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**

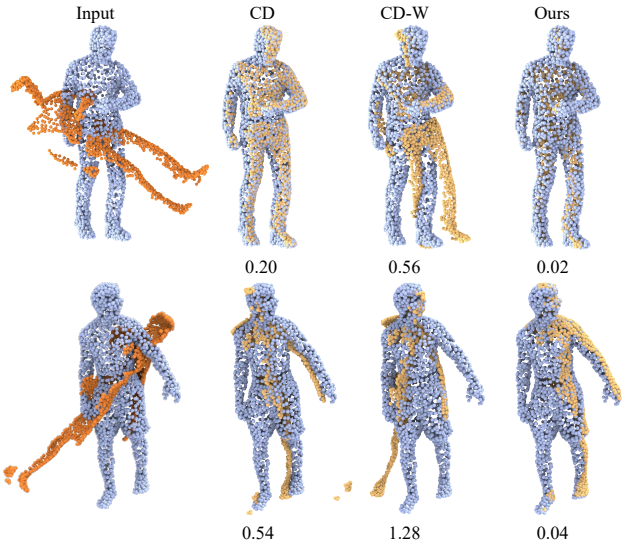

Figure 10. Comparison of different variants of chamfer distance on the Human dataset.

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.

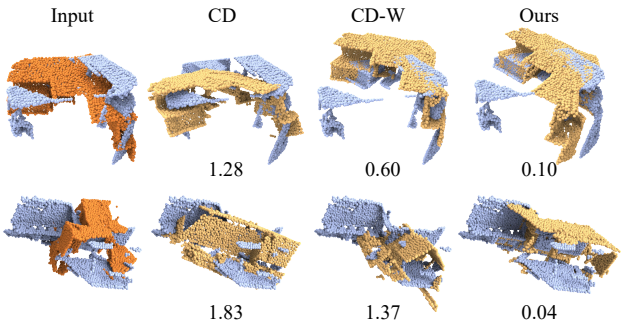

Figure 11. Comparison of different variants of chamfer distance on 3D-Macth datasets.

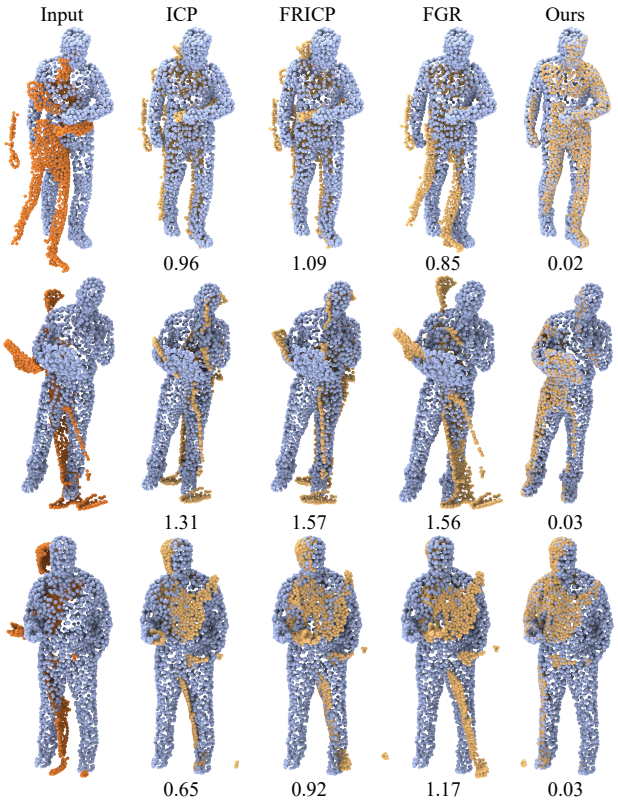

Figure 12. Comparison with different traditional methods on the Human dataset.

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

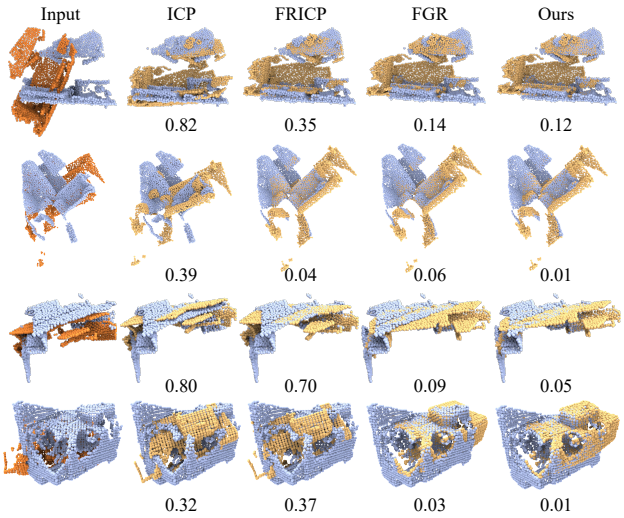

Figure 13. Comparison with different traditional methods on the 3d-Match dataset.

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