CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

ORCA – Online Research @ Cardiff

This is an Open Access document downloaded from ORCA, Cardiff University's institutional repository:https://orca.cardiff.ac.uk/id/eprint/137995/

This is the author's version of a work that was submitted to / accepted for publication.

Citation for final published version:

Zhang, Juyong, Yao, Yuxin and Deng, Bailin 2022. Fast and robust iterative closest point. IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (7) , pp. 3450-3466. 10.1109/TPAMI.2021.3054619

Publishers page: http://dx.doi.org/10.1109/TPAMI.2021.3054619

Please note:
Changes made as a result of publishing processes such as copy-editing, formatting and page numbers may not be reflected in this version. For the definitive version of this publication, please refer to the published source. You are advised to consult the publisher's version if you wish to cite this paper.

This version is being made available in accordance with publisher policies. See http://orca.cf.ac.uk/policies.html for usage policies. Copyright and moral rights for publications made available in ORCA are retained by the copyright holders.

Fast and Robust Iterative Closest Point

Juyong Zhang, *Member, IEEE,* Yuxin Yao, Bailin Deng†, *Member, IEEE*

Abstract—The Iterative Closest Point (ICP) algorithm and its variants are a fundamental technique for rigid registration between two point sets, with wide applications in different areas from robotics to 3D reconstruction. The main drawbacks for ICP are its slow convergence as well as its sensitivity to outliers, missing data, and partial overlaps. Recent work such as Sparse ICP achieves robustness via sparsity optimization at the cost of computational speed. In this paper, we propose a new method for robust registration with fast convergence. First, we show that the classical point-to-point ICP can be treated as a majorization-minimization (MM) algorithm, and propose an Anderson acceleration approach to speed up its convergence. In addition, we introduce a robust error metric based on the Welsch's function, which is minimized efficiently using the MM algorithm with Anderson acceleration. On challenging datasets with noises and partial overlaps, we achieve similar or better accuracy than Sparse ICP while being at least an order of magnitude faster. Finally, we extend the robust formulation to point-to-plane ICP, and solve the resulting problem using a similar Anderson-accelerated MM strategy. Our robust ICP methods improve the registration accuracy on benchmark datasets while being competitive in computational time.

Index Terms—Rigid Registration, Robust Estimator, Fixed-point iterations, Majorlazer Minimization method, Anderson Acceleration.

1 Introduction

Rigid registration, which finds an optimal rigid transformation to align a source point set with a target point set, is a fundamental problem in computer vision and many other areas. The iterative Closest Point (ICP) algorithm [1] is a classical method for rigid registration. It alternates between closest point query in the target set and minimization of distance between corresponding points, and is guaranteed to converge to a locally optimal alignment. However, classical ICP can suffer from slow convergence due to its linear convergence rate [2]. Other registration methods have been developed with faster convergence. For example, in [3] the alignment is performed by minimizing a point-to-plane distance, whereas in [4] a locally quadratic approximant of the squared distance function is minimized. Both approaches are shown to have faster convergence rate than classical ICP [2]. Another issue with ICP is that the alignment accuracy can be affected by imperfections of the point sets such as noises, outliers and partial overlaps, which often occur in real-world acquisition processes. Various techniques have been developed to address this problem. One popular approach is to disregard erroneous correspondence between points, using heuristics based on their distance or the angle between their normals [5]. Recently, an ℓ_p-norm minimization approach is proposed in [6] to induce sparsity of the distance between corresponding point pairs, which aligns the points in true correspondence while allowing large distance due to outliers and incomplete data.

In this paper, we propose a novel and simple approach to address these two issues. Our key observation is that classical ICP is a majorization-minimization (MM) algorithm [7] for minimizing ℓ_2 distance between the two point sets, which iteratively constructs and minimizes a surrogate function and ensures monotonic decrease of the target energy. By treating this process as a fixed-point iteration, we speed up its convergence using *Anderson acceleration* (AA) [8], an established numerical technique that proves effective for a variety of optimization problems in computer graphics [9]. In each iteration, Anderson acceleration computes an accelerated iterate based on the history of m previous iterates. Compared with existing approaches such as [3], [4], our method does not require higher-order information such as normal or curvature which may not be available from the point cloud data and need to be estimated carefully in the presence of noise [10]. Moreover, different from previous attempt on Anderson acceleration of ICP [11] that uses Euler angles to represent rotation, we adopt a parameterization of rigid transformation that does not suffer from singularity of Euler angles. Using the same MM framework, we can replace the squared distance metric used in classical ICP with a robust metric that is insensitive to noises, outliers, and partial overlaps. In particular, we adopt a robust metric based on the Welsch's function [12], which allows for a simple quadratic surrogate function and can be minimized efficiently. Compared to the sparse ICP algorithm [6], our approach does not require introducing auxiliary variables for the solver, which leads to lower memory footprint and significantly faster convergence. We conduct a variety of experiments on both synthetic and real data, where our method improves the speed and robustness for alignment. Our approach can also be extended to other ICP formulations. In particular, we apply it to the point-to-plane ICP from [3], and achieve better registration accuracy than the original method on benchmark datasets. This illustrates the effectiveness of our method in improving robustness of ICP-type registration algorithms.

To summarize, our main contributions include:

- We propose a new formulation for Anderson-accelerated point-to-point ICP method. We parameterize rigid transformations via Lie algebra instead of Euler angles as in [11], and use a more simple stabilization strategy than [11] that is easier to implement while guaranteeing monotonic decrease of the target energy.

- *J. Zhang and Y. Yao are with School of Mathematical Sciences, University of Science and Technology of China.*
- *B. Deng is with School of Computer Science and Informatics, Cardiff University.*

†*Corresponding author. Email:* DengB3@cardiff.ac.uk.

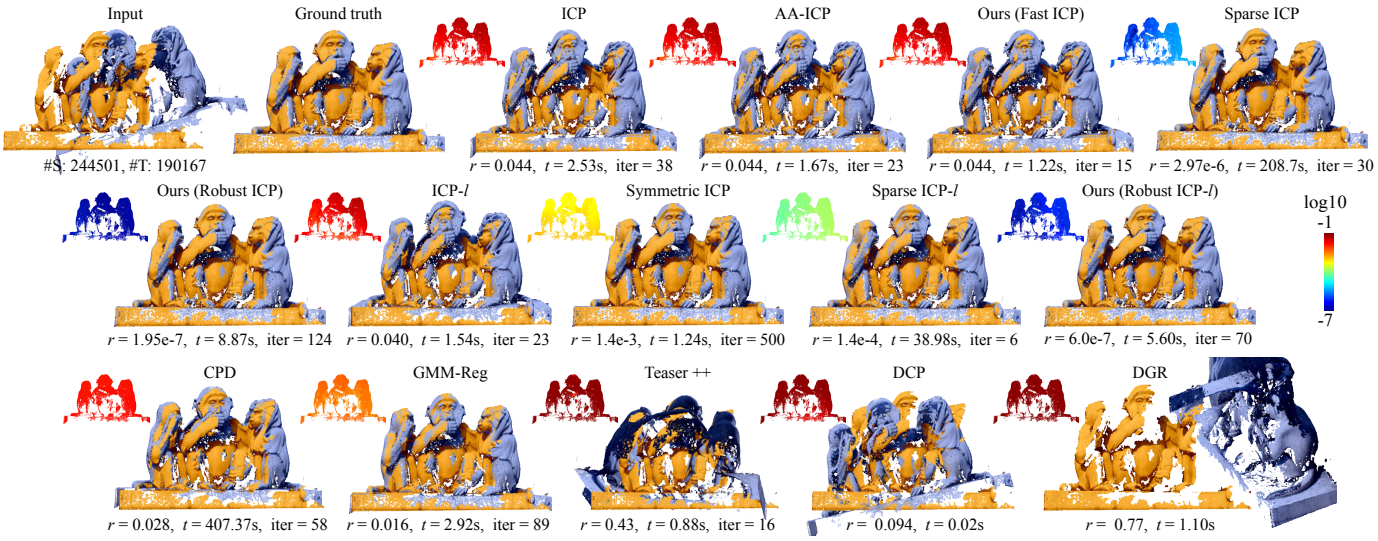

Fig. 1. Comparison between different registration methods (see Section 6) on a pair of partially overlapping point clouds constructed using the monkey model from the EPFL statue dataset [13]. #S and #T denote the number of points in the source and the target point clouds, respectively. Below each result we show the RMSE according to Eq. (12), the computational time, and the number of iterations. The log-scale color-coding illustrates the deviation between the transformed source point clouds using the computed alignment and the ground-truth alignment. Our robust point-to-point and point-to-plane ICP methods result in the lowest RMSE values, while being an order of magnitude faster than Sparse ICP.

- We propose a robust metric for point-to-point alignment based on the Welsch's function, which is less sensitive to outliers and partial overlaps and can be solved efficiently using the MM framework with Anderson acceleration. Our method achieves similar or better registration accuracy than sparse ICP, while being significantly faster.
- We extend the formulation to point-to-plane ICP, using the Welsch's function to define a robust error metric that is minimized with an Anderson-accelerated MM solver. Our formulation improves the robustness of point-to-plane ICP without the need for point pair rejection.

2 Related Work

Registration is a classical research topic in computer vision and robotics due to its numerous applications such as 3D scene reconstruction and localization. For a comprehensive review of rigid and nonrigd registration, the reader is referred to [14], [15]. Here, we focus on ICP for rigid registration. ICP and its variants [1], [3], [5], [16], [17] start from an initial alignment, and alternate between correspondence update using closest-points lookup and alignment update based on the correspondence. Using this framework, an accurate registration relies on a good initial alignment as well as a robust way to update the alignment.

For the initial alignment, Gelfand et al. [18] computed shape descriptors on the point clouds, and used the descriptors to match feature points and determine a coarse alignment. Rusu et al. [19] performed similar matching using the Point Feature Histograms defined at each point. Aiger et al. [20] aligned two point clouds by matching a pair of co-planar 4-point sets from them that are approximately congruent. Later, Mellado et al. [21] proposed a more efficient approach for such alignment with linear time complexity.

To update the alignment, ICP minimizes the ℓ_2 distance from the source points to their corresponding points [1] or to the tangent planes at the corresponding points [3]. Mitra et al. [22] proposed a framework that determines the alignment by minimizing a squared distance function between the two point clouds, as well as a local quadratic approximant for efficient update of the alignment. A similar approach was taken in [4] for aligning a point cloud to a surface. It was later shown in [2] that such a local quadratic approximant can lead to quadratic convergence. Recently, Rusinkiewicz [17] proposed a symmetrized objective function for ICP that yields faster convergence than point-to-point and point-to-plane ICP. Besides the convergence rate, another consideration for registration algorithms is their robustness to noises, outliers, and partial overlaps. A popular approach is to discard some point pairs from the alignment problem based on heuristics regarding their distance [5], [23], [24]. Other methods take a statistical approach and align two point sets via their Gaussian mixture representations [25], [26]. Another approach is to optimize a robust objective that reduces the influence from point pairs that are far apart [6], [27], [28], [29], [30]. In [6], the objective is defined using the ℓ_p-norm ($p < 1$) to induce sparsity of the point-wise distances. Our robust metric is defined using Welsch's function instead, which also induces sparsity while allowing for a more efficient solver that guarantees convergence.

Besides ICP, other methods formulate registration as a global optimization problem [31], [32], [33], which produces globally optimal results at the expense of higher computational costs. In [34], a truncated least squares optimization is proposed to make the registration insensitive to outliers. Recently, deep learning has also been applied to registration problems [35], [36].

Anderson acceleration was originally proposed in [37] for iterative solution of nonlinear integral equations, and has proved effective for accelerating fixed-point iterations [8], [38], [39], [40], [41], [42], [43], [44], [45]. In computer graphics, Anderson acceleration has been applied recently to accelerate local-global solvers [9] and ADMM solvers [46], [47].

Classical Anderson acceleration can become unstable or stagnate [8], [48]. Peng et al. [9] proposed an stabilization strategy on optimization solvers based on the decrease of the target function. Recently, Anderson acceleration has been used in [11] to speed up the convergence of ICP. We also apply Anderson acceleration to ICP, but using a different representation of the transformation together with the stabilization strategy from [9].

3 Classical ICP Revisited

Given two sets of points $P = \{\mathbf{p}_1, \dots, \mathbf{p}_M\}$ and $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_N\}$ in $\mathbb{R}^d$, we optimize a rigid transformation on P (represented using a rotation matrix $\mathbf{R} \in \mathbb{R}^{d\times d}$ and a translation vector $\mathbf{t} \in \mathbb{R}^d$) to align P with Q:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M} \big(D_i(\mathbf{R},\mathbf{t})\big)^2 + I_{SO(d)}(\mathbf{R}), \tag{1}$$

where $D_i(\mathbf{R},\mathbf{t}) = \min_{\mathbf{q}\in Q} \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}\|$ is the distance from the transformed point $\mathbf{R}\mathbf{p}_i + \mathbf{t}$ to the target set Q, and $I_{SO(d)}(\cdot)$ is an indicator function for the special orthogonal group $SO(d)$, which requires $\mathbf{R}$ to be a rotation matrix:

$$I_{SO(d)}(\mathbf{R}) = \begin{cases} 0, & \text{if } \mathbf{R}^T\mathbf{R} = \mathbf{I} \text{ and } \det(\mathbf{R}) = 1, \\ +\infty, & \text{otherwise.} \end{cases} \tag{2}$$

The ICP algorithm [1] solves this problem using an iterative approach that alternates between the following two steps:

- *Correspondence step*: find the closest point $\widehat{\mathbf{q}}_i^{(k)}$ in Q for each point $\mathbf{p}_i \in P$ based on transformation $(\mathbf{R}^{(k)}, \mathbf{t}^{(k)})$:

$$\widehat{\mathbf{q}}_i^{(k)} = \operatorname*{argmin}_{\mathbf{q}\in Q} \left\| \mathbf{R}^{(k)}\mathbf{p}_i + \mathbf{t}^{(k)} - \mathbf{q} \right\|. \tag{3}$$

- *Alignment step*: update the transformation by minimizing the ℓ_2 distance between the corresponding points:

$$\begin{aligned} &(\mathbf{R}^{(k+1)}, \mathbf{t}^{(k+1)}) \\ &= \operatorname*{argmin}_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M} \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)} \right\|^2 + I_{SO(d)}(\mathbf{R}). \end{aligned} \tag{4}$$

The alignment step can be solved in closed form via SVD [49]. This approach can be considered as a majorization-minimization (MM) algorithm [50] for the problem (1). To minimize a target function $f(x)$, each iteration of the MM algorithm constructs from the current iterate $x^{(k)}$ a surrogate function $g(x \mid x^{(k)})$ that bounds $f(x)$ from above, such that:

$$f(x^{(k)}) = g(x^{(k)} \mid x^{(k)}), \text{ and } f(x) \le g(x \mid x^{(k)}) \;\forall\; x \neq x^{(k)}. \tag{5}$$

The surrogate function is minimized to obtain the next iterate

$$x^{(k+1)} = \operatorname*{argmin}_{x} g(x \mid x^{(k)}). \tag{6}$$

Equations (5) and (6) imply that

$$f(x^{(k+1)}) \le g(x^{(k+1)} \mid x^{(k)}) \le g(x^{(k)} \mid x^{(k)}) = f(x^{(k)}).$$

Therefore, the MM algorithm decreases the target function monotonically until it converges to a local minimum. To see that ICP is indeed an MM algorithm, note that the target function for the alignment step is a surrogate function for the target function in problem (1) and satisfies the conditions (5). Specifically, since the closest point $\widehat{\mathbf{q}}_i^{(k)}$ is determined from $\mathbf{R}^{(k)}, \mathbf{t}^{(k)}$, we denote each distance value in (4) as

$$d_i(\mathbf{R},\mathbf{t} \mid \mathbf{R}^{(k)}, \mathbf{t}^{(k)}) = \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)} \right\|.$$

Then from Eq. (3) and the definition of D_i, we have

$$d_i(\mathbf{R}^{(k)}, \mathbf{t}^{(k)} \mid \mathbf{R}^{(k)}, \mathbf{t}^{(k)}) = D_i(\mathbf{R}^{(k)}, \mathbf{t}^{(k)}).$$

Moreover, from the definition of D_i, for any $\mathbf{R}, \mathbf{t}$:

$$\begin{aligned} D_i(\mathbf{R}\mathbf{p}_i + \mathbf{t}) &= \min_{\mathbf{q}\in Q} \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}\|^2 \\ &\le \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)} \right\| = d_i(\mathbf{R},\mathbf{t} \mid \mathbf{R}^{(k)}, \mathbf{t}^{(k)}). \end{aligned}$$

Thus each squared distance term in Eq. (4) is a surrogate function for the corresponding term $\big(D_i(\mathbf{R},\mathbf{t})\big)^2$ in Eq. (1), and the target function in Eq. (4) is a surrogate function for the overall target function in Eq. (1) constructed from $\mathbf{R}^{(k)}$ and $\mathbf{t}^{(k)}$. Therefore, ICP is an MM algorithm that decreases the target function of (1) monotonically until convergence.

4 Fast and Robust ICP

Despite its simplicity, classical ICP can be slow to converge to a local minimum due to its linear convergence rate [2]. In this section, we interpret ICP as a fixed-point iteration, and propose a method to improve its convergence rate using Anderson acceleration [8], [37], an established technique for accelerating fixed-point iterations. In addition, classical ICP can lead to erroneous alignment in the presence of outliers and partial overlaps, due to the use of ℓ_2 distance as the error metric in the alignment step. We adopt a robust error metric based on Welsch's function instead, and derive an MM solver for the resulting optimization problem, with Anderson acceleration to speed up its convergence. In the following, we first review the basics of Anderson acceleration.

4.1 Preliminary: Anderson Acceleration

Given a fixed-point iteration $x^{(k+1)} = G(x^{(k)})$, we define its *residual* function as $F(x) = G(x) - x$, and denote $F^{(k)} = G(x^{(k)})$. By definition, a fixed-point x^* of the mapping $G(\cdot)$ satisfies $F(x^*) = 0$. Anderson acceleration utilizes the latest iterate $x^{(k)}$ as well as its preceding m iterates $x^{(k-m)}, x^{(k-m+1)}, \dots, x^{(k-1)}$ to derive a new iterate $x_{\text{AA}}^{(k+1)}$ that convergences faster to a fixed point [8]:

$$x_{\text{AA}}^{(k+1)} = G(x^{(k)}) - \sum_{j=1}^{m} \theta_j^* \big(G(x^{(k-j+1)}) - G(x^{(k-j)})\big), \tag{7}$$

where $(\theta_1^*, \dots, \theta_m^*)$ is the solution to the following linear least-squares problem:

$$(\theta_1^*, \dots, \theta_m^*) = \operatorname{argmin} \left\| F^{(k)} - \sum_{j=1}^{m} \theta_j \big(F^{(k-j+1)} - F^{(k-j)}\big) \right\|^2,$$

It has been shown that Anderson acceleration is a quasi-Newton method for finding a root of the residual function [39], and it can improve the convergence rate for fixed-point iterations that converge linearly [51].

4.2 Applying Anderson Acceleration to ICP

The classical ICP explained in Section 3 can be written as a fixed-point iteration of the transformation variables $\mathbf{R}$ and $\mathbf{t}$:

$$(\mathbf{R}^{(k+1)}, \mathbf{t}^{(k+1)}) = G_{\text{ICP}}(\mathbf{R}^{(k)}, \mathbf{t}^{(k)}), \tag{8}$$

where

$$G_{\text{ICP}}(\mathbf{R}^{(k)}, \mathbf{t}^{(k)}) = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_{i=1}^{M} \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \Pi_Q(\mathbf{R}^{(k)}\mathbf{p}_i + \mathbf{t}^{(k)}) \right\|^2 + I_{SO(d)}(\mathbf{R}),$$

and $\Pi_Q(\cdot)$ denotes the closest projection onto the point set Q. However, we cannot directly apply Anderson acceleration to the mapping G_{ICP}. This is because Anderson acceleration will compute the new value of $\mathbf{R}$ as an affine combination of rotation matrices, which is in general not a rotation matrix itself. To address this issue, we can parameterize a rigid transformation using another set of variables $\mathbf{X}$, such that any value of $\mathbf{X}$ corresponds to a valid rigid transformation, and the ICP iteration can be re-written in the form of

$$\mathbf{X}^{(k+1)} = \overline{G}_{\text{ICP}}(\mathbf{X}^{(k)}). \tag{9}$$

Then we can apply Anderson acceleration to the variable $\mathbf{X}$ by performing the following steps in each iteration:

1) From the current variable $\mathbf{X}^{(k)}$, recover the rotation matrix $\mathbf{R}^{(k)}$ and translation vector $\mathbf{t}^{(k)}$.
2) Perform the ICP update $(\mathbf{R}', \mathbf{t}') = G_{\text{ICP}}(\mathbf{R}^{(k)}, \mathbf{t}^{(k)})$.
3) Compute the parameterization of $(\mathbf{R}', \mathbf{t}')$ to obtain $\overline{G}_{\text{ICP}}(\mathbf{X}^{(k)})$.
4) Compute the accelerated value $\mathbf{X}_{\text{AA}}$ with Eq. (7) using $\mathbf{X}^{(k-m)}, \ldots, \mathbf{X}^{(k)}$ and $G_{\text{ICP}}(\mathbf{X}^{(k-m)}), \ldots, G_{\text{ICP}}(\mathbf{X}^{(k)})$.

One possible parameterization of rigid transformations is to concatenate the translation vector and the Euler angles of the rotation [52], [53]. This is the approach taken by the AA-ICP method [11] for applying Anderson acceleration to ICP in $\mathbb{R}^3$. However, it is well known that the Euler angle representation has singularities called the *gimbal lock* [52]. This can affect the performance of AA-ICP when the optimal rotation is close to a gimbal lock (see Fig. 2 for an example). An alternative representation of rotation in $\mathbb{R}^3$ without such singularities is the unit quaternions, which are identified with unit vectors in $\mathbf{R}^4$ [52]. This representation is not suitable either, as an affine combination of unit vectors does not result in a unit vector in general. Rather than the using the above representations, we note that all rigid transformations in $\mathbb{R}^d$ form the *special Euclidean group* $SE(d)$, which is a *Lie group* and gives rise to a *Lie algebra* $se(d)$ that is a vector space [54]. From a differential geometry perspective, $SE(d)$ is a smooth manifold and $se(d)$ is its tangent space at the identity transformation. We can then parameterize rigid transformations using their corresponding elements in $se(d)$.

Specifically, if we represent each point $\mathbf{p} \in \mathbb{R}^d$ using its homogeneous coordinates $\tilde{\mathbf{p}} = [\mathbf{p}^T, 1]^T$, then a rigid transformation in $\mathbb{R}^d$ with rotation $\mathbf{R} \in \mathbb{R}^{d\times d}$ and translation $\mathbf{t} \in \mathbb{R}^d$ can be represented as a transformation matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{(d+1)\times(d+1)}$$

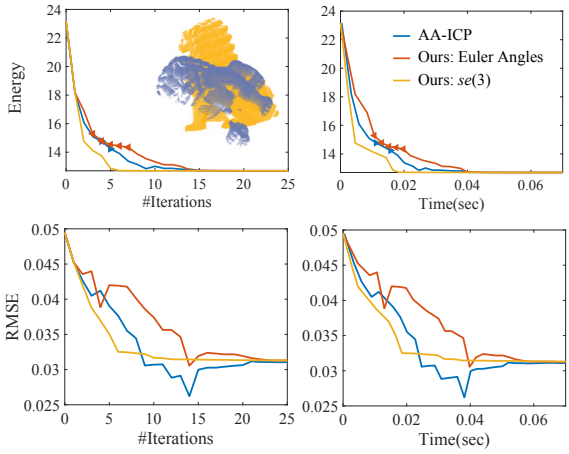

Fig. 2. Target energy and RMSE plots for Anderson-accelerated ICP methods on a pair of point clouds, using different transformation representations and stabilization strategies. Our formulation outperforms AA-ICP [11] as well as a Euler angle-based method using our stabilization strategy. For Euler angle-based methods, we use solid triangle symbols to highlight the iterates that are close to the gimbal lock.

for the homogeneous coordinates. All such matrices form the special Euclidean group $SE(d)$. Its Lie algebra $se(d)$ contains matrices of the following form

$$\check{\mathbf{T}} = \begin{bmatrix} \mathbf{S} & \mathbf{u} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathbb{R}^{(d+1)\times(d+1)}, \tag{10}$$

Each matrix $\check{\mathbf{T}} \in se(d)$ corresponds to a matrix $\mathbf{T} \in SE(d)$ via the matrix exponential:

$$\mathbf{T} = \exp(\check{\mathbf{T}}) = \sum_{i=0}^{\infty} \frac{1}{i!} \check{\mathbf{T}}^i. \tag{11}$$

The matrix exponential can be computed numerically using a generalization of Rodrigues' method [55]. On the other hand, given a matrix $\mathbf{T} \in SE(d)$, there may be more than one matrix $\check{\mathbf{T}} \in se(d)$ that satisfies Eq. (11). In Appendix A, we present a method to determine a unique value of $\check{\mathbf{T}}$. We call it the *logarithm* of $\mathbf{T}$, and denote it by $\check{\mathbf{T}} = \log(\mathbf{T})$. We then perform Anderson acceleration on the logarithms of the transformations. Since $se(d)$ is a vector space, the accelerated value $\check{\mathbf{T}}_{\text{AA}}$—which is computed as an affine combination of elements in $se(d)$—also belongs to $se(d)$ and represents a rigid transformation $\mathbf{T}_{\text{AA}} = \exp(\check{\mathbf{T}}_{\text{AA}}) \in SE(d)$.

Simply applying Anderson acceleration as explained in Section 4.1 is often not sufficient for fast convergence. It is known that Anderson acceleration can suffer from instability and stagnation even for linear problems [48], thus safeguarding steps are often necessary to improve its performance [9], [46], [56]. To this end, we follow the stabilization strategy proposed in [9]: we accept the accelerated value as the new iterate only if it decreases the target function (1) compared with the previous iterate; otherwise, we revert to the unaccelerated ICP iterate as the new iterate. This approach is more simple than the combination of heuristics in [11], while ensuring monotonic decrease of the target energy. Following [9], we set the number of previous iterates for Anderson acceleration to $m = 5$ in all experiments.

Compared to AA-ICP [11] that also applies Anderson acceleration to ICP, our approach differs in two aspects. First,

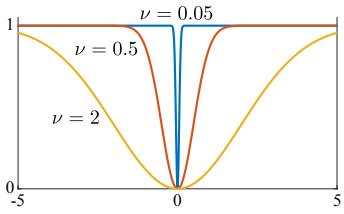

Fig. 3. The graphs of function $\psi_\nu(x)$ with different parameters. As ν decreases, the function ψ_ν approaches the ℓ_0 norm.

we apply Anderson acceleration via the Lie algebra $se(d)$ instead of the Euler angles, which is free from the singularities of gimbal locks. Second, our stabilization strategy is more simple to implement than the multiple heuristics in [11] while ensuring monotonic decrease of the target function. Fig. 2 compares our method with AA-ICP, as well as an alternative Anderson acceleration approach using Euler angle representation and our stabilization strategy. The comparison is done on a synthetic model from [30] for which the ground-truth alignment is known, and the point sets are pre-aligned using Super4PCS [21]. We plot the value of target function (1) with respect to the iteration count and computational time, as well as the following root mean square error (RMSE) between the computed alignment $(\mathbf{R}, \mathbf{t})$ from the ground-truth alignment $(\mathbf{R}^*, \mathbf{t}^*)$:

$$r = \sqrt{\frac{1}{M}\sum\nolimits_{i=1}^{M} \|\mathbf{R}^*\mathbf{p}_i + \mathbf{t}^* - \mathbf{R}\mathbf{p}_i - \mathbf{t}\|_2^2}. \tag{12}$$

Fig. 2 shows that our method using the Lie algebra leads to faster convergence. In the energy-iteration plots, for each Euler angle-based approach we use solid triangles to highlight the iterations that are close to the gimbal lock (with the pitch angle less than 0.01π away from $\pm\pi/2$).

4.3 Robust ICP via Welsch's Function

Classical ICP measures the alignment error using ℓ_2 distance, which penalizes large deviation from *any* point in the source set P to the target set Q. This enables a closed-form solution in the alignment step, but may lead to erroneous alignment in the presence of outliers and partial overlaps: in such cases some points in P may not correspond to any point in Q, and the ground-truth alignment can induce a large error that would be prohibited by the ℓ_2 minimization. The issue can be resolved by adopting error metrics that promote sparsity of the point-wise distance from P to Q. Such metrics penalize the distance between points in true correspondence, while allowing for large deviation induced by outliers and partial overlaps. One example is the ℓ_p-norm of point-wise distance with $p \in (0, 1)$, resulting in the error metric $\sum_{i=1}^{M}\Big(D_i(\mathbf{R}, \mathbf{t})\Big)^p$ that is used in the sparse ICP algorithm [6]. Like classical ICP, the sparse ICP algorithm alternates between closest point query and alignment update. The alignment problem is similar to Eq. (4) but is based on ℓ_p distance instead. The problem is solved using the alternating direction method of multipliers (ADMM) since there is no closed-form solution. Although sparse ICP produces more accurate results, the use of ADMM incurs a much higher computational cost. Moreover, the ADMM solver requires $d \cdot M$ auxiliary variables and $d \cdot M$ dual variables, which can significantly increase the memory footprint.

In this paper, we adopt a different robust error metric that does not incur high computational overhead. Specifically, we formulate the registration problem as

$$\min_{\mathbf{R},\mathbf{t}} \ \sum\nolimits_{i=1}^{M} \psi_\nu\big(D_i(\mathbf{R}, \mathbf{t})\big) + I_{SO(d)}(\mathbf{R}), \tag{13}$$

where ψ_ν is the Welsch's function [12]:

$$\psi_\nu(x) = 1 - \exp\Big(-\frac{x^2}{2\nu^2}\Big), \tag{14}$$

and $\nu > 0$ is a user-specified parameter. Fig. 3 shows the graphs of ψ_ν with different values of ν. Since $\psi_\nu(x)$ is monotonically increasing on $[0, +\infty)$, our formulation penalizes deviation between the point sets. At the same time, ψ_ν is upper bounded by 1, so our metric is not sensitive to large deviations caused by outliers and partial overlaps. Moreover, when ν approaches zero, $\sum_{i=1}^{M} \psi_\nu\big(D_i(\mathbf{R}, \mathbf{t})\big)$ approaches the ℓ_0-norm of the vector $[D_1(\mathbf{R}, \mathbf{t}), \ldots, D_M(\mathbf{R}, \mathbf{t})]$. Thus our formulation promotes sparsity of the point-wise distance between the point sets. Recently, error metrics based on Welsch's function have been applied for robust filtering in image processing [57] and geometry processing [58].

Although our formulation (13) is non-linear and non-convex, the problem can be solved using the same MM framework as classical ICP, by alternating between a correspondence step and an alignment step. The correspondence step is the same as classical ICP. In the alignment step, we utilize the closest points to construct the following surrogate for the target function (13) at the current transformation $(\mathbf{R}^{(k)}, \mathbf{t}^{(k)})$ (see Appendix B for a proof):

$$\sum\nolimits_{i=1}^{M} \chi_\nu\Big(\|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)}\| \ \Big|\ D_i(\mathbf{R}^{(k)}, \mathbf{t}^{(k)})\Big) + I_{SO(d)}(\mathbf{R}), \tag{15}$$

where $\chi_\nu(x \mid y)$ is a quadratic surrogate function for the Welsch's function at y with the following form [57]:

$$\chi_\nu(x \mid y) = \psi_\nu(y) + \frac{x^2 - y^2}{2\nu^2}\exp\Big(-\frac{y^2}{2\nu^2}\Big). \tag{16}$$

We minimize the surrogate function (15) to update the transformation, resulting in the following problem:

$$\begin{aligned}&(\mathbf{R}^{(k+1)}, \mathbf{t}^{(k+1)})\\ =\ &\operatorname*{argmin}_{\mathbf{R},\mathbf{t}} \sum\nolimits_{i=1}^{M} \omega_i \Big\|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)}\Big\|^2 + I_{SO(d)}(\mathbf{R}),\end{aligned} \tag{17}$$

where $\omega_i = \exp\Big(-\|\mathbf{R}^{(k)}\mathbf{p}_i + \mathbf{t}^{(k)} - \widehat{\mathbf{q}}_i^{(k)}\|^2/(2\nu^2)\Big)$. The alignment step (17) minimizes a weighted sum of squared distance between the points $\{\mathbf{p}_i\}$ and $\{\widehat{\mathbf{q}}_i^{(k)}\}$. It can be solved in closed form via SVD [49]. Similar to classical ICP, our MM solver decreases the target energy in each iteration and converges to a local minimum. Using the same approach as in Section 4.2, we improve its convergence rate by applying Anderson acceleration to the parameterization of rigid transformations in $se(d)$, using the same stabilization strategy that checks the target function value for the accelerated value.

Our approach has a similar structure as the iteratively reweighted least squares (IRLS) method that minimizes the ℓ_p-norm ($p < 1$) for compressive sensing [59]. Similar to IRLS, we solve a weighted least squares problem, with the weights ω_i for a point $\mathbf{p}_i$ updated in each iteration according to its current distance to the corresponding point. Since the

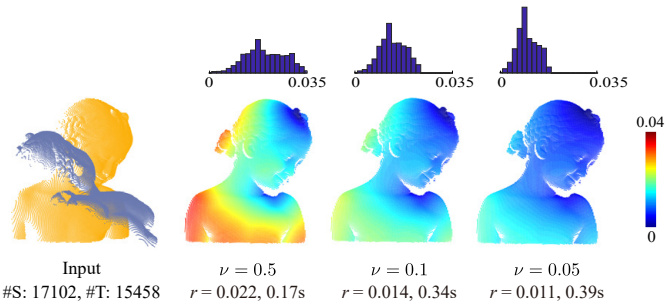

Fig. 4. Registration results via optimization (13) with different values of parameter ν, on a pair of point clouds with partial overlap. The color-coding shows the deviation between the transformed positions of each source point using the computed alignment and ground-truth alignment, with the histograms showing the distribution of the deviation among the source points. For this model, a smaller value of ν leads to a more accurate result.

weight is a Gaussian function with variance ν, a point $\mathbf{p}_i$ with larger distance from the target point set receives a lower weight. Moreover, according to the well-know three-sigma rule, when the distance is larger than 3ν, the weight ω_i is small enough such that the term for $\mathbf{p}_i$ has little influence to the target function and $\mathbf{p}_i$ is effectively excluded from the current alignment problem. In this way, the optimization allows some source points to be far away from the target point set, and is robust to outliers and partial overlaps.

Some ICP variants improve robustness by excluding from the alignment step the point pairs with large deviation between their positions or normals [5]. It was observed in [6] that such methods can be difficult to tune or increase the number of local minima. Our method also excludes point pairs with large positional difference, but using a Gaussian weight that gradually decreases as the point pair becomes further apart. It can be considered as a soft thresholding approach that weakly penalizes outliers, which can lead to more stable results [6]. Indeed, we observe in experiments that our robust methods and sparse ICP tend to produce more accurate results than the symmetric ICP method [17] which is based on outlier rejection; see Section 6 for details.

Compared to ℓ_p-norm minimization ($0 < p < 1$), our formulation and solver also offer benefits in stability and convergence guarantee. For our weighted least-squares problem (17), all the Gaussian weights $\{\omega_i\}$ have values within the range $(0, 1]$. In contrast, an IRLS solver for ℓ_p-norm minimization would assign a weight $\|\mathbf{R}^{(k)}\mathbf{p}_i + \mathbf{t}^{(k)} - \widehat{\mathbf{q}}_i^{(k)}\|^{p-2}$ to the point $\mathbf{p}_i$, which could go to infinity and cause instability when the alignment error for $\mathbf{p}_i$ approaches zero [6]. According to [6], sparse ICP performs ℓ_p-norm minimization using ADMM instead of IRLS because of concern about such instability. In addition, the convergence of IRLS and ADMM for non-convex ℓ_p-norm minimization requires strong assumptions about the problem such as the Kurdyka-Łojasiewicz property [60], [61], whereas our MM solver is guaranteed to converge.

For our algorithm, the parameter ν plays an important role in achieving good performance. A smaller ν helps to attenuate the influence from outliers and partial overlaps (e.g., see Fig. 4). On the other hand, a larger ν in the initial stage helps to include more point pairs in the alignment step and avoid undesirable local minima. Therefore, we gradually decrease ν during the iterations, so that the algorithm first performs more global alignment with a larger number of pairs, and then reduces the influence from the pairs with large deviation to achieve robust alignment. Specifically, we choose two values $\nu_{\max}$ and $\nu_{\min}$ as the upper and lower bounds of ν. We start by setting $\nu = \nu_{\max}$ and running our MM algorithm until the change in the transformation matrix $\mathbf{T}$ is smaller than a threshold (10^{-5} by default) or the iteration count exceeds an upper limit (1000 by default). Then we decrease the value of ν by half, and run the MM algorithm again until the same termination criterion is met. The process is repeated until the lower bound $\nu_{\min}$ is reached. Algorithm 1 summarizes our method with a decreasing ν.

To choose $\nu_{\max}$, we compute the median $\overline{D}^{(0)}$ among all initial point-wise distance $\{D_i(\mathbf{R}^{(0)}, \mathbf{t}^{(0)})\}$, and set $\nu_{\max} = 3 \cdot \overline{D}^{(0)}$. In our experiments, this setting makes $\nu_{\max}$ large enough to include most point pairs into the alignment

Algorithm 1: Robust point-to-point ICP using Welsch's function and Anderson acceleration.

Input: $\mathbf{T}^{(0)}$: initial transformation for P;
m: the number of previous iterates used for Anderson acceleration;
correspondence($\mathbf{T}$): computation of all closest points according via Eq. (3) using transformation $\mathbf{T}$;
alignment($\widehat{\mathbf{Q}}, \mathbf{T}, \nu$): new transformation via Eq. (17) using current transformation $\mathbf{T}$ and closest points $\widehat{\mathbf{Q}}$;
$E_\nu(\widehat{\mathbf{Q}}, \mathbf{T})$: target energy for transformation $\mathbf{T}$ and closest points $\widehat{\mathbf{Q}}$: $E_\nu(\widehat{\mathbf{Q}}, \mathbf{T}) = \sum_{i=1}^{M} \psi_\nu(\|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i\|)$;
I_ν, ϵ_ν: maximum number of iterations and the convergence threshold of $\mathbf{T}$ for a given parameter ν.

$k = 1$; $\nu = \nu_{max}$; $\widehat{\mathbf{Q}}^{(0)} = \mathit{correspondence}(\mathbf{T}^{(0)})$;
while `TRUE` **do**
  $k_{\text{start}} = k - 1$; $E_{\text{prev}} = +\infty$;
  $\mathbf{T}' = \mathit{alignment}(\widehat{\mathbf{Q}}^{(k-1)}, \mathbf{T}^{(k-1)}, \nu)$;
  $\mathbf{T}^{(k)} = \mathbf{T}'$; $\widehat{\mathbf{Q}}^{(k)} = \mathit{correspondence}(\mathbf{T}^{(k)})$;
  $G^{(k-1)} = \log(\mathbf{T}')$; $F^{(k-1)} = G^{(k-1)} - \log(\mathbf{T}^{(k-1)})$;
  while $k - k_{\text{start}} \le I_\nu$ **do**
    `// Ensure` $\mathbf{T}^{(k)}$ `decreases the energy`
    if $E_\nu(\widehat{\mathbf{Q}}^{(k)}, \mathbf{T}^{(k)}) \ge E_{\text{prev}}$ **then**
      $\mathbf{T}^{(k)} = \mathbf{T}'$; $\widehat{\mathbf{Q}}^{(k)} = \mathit{correspondence}(\mathbf{T}^{(k)})$;
    end
    $E_{\text{prev}} = E_\nu(\widehat{\mathbf{Q}}^{(k)}, \mathbf{T}^{(k)})$;
    `// Check convergence`
    $\mathbf{T}' = \mathit{alignment}(\widehat{\mathbf{Q}}^{(k)}, \mathbf{T}^{(k)}, \nu)$;
    if $\|\mathbf{T} - \mathbf{T}'\|_F < \epsilon_\nu$ **then** break ;
    `// Anderson acceleration`
    $G^{(k)} = \log(\mathbf{T}')$; $F^{(k)} = G^{(k)} - \log(\mathbf{T}^{(k)})$;
    $m_k = \min(k - k_{\text{start}}, m)$;
    $(\theta_1^*, \ldots, \theta_{m_k}^*) =$
     $\operatorname{argmin} \|F^{(k)} - \sum_{j=1}^{m_k} \theta_j (F^{(k-j+1)} - F^{(k-j)})\|_F^2$;
    $\mathbf{T}^{(k+1)} =$
     $\exp\big(G^{(k)} - \sum_{j=1}^{m} \theta_j^* (G^{(k-j+1)} - G^{(k-j)})\big)$;
    $\widehat{\mathbf{Q}}^{(k+1)} = \mathit{correspondence}(\mathbf{T}^{(k+1)})$;
    $k = k + 1$;
  end
  if $\nu == \nu_{\min}$ **then return** $\mathbf{T}^{(k)}$;
  $\nu = \max(\nu/2, \nu_{\min})$; $k = k + 1$;
end

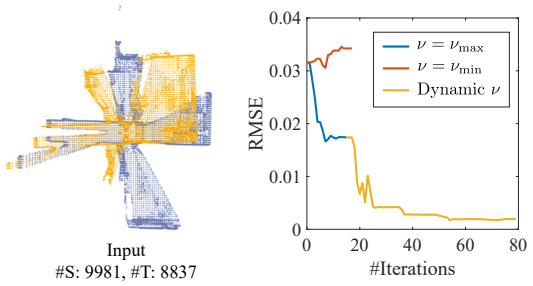

Fig. 5. RMSE plots for optimization (13) with different settings of ν, on a pair of point clouds in the 'Apartment' sequence from the ETH laser registration dataset [62]. Gradually reducing ν from $\nu_{\max}$ to $\nu_{\min}$ results in the lowest RMSE.

process except for outliers with significant deviation. For $\nu_{\min}$, we note that the two point sets may sample the same surface at different locations, and $\nu_{\min}$ should be large enough to accommodate the deviation due to sampling. Therefore, we first compute the median distance from each point $\mathbf{q}_i \in Q$ to its six nearest points on Q, and take the median $\overline{E}_Q$ of all such median values. Then we set $\nu_{\min} = \overline{E}_Q/3\sqrt{3}$ (see Appendix C for the rationale).

Fig. 5 illustrates the effectiveness of our ν-update strategy, by comparing its RMSE plot with those resulting from a fixed parameter $\nu = \nu_{\max}$ and $\nu = \nu_{\min}$, respectively. Here a fixed $\nu = \nu_{\min}$ results in a large RMSE, because such a small ν will lead to a small weight for most point pairs, effectively excluding them from the alignment step and producing an erroneous result. Fixing $\nu = \nu_{\max}$ can reduce the final RMSE as it includes more points into the alignment; however, it fails to exclude some outliers so the RMSE is still large. A decreasing ν gradually removes outliers from the alignment process, resulting in a much smaller RMSE.

5 Extension to Point-to-Plane ICP

The classical ICP algorithm discussed in Section 4 is often called the "point-to-point" ICP, since its alignment step minimizes the distance from the source points to their corresponding target points. Another popular ICP variant in $\mathbb{R}^3$, often called the "point-to-plane" ICP, minimizes the distance from the source points to the tangent planes at the target points instead in the alignment step [3]:

$$\begin{aligned}&(\mathbf{R}^{(k+1)}, \mathbf{t}^{(k+1)})\\ &= \operatorname*{argmin}_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M} \left((\mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)}) \cdot \widehat{\mathbf{n}}_i^{(k)}\right)^2 + I_{SO(3)}(\mathbf{R}),\end{aligned} \tag{18}$$

where $\widehat{\mathbf{n}}_i^{(k)}$ is the normal at $\widehat{\mathbf{q}}_i^{(k)}$ for the underlying surface of the target point set. Point-to-plane ICP can be considered as solving an optimization problem

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M} \left(H_i(\mathbf{R},\mathbf{t})\right)^2 + I_{SO(3)}(\mathbf{R}), \tag{19}$$

where $H_i(\mathbf{R},\mathbf{t})$ is the signed distance from the point $\mathbf{R}\mathbf{p}_i + \mathbf{t}$ to the tangent plane at its closest point in Q. Since the tangent plane provides a local linear approximation of the underlying surface, point-to-plane ICP can achieve faster convergence [2]. On the other hand, it suffers from the same issue of robustness to outliers and partial overlaps. Similar to Section 4, we can improve its robustness by adopting a robust metric based on Welsch's function ψ_ν:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M} \psi_\nu\left(H_i(\mathbf{R},\mathbf{t})\right) + I_{SO(3)}(\mathbf{R}). \tag{20}$$

This is solved by alternating between a correspondence step the same as point-to-point ICP, and an assignment step that solves the following problem:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M} \psi_\nu\left((\mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)}) \cdot \widehat{\mathbf{n}}_i^{(k)}\right) + I_{SO(3)}(\mathbf{R}). \tag{21}$$

Similar to Section 4, we replace the target function above with a surrogate function to derive a proxy problem:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M} \gamma_i \left((\mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)}) \cdot \widehat{\mathbf{n}}_i^{(k)}\right)^2 + I_{SO(3)}(\mathbf{R}), \tag{22}$$

where $\gamma_i = \exp\left(-((\mathbf{R}^{(k)}\mathbf{p}_i + \mathbf{t}^{(k)} - \widehat{\mathbf{q}}_i^{(k)}) \cdot \widehat{\mathbf{n}}_i^{(k)})^2/(2\nu^2)\right)$. There is no closed-form solution to this problem. So we rewrite it as an optimization for the $se(3)$ parameterization:

$$\min_{\widetilde{\mathbf{T}}} \sum_{i=1}^{M} \gamma_i \left(B_i^{(k)}(\widetilde{\mathbf{T}})\right)^2. \tag{23}$$

Here $\widetilde{\mathbf{T}} \in \mathbb{R}^6$ denotes the actual variables for the $se(3)$ element $\check{\mathbf{T}}$ in Eq. (10) (three variables for each of the submatrices $\mathbf{S}$ and $\mathbf{u}$, respectively), and $B_i^{(k)}(\widetilde{\mathbf{T}})$ is the signed distance from $\mathbf{R}\mathbf{p}_i + \mathbf{t}$ to the tangent plane at $\widehat{\mathbf{q}}_i^{(k)}$. We then linearize $B_i^{(k)}$ using its first-order Taylor expansion

$$B_i^{(k)}(\widetilde{\mathbf{T}}) \approx B_i^{(k)}(\widetilde{\mathbf{T}}^{(k)}) + (\mathbf{J}_i^{(k)})^T(\widetilde{\mathbf{T}} - \widetilde{\mathbf{T}}^{(k)}), \tag{24}$$

where $\widetilde{\mathbf{T}}^{(k)}$ is the $se(3)$ variable for $(\mathbf{R}^{(k)}, \mathbf{t}^{(k)})$, and $\mathbf{J}_i^{(k)}$ is the gradient of $B_i^{(k)}$ at $\widetilde{\mathbf{T}}^{(k)}$ (see Appendix D for its calculation). Substituting the linearization into Eq. (23), we obtain a quadratic problem that reduces to a linear system

$$\begin{aligned}&\left(\sum_{i=1}^{M} \gamma_i \mathbf{J}_i^{(k)} (\mathbf{J}_i^{(k)})^T\right) \widetilde{\mathbf{T}}\\ &= \sum_{i=1}^{M} \gamma_i \mathbf{J}_i^{(k)} \left(B_i^{(k)}(\widetilde{\mathbf{T}}^{(k)}) - (\mathbf{J}_i^{(k)})^T \widetilde{\mathbf{T}}^{(k)}\right).\end{aligned} \tag{25}$$

The solution $\widetilde{\mathbf{T}}_*^{(k)}$ to this system will be taken as a candidate for the updated transformation. Due to the linearization, $\widetilde{\mathbf{T}}_*^{(k)}$ may increase the target function (20). Therefore, we perform line search along the direction $\widetilde{\mathbf{T}}_*^{(k)} - \widetilde{\mathbf{T}}^{(k)}$ to find a new transformation that decreases the target function. If such a transformation cannot be found after the maximum number of line-search steps is reached, then the step size with the lowest target function value will be used.

Similar to Section 4, we apply Anderson acceleration to speed up the convergence. We note that the mapping from the current variable $\widetilde{\mathbf{T}}^{(k)}$ to the candidate update $\widetilde{\mathbf{T}}_*^{(k)}$, which amounts to finding the closest points $\{\widehat{\mathbf{q}}_i^{(k)}\}$ according to $\widetilde{\mathbf{T}}^{(k)}$ and solving the linear system (25), can be written as

$$\widetilde{\mathbf{T}}_*^{(k)} = G_{\text{ppl}}(\widetilde{\mathbf{T}}^{(k)}). \tag{26}$$

Then for a local minimum $(\mathbf{R}^*, \mathbf{t}^*)$ of the target function (20), the corresponding $se(3)$ variable $\widetilde{\mathbf{T}}^*$ should be a fixed point of G_{ppl}. Therefore, we apply Anderson acceleration to $\widetilde{\mathbf{T}}^{(k-m)}, \ldots, \widetilde{\mathbf{T}}^{(k)}$ and $G_{\text{ppl}}(\widetilde{\mathbf{T}}^{(k-m)}), \ldots, G_{\text{ppl}}(\widetilde{\mathbf{T}}^{(k)})$ to obtain an accelerated value $\widetilde{\mathbf{T}}_{\text{AA}}$. If $\widetilde{\mathbf{T}}_{\text{AA}}$ decreases the target function (20), then we accept it as the new iterate $\widetilde{\mathbf{T}}^{(k+1)}$.

Algorithm 2: Robust point-to-plane ICP using Welsch's function and Anderson acceleration.

Input: $\widetilde{\mathbf{T}}^{(0)}$: initial transformation parameters;
m: the number of previous iterates used for Anderson acceleration;
$G_{\text{ppl}}(\cdot)$: the mapping defined in Eq. (26);
$\widetilde{E}_\nu(\widetilde{\mathbf{T}})$: target energy of problem (20) for parameters $\widetilde{\mathbf{T}}$;
$l_{\max}$: maximum number of inner line search steps;
I_ν, ϵ_ν: maximum number of iterations and the convergence threshold for a given parameter ν.

$k = 1$; $\nu = \nu_{max}$;
while *TRUE* **do**
  $k_{\text{start}} = k - 1$; $E_{\text{prev}} = +\infty$; $\widetilde{\mathbf{T}}_*^{(k)} = G_{\text{ppl}}(\widetilde{\mathbf{T}}^{(k-1)})$;
  $G^{(k-1)} = \widetilde{\mathbf{T}}^{(k)} = \widetilde{\mathbf{T}}_*^{(k)}$; $F^{(k-1)} = G^{(k-1)} - \widetilde{\mathbf{T}}^{(k-1)}$;
  while $k - k_{\text{start}} \leq I_\nu$ **do**
    // Check energy decrease
    $E = \widetilde{E}_\nu(\widetilde{\mathbf{T}}^{(k)})$;
    if $E \geq E_{\text{prev}}$ **then**
      // Perform line search
      $\tau = 1$; $l = 1$;
      while $l \leq l_{\max}$ **do**
        $\widetilde{\mathbf{T}}_{\text{trial}} = \widetilde{\mathbf{T}}^{(k-1)} + \tau(\widetilde{\mathbf{T}}_*^{(k)} - \widetilde{\mathbf{T}}^{(k-1)})$;
        $E_{\text{trial}} = \widetilde{E}_\nu(\widetilde{\mathbf{T}}_{\text{trial}})$;
        if $E_{\text{trial}} < E$ **then**
          $E = E_{\text{trial}}$; $\widetilde{\mathbf{T}}^{(k)} = \widetilde{\mathbf{T}}_{\text{trial}}$;
        end
        if $E_{\text{trial}} < E_{\text{prev}}$ **then** break ;
      end
    end
    $E_{\text{prev}} = E$;
    // Check convergence
    $\widetilde{\mathbf{T}}_*^{(k+1)} = G_{\text{ppl}}(\widetilde{\mathbf{T}}^{(k)})$;
    if $\|\widetilde{\mathbf{T}}_*^{(k+1)} - \widetilde{\mathbf{T}}^{(k)}\| < \epsilon_\nu$ **then** break; ;
    // Anderson acceleration
    $G^{(k)} = \widetilde{\mathbf{T}}_*^{(k+1)}$; $F^{(k)} = G^{(k)} - \widetilde{\mathbf{T}}^{(k)}$;
    $m_k = \min(k - k_{\text{start}}, m)$;
    $(\theta_1^*, \ldots, \theta_{m_k}^*) =$
     $\operatorname{argmin} \|F^{(k)} - \sum_{j=1}^{m_k} \theta_j (F^{(k-j+1)} - F^{(k-j)})\|^2$;
    $\widetilde{\mathbf{T}}^{(k+1)} =$
     $\exp\big(G^{(k)} - \sum_{j=1}^{m} \theta_j^* (G^{(k-j+1)} - G^{(k-j)})\big)$;
    $k = k + 1$;
  end
  if $\nu == \nu_{\min}$ **then return** $\widetilde{\mathbf{T}}^{(k)}$;
  $\nu = \max(\nu/2, \nu_{\min})$; $k = k + 1$;
end

Otherwise, we perform line search as described previously. Algorithm 2 summarizes our robust point-to-plane ICP solver. Similarly to Algorithm 1, we start with $\nu = \nu_{\max}$ and gradually decreases it until the lower bound $\nu_{\min}$ is reached. For each given ν value, the solver is run until the change in the transformation matrix is smaller than a threshold (10^{-5} by default) or the iteration count reaches a limit (6 for $\nu_{\max}$, then incremented by 1 each time ν is changed, but no larger than 10). We set $\nu_{\max}$ to be three times the median distance from the source points to the tangent planes at their corresponding points in the initial iteration. To determine $\nu_{\min}$, we first compute for each point $\mathbf{q} \in Q$ the median distance from its six nearest neighbors in Q to its tangent plane; then we take the median $\overline{H}_Q$ of all such values and set $\nu_{\min} = \overline{H}_Q/6$ (see Appendix C for the rationale).

6 Results

In this section, we compare the performance of our methods with existing ICP-based methods including AA-ICP [11], sparse ICP [6], and symmetric ICP [17]. Our comparison includes both point-to-point and point-to-plane ICP methods and their variants. In the following, we will denote the point-to-point ICP and its variants as "ICP", whereas point-to-plane ICP and its variants will be denoted as "ICP-*l*". For sparse ICP and symmetric ICP, we use the source codes released by the authors[1] [2]. For symmetric ICP, we use the formulation that does not rotate the normals ($\mathcal{E}_{symm}$ as defined in the paper). Besides ICP-based methods, we also compare with other methods including CPD [25] and GMM-Reg [26] based on statistical frameworks, Teaser++ [34] which uses truncated least squares optimization, as well as DCP [35] and DGR [36] based on deep learning, using their open-source implementations [3] [4] [5] [6] [7]. We implement our methods in C++, using the Eigen library [63] for linear algebra operations. For each test problem, we normalize the input data by aligning the centroids of the point clouds and uniformly scaling them such that their bounding box has diagonal length 1. Unless stated otherwise, the point clouds are pre-aligned using Super4PCS [21] before the alignment is refined using different methods. We test the methods on both synthetic and real-world datasets. For problems where the ground-truth alignment is known, we evaluate the registration accuracy using the RMSE value in Eq. (12). Some methods (point-to-plane ICP and its variants, as well as symmetric ICP) require normals at the points. For synthetic data where the underlying surface is known, we use the surface normals as the normals at the points. Otherwise, we use the Point Cloud Library[8] to estimate the normals using 30 nearest neighbors. The source codes for our methods are available at https://github.com/yaoyx689/Fast-Robust-ICP. The two deep learning-based methods are run on a PC with a 20-core CPU at 3.3GHz, an NVIDIA RTX 2080 Ti and 128GB of RAM, whereas all other methods are run on a PC with a 6-core CPU at 3.6GHz and 16GB of RAM. Detailed settings for each method are provided in Appendix E.

6.1 Synthetic Data

In Fig. 1, we perform registration on two point sets with a small overlap, constructed using the monkey model from the EPFL statue dataset [13]. From the full model, we take the first 60% of the points to create the source set, and the last 47% with a random rigid transformation to construct the target set. Our robust point-to-point and point-to-plane ICP methods achieve the lowest RMSE values among all methods, while being significantly faster than point-to-point ICP, the method that achieves the next lowest RMSE. In addition, our fast ICP achieves the same RMSE as classical

1. https://github.com/OpenGP/sparseicp
2. https://gfx.cs.princeton.edu/pubs/Rusinkiewicz_2019_ASO/icptests-1.0.zip
3. https://github.com/gadomski/cpd
4. https://github.com/bing-jian/gmmreg
5. https://github.com/MIT-SPARK/TEASER-plusplus
6. https://github.com/WangYueFt/dcp
7. https://github.com/chrischoy/DeepGlobalRegistration
8. https://pointclouds.org/

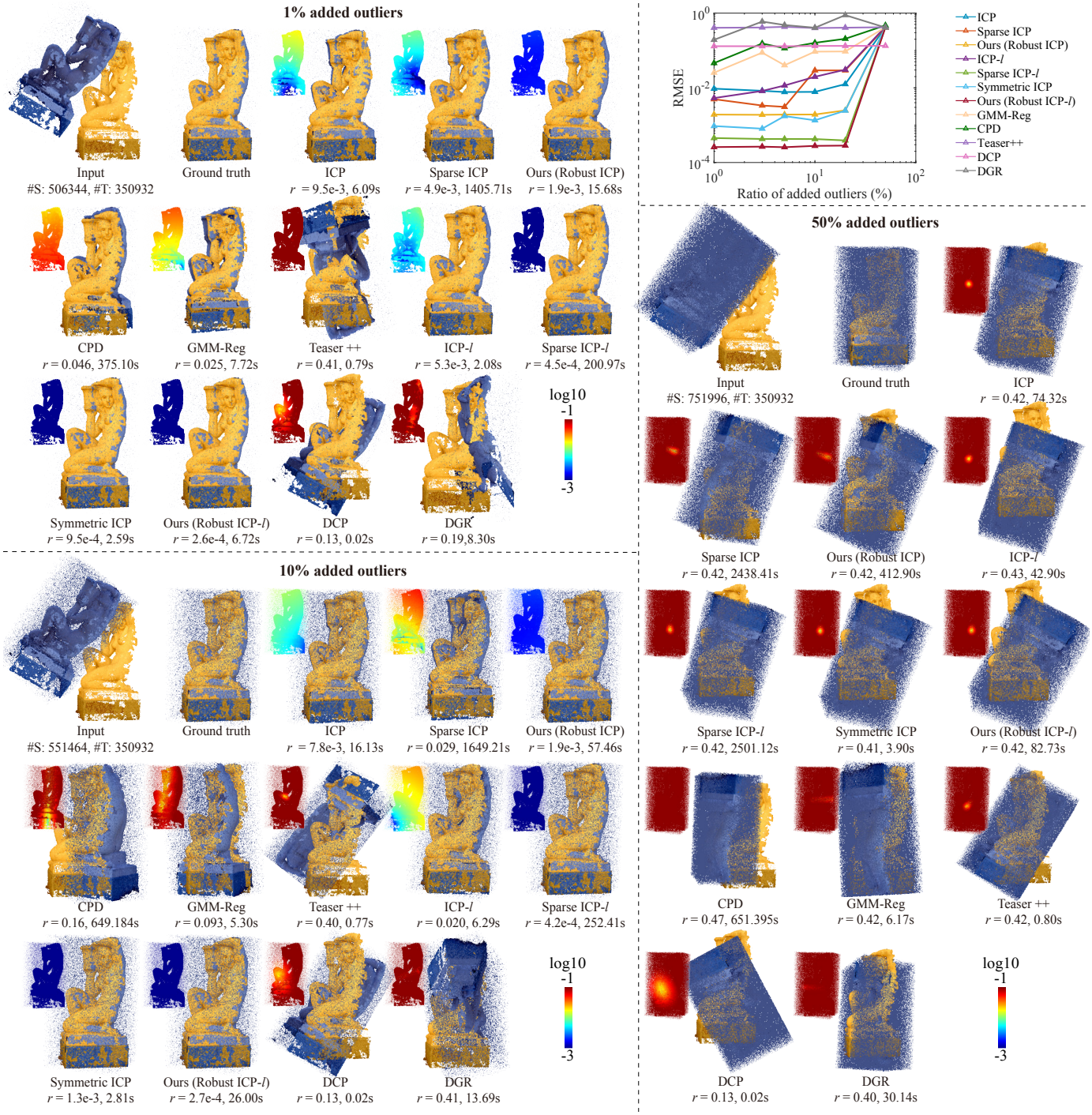

Fig. 6. Comparison between different registration methods on partially overlapping point clouds with added noises and outliers, constructed using the Aquarius model from the EPFL statue dataset [13]. The plot on the top-right shows the resulting RMSE values with different ratios (1%, 3%, 5%, 20% and 50%) of outliers added to the source point cloud.

ICP and AA-ICP with less computational time. Symmetric ICP also achieves better accuracy than point-to-point and point-to-plane ICP and their accelerated versions; it is faster than sparse ICP and our robust methods but with worse accuracy. The saving in computational time from symmetric ICP is partly because its implementation only samples 200 pairs of valid corresponding points for the alignment step, which reduces the computational cost for large point clouds.

In Fig. 6, we test the methods on point sets that contain noises and outliers, which are constructed using the Aquarius model from the EPFL statue dataset. Starting from the clean point cloud of the full model, we take the last 42% of the points as the target point cloud, add Gaussian noises along their normal directions with the standard deviation being the average value of all points' median distance to their six nearest neighbors, and apply a random transformation. For the source point cloud, we take the first 60% of the points from the full model, and add Gaussian noises in the same way as the target point could. To emulate outliers, we add $\mu \cdot M$ random points to the source point cloud using a

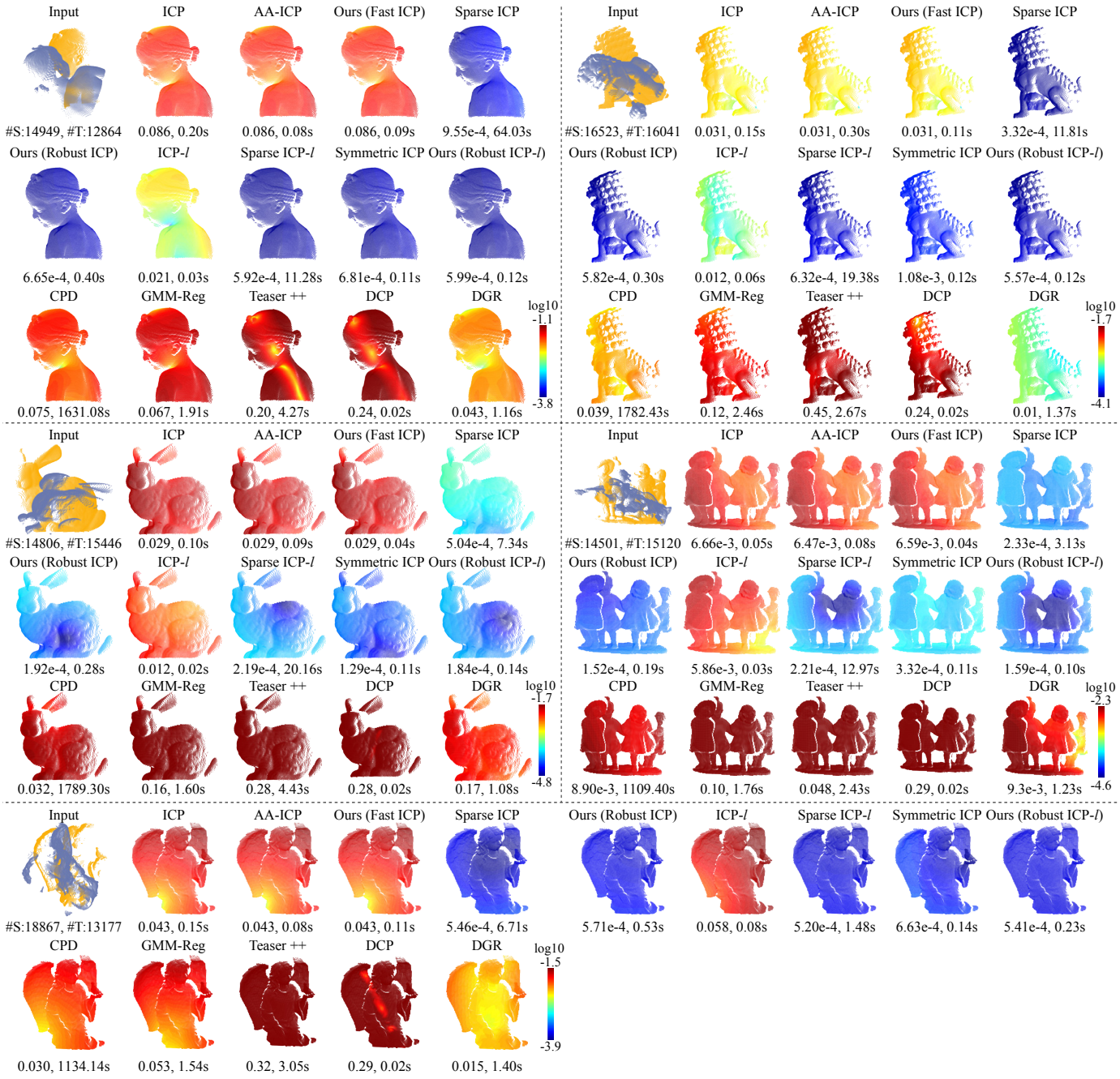

Fig. 7. Examples of registration results using different methods on partially overlapping point clouds, with RMSE and computational time shown below each result. The log-scale color-coding visualizes the deviation between the computed alignment and the ground-truth alignment.

uniform distribution within its bounding box, where $\overline{M}$ is the number of source points before the addition, and μ is chosen to be 1%, 3%, 5%, 20% and 50%, respectively. For problems with up to $\mu = 20\%$ added outliers, our robust point-to-point and point-to-plane methods outperform other point-to-point and point-to-plane ICP variants respectively in accuracy, with our robust point-to-plane method achieving the best accuracy among all methods. For 50% added outliers, all methods result in similar RMSE values that indicate large registration error. For ICP-based methods, this is partly due to poor initial alignment produced by Super4PCS in the presence of outliers. With better initialization, our methods can still produce reasonable registration results (see Section 6.3 for details). It is also worth noting that Teaser++, which is aimed at problems with a large amount of outliers, performs poorly in this example. This is potentially because Teaser++ assumes a generative model where the deviation between corresponding points is due to a bounded noise, which is not obeyed by the randomly generated outliers here.

We further test the methods on 25 pairs of partially overlapping point clouds constructed from five models in [30], with five pairs for each model. Fig. 7 compares the results on some problem instances, showing their computational time and RMSE, and using color-coding to visualize the deviation from the ground-truth alignment. Tab. 1 shows the average computational time and average/median RMSE on

TABLE 1
Average computational time (in seconds) and average/median RMSE ($\times 10^{-3}$) for different registration methods on partially overlapping point cloud pairs constructed from five models, with five pairs for each model (see Fig. 7). Best performance numbers are highlighted in bold fonts.

Dataset	Bimba		Children		Dragon		Angle		Bunny	
	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE
ICP	0.33	68/60	0.12	9.8/6.7	0.18	21/19	0.12	13/5.6	0.11	26/28
AA-ICP	0.13	68/60	0.10	9.8/6.5	0.16	21/19	0.08	13/5.6	0.10	26/28
Ours (Fast ICP)	0.12	68/60	0.07	9.8/6.6	0.12	21/19	0.11	13/5.6	0.06	26/28
Sparse ICP	37.90	67/27	8.59	0.96/0.81	24.42	**0.92**/0.95	15.45	0.83/**0.97**	24.06	0.94/0.71
Ours (Robust ICP)	0.96	**0.87**/0.67	0.26	0.89/**0.62**	0.27	0.93/**0.92**	0.27	0.83/0.98	0.34	0.85/0.69
ICP-*l*	0.74	78/20	0.07	16/5.9	1.06	14/11	0.07	13/2.5	0.06	13/12
Sparse ICP-*l*	20.97	3.4/**0.59**	20.69	0.92/0.67	14.25	0.96/0.94	6.23	**0.82**/0.98	10.68	0.81/**0.56**
Symmetric ICP	0.25	34/0.6	0.19	**0.88**/0.64	0.20	0.92/0.93	0.20	0.82/0.98	0.18	**0.79**/0.56
Ours (Robust ICP-*l*)	0.36	57/0.6	0.20	0.88/0.64	0.21	0.92/0.93	0.18	0.82/0.98	0.17	0.79/0.56
GMM	4.37	130/120	3.52	29/30	5.66	63/49	4.02	19/12	5.73	110/120
CPD	2053.87	59/75	1115.53	17/9.9	2646.56	18/13	2056.87	13/8.8	1729.61	34/32
Teaser++	3.07	180/180	2.80	38/20	2.83	230/90	2.75	230/320	2.99	250/260
DCP	**0.02**	220/210	**0.02**	320/360	**0.02**	300/270	**0.02**	330/350	**0.02**	220/220
DGR	1.10	26/28	1.12	6.5/5.7	1.29	9.8/9.8	0.94	6.5/5.2	1.25	11/8.9

each model for each method. Overall, our robust methods and sparse ICP lead to more accurate results. Our methods achieve best average/median RMSE measures in more instances, while being significantly faster than sparse ICP.

In Fig. 8, we evaluate how partial overlaps and initialization affect the registration accuracy of different methods. For the Stanford bunny model, we use the method from [64] to simulate four point clouds captured using Kinect from different locations on the same horizontal plane as the model. We take one of the point clouds as the source, and each of the remaining three as the target for registration. For each pair of point clouds, we first place them according to their ground-truth alignment and perform PCA on the points, then rotate the target point cloud around the PCA axis with the smallest variance by an angle β as initial alignment. As β increases, the initialization deviates more from the ground-truth alignment. We test the methods with $\beta = 10°$, $20°$, $50°$, $60°$, $80°$ and $100°$, respectively. Fig. 8 plots the resulting RMSE values on each pair of point clouds with different values of β, together with the overlapping ratio with respect to the source point cloud. For all methods, the registration accuracy deteriorates as the overlap ratio decreases and the rotation angle increases. For an overlap ratio of 59%, our robust methods and symmetric ICP can achieve small RMSE values at the scale of 1×10^{-3} with a rotation angle up to 60%. For an overlap ratio of 19%, our robust point-to-point ICP can still achieve an RMSE at the scale of 1×10^{-2} with a rotation angle up to 60%, while other methods perform notably worse. With 1% overlap, all methods result in large RMSE values regardless of the rotation angle.

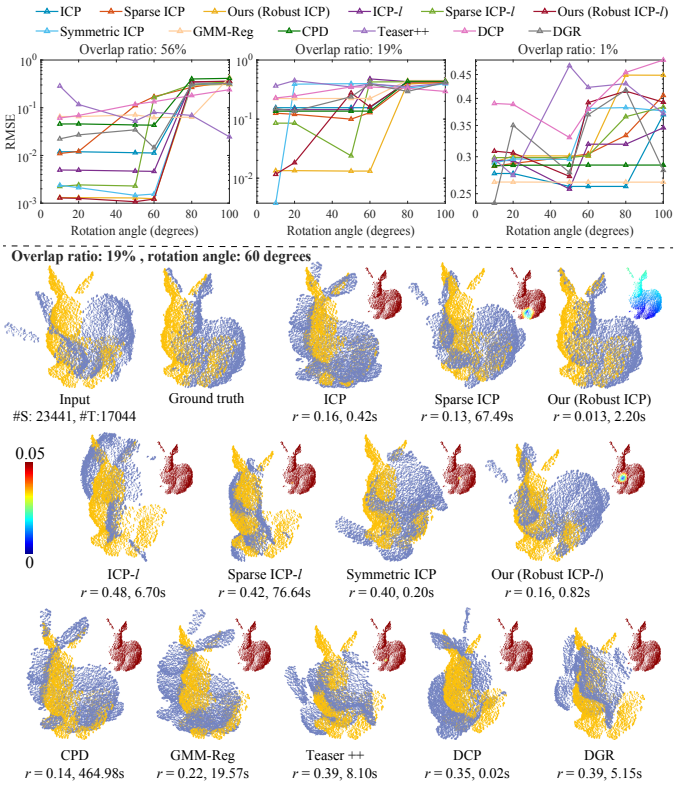

Fig. 8. Registration results on simulated Kinect point clouds from the Stanford bunny model, with different overlap ratios, and different amounts of rotation between the initial alignment to the ground truth.

6.2 Real-World Data

To evaluate their performance on real-world problems, we test the methods on the RGB-D SLAM dataset [65], the ETH laser registration dataset [62], and the 3DMatch dataset [66]. For the RGB-D SLAM dataset, We use eight point cloud sequences captured with two cameras ("xyz", "360","teddy", "desk" and "plant" for camera 1; 'dishes", "coke" and "flowerbouquet" for camera 2). For each sequence, we register pairs of point clouds that are a fixed number of frames apart (five frames for camera 1, and 20 frames for camera 2, taking into consideration the different velocities of the two cameras). As the two point clouds are already close to each other, we directly apply the registration methods without pre-alignment. For the ETH laser registration dataset, we test all of its eight point cloud sequences each containing between 31 and 45 point clouds, and we align each pair of adjacent point clouds from each sequence. For the 3DMatch dataset, we use the point cloud pairs in their geometric registration benchmark, and divide them into five categories according to the overlapping ratio with respect to the source point cloud: $[0\%, 20\%)$, $[20\%, 40\%)$, $[40\%, 60\%)$, $[60\%, 80\%)$,

TABLE 2
Average computational time (in seconds) and average/median RMSE ($\times 10^{-2}$) using different registration methods for eight sequences from the RGB-D SLAM dataset [65], with best performance numbers highlighted in bold fonts.

Dataset	xyz		360		teddy		desk		plant		dishes		coke		flowerbouquet	
	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE
ICP	0.23	2.1/0.89	0.76	5.1/4	0.68	2.1/1.4	0.26	2.3/1.2	0.51	1.6/1.1	0.80	3.7/2.8	0.93	3.1/2.5	0.91	2.7/2.2
AA-ICP	0.16	2.1/0.9	0.48	5.1/4	0.38	2.1/1.4	0.17	2.3/1.2	0.32	1.6/1.1	0.43	3.7/2.8	0.51	3.1/2.5	0.59	2.7/2.1
Ours (Fast ICP)	0.14	2.1/0.87	0.36	5.1/4	0.35	2.1/1.4	0.15	2.3/1.2	0.26	1.6/1.1	0.37	3.7/2.8	0.43	3.1/2.5	0.43	2.7/2.2
Sparse ICP	11.2	1.6/0.86	36.4	4.8/3.7	51.7	1.8/1.1	26.7	1.8/1.1	71.9	0.88/0.67	57.3	3.9/3.6	77.1	3.3/3	66.8	3.2/3
Ours (Robust ICP)	0.60	**0.5**/0.43	2.86	2.2/0.75	2.69	**1/0.76**	0.93	1.2/0.77	2.25	**0.65/0.56**	2.36	3.2/2.6	3.73	2.4/2.1	3.52	2.4/1.8
ICP-*l*	0.76	2.6/0.63	1.93	4.3/2.2	1.31	1.9/1.2	0.66	14/0.9	1.13	1.3/0.89	1.28	11/2.7	0.95	3.1/2.6	1.00	2.7/2.1
Sparse ICP-*l*	31.0	0.85/**0.43**	64.0	2.6/0.86	86.3	1.3/0.79	33.1	**1.2/0.63**	62.4	0.83/0.6	65.6	790/3.6	108	3.5/3	92.2	3.4/3.3
Symmetric ICP	0.18	1.2/0.44	0.36	1.8/**0.73**	0.47	1.1/0.82	0.19	1.7/0.7	0.42	0.7/0.59	0.39	3.8/3	0.54	3.4/2.9	0.53	3.3/3
Ours (Robust ICP-*l*)	0.43	1.1/0.43	1.25	2.6/0.84	1.34	1.3/0.82	0.56	1.5/0.64	1.16	0.71/0.57	1.11	3.8/3.4	1.50	3.3/3	1.46	3.2/3
GMM-Reg	1.88	4.4/3.7	2.31	6.3/4.6	1.63	3.2/2.6	1.95	5.7/4.6	1.72	2.7/2.1	1.65	3.7/2.7	1.73	3.1/2	1.63	2.4/1.8
CPD	427	4.6/3.6	423	5.3/3.3	442	2.2/1.5	348	2.1/1.8	442	1.4/1.1	434	3.4/2.7	433	2.6/1.8	430	2.3/1.8
Teaser++	1.36	3.4/2	1.05	20/14	0.77	11/3.2	5.89	8/2.5	0.83	6.1/2.4	0.73	21/15	0.58	23/21	0.59	20/11
DCP	**0.02**	6.5/5.4	**0.02**	10/9.9	**0.02**	6.6/5.5	**0.02**	7/6	**0.02**	5.6/4.9	**0.02**	7.4/6	**0.02**	7.5/6	**0.02**	6.3/5.7
DGR	0.68	0.6/0.53	1.14	**1.4**/0.86	1.19	1/0.84	0.77	1.2/0.96	1.15	0.71/0.65	3.54	**2.6/1.7**	4.69	**2.1/1.2**	3.96	**1.9/1.1**

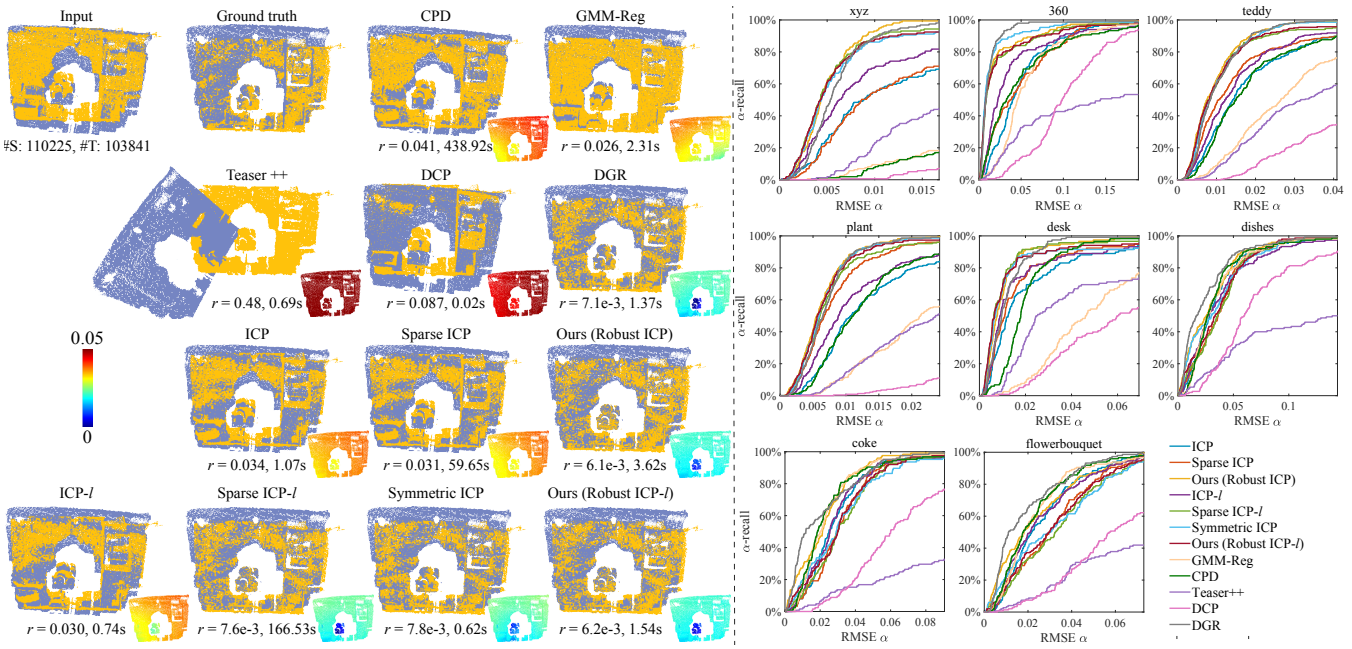

Fig. 9. Examples of registration results using different methods on eight sequences from the RGB-D SLAM dataset [65], with color-coding to visualize the deviation from the ground-truth alignment. The plots on the right show the α-recall rates of different methods for each point cloud sequence from the dataset.

and $[80\%, 100\%]$. Within each category we sample 50 pairs to perform registration. All point clouds are pre-processed using a box grid filter to make the density more uniform, which is the same as the pre-processing operation in [67].

Tables 2, 3 and 4 show the average computational time and average/median RMSE for each method on the datasets, whereas Figures 9, 10 and 11 show examples of registration results together with color-coding of their deviation from the ground truth. To visualize the distribution of RMSE within each dataset, we also compute the *α-recall rate* $|S_\alpha|/|S|$ for each method, where $|S|$ is the total number of test cases, and $|S_\alpha|$ is the number of test cases where the RMSE is less than α [30]. Intuitively, for a given α, a higher α-recall rate indicates more test cases with RMSE values lower than α. The plots of α-recall rates for each method are included in Figures 9, 10 and 11. For the RGB-D SLAM dataset and the ETH laser registration dataset, the majority of the lowest average/median RMSE values are achieved by our robust methods, sparse ICP and symmetric ICP. Like previous examples, our methods achieve similar or better accuracy than sparse ICP with much lower computational cost. DGR has good performance on both datasets: on the RGB-D SLAM dataset it achieves the lowest average and median RMSE for all the camera-2 sequences, whereas on the ETH laser registration dataset it achieves the lowest average RMSE on many sequences. This is potentially due to similar characteristics between its training data and the test cases. For the 3DMatch dataset, ICP-based methods perform better on problems with overlap ratios higher than 40%, with our robust point-to-point ICP and symmetric ICP attaining four out of the six lowest average/median RMSE values. For lower overlap ratios, DGR achieves the best accuracy

TABLE 3
Average computational time (in seconds) and average/median RMSE ($\times 10^{-3}$) for different registration methods on point cloud pairs in eight sequences from the ETH laser registration dataset [62]. Best performance numbers are highlighted in bold fonts.

Method	Apartment		ETH Hauptgebaude		Stairs		Mountains		Gazebo in summer		Gazebo in winter		Wood in summer		Wood in winter	
	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE
ICP	0.06	36/12	0.23	33/2.5	0.15	16/2.4	0.12	12/6.1	0.14	15/7.5	0.17	14/1.7	0.22	9.7/1.4	0.16	8.8/1.3
AA-ICP	0.08	45/12	0.24	35/2.6	0.14	16/2.4	0.12	12/6.2	0.15	15/7.5	0.21	14/1.7	0.18	9.7/1.4	0.25	8.8/1.3
Ours (Fast ICP)	0.05	36/12	0.17	33/2.6	0.08	17/2.6	0.08	12/6.2	0.14	15/7.5	0.14	14/1.7	0.23	9.7/1.4	0.15	8.8/1.3
Sparse ICP	1.23	13/2	13.6	39/10	2.33	6/1.2	4.22	4.7/0.62	7.18	11/0.75	12.8	17/0.51	13.6	11/0.44	13.4	7.2/0.46
Ours (Robust ICP)	0.24	13/1.9	0.83	15/0.55	0.25	5.6/1.2	0.22	3/0.6	0.40	0.77/0.72	0.63	10/0.35	0.72	8.3/**0.42**	0.48	**0.4**/**0.36**
ICP-*l*	0.74	41/7.1	0.10	33/1.4	0.36	8.3/1.2	0.08	6/3.1	0.28	12/6.5	0.53	18/1.5	0.34	14/1.5	0.64	9.1/1.5
Sparse ICP-*l*	11.0	12/0.83	14.1	35/**0.41**	14.0	4.8/**0.27**	13.0	1.7/**0.59**	10.3	**0.63**/0.58	11.6	12/**0.29**	17.7	11/0.44	18.5	7.2/0.47
Symmetric ICP	0.09	11/1.1	0.22	33/0.44	0.10	6/0.49	0.10	**0.84**/0.82	0.17	6.4/0.67	0.19	10/0.39	0.24	8.5/0.65	0.23	7.6/0.73
Ours (Robust ICP-*l*)	0.13	**11**/**0.79**	0.35	35/0.43	0.17	4.8/0.32	0.15	4.3/0.63	0.27	2.6/**0.56**	0.34	18/0.37	0.48	11/0.43	0.48	7.2/0.38
GMM-Reg	1.95	28/15	1.78	43/9.7	2.27	20/8.6	2.46	15/11	1.45	12/10	1.58	20/7.9	1.19	22/11	1.13	19/11
CPD	127	28/6.4	465	43/12	246	8.2/3.5	150	9.6/5.7	454	12/8.5	453	20/8	458	23/13	456	19/13
Teaser++	0.53	260/310	0.46	190/180	0.48	150/130	0.45	190/190	0.45	230/250	0.45	140/160	0.47	180/180	0.45	200/190
DCP	**0.02**	100/100	**0.02**	20/15	**0.02**	44/39	**0.02**	40/34	**0.02**	58/50	**0.02**	32/27	**0.02**	28/19	**0.02**	45/32
DGR	5.85	15/1.7	27.28	**7.4**/8.6	0.61	**3.5**/1.8	0.78	15/6	0.87	1.7/1.1	1.05	**0.6**/0.55	1.37	**1**/0.94	1.30	2.5/0.81

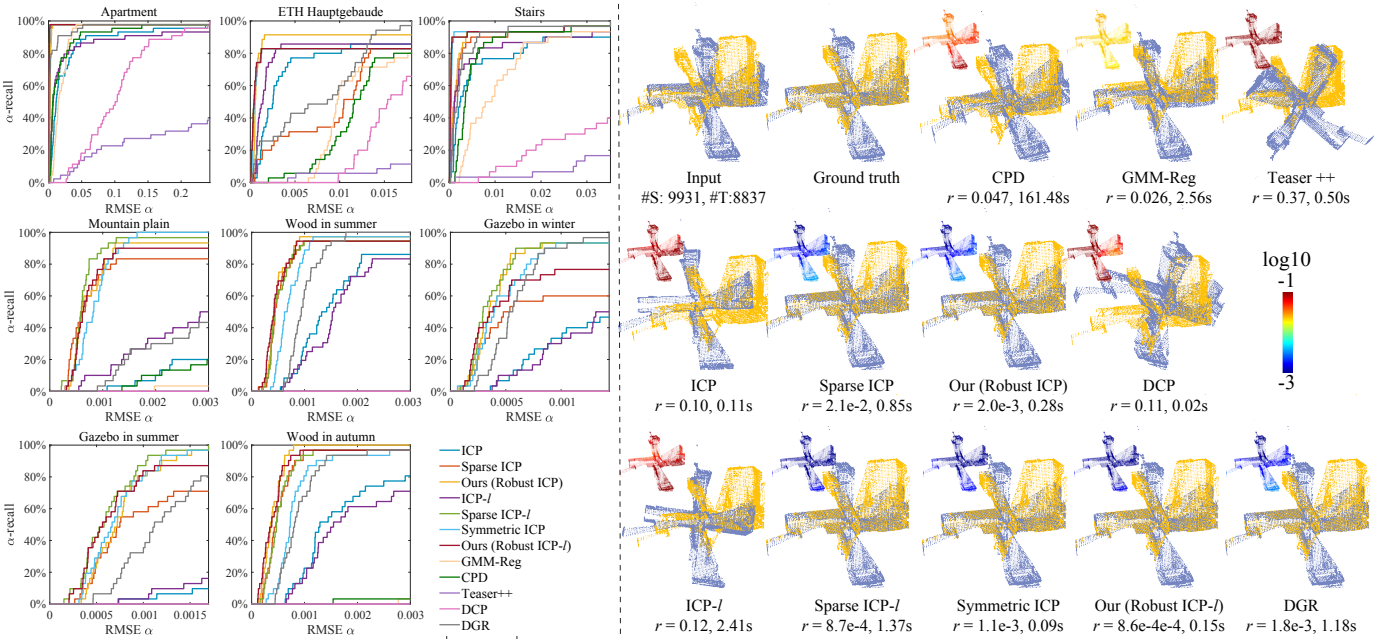

Fig. 10. Examples of registration results with different methods on the ETH laser registration dataset [62], with color-coding for the deviation from the ground-truth alignment. The plots on the left show the α-recall rates of each method for the point cloud sequences in the dataset.

TABLE 4
Average computational time (in seconds) and average/median RMSE ($\times 10^{-2}$) using different registration methods for the 3DMatch dataset [66], with best performance numbers highlighted in bold fonts.

Overlap	0-20%		20%-40%		40%-60%		60%-80%		80%-100%	
	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE
ICP	0.30	43/42	0.28	26/23	0.21	16/11	0.16	6.4/4	0.06	1.8/1.1
AA-ICP	0.25	43/42	0.27	26/23	0.22	16/11	0.13	6.4/4	0.06	1.8/1.1
Ours (Fast ICP)	0.19	43/42	0.16	26/23	0.12	16/11	0.09	6.4/4	0.05	1.8/1.1
Sparse ICP	45.92	41/43	42.54	22/18	29.20	10/2.5	20.38	3.1/0.74	9.43	0.75/0.34
Ours (Robust ICP)	2.18	43/44	1.27	22/19	0.69	9.2/1	0.38	2.5/**0.51**	0.17	0.76/**0.33**
ICP-*l*	3.91	44/44	2.67	25/23	1.53	14/7.9	0.87	4.7/2	0.13	1.4/0.69
Sparse ICP-*l*	73.05	43/45	67.49	21/16	53.82	10/1.4	35.63	2.6/0.54	26.19	0.7/0.33
Symmetric ICP	0.17	40/45	0.15	16/2.6	0.12	9/**0.91**	0.11	2.7/0.59	0.09	**0.36**/0.34
Ours (Robust ICP-*l*)	0.59	42/43	0.40	20/15	0.29	9.8/1.3	0.22	2.5/0.56	0.15	0.71/0.35
GMM-Reg	2.08	52/50	2.33	33/32	2.25	20/12	2.01	11/7.2	1.74	8.2/6.9
CPD	308.72	43/45	390.49	30/27	361.42	20/16	291.68	9.4/6.2	181.80	6.1/4.6
Teaser++	**0.001**	50/52	**0.001**	42/44	**0.001**	34/38	**0.001**	20/18	**0.001**	18/14
DCP	0.02	37/38	0.02	29/28	0.02	22/20	0.02	13/13	0.02	8.6/8.3
DGR	2.94	**14**/**2.7**	2.61	**1.5**/**1.3**	1.33	**2.7**/0.96	1.12	**0.75**/0.64	0.86	0.51/0.43

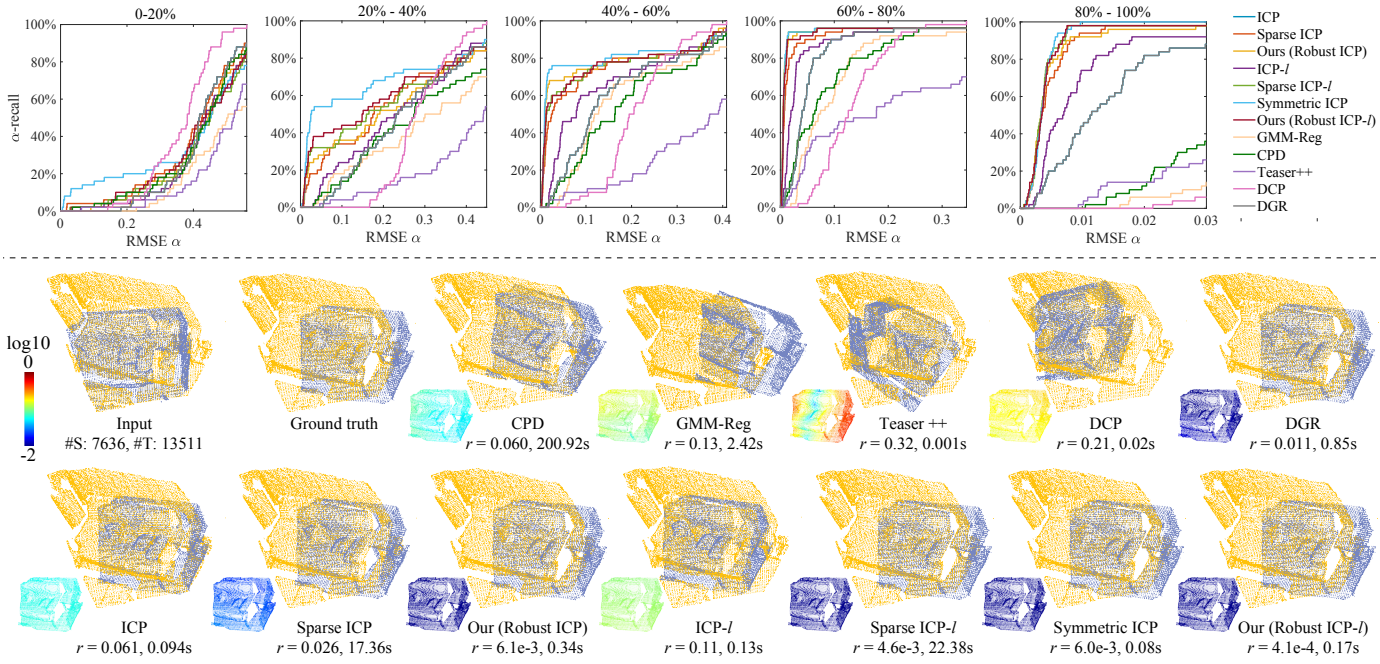

Fig. 11. Examples of registration results for point clouds from the 3DMatch dataset [66], with color-coding to visualize the deviation from the ground-truth alignment. The plots on the top show the α-recall rates of different methods on point cloud pairs in each range of overlapping ratio.

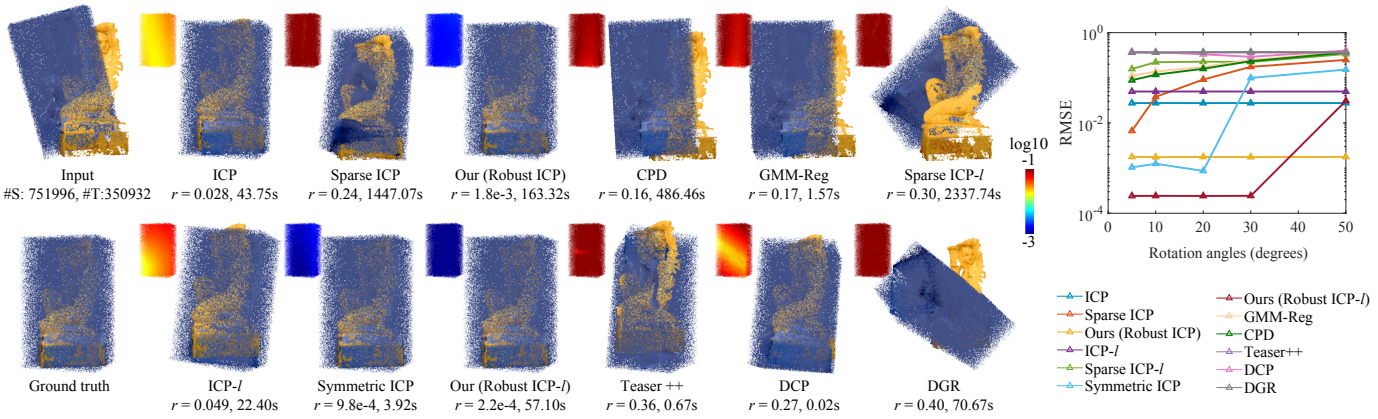

Fig. 12. Registration results for the failure case in Fig. 6 with 50% added outliers, using random initial alignments that rotates the source point cloud from its ground-truth position by a fixed angle around a random axis. The plots on the right shows the average RMSE values for each rotation angle.

because it is trained using the training set of the 3DMatch dataset and learns the characteristics of the test cases.

6.3 Limitations

Like other ICP-based methods, our robust methods rely on good initial alignment. As shown in Fig. 6 and Tab. 4, for some challenging problems, our methods may perform poorly because the initial alignment from Super4PCS deviates significantly from the ground truth. In Fig. 12, we conduct another experiment for the failure case in Fig. 6 with 50% added outliers, using random initial alignments instead of Super4PCS. Specifically, we first rotate the source point cloud from its ground-truth position by a fixed angle β around a random axis, and then perform registration. We test the methods with $\beta = 5°, 10°, 20°, 30°$ and $50°$, respectively. For each value of β, we conduct the experiment 10 times to construct 10 random initial alignments, and compute the average RMSE for each method. Fig. 12 shows that our robust methods can still produce good results for such a challenging case if the initialization is not too far away from the ground truth. In particular, our robust point-to-plane ICP produces an average RMSE at the scale of 1×10^{-4} with a rotation angle up to $30°$, whereas our robust point-to-point ICP produces an average RMSE at the scale of 1×10^{-3} with a rotation angle up to $50°$. It verifies that the poor performance of our methods in Fig. 6 is due to initialization.

For point clouds with a very small overlap, our methods may produce an incorrect result even with a good initial alignment (e.g., see Fig. 8). This is partly due to our choice of the parameter ν. Its initial value $\nu_{\max}$ is chosen based on the median initial alignment error, which is affected by the 50% of source points that are closest to the target point cloud. If the proportion of source points in the overlapping region is significantly less than 50%, then $\nu_{\max}$ may be much larger

than the true initial distance. This may include too many source points into the initial iterations of the solver and lead it towards an incorrect result.

7 Conclusion and Future Work

We proposed methods to improve the convergence speed and robustness of point-to-point and point-to-plane ICP methods. We first propose an Anderson-accelerated point-to-point ICP based on Lie algebra parameterization of rigid transformations, together with a stabilization strategy that ensures monotonic decrease of target energy. We also develop a robustified point-to-point ICP formulation based on the Welsch's function, and solve it using an Anderson-accelerated MM solver. Finally, we extend the robust formulation and the accelerated numerical solver to point-to-plane ICP. The resulting robust ICP schemes achieve similar or better accuracy than sparse ICP, while being significantly faster. The methods provide efficient and robust solutions to rigid registration problems where the data may be noisy, contain outliers, and overlap partially.

Our methods can be further improved in a few directions. First, to obtain good initial alignment for challenging cases, we can potentially adopt a machine learning-based method for determining a coarse alignment; this is similar to the practice in [35] that uses ICP to refine a DCP alignment. Second, we need a more sophisticated way to control the ν parameter and make our solver more robust on point clouds with a very small overlap; a data-driven approach could be a potential solution. Finally, symmetric ICP shows promising performance in many of our comparisons; one interesting future work is to extend our approach to the symmetric ICP formulation, e.g. by replacing their ℓ_2 target function with a robust error metric.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61672481), and Youth Innovation Promotion Association CAS (No. 2018495).

References

[1] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[2] H. Pottmann, Q. Huang, Y. Yang, and S. 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.

[3] 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.

[4] H. Pottmann, S. Leopoldseder, and M. Hofer, "Registration without ICP," *Computer Vision and Image Understanding*, vol. 95, no. 1, pp. 54–71, 2004.

[5] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *3rd International Conference on 3D Digital Imaging and Modeling (3DIM 2001)*, 2001, pp. 145–152.

[6] S. Bouaziz, A. Tagliasacchi, and M. Pauly, "Sparse iterative closest point," *Comput. Graph. Forum*, vol. 32, no. 5, pp. 113–123, 2013.

[7] K. Lange, *MM Optimization Algorithms*. SIAM, 2016.

[8] 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.

[9] Y. Peng, B. Deng, J. Zhang, F. Geng, W. Qin, and L. Liu, "Anderson acceleration for geometry optimization and physics simulation," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 42:1–42:14, 2018.

[10] N. J. Mitra and A. Nguyen, "Estimating surface normals in noisy point cloud data," in *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, 2003, pp. 322–328.

[11] A. L. Pavlov, G. V. 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)*, 2018, pp. 1–6.

[12] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics - Theory and Methods*, vol. 6, no. 9, pp. 813–827, 1977.

[13] EPFL Computer Graphics and Geometry Laboratory, "EPFL statue model repository," https://lgg.epfl.ch/statues_dataset.php, 2012.

[14] G. K. L. Tam, Z. Cheng, Y. Lai, F. C. Langbein, Y. Liu, A. D. Marshall, R. R. Martin, X. Sun, and P. L. Rosin, "Registration of 3d point clouds and meshes: A survey from rigid to nonrigid," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 7, pp. 1199–1217, 2013.

[15] S. Bouaziz, A. Tagliasacchi, H. Li, and M. Pauly, "Modern techniques and applications for real-time non-rigid registration," in *SIGGRAPH ASIA 2016 Courses*, 2016.

[16] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets - open-source library and experimental protocol," *Auton. Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[17] S. Rusinkiewicz, "A symmetric objective function for ICP," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 85:1–85:7, 2019.

[18] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, "Robust global registration," in *Proceedings of the Third Eurographics Symposium on Geometry Processing*, 2005.

[19] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3d registration," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217.

[20] D. Aiger, N. J. Mitra, and D. Cohen-Or, "4-points congruent sets for robust pairwise surface registration," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 85:1–85:10, 2008.

[21] N. Mellado, D. Aiger, and N. J. Mitra, "Super 4PCS fast global pointcloud registration via smart indexing," *Computer Graphics Forum*, vol. 33, no. 5, pp. 205–215, 2014.

[22] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas, "Registration of point cloud data from a geometric optimization perspective," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 2004, pp. 22–31.

[23] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[24] D. Chetverikov, D. Stepanov, and P. Krsek, "Robust Euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm," *Image and Vision Computing*, vol. 23, no. 3, pp. 299–309, 2005.

[25] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010.

[26] B. Jian and B. C. Vemuri, "Robust point set registration using gaussian mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1633–1645, 2011.

[27] T. Masuda and N. Yokoya, "A robust method for registration and segmentation of multiple range images," *Computer Vision and Image Understanding*, vol. 61, no. 3, pp. 295–307, 1995.

[28] E. Trucco, A. Fusiello, and V. Roberto, "Robust motion and correspondence of noisy 3-d point sets with missing data," *Pattern Recognition Letters*, vol. 20, no. 9, pp. 889–898, 1999.

[29] A. W. Fitzgibbon, "Robust registration of 2d and 3d point sets," *Image Vision Comput.*, vol. 21, no. 13-14, pp. 1145–1153, 2003.

[30] Q. Zhou, J. Park, and V. Koltun, "Fast global registration," in *Computer Vision – ECCV 2016*, 2016, pp. 766–782.

[31] H. Li and R. Hartley, "The 3D-3D registration problem revisited," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[32] C. Olsson, F. Kahl, and M. Oskarsson, "Branch-and-bound methods for euclidean registration problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 783–794, 2009.

[33] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A globally optimal solution to 3d ICP point-set registration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2241–2254, 2016.

[34] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and certifiable point cloud registration," *IEEE Transactions on Robotics*, 2020.

[35] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[36] C. Choy, W. Dong, and V. Koltun, "Deep global registration," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[37] D. G. Anderson, "Iterative procedures for nonlinear integral equations," *J. ACM*, vol. 12, no. 4, pp. 547–560, 1965.

[38] V. Eyert, "A comparative study on methods for convergence acceleration of iterative vector sequences," *Journal of Computational Physics*, vol. 124, no. 2, pp. 271–285, 1996.

[39] H.-r. Fang and Y. Saad, "Two classes of multisecant methods for nonlinear acceleration," *Numerical Linear Algebra with Applications*, vol. 16, no. 3, pp. 197–221, 2009.

[40] A. Toth and C. T. Kelley, "Convergence analysis for anderson acceleration," *SIAM Journal on Numerical Analysis*, vol. 53, no. 2, pp. 805–819, 2015.

[41] H. D. Sterck, "A nonlinear gmres optimization algorithm for canonical tensor decomposition," *SIAM Journal on Scientific Computing*, vol. 34, no. 3, pp. A1351–A1379, 2012.

[42] K. Lipnikov, D. Svyatskiy, and Y. Vassilevski, "Anderson acceleration for nonlinear finite volume scheme for advection-diffusion problems," *SIAM Journal on Scientific Computing*, vol. 35, no. 2, pp. A1120–A1136, 2013.

[43] P. P. Pratapa, P. Suryanarayana, and J. E. Pask, "Anderson acceleration of the jacobi iterative method: An efficient alternative to Krylov methods for large, sparse linear systems," *Journal of Computational Physics*, vol. 306, pp. 43–54, 2016.

[44] N. Ho, S. D. Olson, and H. F. Walker, "Accelerating the Uzawa algorithm," *SIAM Journal on Scientific Computing*, vol. 39, no. 5, pp. S461–S476, 2017.

[45] P. Suryanarayana, P. P. Pratapa, and J. E. Pask, "Alternating anderson-richardson method: An efficient alternative to preconditioned krylov methods for large, sparse linear systems," *Computer Physics Communications*, vol. 234, pp. 278–285, 2019.

[46] J. Zhang, Y. Peng, W. Ouyang, and B. Deng, "Accelerating ADMM for efficient simulation and optimization," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 163:1–163:21, 2019.

[47] W. Ouyang, Y. Peng, Y. Yao, J. Zhang, and B. Deng, "Anderson acceleration for nonconvex ADMM based on Douglas-Rachford splitting," *Computer Graphics Forum*, vol. 39, no. 5, pp. 221–239, 2020.

[48] F. A. Potra and H. Engler, "A characterization of the behavior of the anderson acceleration on linear problems," *Linear Algebra and its Applications*, vol. 438, no. 3, pp. 1002–1011, 2013.

[49] O. Sorkine-Hornung and M. Rabinovich. (2017) Least-squares rigid motion using svd. [Online]. Available: https://igl.ethz.ch/projects/ARAP/svd_rot.pdf

[50] K. Lange, *Optimization*. Springer New York, 2004, ch. The MM Algorithm, pp. 119–136.

[51] C. Evans, S. Pollock, L. G. Rebholz, and M. Xiao, "A proof that Anderson acceleration improves the convergence rate in linearly converging fixed-point methods (but not in those converging quadratically)," *SIAM J. Numer. Anal.*, vol. 58, no. 1, pp. 788–810, 2020.

[52] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," 2006.

[53] D. K. Hoffman, R. C. Raffenetti, and K. Ruedenberg, "Generalization of Euler angles to n-dimensional orthogonal matrices," *Journal of Mathematical Physics*, vol. 13, no. 4, pp. 528–533, 1972.

[54] V. S. Varadarajan, *Lie groups, Lie algebras, and their representations*, ser. Graduate Texts in Mathematics. New York: Springer-Verlag, 1984, vol. 102.

[55] J. Gallier and D. Xu, "Computing exponentials of skew symmetric matrices and logarithms of orthogonal matrices," *International Journal of Robotics and Automation*, vol. 18, no. 1, pp. 10–20, 2002.

[56] A. Fu, J. Zhang, and S. Boyd, "Anderson accelerated Douglas-Rachford splitting," *arXiv preprint arXiv:1908.11482*, 2019.

[57] B. Ham, M. Cho, and J. Ponce, "Robust guided image filtering using nonconvex potentials," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 1, pp. 192–207, 2018.

[58] J. Zhang, B. Deng, Y. Hong, Y. Peng, W. Qin, and L. Liu, "Static/dynamic filtering for mesh geometry," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 4, pp. 1774–1787, 2019.

[59] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 3869–3872.

[60] P. Ochs, A. Dosovitskiy, T. Brox, and T. Pock, "On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision," *SIAM Journal on Imaging Sciences*, vol. 8, no. 1, pp. 331–372, 2015.

[61] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

[62] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms," *The International Journal of Robotics Research*, vol. 31, no. 14, pp. 1705–1711, 2012.

[63] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[64] J. Bohg, J. Romero, A. Herzog, and S. Schaal, "Robot arm pose estimation through pixel-wise part classification," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3143–3150.

[65] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[66] A. Zeng, S. Song, M. Niessner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[67] J. Vongkulbhisal, B. I. Ugalde, F. D. la Torre, and J. P. Costeira, "Inverse composition discriminative optimization for point cloud registration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2993–3001.

[68] N. J. Higham, *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, 2008.

[69] G. H. Golub and C. F. V. Loan, *Matrix computions*. The Johns Hopkins University Press, 1983.

[70] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[71] R. M.Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[72] G. Guillermo and Y. Anthony, "A compact formula for the derivative of a 3-d rotation in exponential coordinates," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 3, pp. 378–384, 2015.

APPENDIX A
COMPUTING MATRIX LOGARITHMS

To compute matrix logarithms, Existing numerical methods such as the *inverse scaling and squaring method* [68] requires the matrix to have no negative eigenvalues, which may not hold for the transformation matrices considered in this paper. In the following, we derive a numerical method for computing logarithms of transformation matrices without such restrictions.

Given a transformation matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$$

where $\mathbf{R}$ is a rotation matrix, it can be shown that its real Schur decomposition has the following form [69], [70]:

$$\mathbf{T} = \mathbf{Q}\mathbf{U}\mathbf{Q}^T = \mathbf{Q} \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} & \mathbf{U}_{13} & \dots & \mathbf{U}_{1m} \\ \mathbf{0} & \mathbf{U}_{22} & \mathbf{U}_{23} & \dots & \mathbf{U}_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{U}_{mm} \end{bmatrix} \mathbf{Q}^T,$$

where $\mathbf{Q} \in \mathbb{R}^{(n+1)\times(n+1)}$ is an orthogonal matrix, and each diagonal block $\mathbf{U}_{ii}$ is either a 1-by-1 matrix or a 2-by-2 matrix. Due to the special form of $\mathbf{T}$, we can permute the rows and columns of $\mathbf{U}$ as well as the columns of $\mathbf{Q}$ to obtain the following decomposition:

$$\mathbf{T} = \mathbf{Q}'\mathbf{U}'\mathbf{Q}'^T \tag{27}$$

where

$$\mathbf{Q}' = \begin{bmatrix} \mathbf{Q}_1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{U}' = \begin{bmatrix} \mathbf{D} & \mathbf{y} \\ \mathbf{0} & 1 \end{bmatrix},$$

and $\mathbf{D}$ is a block diagonal matrix [70]:

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & & & \\ & \ddots & & \\ & & \mathbf{D}_p & \\ & & & \mathbf{I}_{n-2p} \end{bmatrix}.$$

Here $\mathbf{D}_i$ is 2-by-2 rotation matrix and can be written as

$$\mathbf{D}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \tag{28}$$

with $\theta \in [0, \pi]$. Using the decomposition (27), the logarithm of $\mathbf{T}$ can be computed as [69], [70]:

$$\log(\mathbf{T}) = \log(\mathbf{Q}'\mathbf{U}'\mathbf{Q}'^T) = \mathbf{Q}' \log(\mathbf{U}')\mathbf{Q}'^T.$$

To compute $\log(\mathbf{U}')$, we first note that the rotation angle θ_i in (28) can be determined from the entries of $\mathbf{D}_i$. We can then calculate the logarithm of $\mathbf{D}_i$ as

$$\mathbf{B}_i = \log(\mathbf{D}_i) = \begin{bmatrix} 0 & -\theta_i \\ \theta_i & 0 \end{bmatrix}, \tag{29}$$

Then we compute $\log(\mathbf{U}')$ according to [55] as

$$\log(\mathbf{U}') = \log \begin{bmatrix} \mathbf{D} & \mathbf{y} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{B} & \mathbf{V}\mathbf{y} \\ \mathbf{0} & 0 \end{bmatrix}, \tag{30}$$

where

$$\mathbf{B} = \log(\mathbf{D}) = \begin{bmatrix} \mathbf{B}_1 & & & \\ & \ddots & & \\ & & \mathbf{B}_p & \\ & & & \mathbf{0}_{n-2p} \end{bmatrix}, \tag{31}$$

and

$$\mathbf{V} = \mathbf{I}_n + \sum_{i=1}^{p} \left(-\frac{\theta_i}{2}\mathbf{B}_i + (1 - \frac{\theta_i \sin\theta_i}{2(1-\cos\theta_i)})\mathbf{B}_i^2 \right). \tag{32}$$

Algorithm 3 provides the psuedo-code for computing $\log(\mathbf{T})$.

Algorithm 3: Logarithm for matrices in $SE(d)$.

Input: Transform matrix $\mathbf{T} \in SE(d)$.
Output: $\hat{\mathbf{T}} = \log(\mathbf{T})$.

1 Compute the real Schur decomposition $(\mathbf{Q}, \mathbf{U})$ of $\mathbf{T}$;
2 Rearrange $\mathbf{Q}, \mathbf{U}$ to obtain matrices $\mathbf{Q}', \mathbf{U}'$ in Eq. (27);
3 Calculate $\log(\mathbf{U}')$ according to Eqs. (29)–(32);
4 $\hat{\mathbf{T}} = \mathbf{Q}' \log(\mathbf{U}')\mathbf{Q}'^T$;

APPENDIX B
SURROGATE FUNCTION FOR EQ. (13)

In this section, we show that the function in Eq. (15) is a surrogate function of the target function in Eq. (13) at the current transformation $\mathbf{R}^{(k)}, \mathbf{t}^{(k)}$. To simplify notation, we denote $D_i = D_i(\mathbf{R}, \mathbf{t})$ and $D_i^{(k)} = D_i(\mathbf{R}^{(k)}, \mathbf{t}^{(k)})$. By definition, since $\chi_\nu(D_i \mid D_i^{(k)})$ is a surrogate function for $\psi_\nu(D_i)$ at $D_i^{(k)}$, we have

$$\begin{aligned} \psi_\nu(D_i^{(k)}) &= \chi_\nu(D_i^{(k)} \mid D_i^{(k)}), \\ \psi_\nu(D_i) &\le \chi_\nu(D_i \mid D_i^{(k)}), \quad \forall\, D_i \neq D_i^{(k)}. \end{aligned} \tag{33}$$

Note that the function $x_i(\mathbf{R}, \mathbf{t}) = \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)}\|$ satisfies

$$D_i \le x_i(\mathbf{R}, \mathbf{t}), \quad D_i^{(k)} = x_i(\mathbf{R}^{(k)}, \mathbf{t}^{(k)}). \tag{34}$$

Moreover, $\chi_\nu(x \mid y)$ is a monotonically increasing function on $x \in [0, +\infty)$, which together with Eqs. (33) and (34) means that

$$\begin{aligned} \psi_\nu(D_i^{(k)}) = \chi_\nu(D_i^{(k)} \mid D_i^{(k)}) &= \chi_\nu(x_i(\mathbf{R}^{(k)}, \mathbf{t}^{(k)}) \mid D_i^{(k)}), \\ \psi_\nu(D_i) \le \chi_\nu(D_i \mid D_i^{(k)}) &\le \chi_\nu(x_i(\mathbf{R}, \mathbf{t}) \mid D_i^{(k)}). \end{aligned}$$

It shows that $\chi_\nu(x_i(\mathbf{R}, \mathbf{t}) \mid D_i^{(k)})$ is a surrogate function for $\psi_\nu(D_i)$ at $D_i^{(k)}$. Then substituting each term $\psi_\nu(D_i)$ in Eq. (13) with $\chi_\nu(x_i(\mathbf{R}, \mathbf{t}) \mid D_i^{(k)})$, we can see that Eq. (15) is a surrogate function at $(\mathbf{R}^{(k)}, \mathbf{t}^{(k)})$.

APPENDIX C
CHOICES OF $\nu_{\min}$

In this section, we explain the rationale for our choices of $\nu_{\min}$ for our robust ICP methods.

For the point-to-point method, our intention is to set $\nu_{\min}$ large enough such that point pairs with deviation due to difference in sampling locations will be included in the alignment step. For ease of discussion, we first assume that the target set has uniform sampling density, and the source set is sampled from a triangulated surface using the target set as vertices. Then within the overlapping region the distance from a point in $\mathbf{p} \in P$ to the set Q can be up to $E/\sqrt{3}$ where E is the distance between neighboring points within Q (e.g., when $\mathbf{p}$ lies at the center of an equilateral triangle T with edge length E from the triangulation of Q). In order to include $\mathbf{p}$ and its closest point into the alignment step, $\nu_{\min}$ should be no smaller than $1/3$ of the distance between them due to the three-sigma rule, i.e., $\nu_{\min} \ge E/(3\sqrt{3})$. In practice, the sampling density of Q may not be uniform. Therefore, we compute the representative distance $\overline{E}_Q$ between neighboring points in Q as explained in Section 4.3, and set $\nu_{\min} = \overline{E}_Q/(3\sqrt{3})$.

For the point-to-plane method, we first use the same assumption as the point-to-point method: the target set has uniform sampling density, and the source set is sampled from a surface triangulated from the target set. For a source point $\mathbf{p} \in P$, suppose its closest point in Q is $\mathbf{q}$. Then for

Such a point should be included into the alignment process. Recall that a point $\mathbf{p}_i \in P$ will be effectively excluded from the alignment step if the distance between its current transformed position and the point set Q is larger than 3ν. Therefore, to ensure the points in the overlapping region are included for alignment, $\nu_{\min}$ should be no smaller than $\frac{E}{3\sqrt{3}}$. In reality, the sampling density of the point sets may not be uniform, thus we adapt the above heuristics as follows. We first compute the median distance from each point $\mathbf{p}_i \in P$ to its six nearest neighbors in P, and take the median $\overline{E}_P$ of these median distance values across P. In the

same way, we compute a value $\overline{E}_Q$ for the set Q. Let $\mathbf{s} \in Q$ be the neighbor point of $\mathbf{q}$ that is the farthest away from the tangent plane at $\mathbf{q}$. Denote by $H_{\mathbf{q}}(\cdot)$ the distance to the tangent plane at $\mathbf{q}$. Then we must have $H_{\mathbf{q}}(\mathbf{p}) \leq \frac{1}{2}H_{\mathbf{q}}(\mathbf{s})$, otherwise $\mathbf{q}$ would not be the closest point to $\mathbf{p}$ in Q. Then order to include the pair $(\mathbf{p}, \mathbf{q})$ into the alignment step, we can set $\nu_{\min} \geq \frac{1}{6}H_{\mathbf{q}}(\mathbf{s}) \geq \frac{1}{3}H_{\mathbf{q}}(\mathbf{p})$. To handle non-uniform sampling of Q, we compute the representative distance $\overline{H}_Q$ to a neighboring point's tangent plane in Q, and set $\nu_{\min} = \overline{H}_Q/6$.

APPENDIX D
CALCULATION OF GRADIENT $\mathbf{J}^{(k)}$ IN EQ. (24)

In this section, we show how to calculate the gradient $\mathbf{J}^{(k)}$ at $\mathbf{T}^{(k)}$ in Eq. (24). We denotes the actual variables $\widetilde{\mathbf{T}} = [\boldsymbol{\delta}^T, \mathbf{u}^T]^T$, where $\boldsymbol{\delta} = [\delta_1, \delta_2, \delta_3]$, then the gradient of $B_i^{(k)}$ can be represented as

$$\mathbf{J}_i^{(k)} = \left[\frac{\partial B_i^{(k)}}{\partial \boldsymbol{\delta}^T}, \frac{\partial B_i^{(k)}}{\partial \mathbf{u}^T}\right]^T,$$

and

$$\begin{cases} \dfrac{\partial B_i^{(k)}}{\partial \delta_j} = \left(\dfrac{\partial B_i^{(k)}}{\partial \mathbf{R}} \circ \dfrac{\partial \mathbf{R}}{\partial \delta_j}\right)_{\text{sum}} + \left(\dfrac{\partial B_i^{(k)}}{\partial \mathbf{t}} \circ \dfrac{\partial \mathbf{t}}{\partial \delta_j}\right)_{\text{sum}}, \\ \dfrac{\partial B_i^{(k)}}{\partial \mathbf{u}_j} = \left(\dfrac{\partial B_i^{(k)}}{\partial \mathbf{t}} \circ \dfrac{\partial \mathbf{t}}{\partial \mathbf{u}_j}\right)_{\text{sum}}, \end{cases}$$

where $1 \leq j \leq 3$, $\circ$ is element-wise multiplication of two matrices, and $(\cdot)_{\text{sum}}$ is to add up each element of the matrix. According to Eq.(22), we can calculate

$$\frac{\partial B_i^{(k)}}{\partial \mathbf{R}} = \widehat{\mathbf{n}}_i^{(k)} \mathbf{p}_i^T, \quad \frac{\partial B_i^{(k)}}{\partial \mathbf{t}} = \widehat{\mathbf{n}}_i^{(k)}.$$

Then we compute the derivative of $(\mathbf{R}, \mathbf{t})$ about $\boldsymbol{\delta}$ and $\mathbf{u}$. According to [71], defining

$$\mathbf{a}^\wedge = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix},$$

where $\mathbf{a} = (a_1, a_2, a_3)^T$. Let $\|\boldsymbol{\delta}\|$ denates the ℓ_2-norm of $\boldsymbol{\delta}$. When $\|\boldsymbol{\delta}\| \neq 0$, according to [72], the derivatives of rotation matrix $\mathbf{R}$ about $\boldsymbol{\delta}$ is

$$\frac{\partial \mathbf{R}}{\partial \delta_j} = \frac{\delta_j \boldsymbol{\delta}^\wedge + (\boldsymbol{\delta} \times (\mathbf{I} - \mathbf{R})\mathbf{e}_j)^\wedge}{\|\boldsymbol{\delta}\|^2}\mathbf{R},$$

where $\mathbf{e}_j$ is the j-th vector of the standard basis in $\mathbb{R}^3$ and $\mathbf{I}$ is the identity matrix in $\mathbb{R}^{3\times 3}$. The translation in $se(3)$ can be represented as [71]

$$\mathbf{t} = \left((\mathbf{R} - \mathbf{I})\mathbf{u}^\wedge \boldsymbol{\delta} + \boldsymbol{\delta}\boldsymbol{\delta}^T \mathbf{u}\right)/\|\boldsymbol{\delta}\|^2.$$

We can compute the derivatives of $\mathbf{t}$ about $\boldsymbol{\delta}$ is

$$\frac{\partial \mathbf{t}}{\partial \boldsymbol{\delta}} = \frac{1}{\|\boldsymbol{\delta}\|^2}\left(\mathbf{M} + \boldsymbol{\delta}\mathbf{u}^T + \mathbf{D}\right) - \frac{2}{\|\boldsymbol{\delta}\|^4}\mathbf{M}\boldsymbol{\delta}^T\boldsymbol{\delta}$$

where

$$\mathbf{M} = (\mathbf{R} - \mathbf{I})\mathbf{u}^\wedge + \boldsymbol{\delta}^T\mathbf{u}\mathbf{I}$$

and the j-th column of $\mathbf{D}$ is

$$\mathbf{D}_j = \frac{\partial \mathbf{R}}{\partial \delta_j}\mathbf{u}^\wedge \boldsymbol{\delta}$$

And the derivatives of $\mathbf{t}$ about $\mathbf{u}$ is

$$\frac{\partial \mathbf{t}}{\partial \mathbf{u}} = \frac{1}{\|\boldsymbol{\delta}\|^2}\left((\mathbf{I} - \mathbf{R})\boldsymbol{\delta}^\wedge + \boldsymbol{\delta}\boldsymbol{\delta}^T\right).$$

When $\|\boldsymbol{\delta}\| = 0$,

$$\frac{\partial \mathbf{R}}{\partial \delta_j} = \mathbf{e}_j^\wedge, \quad \mathbf{t} = \mathbf{u}, \quad \frac{\partial \mathbf{t}}{\partial \delta_j} = \mathbf{0}, \quad \frac{\partial \mathbf{t}}{\partial \mathbf{u}} = \mathbf{I}.$$

APPENDIX E
SETTINGS OF EXPERIMENTS

We follow the default settings of the open-source implementations for each method, except for the following changes:

- ICP, ICP-*l* and AA-ICP: For a fair comparison, we use the same termination criteria as in Algorithm 1 for a fixed parameter ν: we terminate the solver if it reaches the maximum number of iterations (1000), or $\|\Delta\mathbf{T}\|_F^2 < 10^{-5}$, where $\Delta\mathbf{T}$ denotes the difference between the transformation from two consecutive iterations.
- Sparse ICP and Sparse ICP-*l*: For Sparse ICP, we choose $p = 0.8$ for the RGB-D SLAM dataset, and $p = 0.4$ for other experiments. For Sparse ICP-*l*, we choose $p = 0.4$ for all experiments.
- CPD: Due to the high computational cost of CPD, for any point cloud with more than 15000 points, we downsample it to 15000 points using farthest point sampling.
- GMM-Reg: The documentation of the implementation recommends downsampling a point cloud to 5000 points for better performance. Therefore, for any point cloud with more than 5000 points, we downsample it to 5000 points using farthest point sampling.
- Teaser++: The implementation incurs high memory consumption, and requires the source and target point clouds to have the same number of points. Therefore, we first use farthest point sampling to downsample 5000 points on any point cloud containing more than 5000 points. Afterwards, if the source and target point clouds contain different numbers of points, we downsample the point cloud with more points to the same number of points as the other. For synthesized data with a known noise level, we set the noise bound parameter according to the noise level.
- DCP: We train the model using the 10000 pairs of point clouds from the training set of the 3DMatch dataset. For both the training and test data, we downsample the point clouds to 1024 points using farthest point sampling.
- DGR: We use the two pre-tained models (trained with 3DMatch and KITTI, respectively) released by the authors to test our examples. For use the KITTI-based model on five outdoor datasets (Mountains, Gazebo in summer, Gazebo in winter, Wood in summer, Wood in winter) and a mixed dataset (Stairs) in the ETH dataset. For all other problems, we use the model trained with 3DMatch.

In Fig. 1, we count the number of iterations for each method as follows. For CPD, we count the iterations of the EM algorithm. For GMM-Reg method, we count the number of times for constructing the objective function. For Teaser++, we count the iterations for calculating the rotation matrix. For others, we count the number of times for the updating the corresponding points.

Juyong Zhang is an associate professor in the School of Mathematical Sciences at University of Science and Technology of China. He received the BS degree from the University of Science and Technology of China in 2006, and the PhD degree from Nanyang Technological University, Singapore. His research interests include computer graphics, computer vision, and numerical optimization. He is an associate editor of The Visual Computer.

Yuxin Yao is currently working toward the master's degree in the School of Mathematical Sciences, University of Science and Technology of China. Her research interests include computer graphics and 3D registration.

Bailin Deng is a lecturer in the School of Computer Science and Informatics at Cardiff University. He received the BEng degree in computer software (2005) and the MSc degree in computer science (2008) from Tsinghua University (China), and the PhD degree in technical mathematics (2011) from Vienna University of Technology (Austria). His research interests include geometry processing, numerical optimization, computational design, and digital fabrication. He is a member of the IEEE.