Inverse Composition Discriminative Optimization for Point Cloud Registration

Jayakorn Vongkulbhisal[1,2], Beñat Irastorza Ugalde[2], Fernando De la Torre[2,3], João P. Costeira[1]
[1]ISR - IST, Universidade de Lisboa, Lisboa, Portugal
[2]Carnegie Mellon University, Pittsburgh, PA, USA
[3]Facebook Inc., Menlo Park, CA, USA
jvongkul@andrew.cmu.edu, birastor@andrew.cmu.edu, ftorre@cs.cmu.edu, jpc@isr.ist.utl.pt

Abstract

*Rigid Point Cloud Registration (PCReg) refers to the problem of finding the rigid transformation between two sets of point clouds. This problem is particularly important due to the advances in new 3D sensing hardware, and it is challenging because neither the correspondence nor the transformation parameters are known. Traditional local PCReg methods (*e.g.*, ICP) rely on local optimization algorithms, which can get trapped in bad local minima in the presence of noise, outliers, bad initializations,* etc. *To alleviate these issues, this paper proposes Inverse Composition Discriminative Optimization (ICDO), an extension of Discriminative Optimization (DO), which learns a sequence of update steps from synthetic training data that search the parameter space for an improved solution. Unlike DO, ICDO is object-independent and generalizes even to unseen shapes. We evaluated ICDO on both synthetic and real data, and show that ICDO can match the speed and outperform the accuracy of state-of-the-art PCReg algorithms.*

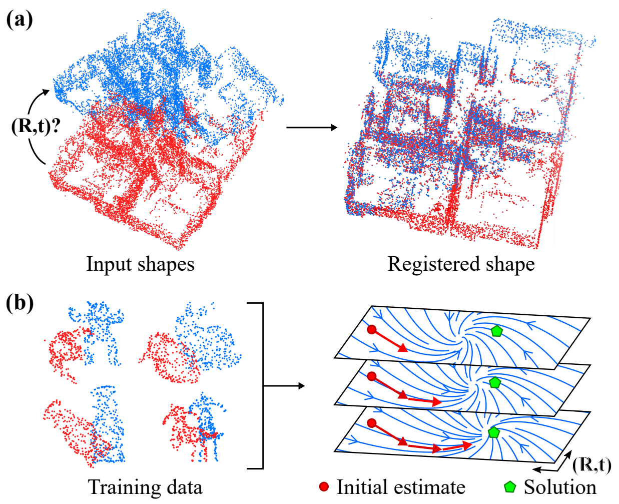

Figure 1. Rigid Point Cloud Registration (PCReg) with ICDO. (a) The goal of PCReg is to estimate rigid transformation parameter that registers two point clouds together. (b) ICDO learns an inverse composition update rule that searches for the solution from PCReg examples. The learned update rule of ICDO generalizes to shapes that are not even in the training set.

1. Introduction

Rigid Point Cloud Registration (PCReg) refers to the problem of finding the rigid transformation between two or more point clouds without correspondence (Fig. 1a). PCReg algorithms are fundamental to 3D data processing, especially nowadays with the ever increasing access to 3D sensors (*e.g.*, iPhone X, Kinect, LIDAR). Applications of PCReg span 3D reconstruction, pose estimation, tracking, *etc*. Many successful approaches formulate PCReg as an optimization problem and solve it with local optimization algorithms. Unfortunately, these local methods tend to get trapped in bad local optima without a good initialization.

To achieve robustness against local optima, several authors proposed different formulations and algorithms. A major class of successful algorithms relies on *deterministic annealing* strategies [16, 20, 7]. In short, these algorithms first optimize a coarse, non-robust cost function with a small number of local optima, then continually increase the robustness by modifying the cost function. One issue with annealing approaches is that they require a schedule for such modification, *i.e.*, how fast and to what shape the cost should be modified. Setting this schedule is not trivial. With an improper schedule, the optimization might take longer than necessary or even skip the correct solution to a different optimum altogether.

Recently, Discriminative Optimization (DO) [38] has been proposed as a learning-based technique to solve PCReg. DO searches for a solution by mapping a feature vector to a parameter update vector, where the maps are learned from a set of training data. Although it was shown to be very robust, DO has a significant limitation: the features and the maps are object-specific, *i.e.*, they only work for the particular object they were trained on. This limits the usefulness of DO because it cannot be efficiently applied to problems that only register each point cloud once, *e.g.*, merging point clouds for reconstruction. Meanwhile, a general framework for deriving feature vectors for DO has been proposed, allowing DO to solve other computer vision problems, such as camera calibration and image denoising [37]. At first glance, it seems this framework can be applied to solve object-independent PCReg. However, the dimensions of the derived feature is exponential in the dimensions of the point clouds, rendering it impractical even for 3D PCReg.

In this work, we reformulate DO to solve object-independent PCReg. We modify the feature derivation in [37] to represent the interaction between any two 3D point clouds[1] such that the feature's dimensions are independent of the point cloud's dimensions. This allows us to train a single set of maps and use it to register arbitrary shapes, including unseen ones. Since our update rule is the inverse composition operation, we call our approach Inverse Composition Discriminative Optimization (ICDO). We also show that ICDO can be interpreted as learning an annealing schedule, allowing fast convergence compared with other annealing-based local PCReg algorithms while maintaining high accuracy.

2. Previous Work

Rigid Point Cloud Registration (PCReg): PCReg algorithms can be classified into two classes. *(i) Local approaches* use local search algorithms that search around the current estimated parameters. They are typically fast but can get trapped in local optima without good initializations. *(ii) Global approaches* search the whole configuration space using globally optimal algorithms, such as branch-and-bound [40, 8], or formulate the problem with a convex relaxation [23, 5]. They do not require any initialization but are generally slower than local algorithms. Rather than using just points, there are also algorithms that use other information, *e.g.*, colors [24, 29], lines [9, 6], planes [22, 28], and local features from point clouds [41, 21, 15, 13]. For these algorithms, their optimization module may be either local [9, 24, 29, 22, 41], global [15, 6, 28], RANSAC-based [21, 13], or their combinations. Since ICDO is a local algorithm, we will focus our review in this class.

Local PCReg approaches generally rely on local search algorithms. Different techniques are used depending on how the rotation is parametrized. One major class directly uses rotation matrix, and alternately solve for the parameter (with Procrustes analysis [36]) and the correspondence weights. These methods include ICP [3, 10] and their variants [32, 2]. While ICP uses binary weights, many algorithms use the Gaussian of the distance between the points as weights, *e.g.*, Robust Point Matching (RPM) [16], EM-ICP [18], and Coherent Point Drift (CPD) [26]. More recently, [17] models PCReg as objects moving under a gravitational field. Rather than using rotation matrix, another class of methods relies on other parametrization, *e.g.*, axis-angle or quaternion, and uses gradient-based techniques to solve for the parameter. LM-ICP [14] minimizes robust cost functions with the Levenberg-Marquardt algorithm. Kernel Correlation (KC) [34] and Gaussian Mixture Registration (GMR) [20] minimize the L_2 distance between the Gaussian mixtures of the point clouds. We note that the algorithms that rely on Gaussian functions require setting their widths. They either estimate such widths in each iteration [26, 18] or use deterministic annealing [16, 18, 20].

Discriminative Optimization (DO): DO [38] was proposed as a learning-based approach for local PCReg. It learns an update rule as a linear mapping from a feature vector to an update parameter[2]:

$$\mathbf{x}_\tau = \mathbf{x}_{\tau-1} - \mathbf{D}_\tau \mathbf{h}(\mathbf{x}_{\tau-1}) \tag{1}$$

where $\mathbf{x}_\tau \in \mathbb{R}^p$ is the transformation parameter in step $\tau = 1, \ldots$; $\mathbf{h} : \mathbb{R}^p \to \mathbb{R}^f$ extracts features from the point clouds at $\mathbf{x}_\tau$; and $\{\mathbf{D}_\tau\} \subset \mathbb{R}^{f\times p}$, which map the feature to an update vector, are learned from training data and are specific to a single shape. The concept of DO is similar to cascaded regression [12] and supervised descent [39], which are widely used in face alignment. Recently, a framework for deriving feature functions based on gradient descent has been proposed [37], and DO was further applied to image denoising and camera pose estimation. Note that while some works use learning-based techniques for PCReg (*e.g.*, support vector regression [7] and deep neural network [13]), they are used to learn new shape representations, not for the estimation of parameters.

In this work, we build upon DO and extend it in several ways. *(i)* Instead of the summation rule in (1), ICDO uses inverse composition as the update rule. Learning-based composition rules have been used for image-based pose estimation and tracking [12, 35], but they have not been applied to PCReg. *(ii)* Unlike DO which is shape-specific, ICDO generalizes across different shapes, even those not included in the training data. *(iii)* We show how to derive the feature function with much smaller dimensions than the framework in [37]. *(iv)* We also show that the learned maps can be interpreted as an annealing schedule, avoiding the need to manually set one like in previous PCReg methods.

3. Inverse Composition DO (ICDO)

In this section, we introduce our PCReg algorithm called Inverse Composition Discriminative Optimization

[1] We focus on 3D PCReg. The 2D case can be derived in a similar way.

[2] Bold capital letters denote a matrix $\mathbf{X}$, bold lower-case letters a column vector $\mathbf{x}$, non-bold letters a scalar x. $\mathbf{0}_n, \mathbf{1}_n \in \mathbb{R}^n$ are the vector of zeros and ones. Vector $\mathbf{x}_i$ denotes the i^{th} column of $\mathbf{X}$. Bracket subscript $[\mathbf{x}]_i$ denotes the i^{th} element of $\mathbf{x}$. $\|\mathbf{x}\|$ denotes ℓ_2-norm $\sqrt{\mathbf{x}^\top\mathbf{x}}$.

(ICDO). We first describe our motivation from gradient-based PCReg and inverse composition PCReg, then we describe how to combine them with the DO framework [37] to solve shape-independent PCReg. We provide the interpretation of ICDO, its computational complexity, and implementation details at the end of the section.

3.1. Motivation: Gradient-based PCReg

Given two point clouds represented by the matrices $\mathbf{M} \in \mathbb{R}^{3\times N_M}$ for the model shape and $\mathbf{S} \in \mathbb{R}^{3\times N_S}$ for the scene shape, the goal of 3D rigid PCReg is to find a transformation parameter $\mathbf{x}$ such that the transformed scene is similar to the model: $T(\mathbf{S},\mathbf{x}) \sim \mathbf{M}$. Here, we consider $\mathbf{x} = [\mathbf{r}^\top, \mathbf{t}^\top]^\top$, where $\mathbf{r}$ parametrizes rotation such that $\mathcal{R}(\mathbf{r})$ is a rotation matrix (*e.g.*, $\mathbf{r}$ can be an axis-angle vector, a quaternion, or a rotation matrix itself); $\mathbf{t} \in \mathbb{R}^3$ is a translation vector; and T transforms a point cloud as

$$T(\mathbf{S},\mathbf{x}) = \mathcal{R}(\mathbf{r})\mathbf{S} + \mathbf{t}\mathbf{1}_{N_S}^\top. \tag{2}$$

To solve PCReg, many works formulate it as an optimization problem. For example, the ICP algorithm solves

$$\underset{p_{ij}\in\{0,1\},\mathbf{x}}{\text{minimize}} \sum_{i=1}^{N_M}\sum_{j=1}^{N_S} p_{ij}\|\mathbf{m}_i - T(\mathbf{s}_j;\mathbf{x})\|^2, \tag{3}$$

where $p_{ij} \in \{0,1\}$ denote correspondences. We can see the cost function of (3) is not continuous, so ICP requires to alternate between solving for $\mathbf{x}$ and p_{ij}. On the other hand, KC [34] and GMR [20] propose to solve

$$\underset{\mathbf{x}}{\text{minimize}} \; -\sum_{i=1}^{N_M}\sum_{j=1}^{N_S} \exp(-(1/\sigma)\|\mathbf{m}_i - T(\mathbf{s}_j;\mathbf{x})\|^2), \tag{4}$$

where σ controls the width of the Gaussian function. The cost function in (4) is continuous and differentiable, allowing gradient-based algorithms, such as gradient descent, to solve PCReg. One problem with (4) is that it is not easy to set σ: If σ is too large then the cost function could be too coarse and disregard details of the shapes, while a small σ could lead to a large number of local minima [20]. To handle this issue, GMR uses deterministic annealing, *i.e.*, it initializes with large σ then reduces it as the problem is solved. This scheduling can be difficult to set, and may lead to more computation time than necessary.

More generally, instead of using a Gaussian with specific widths, we may consider a generalization of (4):

$$\underset{\mathbf{x}}{\text{minimize}} \sum_{i=1}^{N_M}\sum_{j=1}^{N_S} \psi(\|\mathbf{m}_i - T(\mathbf{s}_j;\mathbf{x})\|), \tag{5}$$

where ψ is a 1D penalty function. We can see that other functions can be used in place of the Gaussian, but this makes it more complicated to select a function ψ and how to modify it to obtain a robust PCReg algorithm. In this work, we tackle this issue by learning from training data. Specifically, our algorithm learns to search for the solution of PCReg, where each step imitates the gradient descent on a ψ which is not expressed explicitly. We will show that our algorithm can be interpreted as learning the annealing of ψ from the training data, bypassing the need to manually design and modify it. Before we describe our algorithm, we look at the inverse composition operation, which we will use to update our parameters.

3.2. Inverse composition PCReg

Our algorithm is based on the inverse composition (IC) framework [1]. For PCReg, The composition operation can be described as follows. Two parameter vectors $\mathbf{x}_1 = [\mathbf{r}_1^\top, \mathbf{t}_1^\top]^\top$ and $\mathbf{x}_2 = [\mathbf{r}_2^\top, \mathbf{t}_2^\top]^\top$ are composed as

$$\mathbf{x}_1 \oplus \mathbf{x}_2 = \begin{bmatrix} \mathcal{R}^{-1}(\mathcal{R}(\mathbf{r}_2)\mathcal{R}(\mathbf{r}_1)) \\ \mathcal{R}(\mathbf{r}_2)\mathbf{t}_1 + \mathbf{t}_2 \end{bmatrix}, \tag{6}$$

where $\mathcal{R}^{-1}$ reverts a rotation matrix back to its parametrization. We also define $(\mathbf{x})^{-1}$ to be the inverse of $\mathbf{x}$: $\mathbf{x}_1 \oplus \mathbf{x}_2 \oplus (\mathbf{x}_2)^{-1} = \mathbf{x}_1$. In terms of transformation T, we can see that

$$T(\mathbf{S}, \mathbf{x}_1 \oplus \mathbf{x}_2) = T(T(\mathbf{S},\mathbf{x}_1),\mathbf{x}_2). \tag{7}$$

With the above notation, we compare IC with the forward composition (FC). Consider (5) with a differentiable ψ, and let $\mathbf{x}$ and $\mathbf{x}_+$ denote the current and the next estimates, *resp*. FC operates by alternately computing *(i)* the gradient that transforms $T(\mathbf{S};\mathbf{x})$ towards $\mathbf{M}$ and *(ii)* the FC update (we disregard the step size in $\Delta\mathbf{x}$):

$$\Delta\mathbf{x} = -\sum_{i=1}^{N_M}\sum_{j=1}^{N_S} \frac{\partial T(\mathbf{s}_j;\mathbf{x}\oplus\tilde{\mathbf{x}})}{\partial\tilde{\mathbf{x}}} \frac{\partial\psi(\|\mathbf{m}_i - T(\mathbf{s}_j;\mathbf{x}\oplus\tilde{\mathbf{x}})\|)}{\partial T(\mathbf{s}_j;\mathbf{x}\oplus\tilde{\mathbf{x}})}\Bigg|_{\tilde{\mathbf{x}}=\mathbf{0}} \tag{8}$$

$$\mathbf{x}_+ = \mathbf{x} \oplus \Delta\mathbf{x}. \tag{9}$$

In contrast, IC alternately computes *(i)* the gradient that transforms $\mathbf{M}$ towards $T(\mathbf{S};\mathbf{x})$ and *(ii)* the IC update:

$$\Delta\mathbf{x} = -\sum_{i=1}^{N_M}\sum_{j=1}^{N_S} \frac{\partial T(\mathbf{m}_i;\tilde{\mathbf{x}})}{\partial\tilde{\mathbf{x}}} \frac{\partial\psi(\|T(\mathbf{m}_i;\tilde{\mathbf{x}}) - T(\mathbf{s}_j,\mathbf{x})\|)}{\partial T(\mathbf{m}_i;\tilde{\mathbf{x}})}\Bigg|_{\tilde{\mathbf{x}}=\mathbf{0}} \tag{10}$$

$$\mathbf{x}_+ = \mathbf{x} \oplus (\Delta\mathbf{x})^{-1}, \tag{11}$$

where $\mathbf{0}$ denotes the identity transformation parameter. Similar to image alignment [1], we see that FC requires recomputing $\frac{\partial T(\mathbf{s}_j;\mathbf{x}\oplus\tilde{\mathbf{x}})}{\partial\tilde{\mathbf{x}}}$ at every iteration as it depends on $\mathbf{x}$, while IC's $\frac{\partial T(\mathbf{m}_i;\tilde{\mathbf{x}})}{\partial\tilde{\mathbf{x}}}$ is constant at $\tilde{\mathbf{x}} = \mathbf{0}$. The IC framework allows $\frac{\partial T(\mathbf{m}_i;\tilde{\mathbf{x}})}{\partial\tilde{\mathbf{x}}}$ to be computed only once, leading to less computation than FC. In this work, we rely on an update similar to IC, but instead of using the gradient of ψ in $\Delta\mathbf{x}$, we will learn the update from training data.

3.3. Learning and performing update steps

In order to learn the update step under the IC update rule, we follow the DO framework from [37], which is based on mapping a feature vector to an update vector. First, we describe our update rule, followed by how to learn the maps and apply them to solve PCReg.

Update rule: Given an initialization $\mathbf{x}_0 = \mathbf{0}$, the two point clouds denoted as $\theta = (\mathbf{M}, \mathbf{S})$, and a function $\mathbf{h}$ that extracts features from the point clouds, ICDO updates the estimated parameter at step τ using the IC operation

$$\mathbf{x}_\tau = \mathbf{x}_{\tau-1} \oplus (\mathbf{D}_\tau \mathbf{h}(\mathbf{x}_{\tau-1}; \theta))^{-1}, \tag{12}$$

where $\mathbf{D}_\tau, \tau = 1, 2, \dots$ are matrices that map the feature function $\mathbf{h}(\mathbf{x}_{\tau-1}; \theta)$ to an update vector $\Delta\mathbf{x}$. This update rule differs from the additive update rule of DO. Another major difference lies in the fact that the features and the maps of DO are shape-specific, while here we will show how to derive a shape-independent function $\mathbf{h}$ in Sec. 3.4. Next, we describe how we learn the maps.

Learning update maps: Suppose we are given a training set $\{(\mathbf{x}_*^k, \theta^k)\}_{k=1}^K$, where $\theta^k = (\mathbf{M}^k, \mathbf{S}^k)$ contains the two point clouds of the k^{th} training instance, and $\mathbf{x}_*^k$ is the ground truth registration parameter satisfying $T(\mathbf{S}^k; \mathbf{x}_*^k) \sim \mathbf{M}^k$. At step τ, we wish to learn a map $\mathbf{D}_\tau$ such that the updated $\mathbf{x}_\tau^k, k = 1, \dots, K$ in (12) move towards $\mathbf{x}_*^k$. Similar to [37], this is done using the following regularized linear least-squares regression:

$$\mathbf{D}_\tau = \arg\min_{\tilde{\mathbf{D}}} \frac{1}{K} \sum_{k=1}^K \|((\mathbf{x}_*^k)^{-1} \oplus \mathbf{x}_{\tau-1}^k) - \tilde{\mathbf{D}}\mathbf{h}(\mathbf{x}_{\tau-1}^k; \theta^k)\|_2^2 + \lambda \|\tilde{\mathbf{D}}\|_F^2. \tag{13}$$

Here, $((\mathbf{x}_*^k)^{-1} \oplus \mathbf{x}_{\tau-1}^k)$ is the difference between $\mathbf{x}_{\tau-1}^k$ and $\mathbf{x}_*^k$ under the IC operation. After a map is learned, we update the training instances using the update rule (12). We repeat this process until a terminating criteria is reached, *e.g.*, a maximum number of maps. Alg. 1 shows a pseudocode for training ICDO.

Solving PCReg: The pseudocode for solving PCReg with ICDO is summarized in Alg. 2. Suppose we trained a total of $\mathcal{T}$ maps. We first perform the update using (12) until step $\mathcal{T}$, then we continue using $\mathbf{D}_\mathcal{T}$ to update until a termination criteria is reached, *e.g.*, the update is small. However, we found that many times using $\mathbf{D}_\mathcal{T}$ to update causes the parameter to bounce around the correct solution without converging to it. This behavior resembles subgradient method with constant step size [4], which may not converge to a minimum. To alleviate this issue, we attempted to scale the update with $1/(\tau - \mathcal{T})$ and $1/\sqrt{\tau - \mathcal{T}}$ for $\tau > \mathcal{T}$ but we found that the updates diminished too fast, leading to a premature termination. The strategy that we found effective is to use $\Delta\mathbf{x}$ from the average of the updates from the current and the previous iterations (line 6 in Alg. 2). This strategy resembles the momentum approach [27] used frequently in first-order optimization.

3.4. From gradient to features

In this section, we describe how to derive the feature function $\mathbf{h}$ based on the gradient of (5). We parametrize rotation with the axis-angle vector in $\mathbb{R}^3$, but the derivation can also be used with other parametrizations. The steps to derive $\mathbf{h}$ is similar to those in [37]. First, we define a cost function (without an explicit expression) that penalizes registration residuals. Then, we take the cost's derivative and represent it as an inner product between two functions. Finally, we discretize the functions into a feature vector $\mathbf{h}$ and a matrix $\mathbf{D}$, allowing us to learn $\mathbf{D}$ from training data. The details are as follows.

Define $\mathbf{g}$ to be the following residual function

$$\mathbf{g}_{ij}(\tilde{\mathbf{x}}; \mathbf{x}) = T(\mathbf{m}_i; \tilde{\mathbf{x}}) - T(\mathbf{s}_j; \mathbf{x}), \tag{14}$$

where $\mathbf{x}$ is the current parameter estimate. To update $\mathbf{x}$ under IC, we consider the following optimization problem

$$\underset{\tilde{\mathbf{x}} \in \mathbb{R}^6}{\text{minimize}}\ J(\tilde{\mathbf{x}}) = \sum_{i=1}^{N_M} \sum_{j=1}^{N_S} \psi(\|\mathbf{g}_{ij}(\tilde{\mathbf{x}}; \mathbf{x})\|), \tag{15}$$

for some 1D function ψ. Next, we compute $\Delta\mathbf{x} = -\frac{\partial J(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}}$ at $\tilde{\mathbf{x}} = \mathbf{0}_6$. For simplicity, we consider a single term (i, j):

$$\begin{aligned} \Delta\mathbf{x}_{ij} &\triangleq -\frac{\partial}{\partial \tilde{\mathbf{x}}} \psi(\|\mathbf{g}_{ij}(\tilde{\mathbf{x}}; \mathbf{x})\|)\Big|_{\tilde{\mathbf{x}}=\mathbf{0}_6} \\ &= -\underbrace{\begin{bmatrix} -[\mathbf{m}_i]_\times \\ \mathbf{I}_3 \end{bmatrix} \frac{\mathbf{g}_{ij}(\mathbf{0}_6; \mathbf{x})}{\|\mathbf{g}_{ij}(\mathbf{0}_6; \mathbf{x})\|}}_{=\mathbf{w}_{ij}} \frac{\partial \psi(\|\mathbf{g}_{ij}(\tilde{\mathbf{x}}; \mathbf{x})\|)}{\partial \|\mathbf{g}_{ij}(\tilde{\mathbf{x}}; \mathbf{x})\|}\Big|_{\tilde{\mathbf{x}}=\mathbf{0}_6} \end{aligned} \tag{16}$$

We can see that only the rightmost term is dependent on ψ. Since we assume we do not have access to ψ, we will learn this term from training data using the algorithm in Sec. 3.3. To do so, we need to express $\Delta\mathbf{x}$ as $\mathbf{D}\mathbf{h}(\mathbf{x})$. This is done by replacing the term with a function $\varphi : \mathbb{R} \to \mathbb{R}$, then factorizing it as a convolution with Dirac delta function δ:

$$\Delta\mathbf{x}_{ij} = -\mathbf{w}_{ij}\varphi(\|\mathbf{g}_{ij}(\mathbf{0}_6; \mathbf{x})\|) \tag{17}$$

$$= -\mathbf{w}_{ij} \int_V \varphi(v)\delta(v - \|\mathbf{g}_{ij}(\mathbf{0}_6; \mathbf{x})\|)dv, \tag{18}$$

where $V = \mathbb{R}$. Consider only an element l of $\Delta\mathbf{x}_{ij}$, we see

$$[\Delta\mathbf{x}_{ij}]_l = -\int_V [\mathbf{w}_{ij}]_l \varphi(v)\delta(v - \|\mathbf{g}_{ij}(\mathbf{0}_6; \mathbf{x})\|)dv. \tag{19}$$

We can see that (19) is an inner product between $-\varphi(v)$ and $[\mathbf{w}_{ij}]_l \delta(v - \|\mathbf{g}_{ij}(\mathbf{0}_6; \mathbf{x})\|)$. This is similar to $[\mathbf{D}\mathbf{h}]_l$, which can be considered as the inner product between $\mathbf{h}$ and row l of $\mathbf{D}$. Following this connection, we will express the product between $\varphi(v)$ and $[\mathbf{w}_{ij}]_l \delta(v - \|\mathbf{g}_{ij}(\mathbf{0}_6; \mathbf{x})\|)$ as a matrix-vector product $[\mathbf{D}\mathbf{h}]_l$. To do so, we discretize the space V into q boxes, leading to (19)'s discretized counterpart:

$$[\Delta\mathbf{x}_{ij}]_l \approx -\boldsymbol{\varphi}^\top [\mathbf{w}_{ij}]_l \mathbf{e}_{\gamma(\frac{q}{r}\|\mathbf{g}_{ij}(\mathbf{0}_6; \mathbf{x})\|)}, \tag{20}$$

where $\gamma : \mathbb{R} \to \{0, 1, \ldots, q\}$ rounds up any number in $[0, q]$, or returns 0 otherwise; $r \in \mathbb{R}$ controls the discretization range; δ is discretized into the standard basis vector $\mathbf{e}_\beta \in \{0, 1\}^q$ (We define $\mathbf{e}_0 = \mathbf{0}_q$); and φ is discretized into a vector $\boldsymbol{\varphi} \in \mathbb{R}^q$. With these discretizations, we can put everything back to the full $\Delta\mathbf{x}$ as

$$\Delta\mathbf{x} = \sum_{i=1}^{N_M} \sum_{j=1}^{N_S} \Delta\mathbf{x}_{ij} \approx \mathbf{Dh}(\mathbf{x}; \theta), \tag{21}$$

$$\mathbf{D} = -\mathbf{I}_6 \otimes \boldsymbol{\varphi}^\top \tag{22}$$

$$\mathbf{h}_{r,q}(\mathbf{x}; \theta) = \sum_{i=1}^{N_M} \sum_{j=1}^{N_S} \bigoplus_{l=1}^{6} [\mathbf{w}_{ij}]_l \mathbf{e}_{\gamma\left(\frac{q}{r}\|\mathbf{g}_{ij}(\mathbf{0}_6; \mathbf{x})\|\right)}, \tag{23}$$

where $\otimes$ is the Kronecker product, and $\bigoplus$ is vector concatenation. We can see that (21) factorizes $\Delta\mathbf{x}$ in (16) into a product of two terms: $\mathbf{D} \in \mathbb{R}^{6\times 6q}$ which contains the unknown φ, and $\mathbf{h}_{r,q} : \mathbb{R}^6 \times (\mathbb{R}^{N_M} \times \mathbb{R}^{N_S}) \to \mathbb{R}^{6q}$ which contains the known information about the two point clouds. This factorization allows us to use $\mathbf{h}$ as the feature function to learn the update maps with the algorithm in Sec. 3.3.

Our derivation of the feature function differs from that in [37]. If we follow [37], we will consider $\hat{\psi}(\mathbf{g}_{ij}(\tilde{\mathbf{x}}; \mathbf{x}))$ with $\hat{\psi} : \mathbb{R}^3 \to \mathbb{R}$ instead of $\psi(\|\mathbf{g}_{ij}(\tilde{\mathbf{x}}; \mathbf{x})\|)$ with $\psi : \mathbb{R} \to \mathbb{R}$. Using $\hat{\psi}$ would allow learning an anisotropic penalty instead of an isotropic one in ψ, but the feature $\mathbf{h}$ of $\hat{\psi}$ will have the dimension of $6q^3$ for 3D cases. This is much larger than $6q$ of (23), which is independent of the point cloud's dimension. Moreover, the maps learned from $\hat{\psi}$ would require a much larger number of training data to prevent overfitting.

3.5. Intrepreting ICDO

We can see from Sec. 3.4 that $\mathbf{h}$ is derived such that $\mathbf{Dh}$ approximates the negative gradient of an unknown ψ from (15). Thus, we can interpret ICDO as imitating gradient descent on (15). In addition, we can also make the following more specific interpretations of ICDO. *(i) Sampling on a gradient field:* Since $\mathbf{h}$ is a weighted sum of the discretized Dirac delta (23) and $\mathbf{D}$ contains the gradient field φ of ψ (22), we can interpret $\mathbf{Dh}$ as performing a weighted sampling from the gradient field. *(ii) Annealing:* In practice, different $\mathbf{D}_\tau$ are used in each step τ. This allows ICDO to learn how such gradient field changes with τ, similar to an annealing schedule (empirical analysis provided in Sec. 4.1). *(iii) Predetermined step sizes:* While many gradient-based algorithms use line search to estimate step sizes, ICDO directly incorporates them into $\mathbf{D}_\tau$. Thus, we can interpret ICDO as using predetermined (by training) step sizes, similar to the subgradient method where the step sizes are set in advance (*e.g.*, to decay in each step [33, 4]).

Algorithm 1 Training ICDO

```
Input: ${\{(\mathbf{x}^k_*, \theta^k)\}^K_{k=1}, \mathcal{T}, \lambda, r_0, q, \alpha}$
Output: ${\{\mathbf{D}_\tau\}^{\mathcal{T}}_{\tau=1}}$
1: Initialize $\mathbf{x}^k_0 := \mathbf{0}, \forall k$; and $r := r_0$
2: for $\tau = 1$ to $\mathcal{T}$ do
3:     Compute $\tilde{\mathbf{h}}^k := \mathbf{h}_{r,q}(\mathbf{x}^k_{\tau-1}; \theta^k), \forall k$ from (23)
4:     Compute $\mathbf{D}_\tau$ with (13)
5:     Compute $\mathbf{x}^k_\tau := \mathbf{x}^k_{\tau-1} \oplus (\mathbf{D}_\tau \tilde{\mathbf{h}}^k)^{-1}, \forall k$
6:     Compute $r := r_0/\alpha^\tau$
7: end for
```

Algorithm 2 Solving PCReg with ICDO

```
Input: $\theta, \{\mathbf{D}_\tau\}^{\mathcal{T}}_{\tau=1}, r_0, q, \alpha$
Output: $\mathbf{x}$
1: Initialize $\mathbf{x} := \mathbf{0}$; $\tau := 1$; and $r := r_0$
2: while not converge do
3:     Compute $\tilde{\mathbf{h}} := \mathbf{h}_{r,q}(\mathbf{x}; \theta)$ with (23)
4:     Compute $\Delta\mathbf{x} := \mathbf{D}_{\min(\tau,\mathcal{T})} \tilde{\mathbf{h}}$
5:     if $\tau > \mathcal{T}$ then
6:         Compute $\Delta\mathbf{x} := (\Delta\mathbf{x} + \Delta\mathbf{x}^-)/2$
7:     end if
8:     Compute $\mathbf{x} := \mathbf{x} \oplus (\Delta\mathbf{x})^{-1}$
9:     Compute $\Delta\mathbf{x}^- := \Delta\mathbf{x}$
10:    Compute $r := r_0/\alpha^\tau$
11:    Compute $\tau := \tau + 1$
12: end while
```

3.6. Computational complexity

We can see that the most demanding step of ICDO is the computation of the feature $\mathbf{h}$, which is $\mathcal{O}(N_M N_S)$ due to the pairwise residual $\mathbf{g}_{ij}$. This is equivalent to straightforward implementations of other PCReg algorithms, as they all require computing the pairwise distances. However, ICP can use kd-tree to find the nearest neighbors, which reduce the complexity to $\mathcal{O}(N_M \log N_S)$. Similarly, Gaussian-based approaches, such as CPD, KC, and GMR, can use fast Gauss transform (FGT) [19] to compute their correlation, which reduces the complexity to $\mathcal{O}(N_M + N_S)$. Unfortunately, the function learned by ICDO can be more general and we do not know of a way to improve its complexity.

3.7. Implementation details

Normalization: PCReg algorithms are generally sensitive to variations in the point clouds, *e.g.*, density and scale. These issues are further complicated by the fact that ICDO is learning-based, thus normalization is very important. First, we remove the mean of $\mathbf{M}$ from both $\mathbf{M}$ and $\mathbf{S}$ to maintain their relative configuration. Next, we perform two normalizations for scale and density. *(i) Scale*: Suppose that we have the registration $\mathbf{RS} + \mathbf{t} \sim \mathbf{M}$. If the shapes are scaled by ρ, *e.g.*, $\hat{\mathbf{M}} = \rho\mathbf{M}$, we will have

$\mathbf{R}\hat{\mathbf{S}} + \rho\mathbf{t} \sim \hat{\mathbf{M}}$: only the translation vector is scaled but not rotation, making it harder to learn effectively. To prevent this effect, we scale both $\mathbf{M}$ and $\mathbf{S}$ by $\sqrt{N_M}/\eta$, where η is the mean of $\mathbf{M}$'s singular values. *(ii) Density*: Consider $\theta^{(1)} = (\mathbf{M}, \mathbf{S})$ and $\theta^{(2)} = ([\mathbf{M}, \mathbf{M}], \mathbf{S})$, *i.e.*, the $\theta^{(2)}$'s model density is doubled. This causes an undesirable effect that $\mathbf{h}(\mathbf{x}, \theta^{(2)}) = 2\mathbf{h}(\mathbf{x}, \theta^{(1)})$, meaning the update step of $\theta^{(2)}$ will be double that of $\theta^{(1)}$, while the shapes are the same. To handle this issue, we divide $\mathbf{h}$ in (23) by $N_M N_S$.

Speeding up computation: We found that the most time-consuming step is the aggregation of $[\mathbf{w}_{ij}]_l$ into $\mathbf{h}$ in (23). To reduce computation, we reduce the number of terms in (23) by reducing the value of r in each iteration (recall that r controls the range of discretization, see (20)). Since we keep q constant, an additional advantage of this reduction is that the discretized boxes become finer as iteration increases, allowing more details to be captured. Note that we do not reduce r beyond iteration $\mathcal{T}$ in test, and this reduction does not affect the fact the ICDO learns an annealing schedule (see Fig. 2). Also, while this reduction speeds up computation, it does not change ICDO's complexity.

Training: We found that the training error in Alg. 1 reduces too fast, which causes the latter maps to have small updates. To handle this issue, we add random rotation in $\mathcal{N}(0, 10)$ degrees and translation vector with the norm in $\mathcal{N}(0, 0.1)$ to the data in each training iteration, and adjust the ground truth $\mathbf{x}_*^k$ accordingly. In addition, notice that $\mathbf{D}$ in (22) is block-diagonal with nonzero values only in the elements of $\boldsymbol{\varphi}$. In practice, we also found that the off-block-diagonal elements have very small values. With these observations, we constrain all elements outside the diagonal blocks to be zero when we learn the maps in (13).

Termination criteria: We terminate the algorithm when the rotation and the total displacement in the past 5 iterations amount to less than 0.5 degrees and 3×10^{-3}, *resp.* We also terminate if the number of iterations reaches 200.

4. Experiments

In this section, we evaluate ICDO with both synthetic and real experiments. We begin this section by describing baseline algorithms and performance measure. Then, we describe how we train the maps, analyze the maps, and show that the annealing effect is inherently learned by our algorithm. Finally, we present the comparison against other PCReg algorithms with synthetic and real data. All experiments were performed in MATLAB on a single thread of a machine with Intel i7-4770K 3.50GHz 16GB RAM.

Baselines: We use 4 baselines.[3] *(i)* ICP [3]. *(ii)* IRLS [2], which is similar to ICP but uses the Huber function as penalty. *(iii)* CPD [26], which maximizes the likelihood that one point cloud is generated by the Gaussian mixture of the other. *(iv)* GMR [20], which minimizes L_2 distance between the two Gaussian mixtures. Recall from Sec. 2 that GMR is gradient-based and uses deterministic annealing. This makes GMR most similar to ICDO. We obtained the MATLAB codes from the authors' websites, except ICP which we used MATLAB's implementation. Note that CPD and GMR's fast Gauss transform (FGT) is in C.

Performance measure: We use the *registration error*, defined as the pointwise RMSE of the model in the ground truth pose and the model in the estimated pose:

$$(1/\sqrt{N_M})\|T(\mathbf{M}; \mathbf{x}) - T(\mathbf{M}; \mathbf{x}_{gt})\|_F, \qquad (24)$$

where $\|\cdot\|_F$ is the Frobenius norm, and $\mathbf{x}$ and $\mathbf{x}_{gt}$ are the estimated and the ground truth poses, *resp.*

4.1. Training and analyzing the maps $\mathbf{D}_\tau$

Training the maps: We generated synthetic data to train the maps from seven 3D shapes: Bunny and armadillo from Stanford's 3D scan repository [11] and all 5 shapes from the UWA dataset [25]. Each training model $\mathbf{M}^k$ was generated by randomly picking a shape; scaling it so that all points are in $[-1, 1]^3$; and randomly rotating in $[0, 180]$ degrees. Next, we copied the model as the scene shape $\mathbf{S}^k$, then added a random rotation in $[0, 85]$ degrees and translation in $[-0.2, 0.2]^3$ to only $\mathbf{S}^k$. Then, we applied the following modifications to $\mathbf{M}^k$ and $\mathbf{S}^k$ independently: Randomly sampling $200-400$ points; adding Gaussian noise with SD in $[0, 0.03]$; and mimicking incomplete shape by randomly sampling a 3D vector $\mathbf{u}$, then removing the points where their dot product with $\mathbf{u}$ are in the top $0-30\%$ (this is done only either $\mathbf{M}$ or $\mathbf{S}$ but not both). No outliers were added for the training data as we found this degraded the results. We found $\lambda = 10^{-8}$, $r = 3$, $q = 100$, $\alpha = 1.15$, and $\mathcal{T} = 20$ work well across all experiments. We used a total of 10^5 training samples, and ICDO took 96 minutes to train.

Analyzing the maps: Fig. 2a shows $\mathbf{D}_5$ as an example of the learned maps. Here, $\boldsymbol{\varphi}_\tau^b$ denotes the vector in the diagonal block b of map $\mathbf{D}_\tau$. We observe that $\boldsymbol{\varphi}_\tau^1$, $\boldsymbol{\varphi}_\tau^2$, and $\boldsymbol{\varphi}_\tau^3$ which map to the update in rotation $\mathbf{r}$ are similar, while $\boldsymbol{\varphi}_\tau^4$, $\boldsymbol{\varphi}_\tau^5$, and $\boldsymbol{\varphi}_\tau^6$ for translation $\mathbf{t}$ are also similar. This is because the distribution of the data is isotropic. Since the maps of the same type are similar, we visualize $\boldsymbol{\varphi}_\tau^1$ and $\boldsymbol{\varphi}_\tau^4$ of different τ in Fig. 2b. We can see that the peaks of the curves move toward 0 as τ increases. Since we can interpret the maps as imitating a gradient field (Sec. 3.4), we also show the numerical integration of $\boldsymbol{\varphi}_\tau^1$ and $\boldsymbol{\varphi}_\tau^4$ in Fig. 2c, where we can see the functions squeeze closer to 0. These visualizations indicate that ICDO is learning an annealing schedule for PCReg from training data, unlike previous works which need to set one manually. Note that since the maps of rotation and translation are different, the vector fields of the updates cannot be integrated into a single cost function[4].

[3]We do not compare with DO [38] as it is shape-specific and requires 3-4 minutes to train each shape, thus DO is impractical for our experiments.

[4]We tried to learn a shared $\boldsymbol{\varphi}$ for all rotation and translation that allows

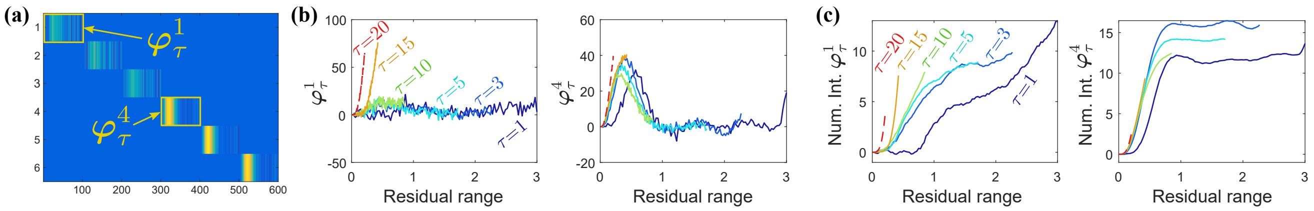

Figure 2. A visualization of the maps $\mathbf{D}_\tau$. (a) The learned matrix $\mathbf{D}_5$ (blue - low value, yellow - high value). (b) Plots of the diagonal blocks φ_τ^1 and φ_τ^4 of $\mathbf{D}_\tau$ for different τ, where we align each element in the vectors to the residual range $[0, r]$ they represent. Note that the length in x-axis of each vector differs since r decreases as τ increases. (c) Numerical integration of φ_τ^1 and φ_τ^4 from (b).

4.2. Synthetic data

We use 7 shapes (cat, centaur, dog, gorilla, gun, horse, and wolf) from TUM 3D object in clutter dataset [31] for testing. These shapes were selected so that they did not overlap with those in training. The initial shapes were normalized to lie in $[-1, 1]^3$. Following [38], we tested 5 modifications: *(i)* Number of points from 100 to 2000 [default = 200 to 400]; *(ii)* Initial angle from 0° to 180° [default = 0° to 60°]; *(iii)* Noise SD from 0 to 0.1 [default = 0 to 0.03]; *(iv)* Outlier ratio against the number of inliers from 0 to 2 [default = 0]; *(v)* Incomplete shape from 0 to 0.9 [default = 0] (generated the same way as in training). All tests included random translation in $[0, 0.3]^3$. Outlier points were randomly generated in $[-1.25, 1.25]^3$. While one parameter was varied, other parameters were set to the default values. For each setting, we tested 500 pairs of point clouds sampled from the 7 shapes. Unlike in training, the model and scene points were independently sampled. Here, we consider a registration successful if the registration error is less than 0.15. We also report the computation time.

Fig. 3 shows the results of the synthetic experiment. We can see that ICDO is comparable to the state-of-the-art algorithms: It performed almost perfectly under varying number of points, noisy data, and outlier ratios. ICDO has less success than CPD and GMR for large initial angles, while being more successful than ICP and IRLS. GMR even has some success with 180° initial angle because its scheduled annealing can smooth the shapes enough to avoid bad optima. However, a downside is GMR can also oversmooth, leading to some failure even with 0° initial angle. In contrast, ICDO with learned annealing has more success with lower angles and less success with high angles. Interestingly, ICDO works well with outliers even it was not trained with them. This is because ICDO (and also GMR) use a predetermined annealing schedule, so outliers have little effect on its performance. In contrast, outliers can thwart CPD's Gaussian width estimation and create more local minima for ICP and IRLS which use closest matches, leading to bad registration. Under similar reasons, ICDO and GMR are the most robust to incomplete shapes. In terms of computation time, ICDO is generally slightly slower than IRLS and CPD while being much faster then GMR (Recall that CPD and GMR use C code for FGT while ICDO is completely written in MATLAB, so their times are not directly comparable). This experiment demonstrates that ICDO can be trained and tested on different sets of shapes, while being able to obtain competitive success and time as state-of-the-art algorithms.

numerical integration to a cost function, but its result was not good.

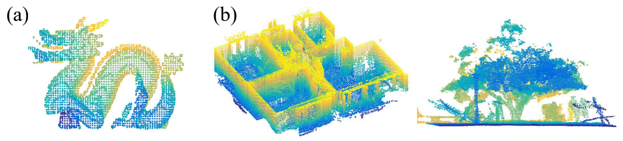

Figure 4. Real data examples (modified for visualization). (a) Stanford's dragon. (b) ETH laser registration dataset (Apartment and Gazebo Summer).

4.3. Real data

We perform experiments on two real datasets to evaluate ICDO. *(i)* Stanford's dragon [11] and *(ii)* ETH laser registration dataset [30]. Fig. 4 shows examples from the datasets. We provide the details and results below.

Stanford's dragon [11]: This dataset comprises 15 scans at every 24° of a dragon statue. Following [20, 7], we attempted to merge scans at $\pm24°$, $\pm48°$, $\pm72°$, $\pm96°$, with a total of 30 pairs for each angle. A registration is successful if $\mathbf{q}^\top\mathbf{q}_{gt} > 0.99$ where $\mathbf{q}$ and $\mathbf{q}_{gt}$ are the estimated and the ground truth unit quaternions, *resp.* Each point cloud was downsampled to 2000 points. The result is presented in Table 1. The results of the baselines were taken from SVR paper [7], which improves GMR by learning the weight of each Gaussian. We can see that ICDO is second to SVR while outperforming ICP, CPD, and GMR, illustrating the robustness of our approach against methods which consider all point as having equal weights. Our implementation took 7.7 seconds to register each pair on average.

ETH laser registration dataset [30]: This dataset consists of 3D laser scans from 8 outdoor and indoor environments. Each environment has 31 to 45 scans (total 275), and contains dynamic objects such as people and furniture displacement, which can be considered as outliers. The scans were recorded sequentially as the scanner traversed the en-

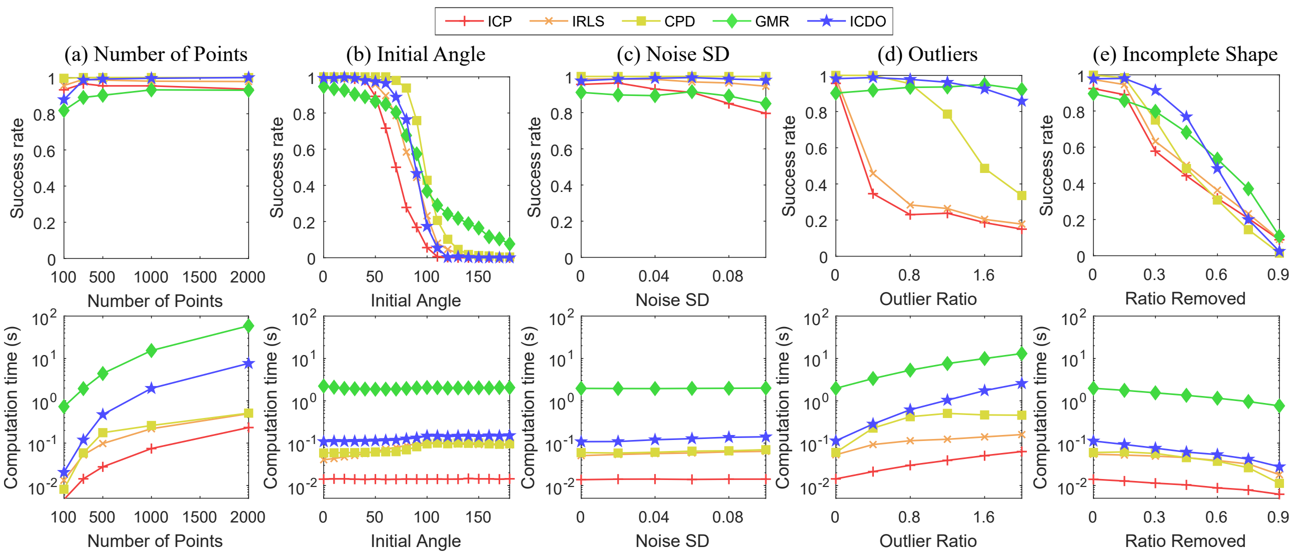
Figure 3. Results for synthetic data experiment over different modifications. (Top) Success rate. (Bottom) Computation time.

Table 1. Successful registration on Stanford's dragon.

Pose	ICP	CPD	GMR	SVR	ICDO
±24°	28	26	29	**30**	**30**
±48°	19	18	20	**29**	26
±72°	13	14	13	**16**	15
±96°	1	3	2	**4**	0

vironments. In this experiment, we merge consecutive scans in both forward and backward directions (total 534 pairs). We preprocessed each point cloud by using a box grid filter (MATLAB's `pcdownsample`) at 10cm interval to make the density more uniform, then subsampled to 1000 points.

Fig. 5 shows the cumulative error plots in terms of the absolute registration error (in meters) and the relative registration error. The latter is defined as the registration error divided by the largest distance between any two model points. We can see that ICDO achieved the best result in both measures. Recall that ICDO was trained with synthetic data synthesized from 7 shapes, which have no similarity to the data in this section. This demonstrates the potential of ICDO as a robust learning-based PCReg algorithm which can generalize to different classes of objects. In terms of the average computation time, we have ICP at 0.06s, IRLS at 0.35s, CPD at 1.62s, GMR at 18.66s, and ICDO at 2.14s.

5. Conclusion

We proposed Inverse Composition Discriminative Optimization (ICDO) for Point Cloud Registration (PCReg). ICDO learns a set of maps from a feature vector to an update vector, which is inversely composed with the previous estimates. We also derived a feature function where its dimension is independent of the dimension of the point cloud, and

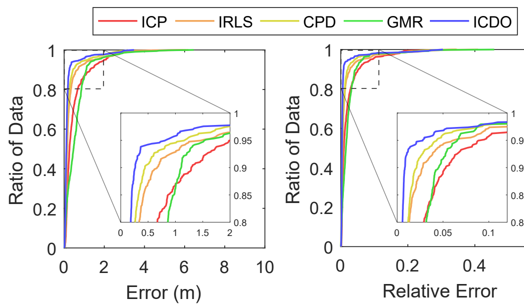
Figure 5. Results of ETH laser registration dataset in cumulative plots. (Left) Absolute registration error. (Right) Relative error.

show that it can learn to register arbitrary shapes even when learned with different ones. We also show that ICDO is essentially learning annealing schedule, avoiding the need to set it manually. Our experiments show that ICDO can match or outperform state-of-the-art algorithms in both synthetic and real data.

A downside of ICDO is that its complexity is quadratic in the number of points, making it unsuitable for large point clouds. This issue may be resolved by subsampling, or learning the weights to reduce the number of relevant points like in [7]. In addition, since ICDO is similar to subgradient method (Sec. 3.5), its convergence can be slow [4]. Finding a way to estimate step sizes, similar to line search, could lead to a fewer number of iterations required to converge.

Acknowledgments This research was supported in part by Fundação para a Ciência e a Tecnologia (project FCT [SFRH/BD/51903/2012] and a PhD grant from the Carnegie Mellon-Portugal program), the National Science Foundation under the grants RI-1617953, and the EU-Horizon 2020 project #731667 (MULTIDRONE). The content is solely the responsibility of the authors and does not necessarily represent the official views of the supporting agencies.

References

[1] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *IJCV*, 56(3):221–255, 2004. 3

[2] P. Bergström and O. Edlund. Robust registration of point sets using iteratively reweighted least squares. *Computational Optimization and Applicat.*, 58(3):543–561, 2014. 2, 6

[3] P. J. Besl and H. D. McKay. A method for registration of 3-D shapes. *IEEE TPAMI*, 14(2):239–256, 1992. 2, 6

[4] S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods. Lecture Notes of EE392o, Stanford University, 2003. 4, 5, 8

[5] J. Briales and J. Gonzalez-Jimenez. Convex global 3d registration with lagrangian duality. In *CVPR*, 2017. 2

[6] M. Brown, D. Windridge, and J.-Y. Guillemaut. Globally optimal 2d-3d registration from points or lines without correspondences. In *ICCV*, 2015. 2

[7] D. Campbell and L. Petersson. An adaptive data representation for robust point-set registration and merging. In *ICCV*, 2015. 1, 2, 7, 8

[8] D. Campbell and L. Petersson. GOGMA: Globally-optimal gaussian mixture alignment. In *CVPR*, 2016. 2

[9] A. Censi. An icp variant using a point-to-line metric. In *ICRA*, 2008. 2

[10] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992. 2

[11] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996. 6, 7

[12] P. Dollár, P. Welinder, and P. Perona. Cascaded pose regression. In *CVPR*, 2010. 2

[13] G. Elbaz, T. Avraham, and A. Fischer. 3d point cloud registration for localization using a deep neural network auto-encoder. In *CVPR*, 2017. 2

[14] A. W. Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13):1145–1153, 2003. 2

[15] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Proc. Eurographics Symposium Geometry Processing*, 2005. 2

[16] S. Gold, A. Rangarajan, C.-P. Lu, P. Suguna, and E. Mjolsness. New algorithms for 2D and 3D point matching: pose estimation and correspondence. *Pattern Recognition*, 38(8):1019–1031, 1998. 1, 2

[17] V. Golyanik, S. Aziz Ali, and D. Stricker. Gravitational approach for point set registration. In *CVPR*, 2016. 2

[18] S. Granger and X. Pennec. Multi-scale em-icp: A fast and robust approach for surface registration. In *ECCV*, 2002. 2

[19] L. Greengard and J. Strain. The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991. 5

[20] B. Jian and B. C. Vemuri. Robust point set registration using gaussian mixture models. *IEEE TPAMI*, 33(8):1633–1645, 2011. 1, 2, 3, 6, 7

[21] H. Lei, G. Jiang, and L. Quan. Fast descriptors and correspondence propagation for robust global point cloud registration. *IEEE TIP*, 2017. 2

[22] K. L. Low. Linear least-squares optimization for point-to-plane icp surface registration. Technical Report TR04-004, Department of Computer Science, University of North Carolina, Chapel Hill, 2004. 2

[23] H. Maron, N. Dym, I. Kezurer, S. Kovalsky, and Y. Lipman. Point registration via efficient convex relaxation. *ACM Transactions on Graphics (TOG)*, 35(4):73, 2016. 2

[24] H. Men, B. Gebre, and K. Pochiraju. Color point cloud registration with 4d icp algorithm. In *ICRA*, 2011. 2

[25] A. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *IJCV*, 89(2):348–361, 2010. 6

[26] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE TPAMI*, 32(12):2262–2275, 2010. 2, 6

[27] Y. Nesterov. *Introductory Lectures on Convex Optimization*. Springer, 2004. 4

[28] D. Pani Paudel, A. Habed, C. Demonceaux, and P. Vasseur. Robust and optimal sum-of-squares-based point-to-plane registration of image sets and structured scenes. In *ICCV*, 2015. 2

[29] J. Park, Q.-Y. Zhou, and V. Koltun. Colored point cloud registration revisited. In *CVPR*, 2017. 2

[30] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart. Challenging data sets for point cloud registration algorithms. *The International Journal of Robotics Research*, 31(14):1705–1711, Dec. 2012. 7

[31] E. Rodolà, A. Albarelli, F. Bergamasco, and A. Torsello. A scale independent selection process for 3d object recognition in cluttered scenes. *IJCV*, 102(1-3):129–145, 2013. 7

[32] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. Int. Conf. 3-D Digital Imaging and Modeling*, 2001. 2

[33] N. Z. Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer-Verlag Berlin Heidelberg, 1985. 5

[34] Y. Tsin and T. Kanade. A correlation-based approach to robust point set registration. In *ECCV*, 2004. 2, 3

[35] O. Tuzel, F. Porikli, and P. Meer. Learning on lie groups for invariant detection and tracking. In *CVPR*, 2008. 2

[36] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE TPAMI*, 13(4):376–380, 1991. 2

[37] J. Vongkulbhisal, F. De la Torre, and J. P. Costeira. Discriminative optimization: Theory and applications to computer vision problems. *arXiv preprint arXiv:1707.04318*, 2017. 2, 3, 4, 5

[38] J. Vongkulbhisal, F. De la Torre, and J. P. Costeira. Discriminative optimization: Theory and applications to point cloud registration. In *CVPR*, 2017. 1, 2, 6, 7

[39] X. Xiong and F. De la Torre. Supervised descent method and its application to face alignment. In *CVPR*, 2013. 2

[40] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-ICP: a globally optimal solution to 3D ICP point-set registration. *IEEE TPAMI*, 38(11):2241–2254, 2016. 2

[41] Q.-Y. Zhou, J. Park, and V. Koltun. Fast global registration. In *ECCV*, 2016. 2