國 立 清 華 大 學

碩 士 論 文

利用重疊區域降採樣的優化ICP算法進行點雲配準

Optimized ICP Algorithm for Point Cloud Registration with Overlapping Region Pruning

系別：計算科學與建模研究所

學號：111026501

研 究 生：茹伊辰 (Yichen Ju)

指導教授：朱家杰,陳啓銘 (Jay Chu, Chi-Ming Chen)

中 華 民 國 一一三 年 七 月

Abstract

The algorithm of tackling Point Cloud Registration (PCR), the Iterative Closest Point (ICP), along with its various adaptations, is a crucial method for rigid registration within two point cloud sets, widely used in fields such as robotics and 3D surface reconstruction. Despite its significance, ICP suffers from slow convergence and is highly sensitive to outliers, incomplete data, and partial overlaps. In this work, we build on our key observation that computing alignment transformation resembles a point pair voting mechanism: point pairs with larger distances vote for larger adjustments to fit their closest corresponding points, while point pairs with smaller distances tend to vote for minimal movement. By removing overlapping point pairs, which are typically very close but incorrect in the first few iterations, from the computation of alignment, we can significantly accelerate the convergence speed. Additionally, we incorporate an insensitive to outliers error metric using the Welsch's function, which significantly mitigates the influence of outliers. Furthermore, to address the computational demands of larger point cloud datasets, we develop a GPU-accelerated version of the algorithm. This GPU implementation leverages parallel processing capabilities, enabling the handling of extensive datasets with greater efficiency and substantially improving the overall speed of the algorithm. These advancements make our approach highly effective for real-world applications requiring precise and rapid point cloud registration.

Acknowledgements

During the writing of this thesis, I received help and support from many people, to whom I would like to extend my sincere gratitude.

First and foremost, I want to thank my parents. They have not only provided me with financial support but have also encouraged me to pursue my dreams with courage. From the beginning of my studies until today, they have been my strongest pillars. In times of difficulty and setbacks, their boundless care and support have helped me regain my strength and continue moving forward. Their love and support have allowed me to focus on my research without any worries, making it possible for me to successfully complete my studies. My parents' support and encouragement are the most precious assets in my life, for which I am deeply grateful.

Secondly, I would like to thank my advisor, Professor Jia-Jie Zhu. At every stage of my research, he provided me with meticulous guidance and invaluable advice. He taught me to think about problems from different perspectives and inspired me to view the world with a broader vision. Throughout my research process, Professor Zhu not only offered rich knowledge and experience in academics but also provided immense care and support in my personal life. He patiently addressed the difficulties I encountered in my research and tirelessly revised my thesis to make it more complete. Additionally, Professor Zhu shared many insights and strategies regarding the application for Ph.D. programs, which gave me a clearer understanding and preparation for my future academic path. His dedicated guidance and enthusiastic encouragement have filled me with confidence and hope in my academic journey. His professional competence and passion for academic research have profoundly influenced me, instilling in me a deep sense of confidence and anticipation for my future research. His encouragement and guidance have

continuously propelled me forward in my academic pursuits.

In addition, I want to especially thank Professor Chi-Ming Chen. Throughout my studies and research, he has provided me with substantial financial aid and unceasing care in my daily life. He also offered many valuable suggestions and support during my search for an advisor. Professor Chen is always willing to share his experience and knowledge, and his patient guidance and advice have been instrumental when I faced confusion and challenges. His assistance and support have allowed me to focus wholeheartedly on my research, and his encouragement has fortified my resolve in my academic journey, for which I am profoundly grateful.

Moreover, I would like to thank the owners of the Feng-He Three Delicacies canteen. They have taken great care of my dietary needs, ensuring that I have sufficient nutrition and energy to cope with demanding academic work, and have extended their warmth and support in my daily life. Their kindness and affection have made me feel the warmth of home amidst the busy academic life, for which I am deeply thankful.

I am also grateful to my girlfriend. She has been by my side in the lab, studying together with me, offering endless support and encouragement. When my shoes were worn out and I was reluctant to replace them, she gifted me a new pair, allowing me to continue my studies and research comfortably. Her love and meticulous care have provided me with immeasurable warmth and support during my busy academic life.

Finally, I would like to thank my lab mate, Zhe Li, for his companionship, discussions, and support during my research. When he returned to China, he brought back many gifts, even though I hardly ate any of them. His gesture deeply moved and touched me. His friendship and collaboration have made my research process more smooth and enjoyable.

Once again, I express my sincerest gratitude to everyone who has helped and

supported me throughout my studies and research.

Ju Yichen

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

INTRODUCTION

Point cloud registration (PCR) plays a vital role in robotics and computer vision by aligning a source point cloud with a target point cloud using a combination of 3D rotation and translation. This technique is crucial for applications like localization [8], 3D reconstruction [25], augmented reality [26],and 6D object pose estimation [39]. In robotics, one of the biggest challenges is localization, which often uses laser scanner data presented as point clouds. In disaster management, Chen [4] uses RGB-D images to classify rock to preprocess the data for disaster prediction, while classification tasks often require different angles of scans to form a complete geometry. To align these scans, algorithms such as Iterative Closest Point (ICP) are commonly employed. Interestingly, ICP isn't just limited to robotics; it's also extensively used in medical image comparison and registration [11,38]. The rise of 3D perception technologies and the increasing availability of point cloud data from devices like LiDAR(Light Detection And Ranging) and stereo cameras have brought PCR into the spotlight in fields such as 3D modeling, augmented reality, autonomous driving, and robot perception. PCR aims to align point clouds captured from different viewpoints or at different times, ensuring consistent coordinate systems or pose information. For instance, in 6D object pose estimation, PCR aligns the observed partial point cloud of an object with

points sampled from its template CAD(Computer-Aided Design) model surfaces, enabling robotic arms to determine the best way to grasp objects [39].

The ICP algorithm [3], a cornerstone in this field, works by alternating between finding the closest points in the target set and minimizing the distance between corresponding points. This iterative process guarantees convergence to a locally optimal alignment. While ICP and its variants [10, 34, 43, 48] are widely used, they have some big drawbacks. They can be slow and need a lot of computing power. They also struggle with noise, outliers, and getting the initial setup right, which makes them less reliable in changing environments. One big issue is the heavy computational load when finding the closest correspondences. To address this, the KD-tree data [12,34] structure was introduced, which speeds up the nearest neighbor search significantly. Additionally, ICP's accuracy can be affected by noise, outliers, and partial overlaps in the point sets. These issues are common in real-world data, making the registration process more complex and less reliable. One way to tackle this is to remove outliers and reduce noise at the start. For instance, [16] proposed a method to clean up point clouds by removing outliers and reducing noise. Another approach is by using feature graph Laplacian regularization for denoising, as demonstrated by [7]. Also, replacing the traditional l_2 norm with the Welsch function as the distance metric can help make the process more robust to outliers and noise [47].

The primary challenge addressed in this work is to determine the optimal parameters $\mathbf{R}$ (rotation matrix) and $\mathbf{t}$ (translation vector) that align two point clouds, $\mathcal{P}$ and $\mathcal{Q}$, with minimal discrepancy. Formally, the problem is defined as finding $\mathbf{R}$ and $\mathbf{t}$ that minimize the following objective function:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N_p} \|\mathbf{p}_i - (\mathbf{R}\mathbf{q}_{\phi(i)} + \mathbf{t})\|^2, \tag{1.1}$$

where $\phi(i)$ is a correspondence function identifying the closest point in $\mathcal{Q}$ to each point $\mathbf{p}_i$ in $\mathcal{P}$. This function is mathematically defined as:

$$\phi(i) = \arg\min_j \|\mathbf{p}_i - \mathbf{q}_j\|. \tag{1.2}$$

The Iterative Closest Points (ICP) algorithm, central to this problem, iteratively refines the transformation parameters $(\mathbf{R}, \mathbf{t})$ through a three-step process:

1. **Find Correspondences**: For each point $\mathbf{p}_i$ in $\mathcal{P}$, determine the closest point $\mathbf{q}_{\phi(i)}$ in $\mathcal{Q}$.

2. **Compute Transformation**: Calculate the optimal rotation $\mathbf{R}$ and translation $\mathbf{t}$ that minimize the objective function.

3. **Apply Transformation**: Update the point cloud $\mathcal{P}$ by applying the computed transformation $(\mathbf{R}, \mathbf{t})$.

This process can be viewed as a voting mechanism where close-distance point pairs in the overlapping area tend to vote to stay the same due to their proximity, while point pairs that are farther apart seek larger adjustments to get closer. However, in the initial iterations, most point pair correspondences are incorrect. We can achieve faster and more accurate alignment by strategically reducing the voting weight of the overlapping areas. This definition encapsulates the core challenge that our proposed methods aim to tackle, ensuring precise and efficient alignment of point clouds in various complex and real-world scenarios.

We also incorporate the Welsch metric to enhance robustness against outliers. By adjusting the variance within the Welsch metric, we effectively tune the algorithm, similar to adjusting the learning rate in machine learning. In the initial iterations, we use a larger variance, comparable to a higher learning rate during the early stages of machine learning training. This larger variance allows for

greater flexibility in escaping local minima and promotes quicker convergence to the global minimum. As iterations progress, we gradually decrease the variance, ensuring more precise and stable alignment by dynamically adapting to the data, and balancing sensitivity and stability in the optimization process.

This work further uses a random down sampling method on the target point cloud before the overlapping pruning procedure. Since it's a rigid motion, it's not necessary to fit all the points to check if the alignment is accurate. By fitting a smaller, representative subset of points, we can achieve a good fit for the entire point cloud and significantly reduce the computational cost.

Additionally, we have developed a GPU-accelerated version of the algorithm, enabling the utilization of parallel computing to handle large amounts of point cloud data in complex registration tasks. This enhancement leverages the computational power of modern GPUs, providing substantial speed improvements and making the algorithm more practical for large-scale applications. Furthermore, we employed several heuristic algorithms in this work, such as a convergence safeguard and the gradual reduction of ν_0 when the safeguard is triggered, acting like a learning rate mechanism in machine learning scenarios. These strategies ensure robust convergence and further optimize the registration process.

Contributions

This thesis presents several key contributions to the field of point cloud registration:

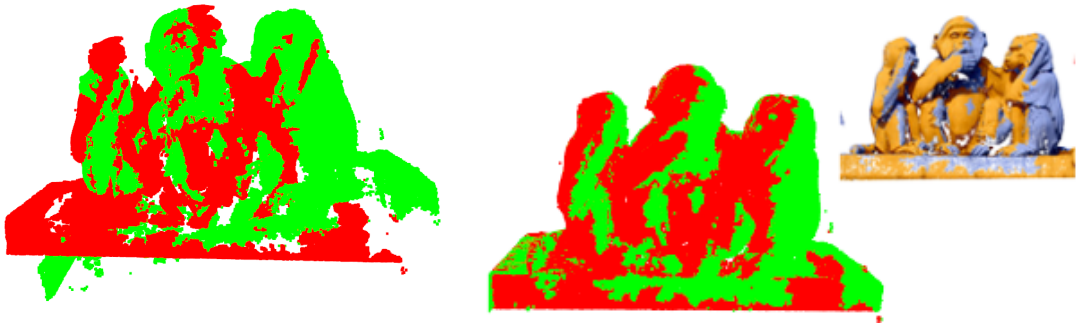
- **Overlapping Area Pruning:** Our key observation is that overlapping areas in spatially close point clouds (with small angle rotation) often contain very close but incorrect corresponding point pairs. By reducing the voting weight of these overlapping areas, we enhance the speed and accuracy of the alignment process.

- **Combine with Robust Distance Metric:** We apply the Welsch function [47] as a distance metric to replace the traditional l_2 norm. This makes the registration process more robust to outliers and noise, improving the quality of the point clouds. Additionally, we gradually decrease the variance within the Welsch function over the iterations. This approach is akin to adjusting the learning rate in machine learning training, where a higher initial variance allows for broader exploration to escape local minima, and a reduced variance over time ensures more precise and accurate alignment.

- **GPU-Accelerated Version for Large-Scale Applications:** We have developed a GPU-accelerated version [18,29]of our algorithm to handle large-scale point cloud registration tasks. This version leverages parallel computing capabilities of modern GPUs, significantly improving the processing speed and efficiency. This enhancement makes the algorithm practical for real-time and large-scale applications, enabling the handling of complex registration tasks involving extensive point cloud data.

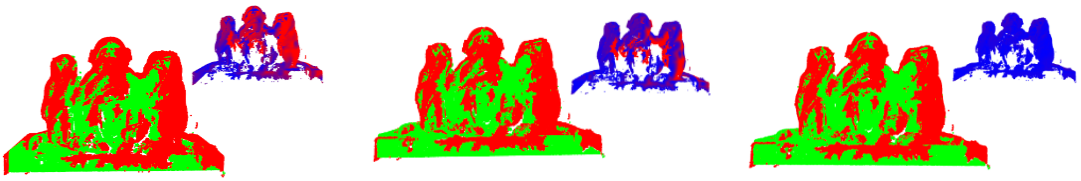
- **Heuristic Algorithms for Robust Convergence:** We employed several heuristic algorithms, such as a convergence safeguard and the gradual reduction of ν_0 when the safeguard is triggered. These methods act like a learning rate mechanism in machine learning scenarios, ensuring robust convergence and optimizing the registration process. By dynamically adjusting the parameters, we enhance the stability and efficiency of the iterative alignment procedure.

The following are comparisons of registration methods: ICP, our robust ICP, and our GPU-accelerated ICP on two overlapping point clouds constructed using the monkey statue model from the EPFL dataset [9].



Input
#S: 244501 #T: 211901

Ground Truth



ICP
RMSE=0.086,
Time=41.53 s

Our ICP
RMSE=0.096,
Time=17.2 s

Our ICP (GPU)
RMSE=0.079,
Time=21.95 s

Figure 1.1: The point clouds numbers are represented as #S and #T for the cloud of source point and the cloud of target point. Each experiments result displays the Root Mean Square Error (RMSE) and the computation time. The color-coding delineates the disparity between the source point clouds that's been transformed, as determined by the computed alignment, and the ground-truth alignment. Our GPU-accelerated ICP method achieves the lowest RMSE values and is nearly twice as fast as the original ICP.

Chapter 2

LITERATURE REVIEW

The Iterative Closest Point (ICP) algorithm, first introduced by Besl and McKay (1992) [3], is a widely used method for aligning 3D point clouds. ICP iteratively refines the transformation (rotation and translation) needed to minimize the distance between corresponding points in two point clouds. The algorithm consists of the following steps:

1. Selection of points (or features) in one or both datasets.
2. Matching of these points to the closest points in the other dataset.
3. Estimation of the transformation that minimizes the distance between corresponding points.
4. Application of the transformation to the points.
5. Repetition of the process until convergence.

Several surveys were proposed to investigate this field [17, 23]. Fig2.1 shows the main process of optimization-based registration.

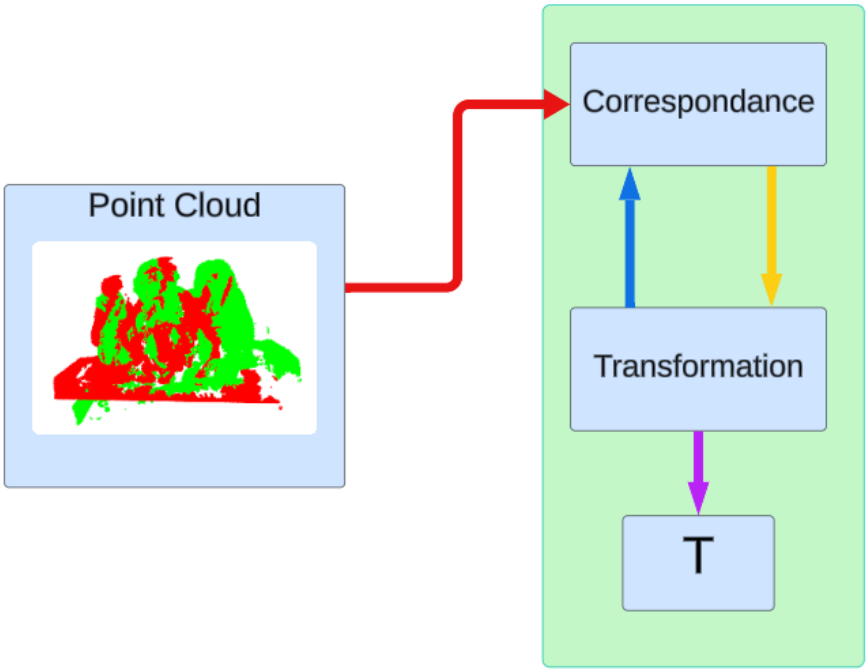


Figure 2.1: The point cloud registration process is driven by an optimization-based framework. This framework iteratively estimates the correspondences and transformations between two input point clouds. Through repeated iterations, the algorithm refines these estimates until it arrives at the optimal transformation, denoted as T, which is then produced as the final solution.

2.1 Variants of ICP

Numerous variants of the Iterative Closest Point (ICP) algorithm have been proposed [17] to improve convergence speed and robustness to noise and outliers. Additionally, partial overlap presents a significant challenge for ICP due to the difficulty in establishing correct correspondences. Hence, specialized methods have been developed to address this issue.

2.1.1 Different Metrics

The choice of metric in the ICP algorithm is critical to its performance. Commonly used metrics include point-to-point distance and point-to-plane distance. The point-to-point metric is straightforward but may not effectively capture surface geometry. In contrast, the point-to-plane metric, which takes surface normals

into account, can achieve better alignment [24]. To enhance robustness against incorrect correspondences, Generalized-ICP was proposed [36]. This method integrates plane-to-plane ICP, combining the strengths of normal ICP and point-to-plane ICP for improved performance.

2.1.2 Feature-based Correspondence

Feature-based correspondence methods aim to match points based on local geometric features rather than just spatial proximity. This approach can improve robustness, particularly in scenarios with repetitive structures or partial overlaps. Commonly used features include Fast Point Feature Histograms (FPFH) and Signature of Histograms of Orientations (SHOT) [19, 32, 35, 44].

2.1.3 Point Sampling

Point sampling techniques reduce the number of points considered in each iteration of the ICP algorithm. Methods such as uniform sampling, random sampling, and voxel grid filtering can be used to select a representative subset of points, thereby reducing computational complexity without significantly compromising accuracy. A dynamic approach, which is widely used to improve network efficiency in computer vision tasks, was proposed to identify regions where matching points cluster, ultimately enabling us to remove noisy points [1]. Another perspective from dynamical system consider point clouds translates and rotates under force and torque, using random perturbation to ensure the globally registration solution [42].

2.1.4 Random Restart

Random restart techniques involve running the ICP algorithm multiple times with different initial conditions. This approach can help mitigate the problem of local minima by exploring a broader range of possible alignments, increasing the likelihood of finding the global optimum [22].

2.1.5 Usage of Index

Indexing methods, such as KD-trees and Octrees, are used to accelerate the nearest neighbor search during the correspondence step of the ICP algorithm. Efficient indexing can significantly reduce the time complexity of this step, particularly in large point clouds [2, 12, 28].

2.1.6 Deep Learning-Based ICP

Feature Extraction Networks

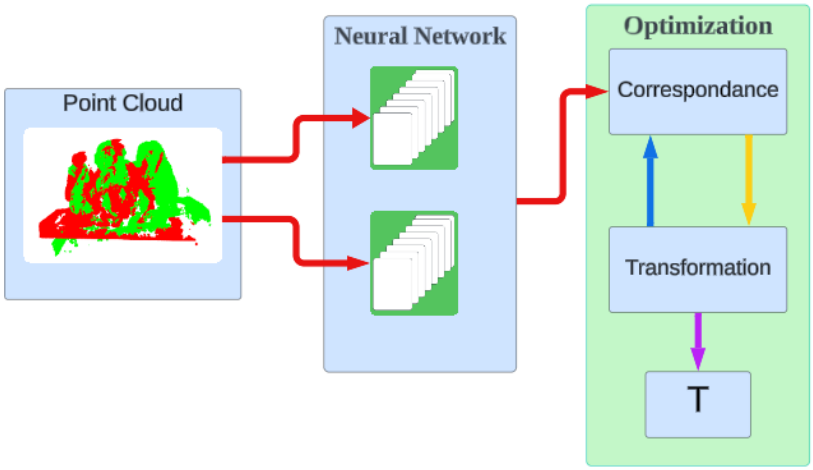
Deep learning methods for point cloud registration often start by extracting robust features from the point clouds. The Deep Closest Point (DCP) method, for example, uses a point cloud embedding network to create strong representations for registration [41]. Similarly, CorsNet combines local and global features to improve accuracy and robustness [21].

Attention Mechanisms

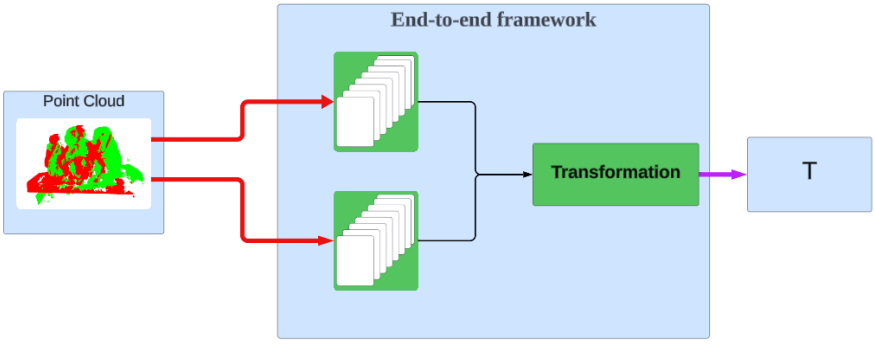
Attention mechanisms can boost the performance of deep learning-based registration methods by focusing on the most relevant parts of the point clouds. DCP includes an attention-based module to better match point clouds, helping to handle occlusions and noise effectively [41]. DOPNet also uses attention modules to extract global features, resulting in accurate point cloud registration [45].

End-to-End Training

Many deep learning-based ICP methods are trained end-to-end, meaning the entire pipeline is optimized simultaneously. IPCR, for instance, integrates global and local features with iterative processes to significantly improve registration accuracy [5]. Similarly, the Points Registration Learning (PREL) algorithm uses deep neural networks trained with sparse autoencoders and distance maps, allowing for high-accuracy, non-iterative registration [31].



(a) A feature learning-based framework for point cloud registration involves using a deep neural network to estimate features from two input point clouds. Subsequently, the framework iteratively estimates the correspondences and transformations to determine the final solution, denoted as T.



(b) An end-to-end learning-based framework for point cloud registration leverages an integrated approach to estimate the final solution, T, directly from two input point clouds.

Figure 2.2: Frameworks for point cloud registration.

Deep learning-based methods have shown impressive results in point cloud registration, but their application on devices with limited computational power, such as Raspberry Pi or other embedded systems, poses significant challenges. These small devices often lack the processing power and memory needed to efficiently run deep learning algorithms, which is especially problematic for real-time applications like autonomous drones, mobile robots, wearable augmented reality devices, and factory robot arms.

In these scenarios, traditional ICP (Iterative Closest Point) algorithms, optimized for speed and efficiency, remain crucial. For example, mobile robots navigating through cluttered environments need fast and reliable point cloud registration to avoid obstacles and map their surroundings effectively. Wearable augmented reality systems depend on efficient point cloud processing to seamlessly overlay digital information onto the physical world in real-time. Similarly, in factory settings, robot arms use traditional ICP [20,40] algorithms to accurately identify and manipulate parts of a product during assembly tasks. These devices, often limited in computational capability, benefit greatly from optimized ICP algorithms.

Recognizing these constraints, our work focuses on enhancing traditional ICP algorithms to improve their performance and usability in such practical applications. By making ICP more efficient, we aim to bridge the gap between the high performance required for point cloud registration and the computational limitations of small, embedded devices.

Chapter 3

METHODOLOGY

3.1 PROBLEM DEFINITION

Iterative Closest Points

The Iterative Closest Points (ICP) [3] algorithm is used to align two point clouds, $\mathcal{P}$ and $\mathcal{Q}$, where $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3 \mid i = 1, \ldots, N_p\}$ and $\mathcal{Q} = \{\mathbf{q}_j \in \mathbb{R}^3 \mid j = 1, \ldots, N_q\}$. The goal is to find a rigid transformation, defined by a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$, that minimizes the distance between the two point clouds.

Mathematically, the problem can be defined as:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N_p} \|\mathbf{p}_i - (\mathbf{R}\mathbf{q}_{\phi(i)} + \mathbf{t})\|^2, \tag{3.1}$$

where $\phi(i)$ is a function that finds the closest point in $\mathcal{Q}$ to the point $\mathbf{p}_i$ in $\mathcal{P}$, defined as:

$$\phi(i) = \arg\min_j \|\mathbf{p}_i - \mathbf{q}_j\|. \tag{3.2}$$

The ICP algorithm iteratively refines the transformation $(\mathbf{R}, \mathbf{t})$ by repeating

the following steps:

1. **Find Correspondences**
2. **Compute Transformation**
3. **Apply Transformation**

3.2 Overview of the ICP Algorithm

The Iterative Closest Point (ICP) [3]algorithm is a widely used iterative method for aligning 2D, 3D point clouds. The core idea of the algorithm is to alternately perform the following two steps until convergence:

1. **Closest Point Correspondence Establishment**
2. **Rigid Transformation Estimation and Update**

Specifically, the ICP algorithm can be divided into the following key steps:

1. **Initialization**: Given the source point cloud $\mathcal{P}$ and the target point cloud $\mathcal{Q}$, along with an initial rigid transformation $\mathbf{T}_0$ that roughly aligns $\mathcal{P}$ to $\mathcal{Q}$.
2. **Iteration**:
   (a) **Nearest Point Correspondence**: For each point $\mathbf{p}_i$ in the current source point cloud $\mathcal{P}$, find its nearest neighbor $\mathbf{q}_i$ in the target point cloud $\mathcal{Q}$, establishing a set of point correspondences $\mathcal{C} = \{(\mathbf{p}_i, \mathbf{q}_i)\}$.
   (b) **Rigid Transformation Estimation**: Based on the set of point correspondences $\mathcal{C}$, compute a new rigid transformation matrix $\mathbf{T}_k$ that minimizes the distance error between the transformed source point cloud $\mathbf{T}_k(\mathcal{P})$ and its corresponding points.

(c) **Transformation Update**: Apply the estimated rigid transformation $\mathbf{T}_k$ to the source point cloud $\mathcal{P}$, generating a new source point cloud $\mathcal{P}' = \mathbf{T}_k(\mathcal{P})$.

(d) **Convergence Check**: Determine if the convergence criteria are met. If not, return to step (a) and repeat the iteration.

3. **Output**: When the algorithm converges or reaches the maximum number of iterations, output the final estimated rigid transformation $\mathbf{T}^*$ that aligns the source point cloud $\mathcal{P}$ to the target point cloud $\mathcal{Q}$.

In the iterative process, steps (a) and (b) are performed alternately until the alignment accuracy can no longer be significantly improved or other termination conditions are met.

3.3 Optimizations

To enhance the performance and accuracy of the ICP algorithm, several optimizations have been applied:

3.3.1 KD-Tree for Nearest Neighbor Search

Implementing a KD-Tree to expedite the nearest neighbor search process in the ICP algorithm [2, 12, 28]. The KD-Tree allows the algorithm to efficiently partition the space and quickly identify the closest points between the target and source datasets. This optimization significantly reduces computational complexity and enhances the overall speed of the convergence process.

The KD-Tree data structure partitions the space into a series of nested hyper-rectangles, which can be searched quickly to find the nearest neighbors.

3.3.2 Outliers Threshold Filtering

Discarding point pairs whose distances exceed a certain threshold to improve robustness. This step ensures that outliers and erroneous matches do not affect the alignment process.

3.3.3 Robust Registration via Welsch's Error Metric

To enhance resilience against outliers and foster sparse solutions in point set alignment, we leverage the Welsch function [15] as our error metric, following the approach proposed in [47]. This function, denoted as $\psi_n(x)$, exhibits monotonic growth over the non-negative real line $[0, +\infty)$ and is formulated as:

$$\psi_n(x) = 1 - \exp\left(-\frac{x^2}{2n^2}\right)$$

where n represents a user-defined parameter.

The monotonically increasing nature of $\psi_n(x)$ on $[0, +\infty)$ induces a penalization of deviations between the point sets within our formulation. Concurrently, ψ_n is upper-bounded by unity, rendering our metric insensitive to substantial deviations stemming from outliers and partially overlapping regions. Furthermore, as n approaches zero, the summation $\sum_{i=1}^{n} \psi_n(r_i)$ approximates the ℓ_0-norm of the vector $[D_M(R,t), \ldots, D_M(R,t)]$, thereby promoting sparsity in the point-wise distances between the point sets. In recent years, error metrics derived from Welsch's function have found applications in robust registration tasks within image processing [13] and geometry processing domains [46].

3.3.4 Overlapping Region Pruning

Pruning the corresponding points in overlapping regions enhances convergence speed. Overlapping regions are prone to incorrect point correspondences due to

close proximity, so pruning these points helps accelerate the convergence process. In overlapping regions, the corresponding points often match to very close but incorrect points. Including these points in the alignment step can lead to suboptimal transformations. For example, minimizing the following objective function including overlapping points:

$$(\mathbf{R}_{\text{incl}}^{(k+1)}, \mathbf{t}_{\text{incl}}^{(k+1)}) = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M_{\text{incl}}} |\mathbf{R}\mathbf{p}_i + \mathbf{t} - \hat{\mathbf{q}}_i|^2$$

where M_{incl} includes all points, including those in overlapping regions. This solution tends to stand still due to the influence of closely matched but incorrect points. On the other hand, excluding the overlapping points:

$$(\mathbf{R}_{\text{excl}}^{(k+1)}, \mathbf{t}_{\text{excl}}^{(k+1)}) = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M_{\text{excl}}} |\mathbf{R}\mathbf{p}_i + \mathbf{t} - \hat{\mathbf{q}}_i|^2$$

where M_{excl} excludes points in overlapping regions, results in a transformation that is more inclined to move and adjust appropriately. By excluding these points, the alignment process becomes more robust and converges faster, as the algorithm avoids being trapped by incorrect close correspondences.

Definition of Overlapping Area

The definition of the overlapping area is based on the distance between point pairs relative to a threshold derived from local point density. The process for identifying overlapping points is as follows:

1. For each point, calculate the average distance to its six nearest neighbors.
2. Take the median of these average distances as the threshold, denoted as $d_{\text{threshold}}$.

3. For each point pair $(\mathbf{p}_i, \hat{\mathbf{q}}_i)$, if their distance is less than $d_{\text{threshold}}$, i.e.,

$$|\mathbf{p}_i - \hat{\mathbf{q}}_i| < d_{\text{threshold}}$$

then this point pair is considered to be in the overlapping region.

3.3.5 Heuristic Convergence Criterion

To enhance the convergence behavior and prevent divergence during the iterative registration process, we employ a heuristic criterion based on monitoring the mean distance between corresponding point pairs across successive iterations. Let $\mathcal{C}_k = \{(\mathbf{p}_i, \mathbf{q}_i)\}$ denote the set of point correspondences obtained in the k-th iteration, and d_k represent the mean distance between corresponding pairs, defined as:

$$d_k = \frac{1}{|\mathcal{C}_k|} \sum_{(\mathbf{p}_i, \mathbf{q}_i) \in \mathcal{C}_k} \|\mathbf{p}_i - \mathbf{q}_i\|_2$$

Our heuristic criterion operates as follows:

1. Compare d_k with the mean distance from the previous iteration, d_{k-1}.
2. If $d_k < d_{k-1}$, proceed to the next iteration, as the registration is improving.
3. If $d_k \geq d_{k-1}$, modify the current iteration strategy by adjusting a parameter ν to improve convergence stability. Here, $\nu = \nu_0 \cdot d_{med}$, where d_{med} is the median distance between all corresponding point pairs and ν_0 is a user-specified parameter [6].

Initially, ν_0 is set to a larger value, acting like a higher learning rate in machine learning scenarios. If the safeguard is triggered (i.e., $d_k \geq d_{k-1}$), we replace the current ν_0 with a smaller value from a predefined list of ν_0 values. This adjustment

continues iteratively, reducing the learning rate-like parameter, ν_0, gradually over successive iterations.

By doing this, we mimic the strategy used in machine learning where a larger learning rate is used at the beginning for faster convergence and gradually decreased to refine the model. This approach ensures that the registration process starts with a larger ν to make significant progress and reduces ν to fine-tune the alignment as iterations progress, ensuring a robust and efficient registration procedure.

3.3.6 Random Downsampling Before Overlapping Region Pruning

In this work, we apply random downsampling to the point cloud before downsampling the overlapping area. The rationale behind this approach is that, since it is a rigid motion registration, it is not necessary to fit all the points to determine if the alignment is accurate. By fitting a smaller, representative subset of points, it is possible to achieve a good fit for the entire point cloud.

This approach helps in reducing the computational load and speeds up the convergence process without compromising the accuracy of the registration. Randomly selecting a subset of points ensures that the selected points are well distributed across the entire point cloud, providing a good representation for the registration process.

1. Randomly downsample the point cloud to a smaller subset.

2. Proceed with the overlapping region pruning on the downsampled subset.

This strategy leverages the efficiency of working with fewer points while maintaining the integrity of the registration process, ensuring both speed and accuracy.

3.3.7 GPU-Accelerated Computing

The GPU acceleration dramatically enhances the performance of our ICP algorithm by exploiting the parallel processing capabilities of modern GPUs [33]. Unlike CPUs, which have a limited number of cores optimized for sequential processing, GPUs boast thousands of smaller cores designed to handle multiple operations simultaneously. This architecture is exceptionally effective for the ICP algorithm, which involves numerous independent computations, such as finding the nearest neighbors and updating transformation matrices. By distributing these tasks across many cores, the GPU significantly accelerates the computation of point correspondences and the iterative refinement process, resulting in much faster convergence times, particularly in large-scale point cloud datasets. The ability to perform a high volume of floating-point operations in parallel empowers our GPU-accelerated ICP to process more data points with greater efficiency, thereby achieving superior performance in both speed and accuracy.

We have implemented GPU acceleration using the CuPy [18] Python package to handle extensive vector and matrix computations. This optimization capitalizes on the parallel computing capabilities of modern GPUs, markedly improving the processing speed and efficiency of the algorithm. By offloading computationally intensive tasks to the GPU, we handle large-scale point cloud registration tasks more effectively, making the algorithm highly practical for real-time and complex applications. This implementation ensures that our ICP algorithm is not only faster but also capable of managing the demands of modern, data-intensive environments.

3.4 Mathematical Formulation

The ICP algorithm consists of two main steps: correspondence and alignment, which are optimized as follows:

3.4.1 Correspondence

In the correspondence step, the KD-Tree data structure accelerates the search for nearest neighbors between the 3D point sets. Given the datasets P (source) and Q (target), the nearest neighbor for each point $p \in P$ in Q is determined efficiently using the KD-Tree, leveraging the spatial partitioning properties of the data structure.

Mathematically, this can be represented as:

$$\text{NN}(p_i) = \arg\min_{q_j \in Q} \|p_i - q_j\|$$

where $\text{NN}(p_i)$ denotes the nearest neighbor of point p_i in dataset Q.

3.4.2 Alignment

Once the corresponding points are identified, the alignment process begins. Here, point pairs with distances greater than a specified threshold are rejected to ensure robustness against outliers and erroneous matches. Additionally, points in overlapping regions are pruned to accelerate convergence, as these regions are prone to incorrect point correspondences due to close proximity.

The transformation matrix T that aligns P to Q is computed using the Singular Value Decomposition (SVD) of the cross-covariance matrix between the corresponding points, providing a closed-form solution [37]. This matrix T minimizes the squared distance between the datasets under the chosen distance metric.

The optimization can be formulated as:

$$T = \arg\min_{T} \sum_{i} \rho(\|T(p_i) - q_i\|^2)$$

where ρ is the distance metric. Using the Welsch function as the distance metric helps to reduce the influence of outliers, and it is defined as:

$$\rho(x^2) = 1 - \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

Here, σ is a scaling parameter.

The alignment problem with the Welsch function as the distance metric is formulated as [47]:

$$(\mathbf{R}^{(k+1)}, \mathbf{t}^{(k+1)}) = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M} \omega_i \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \hat{\mathbf{q}}_i^{(k)} \right\|^2$$

where ω_i is defined as:

$$\omega_i = \exp\left(-\frac{\left\| \mathbf{R}^{(k)}\mathbf{p}_i + \mathbf{t}^{(k)} - \hat{\mathbf{q}}_i^{(k)} \right\|^2}{2\nu^2}\right)$$

We can obtain a closed form via SVD [37].

This formulation provides a comprehensive and optimized approach to implementing the ICP algorithm, ensuring better accuracy and faster convergence in practical applications.

ORDS-ICP Algorithm Pseudocode

Algorithm 1: ORDS-ICP Algorithm

Input : A, B, *init_pose*, *max_iterations*, *tolerance*, *distance_threshold*, *overlap_threshold*, *overlap_decline*

Output: T, *valid_distances*, *iterations*, *mean_distances*

mean_distances $\leftarrow$ [ ];
Downsample A to 5000 points;
Initialize *src* and *dst* as homogeneous coordinates of A and B;
if *init_pose* $\neq$ *None* **then**
    Apply *init_pose* to *src*;
Fit nearest neighbors for *dst*;
Compute initial mean error and append to *mean_distances*;
for $i \leftarrow 0$ **to** *max_iterations* **do**
    Find nearest neighbors for *src*;
    Filter out point pairs with distance greater than *distance_threshold*;
    Find overlapping points between *src* and *dst*;
    Randomly reduce a certain percentage of overlapping points based on *overlap_decline*^{i};
    Remove overlapping points from valid indices;
    Compute weight matrix W based on distances and iteration i;
    Compute best fit transform T using *src* and *dst* with weighting;
    Apply transformation T to *src*;
    Compute new mean error and append to *mean_distances*;
    if *convergence criteria met* **then**
        Break;
Compute final transform T using normal best fit transform;
return T, *valid_distances*, *iterations*, *mean_distances*;

Chapter 4

EXPERIMENT RESULTS

This section presents a comparison of the performance of this works with the original ICP method. Becuase of the high computational cost, we employ KD-tree ICP to accelerate the convergence speed. Our implementation is carried out in Python, leveraging the NumPy [14] library for linear algebra operations and the CuPy [29] library for GPU-accelerated computations. For every test case, in order to standardize the input data, we align the centre of the point clouds. and applying uniform scaling so that the diagonal length of their bounding box is 1. In other work [47], point clouds were pre-aligned using Super4PCS [27] before applying different refinement methods. We skip the pre-alignment step and directly use relatively close point clouds for our experiments.

This work examine the proposed algorithms on synthetic datasets such as the EPFL statue dataset [9] and Open3D dataset [30]. For problems that the ground-truth alignment is known, we measure the alignment accuracy using the Root Mean Square Error (RMSE). All algorithms are executed on a computer equipped with a 6-core CPU at 2.2GHz and 8GB of RAM.

4.1 Synthetic Data

In Figure 1.1, we demonstrate the registration of two point clouds with added noise, using the monkey statue model from the EPFL statue dataset [9]. The source point cloud comprises the first 60% of the model's points, while the target point cloud is constructed from the last 52% after applying a rigid transformation. Our GPU-accelerated ICP achieves the lowest RMSE value of 0.079 and completes the process in 21.95 seconds, proving to be twice as fast as classical ICP. Although our ICP achieves the highest RMSE value of 0.096, it converges the fastest at 17.2 seconds, due to our heuristic algorithm that detects divergence trends and terminates iterations early. Our non-GPU-accelerated ICP was applied to a point cloud reduced by one-third due to high computational costs, resulting in the poorest RMSE among all methods. Classical ICP, with an RMSE of 0.086 and a time of 41.53 seconds, delivers the second-lowest RMSE but struggles with slow convergence in large point cloud registration tasks.

In Figure 4.1, we rigorously evaluate the methods on point cloud sets with varying noise levels and outliers. Table 4.1 provides a summary of the test data. These sets are derived from the Eminescu model in the EPFL statue dataset [9]. We extract the last 52% of the points to form the cloud of target points. Gaussian noise is systematically added alongside their normal directions, with the std precisely set to the average median distance of all points to their six nearest neighbors [48]. Finally, a rigid transformation is meticulously applied.

We begin by extracting the first 60% of points from the full model for the source point cloud and applying Gaussian noise, consistent with the noise added to the target point cloud. To accurately simulate outliers, we introduce $\mu \cdot M$ random points, uniformly distributed within the bounding box of the source point cloud. Here, M represents the number of source points before adding outliers, and

μ denotes the outlier percentage. We specifically select μ values of 1%, 5%, 10%, and 20% of the original number of source points.

The bounding box of a point cloud is the smallest rectangular box that can completely contain all the points in the cloud. Mathematically, if the point cloud consists of points $\{(x_i, y_i, z_i)\}$, the bounding box is defined by the minimum and maximum coordinates in each dimension:

$$\text{Bounding Box} = \{(x, y, z) \mid x_{\min} \leq x \leq x_{\max},\ y_{\min} \leq y \leq y_{\max},\ z_{\min} \leq z \leq z_{\max}\}$$

where

$$x_{\min} = \min_i(x_i), \quad x_{\max} = \max_i(x_i)$$

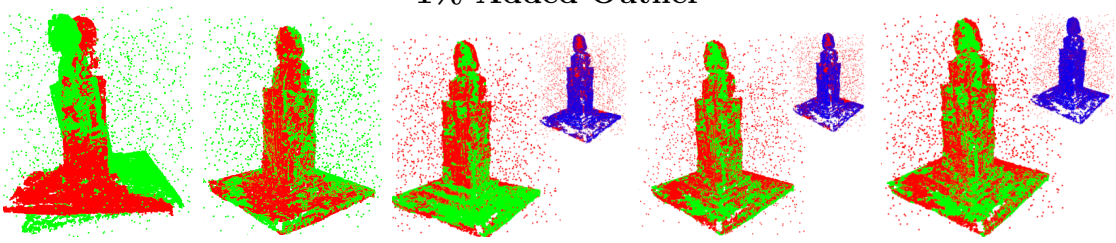
$$y_{\min} = \min_i(y_i), \quad y_{\max} = \max_i(y_i)$$

$$z_{\min} = \min_i(z_i), \quad z_{\max} = \max_i(z_i)$$

Random points are generated within this bounding box to serve as outliers.

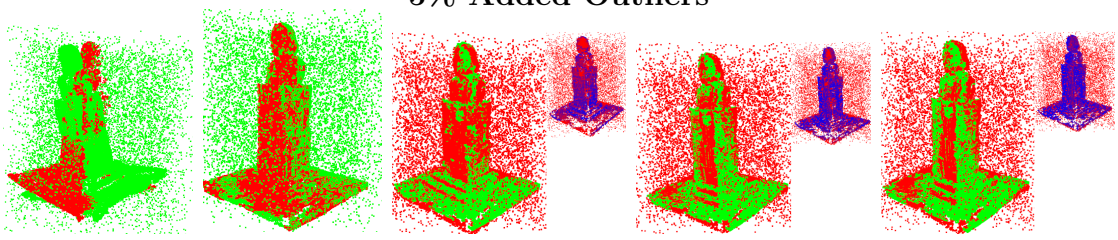
For problems with up to $\mu = 10\%$ added outliers, our robust registration methods outperform traditional ICP by an order of magnitude, with our GPU-accelerated ICP achieving the best accuracy among all methods. For $\mu = 20\%$ added outliers, both our non-GPU ICP and GPU-accelerated ICP converge 1.5 times faster than traditional ICP, while maintaining an order of magnitude better RMSE value. Our non-GPU ICP converges faster in cases with $\mu = 10\%$ and $\mu = 20\%$ due to hardware limitations, where we reduce the point cloud size by half, significantly decreasing computational time. Additionally, in GPU-accelerated ICP, the need for vectors to commute between CPU and GPU contributes to slower convergence times in small-scale point cloud scenarios.

1% Added Outlier



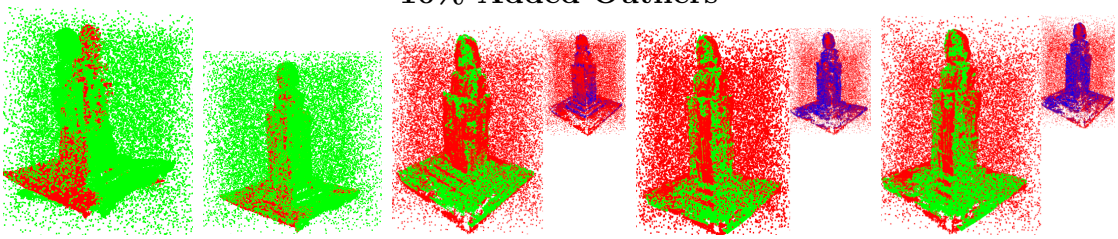
Input	Ground Truth	ICP	Our ICP	Our ICP(GPU)
#S:157199		r=7.6×10^{-3}	r=7.5×10^{-3}	r=7.5×10^{-3}
#T:136239		t=18s	t= 17.6s	t= 14.5s

5% Added Outliers



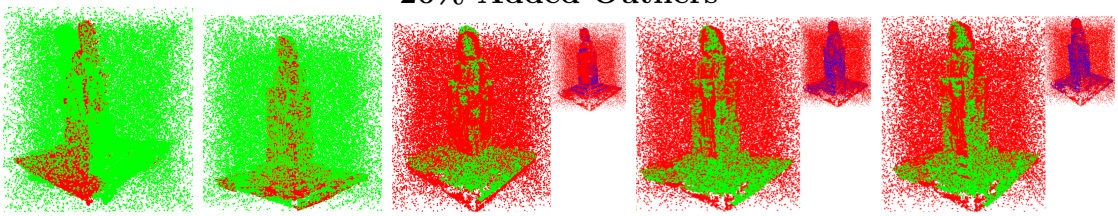
Input	Ground Truth	ICP	Our ICP	Our ICP(GPU)
#S:157199		r=8.53×10^{-2}	r=7.9×10^{-3}	r=7.75×10^{-3}
#T:136239		t=18.42s	t= 23.48s	t= 21.3s

10% Added Outliers



Input	Ground Truth	ICP	Our ICP	Our ICP(GPU)
#S:157199		r=1.03×10^{-2}	r=8.45×10^{-3}	r=8.35×10^{-3}
#T:136239		t=20.45s	t= 15.45s	t= 17.54s

20% Added Outliers



Input	Ground Truth	ICP	Our ICP	Our ICP(GPU)
#S:157199		r=1.37×10^{-2}	r=9×10^{-3}	r=9.21×10^{-3}
#T:136239		t=24.05s	t= 16.14s	t= 16.74s

Figure 4.1: Comparison of traditional ICP with our ICP variants method on point cloud with added noises and outliers, constructed using the Eminescu statue model from EPFL statue dataset [9]. The graph at different outlier levels(1%, 5%, 10%, and 20%). #S and #T represent the number of Source point cloud and Target point cloud respectively. The blue figure is color coding from close to far distance representing from blue to red.

Table 4.1: Comparison of ICP methods with different levels of outliers

	Our ICP	**Our ICP_GPU**	**Normal ICP**
1% add outlier	7.5×10^{-3}/18s	7.5×10^{-3}/14.5s	7.6×10^{-3}/17.6s
5% add outlier	7.9×10^{-3}/23.48s	7.75×10^{-3}/21.3s	8.5×10^{-3}/18.42s
10% add outlier	8.45×10^{-3}/15.45s	8.35×10^{-3}/17.5s	1.03×10^{-2}/20.45s
20% add outlier	9×10^{-3}/16.14s	9.21×10^{-3}/16.74s	1.37×10^{-2}/24.05s

4.2 Limitations

Our robust methods, like other ICP-based techniques, depend heavily on good initial alignment. For particularly challenging problems involving large rotation angles or minimal overlap, deploying a global registration method is imperative before applying more advanced refinement techniques. Our heuristic algorithm is designed to reject corresponding point pairs with large distances and to prune points that merely overlap, which can result in a lack of representative points for refinement. In scenarios with synthetic data where the overlapping boundary area is less than 10%, our method may quickly diverge within a few iterations. This divergence triggers the safeguard mechanism in our heuristic algorithm, leading to an early termination and ultimately resulting in incorrect alignment.

Chapter 5

CONCLUSION AND FUTURE WORKS

We have proposed methods to enhance the convergence speed and robustness of point-to-point ICP methods. Our approach includes several optimizations, such as employing a KD-tree data structure to expedite the nearest corresponding point search, which is a challenging task when using brute force search on large point clouds. By avoiding the complexity of the worst-case $O(N^2)$ in brute force search and replacing it with the worst-case complexity $O(N)$ in KD-tree search, we significantly improve efficiency.

Additionally, we introduce an overlapping area pruning technique based on the key observation that, in the first few iterations, overlapping areas often have very close and incorrect corresponding point pairs. This acts like a voting mechanism where close point pairs vote for stability, while farther point pairs vote for significant adjustments to minimize the distance between corresponding points.

Furthermore, we apply the Welsch function as the metric [47], adjusting the parameter ν to control convergence time and outlier robustness. The parameter ν in the Welsch metric functions like a learning rate; we use a larger ν in the initial

iterations and gradually decrease it to achieve more precise registration.

Finally, we leverage the parallel computing capabilities of modern GPUs, utilizing the CuPy Python package to handle large-scale point clouds efficiently. These optimizations collectively contribute to substantial improvements in both the speed and robustness of ICP methods.

5.1 Future Work

Leveraging the computational power of modern GPUs, a promising direction for future work is to integrate deep neural networks for identifying representative points and pruning the rest, thereby creating an end-to-end model. This approach will fully exploit GPU capabilities, enabling more precise results and faster convergence times for large-scale registration tasks.

One specific avenue to explore is training the neural network on datasets with minimal overlap. By exposing the network to a diverse and extensive set of challenging scenarios, the model can learn robust alignment strategies through a massive amount of data. This training process will enhance the network's ability to handle real-world cases where traditional methods struggle due to large rotation angles or minimal overlap.

Furthermore, developing adaptive algorithms that can dynamically adjust parameters, such as the Welsch function's ν, based on real-time feedback from the registration process, could further improve the efficiency and accuracy of the method. Integrating these adaptive mechanisms with deep learning techniques will result in a more resilient and versatile registration framework.

In addition, expanding the application of our GPU-accelerated techniques to other point cloud processing tasks, such as segmentation, classification, and object detection, will be an essential part of future research. By doing so, we can create a

comprehensive suite of tools for various applications in computer vision, robotics, and autonomous systems.

Overall, the combination of deep learning and GPU acceleration holds great potential for advancing the field of point cloud registration, offering substantial improvements in performance and capability.

Bibliography

[1] Yang Ai and Xi Yang. A dynamic network for efficient point cloud registration. *ArXiv*, abs/2312.02877, 2023.

[2] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

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

[4] Bo-Rui Chen. Riverbed grain size survey using rgb-d images and deep learning model. Master's thesis, National Yang Ming Chiao Tung University, 2024. Unpublished master's thesis.

[5] Jiaqi Chen, Zhibao Su, and Xijun Zhao. Ipcr: an end-to-end iterative point cloud registration network. *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, pages 894–899, 2022.

[6] Zhi Deng, Yuxin Yao, Bailin Deng, and Juyong Zhang. A robust loss for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6138–6147, 2021.

[7] Chinthaka Dinesh, Gene Cheung, and Ivan V Bajić. Point cloud denoising via feature graph laplacian regularization. *IEEE Transactions on Image Processing*, 29:4143–4158, 2020.

[8] Li Ding and Chen Feng. Deepmapping: Unsupervised map estimation from multiple point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8650–8659, 2019.

[9] EPFL Computer Graphics and Geometry Laboratory. EPFL statue model repository, 2012. Accessed: 2024-06-13.

[10] Andrew W Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and vision computing*, 21(13-14):1145–1153, 2003.

[11] Yaorong Ge, Calvin R Maurer Jr, and J Michael Fitzpatrick. Surface-based 3d image registration using the iterative closest-point algorithm with a closest-point transform. In *Medical Imaging 1996: Image Processing*, volume 2710, pages 358–367. SPIE, 1996.

[12] Michael Greenspan and Mike Yurick. Approximate kd tree search for efficient icp. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 442–448. IEEE, 2003.

[13] Bumsub Ham, Minsu Cho, and Jean Ponce. Robust guided image filtering using nonconvex potentials. *IEEE transactions on pattern analysis and machine intelligence*, 40(1):192–207, 2017.

[14] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[15] Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977.

[16] Wenguang Hou, Taiwai Chan, and Mingyue Ding. Denoising point cloud. *Inverse Problems in Science and Engineering*, 20(3):287–298, 2012.

[17] Xiaoshui Huang, Guofeng Mei, Jian Zhang, and Rana Abbas. A comprehensive survey on point cloud registration. *arXiv preprint arXiv:2103.02690*, 2021.

[18] Preferred Networks Inc. and Preferred Infrastructure Inc. Cupy: Numpy & scipy for gpu-accelerated computing with python. `https://docs.cupy.dev/`, 2024. Accessed: 2024-06-13.

[19] J Jaw, T Chuang, et al. Feature-based registration of terrestrial lidar point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37(303-308):2, 2008.

[20] Suhui Ji, Wentao Li, Zhen Zhang, Shijun Zhou, Zhiyuan Cai, and Jiandong Tian. Robotic arm grasping through 3d point clouds recognition. In *2021 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 347–352. IEEE, 2021.

[21] Akiyoshi Kurobe, Yusuke Sekikawa, Kohta Ishikawa, and H. Saito. Corsnet: 3d point cloud registration by deep neural network. *IEEE Robotics and Automation Letters*, 5:3960–3966, 2020.

[22] X. Leng, Jun Xiao, and D. Li. An initial registration method of point clouds based on random sampling. *Applied Mechanics and Materials*, 513-517:3680 – 3683, 2014.

[23] Leihui Li, Riwei Wang, and Xuping Zhang. A tutorial review on point cloud registrations: principle, classification, comparison, and technology challenges. *Mathematical Problems in Engineering*, 2021(1):9953910, 2021.

[24] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4(10):1–3, 2004.

[25] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-net: Towards learning based lidar localization for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6389–6398, 2019.

[26] Bilawal Mahmood and SangUk Han. 3d registration of indoor point clouds for augmented reality. In *ASCE International Conference on Computing in Civil Engineering 2019*, pages 1–8. American Society of Civil Engineers Reston, VA, 2019.

[27] Nicolas Mellado, Dror Aiger, and Niloy J Mitra. Super 4pcs fast global point-cloud registration via smart indexing. In *Computer graphics forum*, volume 33, pages 205–215. Wiley Online Library, 2014.

[28] Andreas Nuchter, Kai Lingemann, and Joachim Hertzberg. Cached kd tree search for icp algorithms. In *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007)*, pages 419–426. IEEE, 2007.

[29] NVIDIA. Gpu acceleration in python using cupy and numba. In *NVIDIA GTC*. NVIDIA, 2021.

[30] Open3D. Open3d dataset, 2024. Accessed: 2024-06-13.

[31] J. Perez-Gonzalez, Fernando Luna-Madrigal, and O. Piña-Ramírez. Deep learning point cloud registration based on distance features. *IEEE Latin America Transactions*, 17:2053–2060, 2019.

[32] Tomas Petricek and Tomas Svoboda. Point cloud registration from local feature correspondences—evaluation on challenging datasets. *PLoS One*, 12(11):e0187943, 2017.

[33] Md Mushfiqur Rahman, Panagiota Galanakou, and Georgios Kalantzis. A fast gpu point-cloud registration algorithm. In *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 111–116, 2018.

[34] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001.

[35] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[36] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[37] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1):1–5, 2017.

[38] Charles V Stewart, Chia-Ling Tsai, and Badrinath Roysam. The dual-bootstrap iterative closest point algorithm with application to retinal image registration. *IEEE transactions on medical imaging*, 22(11):1379–1394, 2003.

[39] Meng Tian, Liang Pan, Marcelo H Ang, and Gim Hee Lee. Robust 6d object pose estimation by learning rgb-d features. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6218–6224. IEEE, 2020.

[40] Petras Vestartas and Yves Weinand. Laser scanning with industrial robot arm for raw-wood fabrication. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 37, pages 773–780. IAARC Publications, 2020.

[41] Yue Wang and J. Solomon. Deep closest point: Learning representations for point cloud registration. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3522–3531, 2019.

[42] Heng Yang. A dynamical perspective on point cloud registration. *ArXiv*, abs/2005.03190, 2020.

[43] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-icp: Solving 3d registration efficiently and globally optimally. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1457–1464, 2013.

[44] Jiaqi Yang, Zhiguo Cao, and Qian Zhang. A fast and robust local descriptor for 3d point cloud registration. *Information Sciences*, 346:163–179, 2016.

[45] R. Yi, Jinlong Li, Lin Luo, Yu Zhang, Xiaorong Gao, and Jianqiang Guo. Dopnet: Achieving accurate and efficient point cloud registration based on deep learning and multi-level features. *Sensors (Basel, Switzerland)*, 22, 2022.

[46] Juyong Zhang, Bailin Deng, Yang Hong, Yue Peng, Wenjie Qin, and Ligang Liu. Static/dynamic filtering for mesh geometry. *IEEE transactions on visualization and computer graphics*, 25(4):1774–1787, 2018.

[47] Juyong Zhang, Yuxin Yao, and Bailin Deng. Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3450–3466, 2021.

[48] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 766–782. Springer, 2016.