions (concentrated high-curvature regions) under which this trade-off is least favorable.

CHAPTER FIVE

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

This thesis set out to address a specific, well-defined inefficiency in classical ICP, the correspondence search step, whose cost scales directly with the source cloud size, is repeated at full resolution on every iteration regardless of how far the current transformation estimate is from convergence. The contributions, developed in Chapter 3, directly target this inefficiency.

The first contribution is the adaptive threshold-based downsampling strategy itself (Section 3.1 – 3.1.1). Unlike fixed-resolution voxelization or uniform subsampling, which set a single working resolution before registration begins and hold it fixed throughout (Section 2.1.3), the proposed threshold is initialized proportional to the source cloud’s own minimum inter-point spacing and halved at every coarse iteration (Eq. 3.2). This produces a working subset that is initially selected using a relatively large distance threshold. As the threshold decreases, the selection criterion becomes less restrictive, generally allowing more source points to participate in subsequent iterations. The proposed method adopts an energy-based acceptance criterion inspired by the approach in [1]. Because the transformation estimated on a filtered subset is only guaranteed to be optimal with respect to that subset, not the full cloud, every candidate update, whether produced by the accelerated Anderson Acceleration step or by the fallback SVD solution is validated against the true, full-cloud registration energy (Eqs. (3.3)–(3.4)) before being accepted.

Together, the two-phase pipeline of (Algorithm 3.2) requires two additional tunable parameters beyond baseline ICP, is designed to integrate with, rather than replace, existing convergence-acceleration techniques, and was evaluated across eight distinct datasets and segment combinations spanning both static and real-world, noisy sensor data.

5.1.1 Summary of Key Findings

The static-dataset evaluation of Section 4.1.1.1 showed that the proposed method achieved its largest gains on Buddha, with a 54% runtime reduction and a negligible RMSE difference, and speedups on Bunny (34%), Dragon (13%) and William Tell (34%), with the latter two showing a small numerical increase in RMSE that Section 4.1.6 traced to near-perfect baseline alignments rather than any visible degradation in the final registration (Figure 4.2). The sensitivity analysis in Section 4.1.3 further showed that this behavior is stable across a wide range of the framework’s two tunable parameters, degrading sharply only at the extreme of $\kappa = 128$ on Bunny, where the coarse filter retains too few points (19 points of the source cloud) to adequately constrain the rigid body estimate, a finding that directly validates the design rationale of Section 3.1.8 for why full-cloud energy validation is required compared to subset-only validation is necessary. The Monkey dataset was the exception to this pattern and proved to be the most informative result in the evaluation rather than simply a negative one. Its concentrated high-curvature geometry such as eye sockets, hands, and ears discussed visually in the curvature comparison of Section 4.1.6 increases nearest-neighbor ambiguity once the cloud is spatially filtered, causing Phase 1 to terminate early via the energy plateau branch and shifting more of the registration burden onto the full cloud refining phase.

The LiDAR evaluation of Section 4.1.4 extended these findings to real-world noisy sensor data collected from an instrumented vehicle, achieving speedups of 44% to 64% across the three drive segments evaluated, where the West segment additionally showing a marginal accuracy improvement over the baseline, the only case in the entire evaluation where the speed and accuracy improved simultaneously. This suggests that the framework’s benefits, rather than being an artifact of synthetic mesh geometry, are extended to dense, irregular structured point clouds produced by real LiDAR sensors.

5.1.2 Limitations

The limitations of the present evaluation are stated in this section, both to properly scope the claims made in this thesis and to motivate the future work discussed in Section 5.2. First, the framework was compared against baseline Fast and Robust ICP (Section 2.1.3); it was not benchmarked directly against voxel-based methods such as VGICP or NDT, deep learning methods, all of which pursue conceptually related but architecturally distinct strategies for reducing registration cost. Chapter 2 explores qualitatively that these methods fix their working resolution in advance rather than adapting it within a single run. Second, the registration energy used throughout this thesis (Eq. (2.1)) is the point-to-point objective; the point-to-plane and other variants discussed in Section 2.1.2, which are known to improve convergence behavior on smooth surfaces and in the presence of outliers respectively, were not incorporated into the adaptive downsampling framework evaluated here. Third, the two tunable parameters κ and $k_{\max}$ (Section 3.1.1) were set manually per data source (static vs LiDAR) based on the sensitivity analysis of Section 4.1.3, rather than selected automatically per dataset. All runtime measurements in this thesis were collected on a single hardware platform using a CPU-only implementation (Section 3.2); the relative speedup of the proposed method over baseline ICP may differ on GPU-accelerated implementations. The LiDAR evaluation of Section 4.1.4, while drawn from real-world sensor data, is limited to three drive segments from a single data collection platform and route.

5.1.3 Practical Implications

Despite the limitations, the magnitude of the runtime reductions observed in Chapter 4 has direct practical relevance to the applications that motivated this thesis in Chapter 2. LiDAR systems such as those discussed in Section 2.1.3 must register incoming scans at or near sensor frame rate to operate in real-time, and a 44–64%

reduction in per scan registration time of the kind observed offline on the LiDAR segments in Section 4.1.4, suggests that a real-time deployment of the proposed method could plausibly support real-time budget saving of SLAM pipeline. Similarly, in high-resolution 3D reconstruction and digital twin applications, where registration is repeated across many overlapping dense scans, the 58% speedup observed on the Buddha mesh (Section 4.1.1.1) suggests meaningful aggregate time savings when scaled across a full reconstruction pipeline. Unlike real-time LiDAR odometry, does not require live operation and is therefore more directly comparable to the offline evaluation protocol used throughout this thesis. Because the framework required no training data, introduces two additional parameters, and is designed to integrate with rather than replace existing acceleration techniques (Section 3.1.8), it is intended to be adoptable within existing ICP-based systems with comparatively modest engineering effort relative to a full pipeline design.

5.2 Future Work

5.2.1 Extension to Point-to-Plane Method

The present framework is developed entirely around the point-to-point registration energy of Eq. 2.1. As discussed in the limitation above, extending the adaptive downsampling and dual-tier acceptance criterion to the point-to-plane objective (Section 2.1.2) is a natural next step, since point-to-plane matching is known to improve convergence on smooth surfaces by allowing correspondences to slide freely along the local tangent plane. The filter of Algorithm 3.1 currently selects points based purely on Euclidean separation, with no notion of surface orientation, so a point-to-plane variant would need to ensure that the filtered subset retains enough diversity in its surface normals to properly constrain translation and rotation.

5.2.2 Curvature-Aware and Geometry-Preserving Downsampling

The Monkey dataset exception discussed in Section 4.1.6 identifies a limitation of the present filtering strategy. Because retention in Algorithm 3.1 is governed purely by distance to the most recently accepted reference point, the filter has no mechanism for preferentially retaining points in geometrically distinctive, high-curvature regions. Instead, it keeps or discards a candidate point based only on how far away it is, never on how geometrically distinctive that point is. An extension is to estimate the local curvature at each candidate point and factor it into the acceptance of Algorithm 3.1, so that points in high-curvature regions are retained preferentially and kept even when they fall closer together than the threshold would normally allow. A standard way to obtain this curvature estimate in point cloud processing is to look at each point’s immediate neighborhood and measure how well a flat plane fits it; a neighborhood that lies almost entirely within a plane indicates a flat, low-curvature region, while a neighborhood that bends noticeably out of any single plane indicates a sharp, high-curvature one. Such a strategy would directly target the mechanism identified as the cause of the Monkey dataset’s early termination of Phase 1, and could plausibly close some of the runtime gap observed on that dataset in Table 4.1.

5.2.3 Learned Adaptive Threshold Selection

The sensitivity analysis of Section 4.1.3 demonstrated that the framework’s two tunable parameters, κ and $k_{\max}$, interact with dataset-specific properties such as point density, curvature complexity, static, dynamic dataset, and the poorly chosen values can noticeably degrade accuracy, as observed at $\kappa = 128$ on Bunny. Rather than requiring a particular value to repeat this kind of sensitivity sweep for every new dataset, a learned or data-driven strategy for selecting the initial threshold and refinement budget is another extension for future work. A training approach like regression model to predict a suitable

κ from summary statistics of the source cloud (e.g., point density, curvature complexity, static, dynamic dataset) or framing threshold selection as a sequential decision problem and applying reinforcement learning to adapt τ dynamically within a single registration run based on the observed trajectory of the registration energy, rather than following the fixed halving schedule of Eq. 3.2.

5.2.4 Integrating Within a Full SLAM Pipeline

This thesis evaluates the proposed method as a standalone registration primitive, following the scope established in Chapter 2; it targets the ICP registration step itself, rather than a complete SLAM system such as those discussed in Section 2.1.3. A natural next step is integrating the framework into a full real-time LiDAR SLAM pipeline, to evaluate its effect on long-term trajectory accuracy and map consistency under sustained operation, rather than on the single scan registration accuracy reported in Chapter 4.

5.2.5 GPU Implementation

All results in this thesis were collected using a CPU implementation on a single, modest hardware platform (Section 3.2). The correspondence search bottleneck motivating this thesis (Section 2.1.1) can be addressed through two largely complementary strategies; algorithmically, by reducing the number of points searched at each iteration, which is the approach taken throughout this thesis, or computationally, by parallelizing the correspondence search itself across many processing cores, as is common in GPU-accelerated ICP implementation. Because these two strategies target the same bottleneck through different means, an important direction for future work is implementing the proposed framework on GPU hardware and characterizing whether the relative speedup observed in Chapter 4 is preserved, diminished, or amplified once the underlying correspondence search is itself massively parallelized.

5.2.6 Extension to Non-Rigid and Multi-Modal Registration

Finally, this thesis addresses rigid registration exclusively, following the problem formulation of Eq. (2.1), in which a single transformation $(\mathbf{R}, \mathbf{t})$ is estimated for the entire source cloud. Extending the adaptive downsampling strategy to non-rigid or deformable registration, where different regions of the source cloud may require different local transformations, would require rethinking the full-cloud energy validation step of Eq. (3.3), since a single global energy value may no longer be sufficient to characterize the quality of a spatially varying deformation field. Similarly, extending the framework to multi-modal registration; e.g., point clouds with color or intensity information, which are increasingly common in combined LiDAR and camera SLAM systems could allow the adaptive filter of Algorithm 3.1 to incorporate appearance-based information alongside spatial separation when selecting the working subset, rather than relying on 3D geometry alone.

REFERENCES

- [1] J. Zhang, Y. Yao, and B. Deng, “Fast and robust iterative closest point,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3450–3466, 2021.
- [2] T. Jost and H. Hugli, “A multi-resolution icp with heuristic closest point search for fast and robust 3d registration of range images,” in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pp. 427–433, 2003.
- [3] S. Rusinkiewicz and M. Levoy, “Efficient variants of the ICP algorithm,” in *Proceedings third international conference on 3-D digital imaging and modeling*, pp. 145–152, IEEE, 2001.
- [4] N. Bhatt, Z. Zhou, K. Mehta, and J. Chen, “Adaptive downsampling and refinement for fast iterative closest point,” *IEEE Systems, Man, and Cybernetics Letters*, vol. 1, pp. 20–26, 2026.
- [5] P. J. Besl and N. D. McKay, “Method for registration of 3D shapes,” in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586–606, Spie, 1992.
- [6] R. B. Rusu and S. Cousins, “3D is here: Point cloud library (pcl),” in *2011 IEEE international conference on robotics and automation*, pp. 1–4, IEEE, 2011.
- [7] F. Pomerleau, F. Colas, and R. Siegwart, “A review of point cloud registration algorithms for mobile robotics,” *Foundations and Trends® in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.
- [8] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
- [9] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3D point sets,” *IEEE Transactions on pattern analysis and machine intelligence*, no. 5, pp. 698–700, 1987.
- [10] H. Pottmann, Q.-X. Huang, Y.-L. Yang, and S.-M. Hu, “Geometry and convergence analysis of algorithms for registration of 3D shapes,” *International Journal of Computer Vision*, vol. 67, no. 3, pp. 277–296, 2006.
- [11] M. Greenspan and M. Yurick, “Approximate KD-tree search for efficient icp,” in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pp. 442–448, IEEE, 2003.
- [12] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.

- [13] S. Rusinkiewicz, “A symmetric objective function for ICP,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–7, 2019.
- [14] S. Bouaziz, A. Tagliasacchi, and M. Pauly, “Sparse iterative closest point,” in *Computer graphics forum*, vol. 32, pp. 113–123, Wiley Online Library, 2013.
- [15] P. Mavridis, A. Andreadis, and G. Papaioannou, “Efficient sparse icp,” *Computer Aided Geometric Design*, vol. 35, pp. 16–26, 2015.
- [16] H. Yang, J. Shi, and L. Carlone, “Teaser: Fast and certifiable point cloud registration,” *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 314–333, 2020.
- [17] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng, S. Ilic, D. Hu, and K. Xu, “Geotransformer: Fast and robust point cloud registration with geometric transformer,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 8, pp. 9806–9821, 2023.
- [18] L. Li, M. Yang, and C. Wang, “Graph correspondence-based point set registration,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 7, pp. 4101–4112, 2024.
- [19] D. G. Anderson, “Iterative procedures for nonlinear integral equations,” *Journal of the ACM (JACM)*, vol. 12, no. 4, pp. 547–560, 1965.
- [20] H. F. Walker and P. Ni, “Anderson acceleration for fixed-point iterations,” *SIAM Journal on Numerical Analysis*, vol. 49, no. 4, pp. 1715–1735, 2011.
- [21] A. L. Pavlov, G. W. Ovchinnikov, D. Y. Derbyshev, D. Tsetserukou, and I. V. Oseledets, “AA-ICP: Iterative closest point with anderson acceleration,” in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3407–3412, IEEE, 2018.
- [22] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, “Voxelized gicp for fast and accurate 3d point cloud registration,” in *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 11054–11059, IEEE, 2021.
- [23] 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)*, pp. 5654–5660, IEEE, 2021.
- [24] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, “KISS-ICP: In defense of point-to-point icp—simple, accurate, and robust registration if done the right way,” *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1029–1036, 2023.

- [25] J. Zhang, S. Singh, *et al.*, “LOAM: LiDAR odometry and mapping in real-time.,” in *Robotics: Science and systems*, vol. 2, pp. 1–9, Berkeley, CA, 2014.
- [26] T. Shan and B. Englot, “LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain,” in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 4758–4765, IEEE, 2018.
- [27] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, “CT-ICP: Real-time elastic lidar odometry with loop closure,” in *2022 international conference on robotics and automation (ICRA)*, pp. 5580–5586, IEEE, 2022.
- [28] Y. Wang and J. M. Solomon, “Deep closest point: Learning representations for point cloud registration,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3523–3532, 2019.
- [29] Ouster, Inc., “OS1-64 product specification.”
<https://ouster.com/products/os1-sensor/>. Accessed: 2026-02-19.
- [30] Stanford Computer Graphics Laboratory, “The Stanford 3D Scanning Repository.”
<http://graphics.stanford.edu/data/3Dscanrep/>. Accessed: 2026-02-19.
- [31] EPFL Computer Graphics and Geometry Laboratory, “EPFL statue model repository.” https://lgg.epfl.ch/statues_dataset.php, 2012.