A Robust Loss for Point Cloud Registration

Zhi Deng[1] Yuxin Yao[1] Bailin Deng[2] Juyong Zhang[1*]
[1]University of Science and Technology of China [2]Cardiff University
{zhideng,yaoyuxin}@mail.ustc.edu.cn DengB3@cardiff.ac.uk juyong@ustc.edu.cn

Abstract

The performance of surface registration relies heavily on the metric used for the alignment error between the source and target shapes. Traditionally, such a metric is based on the point-to-point or point-to-plane distance from the points on the source surface to their closest points on the target surface, which is susceptible to failure due to instability of the closest-point correspondence. In this paper, we propose a novel metric based on the intersection points between the two shapes and a random straight line, which does not assume a specific correspondence. We verify the effectiveness of this metric by extensive experiments, including its direct optimization for a single registration problem as well as unsupervised learning for a set of registration problems. The results demonstrate that the algorithms utilizing our proposed metric outperforms the state-of-the-art optimization-based and unsupervised learning-based methods.

1. Introduction

Rigid registration aligns a source shape $\mathcal{S}$ with a target shape $\mathcal{T}$ by applying a rigid transformation $(\mathbf{R}, \mathbf{t})$, where $\mathbf{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is a translation vector. It is an important task in numerous applications such as 3D scene reconstruction and localization. The transformation is often computed by minimizing a function that measures the alignment error. In practice, the shapes are often represented as point clouds, and the alignment error is measured using a distance metric $D(\cdot,\cdot)$ evaluated between the points on the source surface and their corresponding points on the target surface:

$$h(\mathbf{R},\mathbf{t}) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{C}} D(\widetilde{\mathbf{x}},\mathbf{y}), \tag{1}$$

where $\mathcal{C}$ is the set of corresponding points between $\mathcal{S}$ and $\mathcal{T}$, and $\widetilde{\mathbf{x}}$ denotes the new position of $\mathbf{x}$ after the transformation. To perform registration in this way, we must first define the corresponding point. Many traditional methods such as the iterative closest point (ICP) algorithm [6] defines $\mathbf{y}_{\sigma(i)}$ as the current closest point to $\mathbf{x}_i$, which needs to be updated in each iteration along with the transformation. It is easy for such iterations to fall into a local optimal solution, especially when noises, outliers, and partial overlaps in the point clouds. Some methods [17, 33, 13, 50] compute local shape descriptors for some sample points, and find the corresponding point on the target surface by matching the descriptors. However, the ambiguity of these hand-crafted descriptors can make them challenging to match, especially for the point clouds with noises and outliers.

Besides point correspondence, another key component of the alignment error measure in Eq. (1) is the distance metric $D(\cdot,\cdot)$ between the corresponding pairs. Traditional ICP methods [6, 8] use the ℓ_2-norm of the point-to-point or point-to-plane distance as the metric, where $D(\mathbf{x}_i, \mathbf{y}_{\sigma(i)})$ is the squared Euclidean distance from $\mathbf{x}_i$ to $\mathbf{y}_{\sigma(i)}$ or to the tangent plane of $\mathcal{T}$ at $\mathbf{y}_{\sigma(i)}$. To accommodate noise, outliers, and partial overlaps, other methods [7, 5, 50, 48] applied a robust function to the distance values to disregard or down-weight erroneous corresponding pairs. Although such strategies are more robust against noise and partial overlaps, they still rely on the correct point correspondence to some extent.

In this work, we propose an alignment error metric that does not rely on accurate point correspondence. Our key idea is to intersect the source and target shapes with a random straight line that is uniformly distributed in their bounding sphere. We locate the intersection points from the source shape and the target shape, and use the distance between them as a proxy for the alignment error. We apply Welsch's function [15] to the distance values to obtain a robust measure, and compute its expected value as our alignment error metric. Different from traditional methods, our approach does not assume a specific correspondence rule while still attaining rich information about the alignment from multiple directions thanks to the uniform distribution of the straight lines. Using our metric, optimization-based registration is less susceptible to getting stuck at a local minimum and more likely to obtain a robust solution.

Recently, various deep learning-based approaches for rigid registration have been proposed [51, 41, 42, 16]. How-

*Corresponding author

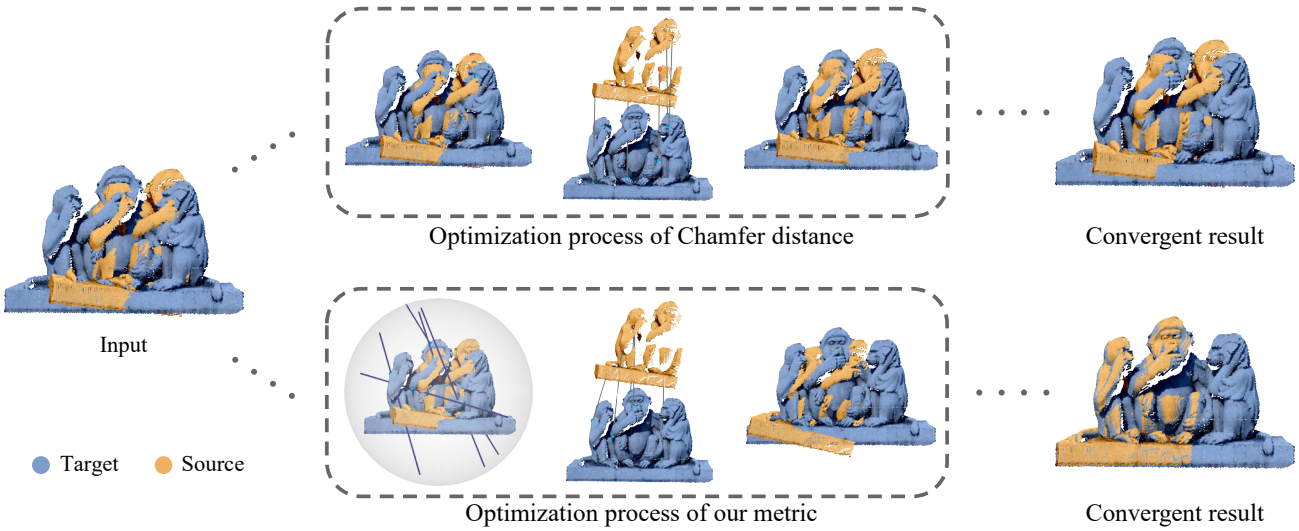


Figure 1. We propose an error metric for rigid registration based on the intersection between the input shapes and random straight lines that are uniformly distributed. Top: registration by minimizing an error metric based on the Chamfer distance leads to a sub-optimal result. Bottom: with our new metric, the optimization becomes more robust to the local minimum and identifies the correct alignment.

ever, most of them train the network in a supervised manner and require ground-truth alignment. Our proposed metric can also be used to replace the ground-truth labels and allow a supervised framework to be trained on unlabeled data. It also can fine-tune the model trained by a supervised metric for use on unlabeled datasets in the real world.

In summary, the main contributions of our work are:

- We propose a novel error metric for rigid alignment based on intersections between the input shapes and a uniform random straight line, which can improve the robustness of optimization-based rigid registration.
- We use the proposed metric to turn various supervised learning frameworks into unsupervised ones that can be trained on real unlabeled data.

2. Related works

Geometry Processing Using Line Intersection Intersection with straight lines has been utilized to process and analyze geometric shapes in the past. In [21, 22], the authors used intersections with random straight lines to compute surface areas of geometric shapes from the perspective of integral geometry [36]. In [31], a method was proposed to sample a point cloud by intersecting with uniformly distributed straight lines. To the best of our knowledge, our work is the first to perform shape registration using intersections with random straight lines.

Optimization-based Registration A classical registration method is Iterative Closest Point (ICP) algorithm [6], which obtains the optimal transformation by alternately finding the closest points and updating the transformation. Many variants of ICP have been proposed to improve its efficiency [8, 33, 28, 32]. Another issue of the classical ICP is its robustness to outliers and partial overlaps that often occur in real-world data. Some methods tackled this issue by disregarding some point pairs using heuristics based on their distance or normals [49, 33, 3]. Another popular approach is to use robust metrics such as the ℓ_p-norm ($p < 1$) [7] or Welsch's function [48] to measure the alignment error and improve robustness. Others solved the problem from a statistical perspective and aligned the point clouds via their Gaussian mixture representations [25, 18]. The above approaches formulate registration as an optimization problem and search for a local minimum using a numerical solver, which require proper initialization. Some other methods formulate a global optimization problem and solve it via either branch-and-bound [45] or semi-definite relaxation [24, 10, 20]. They are often computationally more expensive, especially on large-scale problems. Some methods align point clouds by matching their local shape descriptors [14, 34, 35]. However, the quality of such hand-craft descriptors can be affected by the point density and outliers.

Learning-based Registration Recently, various deep learning approaches have been proposed for registration. PointNetLK [4] uses an iterative framework which combines the PointNet feature [29] and Lucas-Kanade algorithm [23]. DCP [41] utilizes a sub-network to address difficulties in the classical ICP pipeline, which improves the point cloud's features by using DGCNN [43] to extract and merge local

features. RPM-Net [46] extracts the hybrid features by learning from spatial coordinates and local geometry, and uses the differentiable Sinkhorn layer and annealing to obtain soft correspondence. PR-Net [42] uses Gumbel–Softmax with straight-through gradient estimation to obtain a sharp and near-differentiable mapping function. MFG [40] combines the shape features and the spatial coordinates to guide correspondence search independently and fuse the matching results to obtain the full matching. DGR [9] uses a differentiable framework for pairwise registration of real-world 3D scans, adding an optimization module to fine-tune the alignment produced by the weighted Procrustes solver. All of the above approaches train their models in a supervised manner, which restricts their applications on real-world unlabeled data. Recently, FMR [16] takes a semi-supervised approach for point cloud registration, by minimizing a feature-metric projection error. In this paper, we propose a new alignment error metric that is suitable for unsupervised learning and achieves better results than the one used in FMR.

3. Algorithm

3.1. Problem Statement

Point cloud registration is generally posed as an optimization problem. Consider two points clouds $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^m$ on the source surface $\mathcal{S}$ and $\mathbf{Y} = \{\mathbf{y}_j\}_{j=1}^n$ on the target surface $\mathcal{T}$, where $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^3$ are the points. Let $\widetilde{\mathbf{X}} = \{\widetilde{\mathbf{x}}_i\}_{i=1}^m$ denote the deformed source point cloud with the rigid transformation $(\mathbf{R}, \mathbf{t})$, where

$$\widetilde{\mathbf{x}}_i = \mathbf{R}\mathbf{x}_i + \mathbf{t}.$$

Using the alignment error given in Eq. (1), ICP-based methods can be described as

$$(\mathbf{R}^*, \mathbf{t}^*) = \arg\min_{(\mathbf{R},\mathbf{t})} \sum_{\mathbf{x}_i \in \mathbf{X}} D(\widetilde{\mathbf{x}}_i, \mathbf{y}_{\sigma(i)}),$$

where $\sigma(i)$ denotes the index of the corresponding point in $\mathbf{Y}$ for the point $\mathbf{x}_i \in \mathbf{X}$. The above formulation only considers the distance from the source point cloud to the target point cloud. Considering the distance from the target to the source as well, the Chamfer distance has also been used to measure the deviation between two point clouds [11, 16]. The alignment error based on the Chamfer distance can be written as:

$$h(\widetilde{\mathbf{X}}, \mathbf{Y}) = \sum_{\mathbf{x}_i \in \mathbf{X}} D(\widetilde{\mathbf{x}}_i, \mathbf{y}_{\sigma(i)}) + \sum_{\mathbf{y}_j \in \mathbf{Y}} D(\widetilde{\mathbf{x}}_{\rho(j)}, \mathbf{y}_j), \quad (2)$$

where $\rho(j)$ denotes the index of the corresponding point in $\mathbf{X}$ for the point $\mathbf{y}_j \in \mathbf{Y}$. The choices of $\sigma(\cdot)$ and $\rho(\cdot)$ can affect the quality of registration. In ICP-based methods, $\mathbf{y}_{\sigma(i)}$ is chosen to be the closest point to $\mathbf{x}_i$. But such closest-point correspondence is often incorrect when there is large misalignment or a low overlap ratio between the two point clouds. Therefore, we would like to use an alignment error that does not presume a pre-defined rule of point correspondence while still being effective in guiding the alignment. Our key observation is that for two shapes that are perfectly aligned, any straight line that intersects with one shape will also intersect the other shape at the same points. When the two shapes are close, their intersection points with the same line will also be close to each other. Moreover, if we use a set of random straight lines to intersect with the two shapes, then the intersection points along each line can inform us about the difference between the two shapes from a particular viewpoint along a view ray that corresponds to the line. In the past, such random straight lines have been utilized in integral geometry to determine geometric properties of a given shape such as surface area [21, 22]. In the following, we propose an alignment error metric based on the intersection with random straight lines.

3.2. Error Metric Based on Line Intersection

To measure the alignment error between a source shape $\mathcal{S}$ and a target shape $\mathcal{T}$, our basic idea is to intersect both shapes with a set of random straight lines with a uniform distribution, and compare the intersection points along each line. Specifically, given a straight line l that intersects with both shapes, we denote the set of intersection points with the source shape and the target shape as $\mathcal{S}_l = \{\mathbf{x}_i^l\}$ and $\mathcal{T}_l = \{\mathbf{y}_j^l\}$, respectively. Then we measure the deviation between the two sets of intersection points as:

$$F_l(\mathcal{S}, \mathcal{T}) = w_l \left(\sum_{\mathbf{x}_i^l \in \mathcal{S}_l} D(\mathbf{x}_i^l, \mathbf{y}_{\sigma_l(i)}^l) + \sum_{\mathbf{y}_j^l \in \mathcal{T}_l} D(\mathbf{x}_{\rho_l(j)}^l, \mathbf{y}_j^l) \right), \quad (3)$$

where D is an error metric that will be explained later, and

$$\sigma_l(i) = \arg\min_k \|\mathbf{x}_i^l - \mathbf{y}_k^l\|, \quad \rho_l(j) = \arg\min_k \|\mathbf{x}_k^l - \mathbf{y}_j^l\|,$$

i.e., $\mathbf{y}_{\sigma_l(i)}^l$ is the closest point in $\mathcal{T}_l$ to $\mathbf{x}_i^l$, and $\mathbf{x}_{\rho_l(j)}^l$ is the closest point in $\mathcal{S}_l$ to $\mathbf{y}_j^l$. The weight w_l is defined as $w_l = \exp(-|\frac{|\mathcal{S}_l| - |\mathcal{T}_l|}{2}|)$. This reduces the weight for a line with a large difference between the numbers of its intersection points with the two shapes, which may indicate an erroneous correspondence between them along the line. Finally, the alignment error between $\mathcal{S}$ and $\mathcal{T}$ is defined as the expected value of $F_l(\mathcal{S}, \mathcal{T})$ over the distribution of the lines:

$$h(\mathcal{S}, \mathcal{T}) = E(F_l(\mathcal{S}, \mathcal{T})). \quad (4)$$

To apply this in point cloud registration, we evaluate the error metric in Eq. (5) for the transformed source point cloud $\widetilde{\mathbf{X}}$ and the target point cloud $\mathbf{Y}$, and use it as the target function for an optimization-based method or as a loss function term for a learning-based approach. In the following, we present the details for evaluating the error metric on point clouds.

Choice of D in Eq. (3) With Eq. (3), we effectively establish correspondence between points on the source and target shapes along a straight line. However, since the line is chosen randomly, the correspondence may be inaccurate. Therefore, we define D to be a robust metric to alleviate the impact of inaccurate correspondence as well as outliers. We choose Welsch's function as the metric:

$$D(\mathbf{x}, \mathbf{y}) = \psi_\nu(\|\mathbf{x} - \mathbf{y}\|_2), \tag{5}$$

where $\psi_\nu(x) = 1 - \exp(-\frac{x^2}{2\nu^2})$, and $\nu > 0$ is a parameter. To take into account the scale of input point clouds, we set $\nu = \nu_0 d_{\text{med}}$, where d_{med} is the median distance between all corresponding point pairs and ν_0 is a user-specified parameter. We choose ν_0 in all experiments. We treat ν as a constant term during optimization/training and do not evaluate/back-propagate its gradient. The value d_{med} is updated in each iteration according to the latest alignment.

Generation of Random Straight Lines Following [22], we first compute a bounding sphere S_r that covers both the source and the target point clouds. Then we sample two independent uniformly distributed points on S_r and connect them to generate a random straight line. Each point on the sphere can be parameterized as:

$$S_r(u, \alpha) = (r\sqrt{1-u^2}\cos\alpha, r\sqrt{1-u^2}\sin\alpha, ru),$$

where r is the radius, $u \in [-1, 1]$, and $\alpha \in [0, 2\pi)$. The random points on the sphere are generated by uniformly sampling the parameters u and α in their domains [22]. In each iteration of optimization or training, we use this approach to generate 15000 straight lines.

Line Intersection with Point Clouds Since a point cloud contains discrete samples of the underlying shape, a straight line that intersects with the underlying shape does not necessarily intersect with the points in the point cloud. Therefore, we use the following steps to approximate the intersection between the straight line and the underlying shape (see Fig. 2). First, similar to [22], we enlarge the line into a cylinder that is centered at the line and has radius δ, and include all the points contained within the cylinder as candidate points for the intersection (Fig. 2 (b)). Then for each candidate point $\mathbf{p}_0$ whose k-nearest neighbors in the point cloud are also candidate points, we compute a convex combination of $\mathbf{p}_0$ and it's k-nearest neighbors as an intersection point (Fig. 2 (c)):

$$\mathbf{p}_0' = \frac{\sum_{\mathbf{p}\in\mathcal{N}(\mathbf{p}_0)} d_{\mathbf{p}}\mathbf{p}}{\sum_{\mathbf{p}\in\mathcal{N}(\mathbf{p}_0)} d_{\mathbf{p}}}, \tag{6}$$

where the set $\mathcal{N}(\mathbf{p}_0)$ contains $\mathbf{p}_0$ and its k-nearest neighbors, and $d_{\mathbf{p}}$ is the distance from the point $\mathbf{p}$ to the line. In our implementation, we set $k = 2$, and choose $\delta = \frac{\sqrt{3}}{2} d_{\text{nei}}$ where d_{nei} is the average distance across the whole point cloud between a point and its k-nearest neighbors.

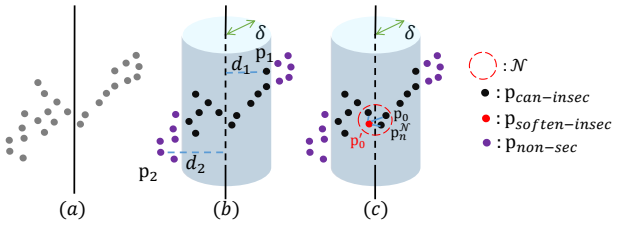


Figure 2. Illustration of the process of generating the intersections between the point cloud and a straight line. (a) shows the local point cloud and the straight line; (b) shows the selection of the candidate points (black dots); (c) shows the process of softening the candidate points to obtain the final intersections (red dot).

4. Experiments

In this section, we apply our alignment error metric to different registration problems, include both optimization-based and learning-based methods, and test it on on both synthetic and real datasets to verify its effectiveness.

4.1. Datasets

Synthetic Datasets Our synthetic datasets are generated from the ModelNet40 dataset [44] and the Axyz-pose human dataset [1]. The ModelNet40 dataset contains CAD models from 40 artificial object categories. We select 625 cases from the Airplane category to construct an Airline dataset, randomly sampling 500 cases for training and using the remaining 125 for testing. The Axyz-pose human dataset contains 110 clothed human mesh models. We randomly choose 110 models to construct a Human dataset, using 100 models for training set and the remaining 10 for testing.

Using the datasets above, we generate point cloud pairs for training and testing. To make the generated point clouds similar to the type of data captured by an RGBD camera, we use the following steps to generate the data. Firstly, we sample a complete model from different perspectives to generate partially overlapping data. Specifically, we choose a certain axis and rotate an imaginary camera around it to derive N camera locations with rotation angles at regular intervals (we set N to 50 for the Human dataset and 18 for the Airplane dataset). We then save the visible part of the model from each camera location as a source point cloud. For each source point cloud of the Airplane dataset, we rotate the camera by a random angle in the range $[-75°, 75°]$ with respective to a random axis, and save the visible part as the corresponding target point cloud. For the Human dataset, we use the whole model to construct the target point cloud. Then we scale each point cloud pair to be contained within $[-1, 1]^3$. Secondly, we generate composite transformations between the source and target point clouds. We follow [41] to generate rotations by sampling three Euler angle rotations in the range $[0, 45°]$ and translations in the range $[-0.2, 0.2]$

Table 1. Comparison between different optimization-based methods on the Human dataset [1].

Method	$\text{Err}_{\mathbf{R}}$ (degrees)	$\text{Err}_{\mathbf{t}}$ $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)	Err_{pw} $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)
ICP [6]	10.015	0.139, 0.093	0.112, 0.082
FRICP [48]	6.001	0.096, 0.064	0.074, 0.054
FGR [50]	46.31	0.411, 0.274	0.616, 0.41
CD	5.863	0.148, 0.132	0.151, 0.14
CD-W ($\nu_0 = 0.5$)	4.84	0.086, 0.078	0.108, 0.087
Ours	**0.576**	**0.017, 0.013**	**0.018, 0.015**

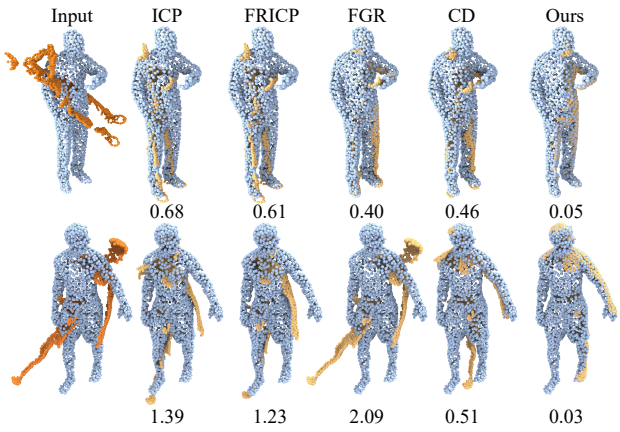


Figure 3. Comparison of registration results on the Human dataset [1] using different optimization-based methods.

on each axis. In total, the Airplane dataset contains 9000 pairs for training and 2250 for testing, while the Human dataset contains 5000 pairs for training and 500 for testing. For each point cloud, we use PCL to compute the point normals [2], and use FPS [30] to sample 1024 points.

Real Dataset To test our metric on unlabeled data, we also construct a real dataset based on the 3D-Match dataset [47], the 7scenes dataset [37] and the RGB-D SLAM dataset [39]. Inevitably, our metric cannot handle point cloud pairs with arbitrary pose differences and overlap ratios. And in practice, it is uncommon to have extremely large differences in poses or extremely small overlap ratios. Therefore, we select point cloud pairs separated by 20 frames from the RGB-D SLAM dataset and the 7scenes dataset, respectively. For the 3D-Match dataset, we collect the pre-processed data pairs from [9][1] where the overlap ratio is greater than 70%. All point cloud pairs are scaled into $[-5, 5]^3$. Finally, the real dataset is divided into 8000 pairs for training and 2000 pairs for testing. Similar to the synthetic dataset, we compute the point normals and sample 2048 points for each point cloud.

Evaluation Criteria We evaluate the registration accuracy on a point cloud pair using the isotropic rotation error $\text{Err}_{\mathbf{R}}$ and translation error $\text{Err}_{\mathbf{t}}$ inspired by [46], as well as the pointwise error Err_{pw}:

$$\begin{aligned} &\text{Err}_{\mathbf{R}} = \angle(\mathbf{R}_{\text{GT}}^{-1}\hat{\mathbf{R}}), \quad \text{Err}_{\mathbf{t}} = \|\mathbf{t}_{\text{GT}} - \hat{\mathbf{t}}\|_*, \\ &\text{Err}_{\text{pw}} = \frac{1}{|\mathbf{X}|}\sum_{\mathbf{x}_i \in \mathbf{X}} \|\mathbf{R}_{\text{GT}}\mathbf{x}_i + \mathbf{t}_{\text{GT}} - \hat{\mathbf{R}}\mathbf{x}_i - \hat{\mathbf{t}}\|_*, \end{aligned} \tag{7}$$

where $\mathbf{R}_{\text{GT}}$ and $\mathbf{t}_{\text{GT}}$ are the ground-truth rotation and translation respectively, $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ are the computed rotation and translation respectively, $\angle(\mathbf{A}) = \arccos(\frac{tr(\mathbf{A})-1}{2})$ is the angle of the rotation matrix $\mathbf{A}$ in degrees, $|\mathbf{X}|$ is the number of points in the source point cloud $\mathbf{X}$, and $\|\cdot\|_*$ is either the ℓ_1-norm or the ℓ_2-norm. We use the mean values of these metrics to measure the performance of a method on a benchmark dataset. For the figures in this section and the supplementary material, the number under each result is the pointwise error with the ℓ_2-norm.

4.2. Effectiveness of Our Metric

Comparison with Optimization-based Methods We optimize the Lie algebraic representation of rigid transformation with our metric as the target function using the Adam optimizer [19] in Pytorch [27]. Using the Human test dataset as the benchmark, we compare our results with other optimization-based methods, including ICP [6], FRICP [48] and FGR [50] with their open-source implementations [2][3]. We also compare with two optimization approaches that use the Chamfer distance in Eq. (2) as the target function, with the metric D chosen to be the Euclidean distance (denoted as CD) and Welsch's function in Eq. (5) (denoted as CD-W, see also the supplementary material for more details), respectively. Tab. 1 shows the performance of different methods on the Human test dataset. We can see our proposed metric can generate more accurate results than other traditional optimization methods. Fig. 3 shows some examples of registration results from different methods, where other methods converge to sub-optimal a local minimum while our metric leads to more accurate alignment. Fig. 4 and Fig. 5 further illustrate the effectiveness of our metric in avoiding local minimum. In Fig. 4, we take the convergent results of other optimization-based methods as initialization for optimization using our metric as the target function. The plot of pointwise ℓ_2 error shows that our approach can often further reduce the alignment error, which indicates its capability to escape

[1] https://github.com/chrischoy/DeepGlobalRegistration

[2] https://github.com/yaoyx689/Fast-Robust-ICP

[3] https://github.com/intel-isl/FastGlobalRegistration

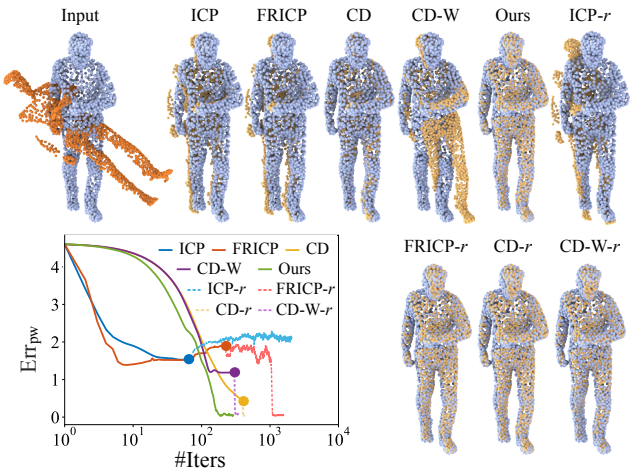
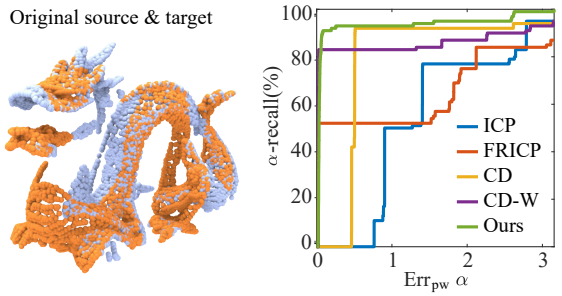


Figure 4. We take convergent results of different optimization-based methods (shown as solid dots in the ℓ_2 Err_{pw}-plot) to initialize a minimization of our metric (denoted as $*$-r where $*$ is the original method). In three out of four cases, our optimization escapes from the local minima of the original method and further reduces Err_{pw}.



Method	ICP	FRICP	CD	CD-W	Ours
Mean Err_{pw}	1.51	1.22	0.80	0.50	**0.20**

Figure 5. α-recall rates and mean ℓ_2-Err_{pw} from different optimization-based methods on a pair of point clouds with 100 random initial alignments. Optimization using our metric is less sensitive to initialization.

from a local minimum of other methods. Fig. 5 demonstrates the robustness of our approach to the initial alignment. We use a point cloud pair generated from the Dragon model in the Stanford 3D Scanning Repository[4], and apply a random transformation to one of them to derive 100 different initial alignments. Then we perform registration using different methods and compare their α-recall rates $|S_\alpha|/|S|$ for different α values, where $|S|$ is the total number of test cases and $|S_\alpha|$ is the number of test cases where the pointwise ℓ_2 error is less than α [50] (i.e., for a given α, a larger α-recall rate indicates better performance). The α-recall plot as well as the mean ℓ_2 pointwise error show that our method is less sensitive to initialization than other methods.

[4] http://graphics.stanford.edu/data/3Dscanrep/

Table 2. Optimization using our metric with different settings on the Human dataset [1], including an alternative line intersection method (Insec1), two alternative line sample methods (Sample1 and Sample2), and different values of ν_0.

Method	$\text{Err}_{\mathbf{R}}$ (degrees)	$\text{Err}_{\mathbf{t}}$ $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)	Err_{pw} $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)
Insec1 ($\nu_0 = 0.5$)	49.27	0.831, 0.744	1.175, 0.772
Sample1	20.482	0.274, 0.175	0.419, 0.276
Sample2	5.201	0.151, 0.093	0.117, 0.075
$\nu_0 = 100$	7.786	0.289, 0.191	0.292, 0.195
$\nu_0 = 10$	8.181	0.288, 0.190	0.291, 0.194
$\nu_0 = 1$	2.814	0.044, 0.029	0.051, 0.034
$\nu_0 = 0.01$	10.814	0.326, 0.217	0.334, 0.223
Ours ($\nu_0 = 0.5$)	**0.576**	**0.017, 0.011**	**0.018, 0.012**

Ablation Studies In Tab. 2, we use the Human test dataset as the benchmark to verify the effectiveness of different components of our metric. The first row shows that the use of convex combination for computing the intersection point in Eq. (6) is important. Here Insec1 denotes an alternative approach where we simply take all the candidate points as the final intersection points, which leads to worse performance than our approach (shown in the last row). The second and the third rows show two alternative sampling approaches (Sample1 and Sample2) for the random straight lines. For Sample1, we uniformly sample a point in the bounding box and uniformly sample a direction, and construct a line that goes through the sampled point along the sampled direction. For Sample2, we sample a point uniformly from the source and the target point cloud, respectively, and make a uniformly small perturbation, and connect them to obtain a straight line. Both approaches are outperformed by our sampling method (the last row). The fourth to the seventh rows show the impact of different ν values on our metric. For Welsch's function, a larger ν makes it closer to the ℓ_2-norm, whereas a smaller ν makes it closer to the ℓ_0-norm. The results show that a choice of ν_0 close to 0.5 is suitable.

4.3. Results for Unsupervised Learning

We also use the proposed metric for deep learning-based registration. Specifically, we replace the alignment term in the loss in the frameworks of DCP [41], FMR [16], and RPM-Net [46], and train them in an unsupervised manner. We compare the results with the original frameworks trained on the same data with ground-truth alignment labels (denoted as DCP-GT, FMR-GT and RPM-GT respectively). For comparison, we also include unsupervised variants of DCP and RPM-Net using the ℓ_2 Chamfer distance as the align-

Table 3. Comparison between different optimization-based and learning-based methods on the Airplane dataset [44].

Method	$\text{Err}_{\mathbf{R}}$ (degrees)	$\text{Err}_{\mathbf{t}}$ $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)	Err_{pw} $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)
ICP [6]	7.223	0.131, 0.087	0.136, 0.105
FRICP [48]	6.91	0.076, 0.051	0.123, 0.094
FGR [50]	13.72	0.099, 0.065	0.156, 0.114
DCP-GT	2.281	0.067, 0.044	0.073, 0.05
DCP-CD	6.612	0.165, 0.11	0.198, 0.139
DCP-Ours	**3.808**	**0.082, 0.056**	**0.093**, **0.063**
FMR-GT	1.977	0.086, 0.055	0.099, 0.065
FMR-CD	5.819	0.215, 0.153	0.247, 0.19
FMR-Ours	**2.51**	**0.117, 0.076**	**0.134, 0.091**
RPM-GT	2.19	0.041, 0.034	0.043,0.03
RPM-CD	2.791	0.103, 0.089	0.102,0.083
RPM-Ours	**1.673**	**0.042, 0.034**	**0.045, 0.031**

Table 4. Comparison between different learning-based methods on the Human dataset [1].

Method	$\text{Err}_{\mathbf{R}}$ (degrees)	$\text{Err}_{\mathbf{t}}$ $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)	Err_{pw} $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)
DCP-GT	3.841	0.061, 0.039	0.068, 0.046
DCP-CD	7.021	0.185, 0.114	0.193, 0.130
DCP-Ours	**4.841**	**0.07, 0.046**	**0.079**, **0.054**
FMR-GT	2.122	0.058, 0.039	0.064, 0.043
FMR-CD	6.207	0.187, 0.123	0.228, 0.134
FMR-Ours	**1.521**	**0.089, 0.051**	**0.091, 0.063**
RPM-GT	1.921	0.030, 0.021	0.033, 0.023
RPM-CD	8.373	0.197, 0.16	0.193,0.133
RPM-Ours	**1.33**	**0.032, 0.024**	**0.034, 0.023**

ment term, as well as the semi-supervised version of FMR from [16] (denoted as DCP-CD, RPM-CD and FMR-CD respectively). For all frameworks, we replace the batch normalization with the group normalization for better-unsupervised training. For the unsupervised variants of RPM-Net, we set the weights of the regularization term and the alignment term to 10 and 1 respectively, and the learning rate to 2×10^{-6}. For the unsupervised variants of DCP, we set the weights of the cycle term and the alignment term to 0.01 and 1.0 respectively, and the learning rate to 10^{-5}. For the unsupervised variant of FMR, we set the weights of the encoder term and the alignment term to 0.001 and 1 respectively, and the learning rate to 10^{-5}. All frameworks are trained using the Adam optimizer from Pytorch for 50 epochs, on a workstation with two Intel Xeon Silver 4110 CPUs at 2.10 GHz, and four Tesla V100 GPUs.

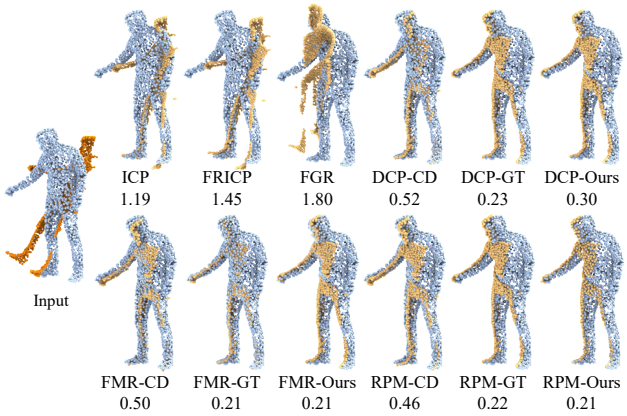


Figure 6. Examples of registration results using different methods on the Human dataset [1].

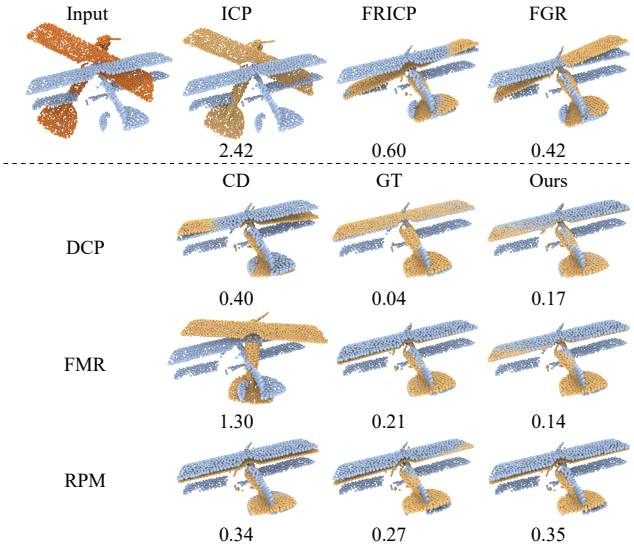


Figure 7. Examples of registration results using different methods on the Airplane dataset [44].

Different from supervised learning, training an unsupervised model often requires suitable initialization [12]. We use the following steps to derive initialization for the synthetic datasets. During preprocessing, we first generate an easier dataset with 100 data pairs and smaller pose differences between the source and target point clouds, and train the model on them for 500 epochs to obtain an overfit model. Then we train the model on 10% of the training dataset. Finally, we train the model on the whole training dataset with a reduced learning rate. For the real dataset, we generate an easier dataset consisting of pairs that are separated by a smaller number of frames during preprocessing, while the remaining training process is the same as the synthetic datasets.

Tab. 3 and Tab. 4 show the performance results on the

Table 5. Comparison between different optimization-based and unsupervised learning methods on the real dataset.

Method	$\text{Err}_{\mathbf{R}}$ (degrees)	$\text{Err}_{\mathbf{t}}$ $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)	Err_{pw} $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)
ICP [6]	17.98	0.337, 0.23	0.238, 0.194
FRICP [48]	11.08	0.199, 0.139	0.151, 0.112
FGR [50]	12.79	0.211, 0.142	0.260, 0.21
FMR-CD	7.559	0.469, 0.34	0.531, 0.384
FMR-Ours	**3.263**	**0.089, 0.065**	**0.101**, **0.075**
RPM-CD	11.28	0.342, 0.246	0.376, 0.272
RPM-Ours	**2.972**	**0.057, 0.04**	**0.068, 0.05**

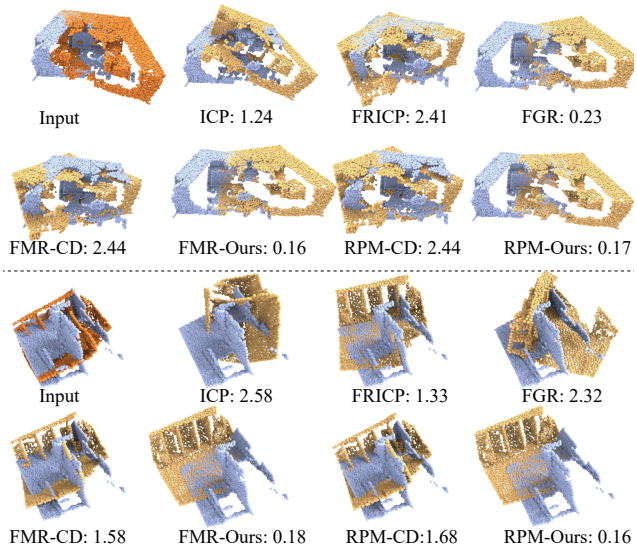


Figure 8. Comparison of different methods on the Real dataset with deep learning frameworks with different metrics.

Airplane dataset and the Human dataset, respectively. They show that our metric is suitable for unsupervised deep learning frameworks, with superior performance compared to unsupervised variants with Chamfer distance, and even better performance than the supervised versions in some cases. Especially for FMR, using our metric to their original unsupervised framework greatly improves the performance. Figs. 6 and 7 show registration results using different methods on two problems from the Human dataset and the Airplane dataset respectively. For both problems, optimization-based methods converge to local minima. Unsupervised learning approaches using our metric produce much better alignment than their counterparts using Chamfer distance.

Tab. 5 and Fig. 8 compare the performance of different methods on our real dataset. Due to the lack of ground-truth alignment labels, supervised learning approaches are no longer applicable. As Fig. 8 shows that ICP-based methods [48, 6] are prone to local minima due to sensitivity to initial values, whereas the feature-based method of [50] performs poorly due to the noisy normals in real data. Unsupervised learning using our metric is more robust than the Chamfer distance and performs the best in this benchmark.

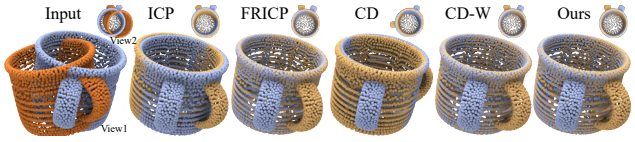


Figure 9. A failure case of optimization using our metric, as well as results from other methods.

5. Conclusion and Future Work

We have proposed a novel metric for point cloud registration. The main contributions of our work are two aspects. First, the proposed metric is based on intersections of uniformly random straight lines set in space, which can obtain richer information and more likely to achieve the global minimum. Second, our proposed metric can turn various supervised learning frameworks into unsupervised and has the ability to train on massive real unlabeled suitable data sets. Extensive ablation studies have verified the effectiveness of each component of our metric. Experiments on synthetic and real datasets show that our metric is competitive compared to the previous metrics and can be used in the loss function of deep learning frameworks.

Fig. 9 shows a failure case for our metric: after the symmetric bodies of two point clouds of a mug are aligned, our method is unable to align the handles. This is because within a set of random sample straight lines, only a small number of them will hit the handles, and their alignment effect will be dominated by other lines that tend to maintain the current alignment between the mug bodies.

In the future, a possible avenue of research is to further investigate why the proposed metric can achieve better performance. Our conjecture is that the intersection with random straight lines introduce randomness to the optimization process and help the solver escape from local minima. This has been observed in our experiments, but will need more rigorous investigation to verify and understand. Another potential direction is to use relevant mathematical theories such as integral geometry to interpret our metric, which can be a challenging and interesting future work.

Acknowledgement This work was supported by NSFC (No. 62122071), the Youth Innovation Promotion Association CAS (No. 2018495), "the Fundamental Research Funds for the Central Universities" (No. WK3470000021), and Guangdong International Science and Technology Cooperation Project (No. 2021A0505030009).

References

[1] Human datasets. https://secure.axyz-design.com// Accessed March 4, 2021. 4, 5, 6, 7, 11

[2] Point cloud library. https://github.com/PointCloudLibrary/pcl// Accessed March 4, 2021. 5

[3] Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. *Image and Vision Computing*, 23(3):299–309, 2005. 2

[4] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust and efficient point cloud registration using pointnet. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[5] Per Bergström and Ove Edlund. Robust registration of point sets using iteratively reweighted least squares. *Computational optimization and applications*, 58(3):543–561, 2014. 1

[6] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 1, 2, 5, 7, 8

[7] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse iterative closest point. *Computer Graphics Forum (Symposium on Geometry Processing)*, 32(5):1–11, 2013. 1, 2

[8] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992. 1, 2

[9] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 5

[10] Nadav Dym, Haggai Maron, and Yaron Lipman. DS++: a flexible, scalable and provably tight relaxation for matching problems. *ACM Transactions on Graphics (TOG)*, 36(6):184:1–184:14, 2017. 2

[11] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2463–2471, 2017. 3

[12] Wanquan Feng, Juyong Zhang, Hongrui Cai, Haofei Xu, Junhui Hou, and Hujun Bao. Recurrent multi-view alignment network for unsupervised surface registration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 7

[13] Andrew Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21:1145–1153, 04 2002. 1

[14] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, and Helmut Pottmann. Robust global registration. In *Third Eurographics Symposium on Geometry Processing, Vienna, Austria, July 4-6, 2005*, volume 255, pages 197–206. Eurographics Association, 2005. 2

[15] Paul W. Holland and Roy E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9):813–827, 1977. 1, 11

[16] Xiaoshui Huang, Guofeng Mei, and Jian Zhang. Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 3, 6, 7, 13

[17] X. Huang, J. Zhang, L. Fan, Q. Wu, and C. Yuan. A systematic approach for cross-source point cloud registration by preserving macro and micro structures. *IEEE Transactions on Image Processing*, 26(7):3261–3276, 2017. 1

[18] Bing Jian and Baba C Vemuri. Robust point set registration using gaussian mixture models. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1633–1645, 2010. 2

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 5

[20] Huu M Le, Thanh-Toan Do, Tuan Hoang, and Ngai-Man Cheung. Sdrsac: Semidefinite-based randomized approach for robust point cloud registration without correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 124–133, 2019. 2

[21] Xueqing Li, Wenping Wang, Ralph R. Martin, and Adrian Bowyer. Using low-discrepancy sequences and the crofton formula to compute surface areas of geometric models. *Comput. Aided Des.*, 35(9):771–782, 2003. 2, 3

[22] Yu-Shen Liu, Jun-Hai Yong, Hui Zhang, Dong-Ming Yan, and Jia-Guang Sun. A quasi-monte carlo method for computing areas of point-sampled surfaces. *Computer-Aided Design*, 38(1):55–68, 2006. 2, 3, 4

[23] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In Patrick J. Hayes, editor, *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679. William Kaufmann, 1981. 2

[24] Haggai Maron, Nadav Dym, Itay Kezurer, Shahar Kovalsky, and Yaron Lipman. Point registration via efficient convex relaxation. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016. 2

[25] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010. 2

[26] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006. 11

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-Workshop*. 2017. 5, 11, 13

[28] Artem L Pavlov, Grigory WV Ovchinnikov, Dmitry Yu Derbyshev, Dzmitry Tsetserukou, and Ivan V Oseledets. Aa-icp: Iterative closest point with anderson acceleration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3407–3412. IEEE, 2018. 2

[29] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2

[30] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on

point sets in a metric space. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 5099–5108, 2017. 5

[31] Jordi Rovira, Peter Wonka, Francesc Castro, and Mateu Sbert. Point sampling with uniformly distributed lines. In *2nd Symposium on Point Based Graphics, PBG 2005, Stony Brook, NY, USA, June 21-22, 2005*, pages 109–118. Eurographics Association, 2005. 2

[32] Szymon Rusinkiewicz. A symmetric objective function for icp. *ACM Trans. Graph.*, 38(4), 2019. 2

[33] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *3rd International Conference on 3D Digital Imaging and Modeling (3DIM 2001), 28 May - 1 June 2001, Quebec City, Canada*, pages 145–152. IEEE Computer Society, 2001. 1, 2

[34] Radu Rusu, Nico Blodow, Zoltan Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. pages 3384–3391, 09 2008. 2

[35] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *IEEE International Conference on Robotics and Automation*, pages 3212–3217, May 2009. 2

[36] Luis A. Santaló and Mark Kac. *Integral Geometry and Geometric Probability*. Cambridge Mathematical Library. Cambridge University Press, 2 edition, 2004. 2

[37] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2013. 5

[38] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1):1–5, 2017. 13

[39] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. 5

[40] Hongyuan Wang, Xiang Liu, Wen Kang, Zhiqiang Yan, Bingwen Wang, and Qianhao Ning. Multi-features guidance network for partial-to-partial point cloud registration. *CoRR*, abs/2011.12079, 2020. 3

[41] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 4, 6, 12, 13

[42] Yue Wang and Justin M. Solomon. Prnet: Self-supervised learning for partial-to-partial registration. In *33rd Conference on Neural Information Processing Systems (To appear)*, 2019. 1, 3

[43] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019. 2

[44] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 4, 7

[45] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2241–2254, 2016. 2

[46] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 5, 6, 12

[47] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5, 11

[48] J. Zhang, Y. Yao, and B. Deng. Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, jan 2021. 1, 2, 5, 7, 8, 12, 13

[49] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13:119–152, 1994. 2

[50] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, volume 9906 of *Lecture Notes in Computer Science*, pages 766–782. Springer, 2016. 1, 5, 6, 7, 8

[51] T. Zodage, R. Chakwate, V. Sarode, R. A. Srivatsan, and H. Choset. Correspondence matrices are underrated. In *2020 International Conference on 3D Vision (3DV)*, pages 603–612, 2020. 1

Supplementary Material

This supplementary material shows some details that were not given in the paper due to constraints of space, including adding an experiment about the variants of Chamfer distance based on Welsch's function [15], more visualization of our comparison.

Comparison with Welsch's Chamfer Distance

As a supplement, we also add an experiment, which compares with variants of the chamfer distance with robust Welsch's function.

$$f(\widetilde{\mathcal{S}}_{(\mathbf{R},\mathbf{t})},\mathcal{T})_{welsch} = \sum_{(\mathbf{x},\mathbf{y})\in C} D_{wel}(\mathbf{R}\mathbf{x}+\mathbf{t},\mathbf{y}),$$

$$D_{wel}(\mathbf{x},\mathbf{y}) = \psi_\nu(\|\mathbf{x}-\mathbf{y}\|_2),$$

where C is a set of closest corresponding pairs, and $\mathbf{x}$ and $\mathbf{y}$ are the sample points on $\widetilde{\mathcal{S}}$ and $\mathcal{T}$ respectively, and $D(\cdot,\cdot)$ is a distance metric between each corresponding point pair, $\psi_\nu(x) = 1-\exp(-\frac{x^2}{2\nu^2})$ is Welsch's function to reduce the influence of corresponding pairs with long distances. Here ν is a hyperparameter that determine the sparseness. We set $\nu = \nu_0 d_{\text{med}}$, where d_{med} represents the median value of all corresponding pairs' distances and ν_0 is a hyperparameter.

We optimize the Lie algebraic representation of rigid transformation with our metric and variants of chamfer distance by gradient descent method [26], and use the Human test dataset as the benchmark in this experiment; we consider the influence of hyperparameters, and thus choose three sets of parameters, which are the $\nu_0 = 2, \nu_0 = 0.5, \nu_0 = 0.1$. The Tab. 5 indicates that our metric is superior to variants of chamfer distance based on Welsch's function, and the Fig. 10 and Fig. **??** are the visualized results of variants of Chamfer distance, and our metric on the Human dataset [1] and the 3d-Match dataset [47], respectively. It shows our metric can generate more robust and global optimal results.

More Visualized Results Compared with Traditional Methods

We provide more visualized results compared with traditional methods, the Fig. 12 and Fig. 13 are the visualized results of other traditional methods and our metric on the Human dataset and the 3D-Match dataset, respectively. The Fig. 12 shows our metric is easier to jump out of the local optimal solution to attain the global optimal solution, but ICP-based methods has reached the local optimum. The Fig. 13 indicates that our metric can also achieve better results for the real data.

Table 6. Comparison with variants of chamfer distance by directly optimizing a Lie algebra on Axyz-pose human dataset [1]. It shows the rotation errors(degree), translation errors and piecewice errors defined in Eq. (7).

Method	$\text{Err}_{\mathbf{R}}$ (degrees)	$\text{Err}_{\mathbf{t}}$ $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)	Err_{pw} $(\cdot 10^{-1})$ (ℓ_1, ℓ_2)
CD	5.863	0.148, 0.132	0.151, 0.14
CD-W ($\nu_0 = 0.1$)	12.574	0.196, 0.152	0.384, 0.304
CD-W ($\nu_0 = 0.5$)	4.84	0.086, 0.078	0.108, 0.087
CD-W ($\nu_0 = 2$)	1.4	0.0267, 0.024	0.051, 0.043
Ours ($\nu_0 = 0.5$)	**0.576**	**0.017, 0.013**	**0.018, 0.015**

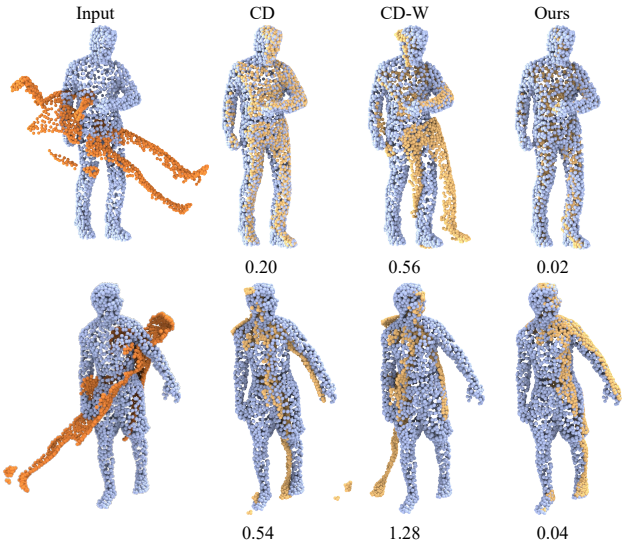


Figure 10. Comparison of different variants of chamfer distance on the Human dataset.

Computation Cost of Our Metric

Tab. 7 shows the training and inference time of our metric per iteration and compares them with the average computational time for FRICP, and model-based inference is very fast. Although our models need take a longer time to train, after the training, the actual registration only involves inference which is much more efficient. Besides, our current metric is implemented with Pytorch [27]. If in the GPU state, the general setting requires about 15G of memory, which is also an interesting direction that can be improved in the future.

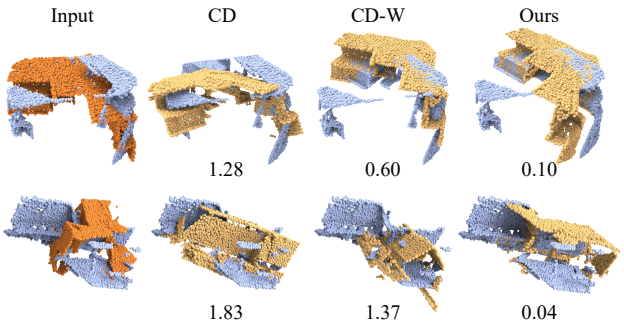


Figure 11. Comparison of different variants of chamfer distance on 3D-Macth datasets.

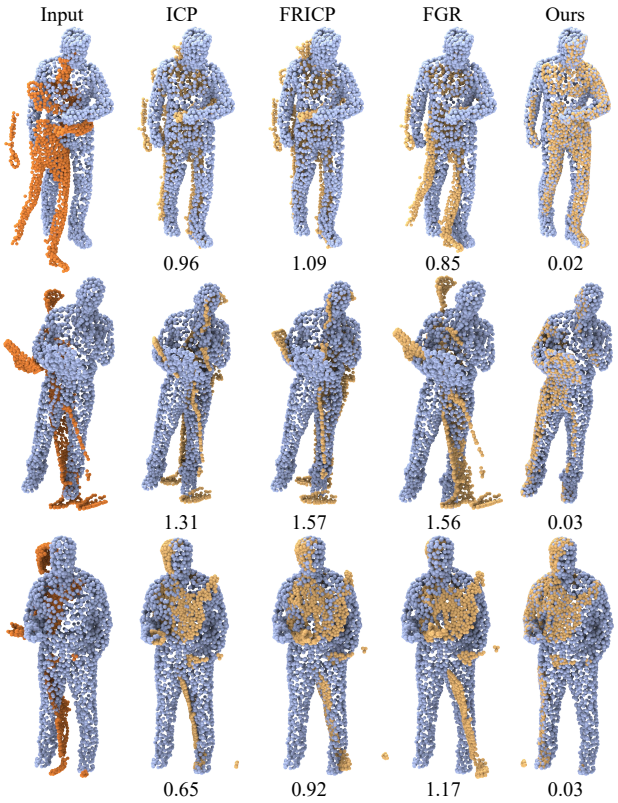


Figure 12. Comparison with different traditional methods on the Human dataset.

More Details of Our Experiments Setting

RPM-net framework [46][5] used a less sensitive initialization and more robust deep learning-based approach for rigid point cloud registration. The metrics which used is l_1 distance between the source point cloud X transformed using the groundtruth transformation $\mathbf{R}_{gt}, \mathbf{t}_{gt}$ and the predicted transformation $\mathbf{R}_{pred}, \mathbf{t}_{pred}$.

$$L_{reg} = \frac{1}{J}\sum_{j}^{J} |(\mathbf{R}_{gt}x_j + \mathbf{t}_{gt}) - (\mathbf{R}_{pred}\mathbf{x}_j + \mathbf{t}_{pred})|.$$

$$L_{inlier} = -\frac{1}{J}\sum_{j}^{J}\sum_{k}^{K} m_{jk} - \frac{1}{K}\sum_{k}^{K}\sum_{j}^{J} m_{jk}.$$

The overall loss is the weighted sum of the two losses: $L_{total} = L_{reg} + \lambda L_{inlier}$ where we use $\lambda = 0.01$ in all our experiments for supervised learning, and use our metric and Chamfer distance replace the L_{reg} for our experiment setting and Chamfer distance experiment setting, respectively. We compute the loss for every iteration i, but weigh the losses by $\frac{1}{2}^{(N_i - i)}$ to give later iterations higher weights, where N_i is the total number of iteration during training.

[5] https://github.com/yewzijian/RPMNet

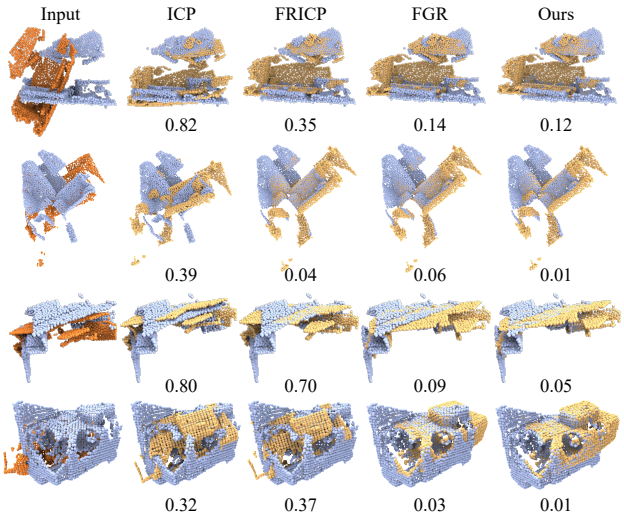


Figure 13. Comparison with different traditional methods on the 3d-Match dataset.

Table 7. The comparison of computation time(ms) per iteration(col 2-6) and of the run in an example(col 7-8) in FRICP [48] and our method. In our method, we count the optimized time of two sub-processes per iteration: random straight line generation(Sam), finding the points of intersection, and expectation calculation(Inter). For the inference time of our method, we use DCP [41] to test. Since this time is related to the number of point clouds and random lines, we show the time under three different numbers of point clouds and two different numbers of random straight lines.

#Points	#Lines: 5000		#Lines: 20000		FRICP(S)	Inference	FRICP(W)
	Sam	Inter	Sam	Inter			
1024	36.6	50.4	46.4	204.4	0.3	19.7	43.9
5000	42.3	145.5	45.6	236.2	1.9	20.7	223.3
10000	45.6	191.9	47.4	544.4	3.2	22.4	379

DCP framework [41][6] is a learning method based on differential SVD, here the loss function to measure the model's agreement to the ground-truth rigid motions:

$$Loss = ||\mathbf{R}_{xy}^T \mathbf{R}_{xy}^g - I|| + ||\mathbf{t}_{xy} - \mathbf{t}_{xy}^g||^2 + \lambda||\theta||^2.$$

Here, g denotes ground-truth. The first two terms define a simple distance on $SE(3)$. The third term denotes Tikhonov regularization of the DCP parameters θ, which reduce the complexity of the network. We replace the first two terms with our metric and Chamfer distance as our experiment setting and Chamfer distance experiment setting, respectively.

FMR framework [16][7] is a fast semi-supervised approach for robust point cloud registration without correspondence. The loss functions in their paper are the

$$loss = loss_{cf} + loss_{pe}.$$

$$loss_{cf} = \sum_{p \in A} \sum_{i=1}^{N} \underset{q \in S^*}{min} ||\phi_{\theta_i(p;x)} - q||_2^2 + \sum_{q \in S^*} \underset{i,\in 1 \ldots N}{min} \underset{p \in A}{min} ||\phi_{\theta_i}(p;x) - q||_2^2,$$

where $p \in A$ is a set of points sampled from a unit square $[0,1]^2$, x is a point cloud feature, ϕ_{θ_i} is the i^{th} component in the MLP parameters, S is the original input 3D points

$$loss_{pe} = \frac{1}{M} \sum_{i=1}^{M} ||f(g_{est} \cdot P) - f(g_{gt} \cdot P)||_2^2,$$

where P is a point cloud, and M is its total point number. For the unsupervised training, we only use the $loss_{cf}$; we replace the $loss$ with our metric as our experiment setting. And we significantly improve the performance of the unsupervised manners.

Extend Our Metric to SVD Solver

Regarding the discrete version of Eq. (5):

$$f(\widetilde{\mathbf{X}}, \mathbf{Y}) = \sum_{l \in A} w_l \Big(\sum_{\mathbf{x}_i^l \in \mathbf{X}_l} D(\widetilde{\mathbf{x}}_i^l, \mathbf{y}_{\pi_i{}^l}^l) + \sum_{\mathbf{y}_j^l \in \mathbf{Y}_l} D(\widetilde{\mathbf{x}}_{\rho_j{}^l}^l, \mathbf{y}_j^l) \Big).$$

As described in the FRICP [48], based on the corresponding points $\{(\mathbf{x},\mathbf{y}), (\mathbf{x},\mathbf{y}) \in C\} = \bigcup_{l \in A} \big(\bigcup_{x_i^l \in \mathbf{X}_l} (x_i^l, y_{\pi_i^l}^l) \bigcup_{y_j^l \in \mathbf{Y}_l} (x_{\rho_j^l}^l, y_j^l) \big)$ between the source intersections $\bigcup_{l \in A} \mathbf{X}_l$ and target intersections $\bigcup_{l \in A} \mathbf{Y}_l$. We can extend our non-linear metric into quadratic by replacing the Welsch's function with quadratic surrogate function, then, we get the sum of squared distance between the points $\mathbf{x}, \mathbf{y}$.

$$(\mathbf{R}^*, \mathbf{t}^*) = \underset{(\mathbf{R},\mathbf{t})}{\arg\min} \sum_{(\mathbf{x},\mathbf{y}) \in \mathbf{C}} w_{(\mathbf{x},\mathbf{y})} ||(\mathbf{R}\mathbf{x} + \mathbf{t} - \mathbf{y})||_2^2,$$

where $w_{(\mathbf{x},\mathbf{y})} = \psi_\nu(\|\mathbf{x}-\mathbf{y}\|_2) * w_l$. It can be solved in closed form via SVD [38], which implemented with Pytorch [27].

[6] https://github.com/WangYueFt/dcp
[7] https://github.com/XiaoshuiHuang/fmr