Meta Learning for Point Cloud Analysis

by

Ahmed Hatem

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

MASTER OF SCIENCE

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
April 2023

Thesis advisor

Dr. Yang Wang

Author

Ahmed Hatem

Meta Learning for Point Cloud Analysis

Abstract

Point cloud has been highly attracting the attention of the research community, due to their numerous applications in 3D computer vision. While learning-based approaches for point cloud problems have achieved impressive progress, generalization to unknown testing environments remains a major challenge due to the large discrepancies of data captured by different 3D sensors. Existing methods typically train a generic model and the same trained model is applied on each test instance. This could be sub-optimal since it is difficult for the same model to handle all the variations during testing. In this thesis, we propose novel frameworks for point cloud problems that adapt the model in an instance-specific manner during inference. Our model is trained using a meta-learning scheme to provide the model with the ability of fast and effective adaptation at test time. First, we consider the problem of point cloud registration. The objective is to estimate the 3D transformation that aligns a pair of partially overlapped point clouds. Next, we investigate the point cloud upsampling problem. In this setting, the goal is to generate high-resolution point clouds from sparse point clouds. Experimental results demonstrate the effectiveness of our proposed frameworks in improving the performance of state-of-the-art models and achieving superior results.

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my supervisor, Dr. Yang Wang, for his continuous encouragement and mentoring throughout the master's research. His immense knowledge and insightful comments were vital to the completion of my thesis. It was my great pleasure to work under his supervision.

I am deeply thankful to Dr. Yiming Qian for his keen guidance and invaluable advice throughout the conducted research, who so generously took a lot of time out of his schedule to participate in our research.

Also, it is an honor to have Dr. Lorenzo Livi and Dr. Noman Mohammed on my thesis committee. I am grateful for their constructive feedback and invaluable suggestions for bringing my thesis to perfection.

I would like to express my sincere gratitude to Dr. Yang Wang for the financial assistance and unconditional support throughout my journey.

Last but not least, I am truly grateful to my family, who always believed in me and pushed me forward to success.

This thesis is dedicated to my family for their endless love and unconditional support.

Publications

Some of the ideas, materials, and figures in this thesis have appeared previously in the following submitted manuscripts:

- **Ahmed Hatem**, Yiming Qian, and Yang Wang. Point-TTA: Test-Time Adaptation for Point Cloud Registration Using Multitask Meta-Auxiliary Learning. *IEEE International Conference on Computer Vision*, 2023. (Under review)

- **Ahmed Hatem**, Yiming Qian, and Yang Wang. Test-Time Adaptation for Point Cloud Upsampling Using Meta-Learning. *IEEE International Conference on Intelligent Robots and Systems*, 2023. (Under review)

Chapter 1

Introduction

With the recent and rapid advancement of 3D acquisition technologies, computer vision has benefited from the increased availability of various scanning devices, such as LiDARs and RGB-D cameras. The captured 3D data is valuable in facilitating scene understanding and decision-making in visual computing systems. Among different 3D data representations, point cloud becomes one of the most popular representations by providing rich geometric information in 3D space. Point clouds are sets of 3D coordinates that accurately represent the surface geometry of scenes and objects in the real world. They play a vital role in a variety of applications, such as autonomous driving [11], augmented reality [7], and robotics [41]. Recently, the emergence of deep learning models has achieved promising results in solving various point cloud problems [55, 16, 88, 46]. However, most learning-based methods have followed a fully supervised learning paradigm. They assume that the training and test data are sampled from the same data distribution. This is unrealistic for real-world scenarios due to the fact that the data captured by different 3D sensors have large discrepancies.

It is challenging for the training data to cover all the variations that can happen during testing. Consequently, trained models usually experience a drastic drop in performance when they are evaluated on unknown test distributions.

In this thesis, we address the above limitation by introducing novel adaptation frameworks for point cloud problems to overcome the out-of-distribution generalization limitations of existing methods and effectively adapt to unseen test data distributions.

Many contemporary methods have been proposed to address the distribution shift issue by minimizing the gap between training and test distributions, e.g. using adversarial learning [58, 79, 19, 37] or self-supervised learning [47, 22, 21, 81]. However, these approaches have some limitations. First, they assume having access to unlabeled samples of the target test distribution during training. This may not be feasible in real-world settings where the information from test distribution is not available in advance. In addition, they do not fully utilize the useful internal information available within the test instance, since the trained model parameters are fixed during inference time for all unseen test instances. In contrast, our proposed approaches do not require prior knowledge about the test data distributions. We adapt the model parameters in an instance-specific manner during inference and obtain a different set of network parameters for each different instance. This allows our model to better capture the uniqueness of each test instance and thus generalize better to unseen data.

First, we consider the problem of point cloud registration. The task is to estimate the 3D transformation that accurately aligns a pair of point clouds, which are usually

acquired from two consecutive frames and have overlapping surface regions. This is a critical step for many applications such as 3D reconstruction, pose estimation, and object detection. Traditional algorithms [25, 93, 80, 12] have been proposed for point cloud registration. However, these approaches take a long time to converge and their accuracy drops in the presence of high outliers. Recently, learning-based approaches [52, 17, 16, 43, 4] have been introduced to overcome the traditional methods limitations. These approaches first learn a model on a labeled dataset. Then the model is fixed when evaluating on unseen test data. The single set of model parameters may not be optimal for different test environments captured with different 3D scanners due to the domain shift. Our proposed method is inspired by the success of test-time training (TTT) [73] in image classification, where an auxiliary task is used to update the model parameters at inference time to learn feature representation specifically for a test instance. Auxiliary learning has been shown to be effective to improve a predefined primary task and is used in multiple 2D computer vision tasks [49, 50, 73, 13]. Recently, there have been some attempts at using auxiliary tasks for improving the representation learning of point clouds [1, 36, 68, 22]. However, using auxiliary tasks for test-time adaptation is still largely unexplored for point cloud data. In this thesis, we introduce three self-supervised auxiliary tasks: point cloud reconstruction, correspondence classification, and contrastive learning. These auxiliary tasks do not require any extra supervision. Given a test instance, we adapt our model using these auxiliary tasks and the updated model is used to perform the inference. Some recent work [13] has shown that naively training the primary task and the auxiliary task together may not be optimal, since the model may be biased towards improving the

auxiliary task rather than the primary task. Therefore, we propose to use meta-learning for a fast and effective adaptation of the model at test time. Meta-learning, also known as learning to learn, has shown great success in learning new tasks quickly with few training samples. In particular, Model-Agnostic Meta-Learning (MAML) [23] has been widely employed for various domain adaptation tasks [53, 14, 13, 49]. MAML [23] is an optimization-based method that aims to learn the model parameters in a way that facilitates fast adaptation at test time within a few gradient updates. We introduce a meta-auxiliary training approach based on MAML [23] to enforce the adapted model via auxiliary tasks to effectively improve the performance of the primary registration task.

Second, we study an essential problem of many downstream applications, namely point cloud upsampling. Point clouds obtained from affordable 3D scanners are usually sparse and non-uniform. Therefore, these sparse point clouds need to be effectively upsampled to produce denser point clouds in order to be used in different downstream applications. Given an input sparse point cloud, our goal is to generate a high-resolution uniform point cloud that adequately represents the underlying surface. We propose a novel solution for this problem beyond the traditional supervised approaches [88, 56, 46, 57, 2]. We introduce a meta-learning approach to learn the model parameters in a way that facilitates fast adaptation at test time within a few gradient updates. We adopt MAML [23] for training the point cloud upsampling networks. During meta-training, each input point cloud is downsampled by a predefined scaling factor and the MAML [23] task is the reconstruction of the input point cloud. At test time, the model is updated by a few gradient updates based on

a self-supervised learning procedure that exploits the internal information of the test point cloud.

1.1 Contributions

Our key contributions are summarized as follows:

- We introduce a novel test-time adaptation framework for point cloud registration using multitask meta-auxiliary learning. We design three self-supervised auxiliary tasks to effectively extract useful features from test instances and adapt the model to unseen test distribution to improve generalization. A meta-auxiliary learning paradigm is used to learn the model parameters, such that adapting the model parameter via the auxiliary tasks during testing improves the performance of the primary task.

- We propose a novel adaptation framework for point cloud upsampling that exploits the complementary advantages of both internal and external learning for point cloud upsampling. We propose to employ meta-learning to allow fast and effective adaptation of model parameters at inference time.

- More importantly, the proposed frameworks are generic and can be applied in a plug-and-play manner with learning-based networks. Extensive experiments validate the effectiveness of the proposed methods in boosting the performance of state-of-the-art models.

1.2 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we discuss the related work and relevant literature. More specifically, we review existing approaches in point cloud registration, point cloud upsampling, domain adaptation, and meta and auxiliary learning. In Chapter 3, we describe the proposed meta-auxiliary framework for point cloud registration. We show that leveraging the advantages of two machine learning paradigms, including meta-learning and auxiliary-learning is beneficial for effective adaptation at test-time. In Chapter 4, we explain the proposed test-time adaptation approach for point cloud upsampling. Here, we propose a meta-learning algorithm to effectively utilize the internal and external features of point clouds to improve the generalization capability of models. Finally, we conclude the thesis in Chapter 5.

Chapter 2

Related Work

Our work is closely related to several lines of research. We briefly review prior work closest related to ours.

2.1 Point Cloud Registration.

Most traditional point cloud registration approaches consist of two modules: feature-based correspondence matching and outlier filtering.

Feature descriptors have been proposed to effectively extract the local and global features of point clouds, which are used to match correspondences in the feature space. Traditional methods use hand-crafted features such as spatial features histogram [38, 75, 26], or geometric features histogram [10, 63]. Recently, learning-based approaches have been proposed to learn 3D feature descriptors including fully convolution methods [28, 17], keypoint detection methods [5, 83, 44], and coarse-to-fine methods [86, 59].

The correspondences obtained from feature matching often include many outliers that must be filtered out for robust point cloud registration. Many traditional approaches [25, 93, 80, 12] have been proposed for robust outlier filtering of correspondences. RANSAC [25] is the most popular method, where a set of correspondences are iteratively sampled to filter outliers. RANSAC variants [69, 18, 6] have been introduced to provide new sampling strategies for fast convergence. However, these methods still have slow convergence rate and low performance in the presence of high outliers. Other methods use robust cost functions that are more effective with high outlier ratio. FGR [93] uses the Geman-McClure cost function and TEASER [80] uses the truncated least squares cost function for robust point cloud registration.

In recent years, deep learning techniques have been employed for outlier filtering. The 3D outlier filtering approaches [52, 16, 43, 4] follow similar ideas in 2D image matching [84, 90, 72], where outlier filtering is defined as an inlier classification problem. 3DRegNet [52] uses the 2D correspondence selection network [84] for point clouds and added a regression module for rigid transformation. DGR [16] proposes a fully convolutional network to better capture global features of correspondences and predict the inlier confidence of each correspondence. DHVR [43] leverages Hough voting in 6D transformation parameter space to identify the confidence of correspondences from Hough space to predict the final transformation. PointDSC [4] uses the spatial consistency between inlier correspondences to better prune the outliers.

2.2 Point Cloud Upsampling

Early work has proposed optimization-based methods [3, 48, 33, 34] for densifying point clouds. Alexa et al. [3] generate dense points by inserting points at the Voronoi diagram's vertices. Lipman et al. [48] introduce a locally optimal projection operator for resampling point clouds based on L1 norm. Later, an improved weighted version of the local optimal projection operator was proposed in [33]. However, these methods usually do not work well around sharp edges. Huang et al. [34] propose an edge-aware resampling algorithm that progressively transfers generated points towards the edge singularities to preserve edge sharpness.

Ever since the emergence of deep learning, recent work has shifted to upsampling point clouds using deep neural networks. PU-Net [88] is the first to apply deep learning in point cloud upsampling. It uses Point-Net++ [55] for feature extraction and expands point features using multi-branch convolutions in feature space. EC-Net [87] proposes an edge-aware point cloud upsampling by directly minimizing distances from points to edges. MPU [85] proposes a patch-based network that learns different levels of point cloud features by progressively upsampling the points in multiple steps. PU-GAN [45] adopts a generative adversarial network to generate high-quality upsampled points. PU-GCN [56] proposes a graph convolutional network for point cloud upsampling. PUGeo-Net [57] incorporates differential geometry to improve point cloud upsampling performance by learning the local geometry of point clouds. Dis-PU [46] introduces disentangled refinement units for point upsampling using two sub-networks, including a dense point generator and a point spatial refiner. PU-Dense [2] proposes a novel feature extraction unit for extracting 3D multiscale features and

adopts U-Net architecture based on sparse convolutions for computationally efficient processing of point clouds.

2.3 Domain Adaptation

Extensive work has been proposed for 2D image domain adaptation [8, 65, 39, 77, 64]. Recently, there is also work exploring domain adaptation for point clouds. Most existing domain adaptation approaches on point clouds [58, 78, 66, 79] mainly rely on adversarial learning to transfer knowledge from labeled source domain to unlabeled target domain. Qin et al. introduce PointDAN [58] to jointly align local and global features of point cloud distributions across different domains for 3D classification. Wang et al. [78] propose a cross-range adaptation to enhance far-range 3D object detection performance. Saleh et al. [66] adopt CycleGAN [94] to adapt projected 2D bird's eye view synthetic images for real-world vehicle detection. Wu et al. [79] use geodesic correlation alignment for minimizing the domain gap between synthetic and real data.

Recently, several studies have proposed to design self-supervised tasks [1, 68, 22, 21] for learning domain invariant features of point clouds. [1] introduces a deformation reconstruction task to learn the underlying structures of 3D objects. [68] defines the self-supervised task as a reconstruction of random partially displaced point clouds. [22] proposes two self-supervised tasks, including a scale prediction task and a 3D/2D projection reconstruction task to transfer global and local features across domains.

The main limitation of most existing domain adaptation approaches is the assumption of the availability of unlabelled target domain data during training, which

is infeasible when the target domain is unknown during training.

2.4 Auxiliary and Meta Learning.

In auxiliary learning, an auxiliary task (often self-supervised) is defined to improve the performance and generalization of a target primary task. This differs from multi-task learning where the goal is to improve performance across all tasks [49]. Auxiliary learning has been proven to be effective in multiple 2D image domain problems [49, 50, 73, 13]. Sun et al. [73] uses the image rotation prediction as a self-supervised auxiliary task to improve image classification. Chi et al. [13] uses image reconstruction as the auxiliary task to improve the primary task of deblurring. Auxiliary learning has also been studied for point clouds [54, 68, 36]. Poursaeed et al. [54] proposes an auxiliary task that estimates the orientation of point clouds to learn complementary features for 3D representations. Huang et al. [36] adopts a contrastive auxiliary task [30] for better representation of point clouds.

Meta-learning has been successfully applied in many computer vision and robotics applications. Existing meta-learning methods can be categorized into model-based [67, 62, 51, 82, 31], metric-based [70, 76, 40, 74] and optimization-based methods [23, 29, 24, 61]. Model-based methods learn to update their model parameters with a few steps either using another meta-learner network for parameters prediction [82, 31] or via its internal architecture [67]. Metric-based methods learn a metric function to measure the similarity between samples. Optimization-based methods learn an optimal model initialization that can rapidly adapt to new tasks.

MAML [23] is a widely used optimization-based algorithm, which has been suc-

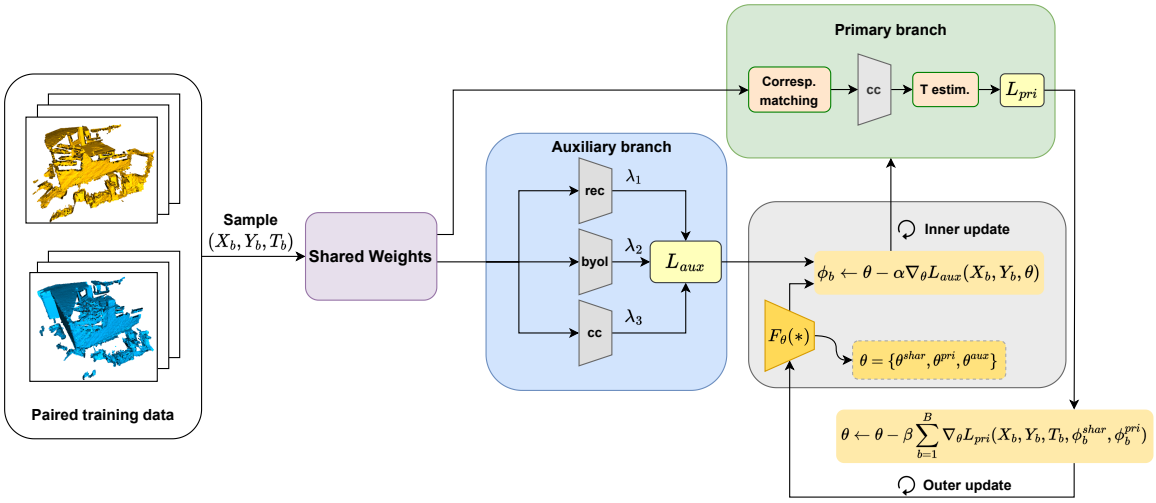
cessfully adopted to many 2D image domain tasks [91, 92, 13, 53, 71]. Zhang et al. [91] propose MetaGaN that integrates the MAML algorithm with GAN network to improve image classification performance with a few number of training samples. Chi et al. [13] introduce meta-auxiliary learning framework based on MAML for image deblurring to enable fast model adaptation. [53, 71] use MAML for image super-resolution, which learns effective pre-trained model weights that can quickly adapt to unseen test images. Relatively little meta-learning work has been proposed for point clouds [82, 31, 32]. Hai et al. [31] introduce a parameter prediction network for point cloud segmentation to enable fast adaptation to new part segmentation tasks. Ye et al. [82] propose a meta-subnetwork for point cloud upsampling that is trained to dynamically adjust the upsampling network parameters to support flexible scale factors.

Chapter 3

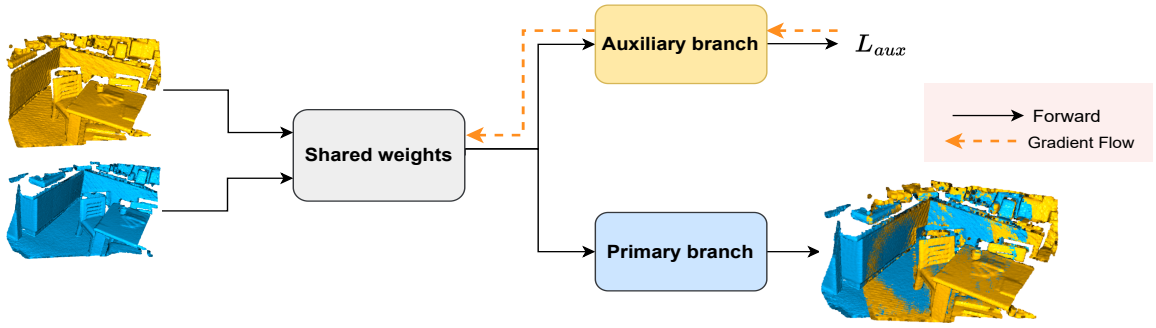
Meta-Auxiliary Learning for Adaptation in Point Cloud Registration

3.1 Problem Statement

Point cloud registration merges two individual scans that are captured from different viewpoints and have overlapping surface regions to precisely align them together into one point cloud. The alignment is done by finding the rigid transformation between the two scanned point clouds, which controls the amount of rotation and translation needed to be applied on one point cloud to be accurately registered with the other point cloud. Given a pair of partially overlapping point clouds $X \in R^{M\times 3}$ with M points and $Y \in R^{N\times 3}$ with N points, our goal is to find an optimal 3D transformation T between the two point clouds that accurately aligns them. Our



(a) Meta-train



(b) Meta-test

Figure 3.1: Overview of the proposed meta-auxiliary framework. (a) Meta-train. Given a pair of input point clouds during training, we first adapt the model by performing a small number of gradient updates using the auxiliary loss calculated via three auxiliary tasks including point cloud reconstruction, BYOL and, correspondence classification. Then the adapted model is used to perform the primary registration task and is evaluated using the meta-objective. Finally, we update the model using the primary loss. (b) Meta-test. At test-time, we use the auxiliary branch to fine-tune the model for each test instance using the auxiliary loss and the adapted model is used to register the input point clouds.

goal is to learn a model $F_\theta(X, Y) \rightarrow T$ parameterized by θ that maps (X, Y) to T. In this work, we propose a framework for point cloud registration that adapts the trained model parameters to each different input at test time, so that our model can improve generalization and performance of point cloud registration. The adaptation is achieved via self-supervised auxiliary tasks.

3.2 Our Approach

We first explain the proposed three self-supervised auxiliary tasks in section 3.2.1. We then describe the model architecture, including the primary branch and auxiliary branch in section 3.2.2. Finally, we present the proposed meta-auxiliary learning approach in section 3.2.3.

3.2.1 Auxiliary Tasks

We propose three different auxiliary tasks in our work. All these auxiliary tasks are self-supervised and do not require extra labels. So they can be used during test-time for adaptation.

Point Cloud Reconstruction. Inspired by the success of using image reconstruction as the auxiliary task [13, 50], we propose to use point cloud reconstruction as one of our self-supervised auxiliary tasks. Given a point cloud P, the features of the point cloud are extracted using the feature encoder. Then, a decoder is used to reconstruct the point cloud P'. Adapting the model parameters at test time using the reconstruction auxiliary loss enables the model to take advantage of the internal features of the test instance before performing the primary task. The reconstruction loss does not

require any supervision, which makes it suitable for test-time adaptation. We use $L1$ reconstruction loss as follows:

$$\ell_{rec} = ||P - P'||_1. \tag{3.1}$$

Self-Supervised Feature Learning. Self-supervised learning (SSL) is an active area of research. The main idea of SSL is to define some proxy self-supervised tasks to learn feature representations from data without manual annotations. We can use any existing SSL task as one of our auxiliary tasks. In our work, we adapt BYOL [30] as our self-supervised task. Different from constrastive learning, BYOL does not require negative samples. This makes it suitable for test-time adaptation. The model architecture of BYOL consists of two networks, namely the online network and the target network. Each network predicts a representation of an augmented view of the same point cloud. The idea is to train the online network to predict representations similar to the target network's predictions, so that the representations of the two augmented views are closely similar.

The online network parameterized by θ consists of a feature encoder f_θ, a feature projector z_θ and a predictor p_θ. Similarly, the target network parameterized by ξ has a feature encoder f_ξ and a feature projector z_ξ . The online network θ is trained based on the regression targets provided by the target network, while the target network ξ is the exponential moving average of the online parameters θ:

$$\xi \leftarrow \tau\xi + (1-\tau)\theta, \tag{3.2}$$

where $\tau \in [0, 1]$ is the target decay rate.

Given a point cloud P, we perform augmentation to produce two augmented versions P_v and $P_{v'}$. The point P_v is passed to the online network to obtain the

projection $z_\theta = g_\theta(P_v)$ and $P_{v'}$ is passed to the target network to obtain the projection $z_\xi = g_\xi(P_{v'})$. Then we minimize the mean squared error between the normalized predictions $q_\theta(z_\theta)$ and target projections z_ξ as follows:

$$L_{\theta,\xi} = 2 - \frac{2q_\theta(z_\theta)^\top z_\xi}{\|q_\theta(z_\theta)\|^2 \|z_\xi\|^2}. \tag{3.3}$$

We define another symmetric loss $L'_{\theta,\xi}$ by similarly passing $P_{v'}$ to the online network and P_v to the target network to compute $L'_{\theta,\xi}$. The final BYOL loss is defined as:

$$\ell_{byol} = L_{\theta,\xi} + L'_{\theta,\xi}. \tag{3.4}$$

Correspondence Classification. We introduce an additional self-supervised auxiliary task designed specifically for point cloud registration. Given a point cloud P, we construct an augmented point cloud P' using a randomly generated 3D transformation T by sampling a random rotation within [0°..360°] and a random translation within [0cm..60cm]. The sampled transformation T is applied on each axis of point cloud to obtain P'. The feature encoder is used to extract the features of P and P'. Then these two sets of points are matched in the feature space using nearest neighbors to obtain the correspondences. Using the same outlier rejection network architecture of the primary task, this auxiliary task is trained to predict whether a correspondence is an inlier or an outlier. Since the transformation T of the point cloud is known, the ground-truth inlier correspondences C are available and the auxiliary loss does not require any manual supervision. Similar to [16, 4], the classification loss is defined as the binary cross entropy loss between the probability $p^i_{(i,j)}$ that a correspondence

$C_{(i,j)}$ is an inlier and the ground-truth inliers C.

$$\ell_{cc} = \frac{1}{|M|}\Big(\sum_{(i,j)\in C} \log p^i_{(i,j)} + \sum \log p^o_{(i,j)}\Big), \tag{3.5}$$

where $p^o = 1 - p^i$.

3.2.2 Model Architecture

Our model architecture consists of a shared feature encoder and two branches for the primary and auxiliary tasks. The primary branch corresponds to the point cloud registration task. The auxiliary branch corresponds to three self-supervised auxiliary tasks defined in Section 3.2.1. We denote the model parameters as $\theta = \{\theta^{shar}, \theta^{pri}, \theta^{aux}\}$, where θ^{shar} corresponds to the shared feature encoder, θ^{pri} is the primary branch and θ^{aux} is the auxiliary branch. Note that θ^{aux} represents the parameters of three auxiliary tasks.

Auxiliary Tasks. Our aim of the auxiliary tasks is to transfer rich and useful knowledge to improve the performance of the primary task. The overall auxiliary loss is the weighted sum of the losses for the three auxiliary tasks:

$$L_{aux} = \lambda_1 \ell_{rec} + \lambda_2 \ell_{byol} + \lambda_3 \ell_{cc}. \tag{3.6}$$

Instead of fixing the values of the balancing weights λ_i $(i = 1, 2, 3)$, we treat them as learnable parameters and learn their values during training. This allows the learning algorithm to automatically choose the right weights that balance the relative importance of each auxiliary task.

To train both primary and auxiliary tasks, we first follow the joint training approach in [73]. The loss of the joint training is simply the combination of the primary

and auxiliary losses:

$$L_{pri}(\theta^{shar}, \theta^{pri}; X, Y, T) + L_{aux}(\theta^{shar}, \theta^{aux}; X, Y). \tag{3.7}$$

Note that since our auxiliary tasks are self-supervised, the auxiliary loss $L_{aux}(\cdot)$ does not need the ground-truth transformation T. To simplify the notation, we have assumed one training instance in Eq. 3.7. It is straightforward to generalize Eq. 3.7 to the entire training set by summing over all training instances.

The model learned from Eq. 3.7 is then used as the initialization for the meta-auxiliary learning.

Primary Task. We follow the standard learning-based network architecture for point cloud registration. First, a pair of point clouds are passed to a fully convolutional network to extract the corresponding geometric pointwise features. Then the points are matched using the nearest neighbor in the feature space to obtain correspondences. These correspondences are fed to an outlier rejection network which predicts the confidence of each correspondence. Finally, given the correspondences with their associated probability weights resulting from the outlier rejection network, the weighted procrustes approach [16] is used to align the paired 3D scans by estimating the transformation between the two point clouds. We use $L_{pri}(\theta^{shar}, \theta^{pri}; X, Y, T)$ to denote the loss function that measures the difference between the ground-truth transformation T and the prediction $F_\theta(X, Y)$.

In this work, we use Fully Convolutional Geometric Features (FCGF) [17] to extract pointwise features of the point clouds. For the outlier rejection network, we adopt the architecture of three state-of-the-art methods including DGR [16], DHVR [43] and PointDSC [4]. However, it is important to note that our proposed meta-

Algorithm 1 Meta-auxiliary training

Require: X, Y, T: training pairs with their transformation

Require: α, β: learning rates

Output: θ: learned parameters

1: Initialize the network with pre-trained weights θ

2: **while** not done **do**

3: Sample a training batch $\{X_b, Y_b, T_b\}_{b=1}^{B}$

4: **for** each example **do**

5: Evaluate the three auxiliary tasks:

$L_{aux} = \lambda_1 \ell_{rec} + \lambda_2 \ell_{byol} + \lambda_3 \ell_{cc}$

6: Compute adapted parameters via gradient descent: $\phi_b \leftarrow \theta - \alpha \nabla_\theta L_{aux}(X_b, Y_b, \theta)$

7: Update auxiliary branch:

$\theta^{aux} \leftarrow \theta^{aux} - \alpha \nabla_\theta L_{aux}(X_b, Y_b, \theta^{aux})$

8: **end for**

9: Evaluate the primary task using the adapted parameters and update:

$\theta \leftarrow \theta - \beta \sum_{b=1}^{B} \nabla_\theta L_{pri}(X_b, Y_b, T_b, \phi_b^{shar}, \phi_b^{pri})$

10: **end while**

11: **return** θ

auxiliary framework is agnostic to these choices and can be applied to any learning-based point cloud registration methods.

3.2.3 Meta-Auxiliary Learning

Our goal is to combine self-supervised auxiliary tasks along with the point cloud registration task to quickly adapt the model parameters for each test instance without the need for any extra supervision. Although jointly training the primary and auxiliary tasks will improve the generalization of our model to unknown test distribution, the updated parameters using auxiliary loss may be more biased during training to improve the auxiliary task and not the primary task. Following [13], we propose to use a meta-auxiliary training scheme.

Training. We use meta-learning to train the model parameters θ to be quickly adaptable to different test distribution data, such that updating model parameters at test-time improve the primary point cloud registration task.

Given a batch of paired point clouds X_b, Y_b, and the pre-trained model parameters θ resulted from jointly training primary and auxiliary tasks. We perform adaptation for small gradient updates using the auxiliary loss, in which all model parameters ($\phi_b^{shar}, \phi_b^{pri}, \phi_b^{aux}$) are updated:

$$\phi_b \leftarrow \theta - \alpha \nabla_\theta L_{aux}(X_b, Y_b, \theta), \tag{3.8}$$

where α is the adaptation learning rate. Note that since Eq. 3.8 is based on the auxiliary task, the adaptation can be done at test-time since it does not require the ground-truth tranformation.

Then, the adapted model($\phi_b^{shar}, \phi_b^{pri}$) will be used to perform the primary task and calculate the primary loss. This will enforce the adapted model to boost the primary task performance. The primary loss will be used to optimize the model

parameters θ as:

$$\theta \leftarrow \theta - \beta \sum_{b=1}^{B} \nabla_{\theta} L_{pri}(X_b, Y_b, T_b, \phi_b^{shar}, \phi_b^{pri}), \tag{3.9}$$

where β is the meta-learning rate and B is the batch size. Note that $L_{pri}(\cdot)$ in Eq. 3.9 is defined in terms of the updated model ϕ_b for each instance, while the optimization is performed on the model parameters θ. The training process is summarized in Algorithm 1 and Figure 3.1.

Testing. During test-time, the optimized meta-learned parameters θ are adapted to a test instance that consists of a pair of point clouds using the auxiliary loss as follows:

$$\phi \leftarrow \theta - \alpha \nabla_{\theta} L_{aux} \tag{3.10}$$

Then, the adapted model $(\phi^{shar}, \phi^{pri})$ is used to perform point cloud registration.

3.3 Experiments

We first evaluate our method on a 3D indoor dataset for the pairwise registration task. Then we analyze the generalization of our proposed model to unseen 3D outdoor dataset. Additionally, we integrate our method into a multi-way registration pipeline and evaluate its performance on generating final 3D reconstruction scenes. Finally, we perform extensive ablation studies to inspect each component of our approach.

3.3.1 Experimental Setup

Dataset. We use the 3DMatch benchmark [89] for indoor pairwise registration. It consists of point cloud pairs with corresponding ground-truth transformations from

real-world indoor scenes scanned by commodity RGB-D sensors. We follow the standard splitting strategy and evaluation protocol in 3DMatch, where the test data contain 1623 partially overlapping point cloud scans from 8 different indoor scenes. For the outdoor dataset, we use the KITTI odometry benchmark [27] which consists of 3D outdoor scenes scanned using a Velodyne laser. We follow the train/test split in [17] to create pairwise splits, since the official benchmark does not have labels for pairwise registration. We perform voxel downsampling to generate point clouds with uniform density and set voxel size to 5cm for indoor dataset and 30cm for outdoor dataset. For multi-way registration experiment, we use the simulated Augmented ICL-NUIM dataset [15] which contains augmented indoor reconstruction scenes from RGB-D videos.

Evaluation Metrics. Following [16, 4], we report Registration Recall (RR), Rotation Error (RE) and Translation Error (TE). RE and TE are defined as:

$$RE = \arccos \frac{Tr(R^T R^*) - 1}{2}, TE = ||t - t^*||^2, \tag{3.11}$$

where R^* and t^* are the ground-truth rotation and translation, respectively. Registration Recall (RR) is the ratio of successful pairwise registration that its rotation error and translation error are below predefined thresholds. These thresholds are set to ($RE = 15$, $TE = 30cm$) for indoor scenes and ($RE = 5$, $TE = 60cm$) for outdoor scenes. Then the average RE and TE are measured on successfully aligned pairs.

Implementation Details. We implement our framework in PyTorch and use the official implementation of DGR [16], DHVR [43], and PointDSC [4] as the backbones of our approach. We first jointly train primary and auxiliary tasks by optimizing the

Table 3.1: Comparison with other state-of-the-art methods on the 3DMatch dataset [89]. ↑ (or ↓) indicates that a higher (or lower) number means better performance.

	Recall ↑	RE (deg) ↓	TE (cm) ↓
FGR [93]	78.56	2.82	8.36
TEASER [80]	85.77	2.73	8.66
GC-RANSAC [6]	92.05	2.33	7.11
RANSAC-1M [25]	88.42	3.05	9.42
RANSAC-2M [25]	90.88	2.71	8.31
RANSAC-4M [25]	91.44	2.69	8.38
CG-SAC [60]	87.52	2.42	7.66
3DRegNet [52]	77.76	2.74	8.13
DGR [16]	91.30	2.40	7.48
Ours + DGR	**92.45**	**1.71**	**6.39**
DHVR [43]	91.40	2.08	6.61
Ours + DHVR	**92.28**	**1.75**	**6.42**
PointDSC [4]	92.85	2.08	6.51
PointDSC-reported [4]	93.28	2.06	6.55
Ours + PointDSC	**93.47**	**1.70**	**6.21**

loss in Eq. 3.7 using the ADAM optimizer with an initial learning rate of 10^{-4} and an exponentially decayed factor of 0.99. For meta-training, the learning rates α and β are set to $2.5 \times 10e^{-5}$. We perform 5 gradient updates during training and testing to adapt the model parameters using the auxiliary loss in Eq. 3.6. All experiments are conducted on an NVIDIA TitanX GPU.

3.3.2 Main Results

We first evaluate our method on the 3DMatch dataset [89] and report the results in Table 3.1. We compare our method with 5 traditional methods: FGR [93], TEASER [80], GC-RANSAC [6], RANSAC [25], CG-SAC [60] and 4 learning-based methods: 3DRegNet [52], DGR [16], DHVR [43], PointDSC [4]. All learning-based methods

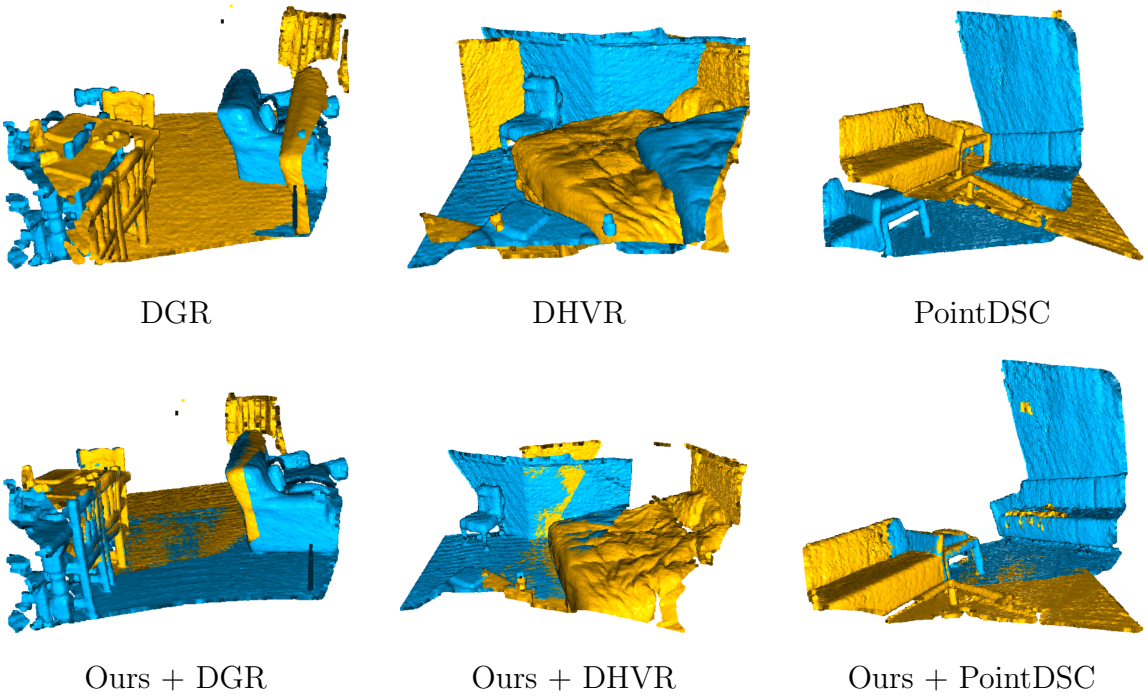


Figure 3.2: We propose a generic test-time adaptation framework that can be applied to boost standard point cloud registration pipeline. Here we show qualitative comparisons between the baselines and our method on 3DMatch dataset [89]. Our proposed framework can successfully align failure examples of DGR [16], DHVR [43], and PointDSC [4].

are trained on the 3DMatch dataset [89] and follow the same experimental setup for a fair comparison. As shown in Table 3.1, Our method improves the registration recall of DGR and DHVR by about 1%, and PointDSC by about 0.5%, as well as the RE and TE have significantly decreased for all three backbones (on average 7.3% and 21%). Our method with PointDSC as backbone outperforms all other state-of-the-art methods. Figure 3.2 shows qualitative results comparison on challenging examples of 3DMatch dataset [89] when applying our TTA method to DGR [16]. Our meta-

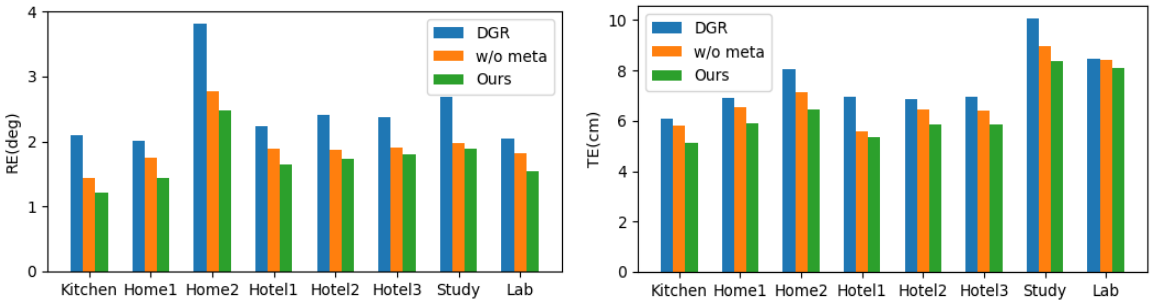


Figure 3.3: Registration results for different scenes on the 3DMatch dataset [89] in terms of RE and TE. "w/o meta" refers to a variant of our method that simply optimizes Eq. 3.7 during training without adopting the meta-auxiliary training paradigm.

auxiliary framework enables the model to better capture the internal features of each test-instance, leading to performance improvement.

Figure 3.3 shows the 3DMatch [89] registration results per scene. As an ablation study, we remove the meta learning diagram and simply optimize the joint loss Eq. 3.7 during training[1]. By doing so, we obtain superior results than the backbone DGR [16], which demonstrates that our proposed auxiliary tasks can complement the primary registration task and thus improves accuracy. Moreover, our final meta-auxiliary framework achieves the best. Figure 3.4 shows the robustness of our approach under different rotation and translation error thresholds.

3.3.3 Ablation and Additional Results

We perform additional experiments and ablation studies to further analyze our proposed method.

Generalization to Unseen Datasets. Although our method achieves the best per-

[1]At test-time, we still perform Eq. 3.10 for TTA.

Table 3.2: Results of cross-dataset generalization experiment on KITTI [27] and 3DMatch [89] datasets.

	KITTI [27]			3DMatch [89]		
	Recall ↑	RE (deg) ↓	TE (cm) ↓	Recall ↑	RE (deg) ↓	TE (cm) ↓
DGR [16]	95.24	0.44	23.25	87.39	2.71	7.58
Ours + DGR	**97.36**	**0.34**	**21.16**	**90.25**	**2.32**	**7.26**
DHVR [43]	95.82	0.39	22.17	87.16	2.74	7.43
Ours + DHVR	**98.01**	**0.32**	**21.18**	**90.48**	**2.25**	**7.04**
PointDSC [4]	97.15	0.36	21.74	89.42	2.15	6.89
Ours + PointDSC	**98.23**	**0.33**	**20.86**	**91.36**	**1.87**	**6.33**

formance over all the traditional and learning-based methods, the main advantage of our method is the ability to generalize to unseen test distribution. In order to evaluate the generalization of our method, we perform a cross-dataset experiment on both 3DMatch [89] and KITTI [27] datasets, where the trained model on 3DMatch is used to test on KITTI and vice versa. As shown in Table 3.2, our method shows a significant improvement on all evaluation metrics, which sheds a light on the generalization capability of our method.

Robustness to Low-Overlapping Point Clouds. To further validate the robustness of our method, we evaluate our method on a dataset with low-overlapping ratio between input point clouds, namely 3DLoMatch [35]. This dataset is constructed from the 3DMatch benchmark [89] and has a low-overlapping ratio (10%-30%) between 3D point cloud fragments. Figure 3.5 compares the inlier ratio between 3DLoMatch and 3DMatch, which shows that 3DLoMatch is more challenging due to the lower inlier ratio. We use the model trained on 3DMatch for evaluation and report our results in Table 3.3. Our approach outperforms all other methods, demonstrating the robustness of our approach to low-overlapping scenarios. More importantly, this validates

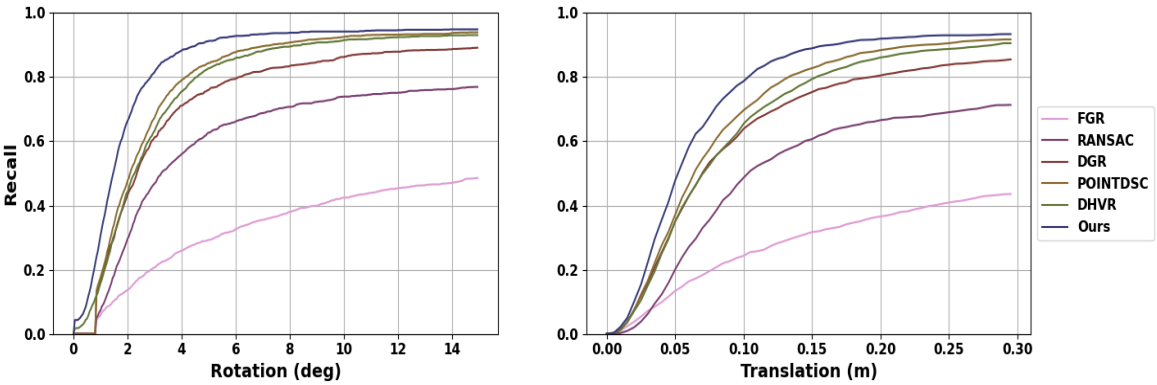


Figure 3.4: Comparison of the registration recall on the 3DMatch dataset [89] between our approach and other state-of-the-art methods by varying the translation error and the rotation error thresholds. Our framework with PointDSC [4] as backbone outperforms all other methods for all thresholds.

Table 3.3: Robustness to low-overlapping point clouds on the 3DLoMatch dataset [35] with low-overlapping ratio between point cloud segments. We train on 3DMatch [89] and evaluate on 3DLoMatch [35].

	Recall ↑	RE (deg) ↓	TE (cm) ↓
DGR [16]	43.80	4.17	10.82
Ours + DGR	**50.73**	**4.06**	**10.52**
DHVR [43]	54.46	4.13	10.54
Ours + DHVR	**57.32**	**3.83**	**10.26**
PointDSC [4]	56.10	3.87	10.39
Ours + PointDSC	**57.81**	**3.79**	**10.15**

the robustness of our approach to the percentage of template and target overlaps where 3DMatch [89] contains overlapping ratios (≥30%) and 3DLoMatch [35] contains low-overlapping ratios (10%-30%). The evaluation results on both datasets show the superiority of our approach among the baselines under different ratios.

Multiway Registration for 3D Reconstruction. Point cloud registration is a

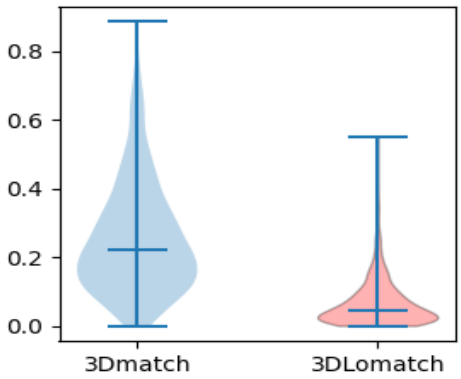


Figure 3.5: Comparison between the distribution of the inlier ratio of correspondences obtained by feature matching on 3DMatch [89] and 3DLoMatch [35] benchmarks. The registration task is more challenging with a lower inlier ratio.

critical step for various 3D applications. In this section, we present the effect of pairwise registration performance on obtaining more accurate and robust 3D reconstruction scenes. Following [16, 4], we integrate our method into a 3D reconstruction pipeline [15]. Given RGB-D scans, 3D fragments are generated from the scene. Next, we perform pairwise registration using our method to align all fragments. Finally, multi-way registration [15] is used to optimize the fragment poses using pose graph optimization [42]. We use the model trained on 3DMatch [89] to further demonstrate the generalization of our method and evaluate our approach on Augmented ICL dataset [15] using Absolute Trajectory Error (ATE). As shown in Table 3.4, our method achieves the lowest error compared to all other methods.

Methodology Components. To study the effectiveness of the proposed framework, we conduct ablation experiments on 3DMatch dataset [89] and evaluate the effect of each component of the proposed framework. We consider DGR [16] as the backbone in our experiments and report the results after applying each component to DGR [16].

Table 3.4: Multiway registeration results on Augmented ICL-NUIM dataset [15] evaluated by ATE(cm) where lower is better.

Method	Living1	Living2	Office1	Office2	AVG
FGR [93]	78.97	24.91	14.96	21.05	34.98
RANSAC [25]	110.9	19.33	14.42	17.31	40.49
DGR [16]	21.06	21.88	15.76	11.56	17.57
Ours + DGR	**18.32**	**16.12**	**12.24**	**10.44**	**14.28**
DHVR [43]	22.91	16.37	12.58	10.90	15.69
Ours + DHVR	**18.46**	**13.59**	**12.43**	**9.56**	**13.51**
PointDSC [4]	20.25	15.58	13.56	11.30	15.18
Ours + PointDSC	**15.73**	**12.07**	**12.15**	**9.78**	**12.43**

Specifically, we compare the results between our method's three major components: Auxiliary Learning, Meta Learning, Test-time Adaptation. We first investigate the effect of our proposed Auxiliary Learning method by jointly training the primary and three auxiliary tasks by optimizing the loss in Eq. 3.7. This shows the strength of the proposed auxiliary tasks acting as a regularizer during training. Then, we study the impact of combining test-time adaptation with auxiliary learning, in which the auxiliary tasks are used to update the model parameters at test-time by optimizing the auxiliary loss in Eq. 3.6. Furthermore, we show the effect of the proposed meta-auxiliary learning paradigm elaborated in Algorithm 1 in learning optimal model parameters. However, we fixed the model parameters at test-time. Finally, we report the results of our final framework integrating Auxiliary Learning, Meta Learning, and Test-time Adaptation.

As reported in Table 3.5, auxiliary learning improves the registration results across all evaluation metrics compared to DGR [16]. This demonstrates that the proposed auxiliary tasks can complement the registration task, leading to performance improve-

Table 3.5: Ablation studies on framework components: Auxiliary Learning, Meta Learning, and Test-time Adaptation.

	Recall ↑	RE (deg) ↓	TE (cm) ↓
DGR	91.31	2.40	7.48
DGR + Aux.	91.42	2.25	7.06
DGR + TTA (w/o meta)	91.86	1.88	6.54
DGR + Meta-Aux. (w/o TTA)	92.28	1.71	6.40
DGR + full framework	**92.45**	**1.71**	**6.39**

ment. Combining auxiliary learning with TTA has further improved the performance of registration recall by 0.44%, and decreased the TE and RE by 0.52cm and 0.37 deg, respectively. As TTA allows the auxiliary tasks to transfer useful features of test-instance to the primary registration task, it enhances the registration performance. Moreover, the proposed meta-auxiliary training method greatly boosts the performance, which demonstrates the effectiveness of training tasks using meta-learning terminology such that the meta-objective enforces the auxiliary tasks to improve the primary task performance. Finally, our final framework further boosts the registeration performance by fine-tuning the model parameters at test-time.

Analysis of Auxiliary Tasks. We conduct additional ablation studies on the 3DMatch dataset [89] to investigate the importance of each auxiliary task in our approach in improving the registration performance. As shown in Table 3.6, the auxiliary reconstruction task significantly boosts the registration recall of DGR [16] by 0.92%. Also, Translation Error (TE) and Rotation Error (RE) greatly drop by 13% and 30%, respectively. These evaluation metrics are further improved when combining the auxiliary correspondence classification task to the reconstruction task. This demonstrates the impact of multiple auxiliary tasks in transferring additional fea-

Table 3.6: Ablation studies on the three auxiliary tasks: Point Cloud Reconstruction (rec), Correspondence Classification (cc), and Feature Learning (byol).

	Recall ↑	RE (deg) ↓	TE (cm) ↓
DGR [16]	91.31	2.43	7.34
DGR + rec	92.24	1.71	6.42
DGR + (rec, cc)	92.38	**1.69**	6.40
DGR + (rec, cc, byol)	**92.45**	1.71	**6.39**

Table 3.7: Ablation studies comparison between learnable and fixed balancing weights. Second row presents results with fixed balancing weights of ($\lambda_1 = 0.5$, $\lambda_2 = 0.3$, $\lambda_3 = 0.2$). Third row presents results with fixed balancing weights of ($\lambda_1 = 0.7$, $\lambda_2 = 0.1$, $\lambda_3 = 0.2$). Our framework with learnable balancing weights achieve better results in all evaluation metrics.

	Recall ↑	RE (deg) ↓	TE (cm) ↓
DGR [16]	91.31	2.43	7.34
DGR + fixed	92.03	1.76	6.48
DGR + fixed	92.26	1.73	6.41
DGR + learnable	**92.45**	**1.71**	**6.39**

tures to the primary task and enhancing registration results. Finally, our final three auxiliary tasks achieve a higher registration recall of 92.45% and a lower Translation Error (TE) of 6.39cm. However, the Rotation Error (RE) was slightly worse when compared to the two auxiliary tasks results.

Impact of Learnable Balancing Weights. In this study, we report the impact of learning the balancing weights. Instead of using the same fixed values of the auxiliary losses balancing weights, we add the balancing weights to the learnable

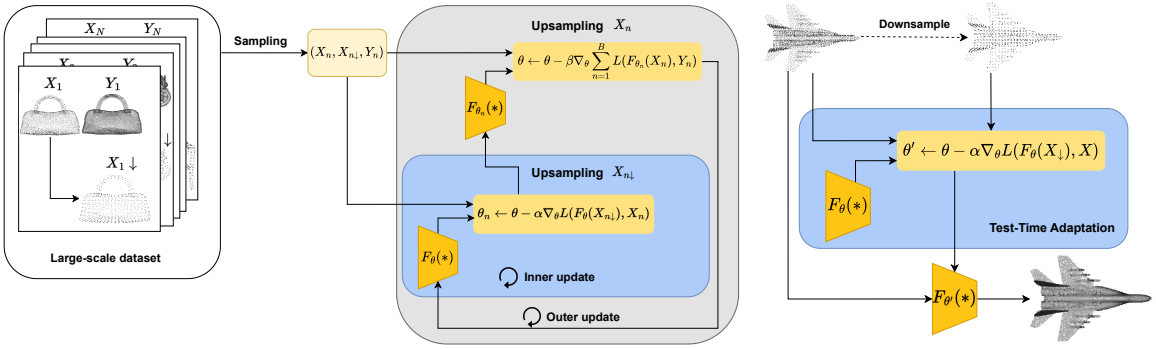
network parameters and learn their values during training. This enables the training algorithm to effectively balance the auxiliary tasks with optimal weights. At test-time, the learned weights are fixed. We perform the experiments on the 3DMatch dataset [89] and adopt DGR [16] as the baseline of the experiment. The results are shown in Table 3.7. In the second and third rows of Table 3.7, we report the results of using fixed balancing weights. Although the registration recall improved by 0.72% and 0.95%, respectively, it is hard to determine the optimal balancing weights without doing numerous experiments. Instead, training with learnable balancing weights effectively balances the auxiliary tasks and greatly improves all evaluation metrics.

Chapter 4

Meta-Learning for Adaptation in Point Cloud Upsampling

4.1 Problem Statement

In this problem, the objective is to generate high-resolution point clouds from sparse point clouds. The popularity of this problem is growing significantly because obtaining high-resolution point clouds is a prohibitively expensive and challenging process. On the other hand, point clouds obtained from affordable 3D LiDAR scanners are often sparse and non-uniform, which negatively affects the performance of many downstream applications, such as 3D reconstruction, robotic manipulation, etc. Therefore, these sparse point clouds need to be effectively densified, before using it in the real-world applications. Given a sparse and noisy point cloud $X \in R^{N \times 3}$ with N points and an upsampling ratio r, our goal is to generate a dense point cloud $Y \in R^{rN \times 3}$ with rN points that adequately cover the underlying surface. More im-



(a) Meta-training (b) Meta-testing

Figure 4.1: Overview of the proposed meta-learning procedure for point cloud upsampling. During each iteration of meta-training, we sample a batch of training pairs. For each sampled training pair (X_n, Y_n), we first downsample X_n to obtain a sparser version $X_{n\downarrow}$. We obtain the adapted parameters by applying our model to upsample $X_{n\downarrow}$ and use X_n as the ground-truth to define as a self-supervised loss. We perform a small number of gradient updates using the computed self-supervised loss in the inner loop. Then we use the adapted model to perform the main task by upsampling X_n. Finally, we update the model in the outer loop based on the calculated upsampling loss on the adapted parameters. Given a new instance X during meta-testing, we perform a few gradient updates using $(X_{\downarrow}, X)$ to adapt the model for this instance and use the adapted model for final prediction.

portantly, the generated points should be uniformly-located on the object surface. We aim to learn a model $F_\theta(X) \rightarrow Y$ parameterized by θ that maps X to Y for a given upsampling ratio r.

4.2 Our Approach

4.2.1 Preliminaries

To leverage the advantages of external learning, we first perform supervised training using pairs of sparse-dense point clouds (X, Y). We adopt the architecture of three state-of-the-art point cloud upsampling networks as the backbones of our approach, including PU-GCN [56], Dis-PU [46], and PU-Dense [2]. Each network is optimized using a standard supervised loss to learn the initial model parameters θ:

$$\min_{\theta} L(F_\theta(X), Y) \tag{4.1}$$

where L is a supervised loss between the prediction $F_\theta(X)$ and the ground truth Y.

At this step, we may directly fine-tune the pre-trained parameters during inference to exploit the internal features of point clouds for test-time adaptation (TTA). For example, given an input point cloud X during inference, we can downsample X to obtain a sparser point cloud $X_\downarrow$. We can then fine-tune the model parameter θ by treating $(X_\downarrow, X)$ as a sparse-dense pair of point clouds in Eq 4.1. In our experiments, we will demonstrate that such naive TTA can already improve the model performance. However, it requires a large number of gradient updates to effectively adapt for each test instance since the model is not explicitly learned to facilitate test-time adaptation.

In our work, we propose a meta-learning approach to explicitly learn the model parameters for test-time adaptation (Fig. 4.1). Our approach consists of a meta-training stage and a meta-testing stage. During meta-training, we learn the model parameters from a set of tasks, where each task is constructed from a sparse-dense pair of point clouds from the training data. Each inner update of meta-training

involves adapting the model to a sampled task. The adapted model is then used for inference on that task. This performance of this task-adaptive model is then used for the loss function for the outer update of meta-training. The goal of the outer update is to optimize the model parameters such that after adapting to a particular task, the adapted model performs well on that task. Through this bi-level optimization, the model parameters are explicitly trained so that they can be effectively adapted to a new test instance with only a few gradient updates. During meta-testing, we are given a new test instance. We first adapt the meta-learned model to this instance, then use the updated model for prediction.

4.2.2 Meta-Training

Inspired by the success of adopting MAML [23] for image super-resolution problem [53, 71], we learn the model using meta-learning such that the model parameters are trained to quickly adapt to unseen data at test time using a small number of gradient updates. First, the model is initialized by the pre-trained weights θ resulting from the standard supervised training. The pre-trained feature representations help in stabilizing meta-training and thus ease the training phase of meta-learning [71]. We further optimize the model parameters using meta-learning. Specifically, we develop a meta-learning algorithm summarized in Algorithm 2 based on MAML [23]. The key to our approach is the construction of a task for the inner update of meta-training. Inspired by prior work in super-resolution [53, 71], we use a pair of point clouds consisting of the input point cloud X and its downsampled version $X_{\downarrow}$ for a task in MAML. During the inner update of MAML, the model is used to upsampled $X_{\downarrow}$ and

X is treated as the ground truth. The loss between $F_\theta(X_\downarrow)$ and X is then used to adapt the model parameters θ by a few gradient updates. This allows the model to be quickly adapted to different data distributions at test time and boosts the overall generalization capability of the model.

Figure 4.1 illustrates the overall scheme of our proposed approach. The external training dataset consists of pairs of sparse-dense point clouds. We optimize the network weights using our proposed meta-learning approach to learn the optimal model parameters that can quickly adapt to new data distributions at test time. At test-time, we adapt the meta-learned parameters for each given test instance and use the adapted parameters to obtain the upsampled point cloud Y.

More specifically, in each iteration of meta-training, we sample a batch B of sparse-dense training pairs $\{X_n, Y_n\}_{n=1}^{B}$. We downsample X_n to a sparser version $X_{n\downarrow}$. In each inner update of meta-training, we perform model adaptation for a small number of gradient updates using ($X_{n\downarrow}$,X_n) pairs as follows:

$$\theta_n \leftarrow \theta - \alpha \nabla_\theta L(F_\theta(X_{n\downarrow}), X_n) \tag{4.2}$$

where α controls the learning rate of the adaptation and θ_n represent the adapted model parameters for the input X_n using internal learning.

The adapted model θ_n is then used to generate the dense point cloud Y and optimize the following meta-objective:

$$\min_\theta \sum_{n=1}^{B} L(F_{\theta_n}(X_n), Y_n) \tag{4.3}$$

Note that we use the adapted model θ_n in the model $F_{\theta_n(\cdot)}$, but the optimization in Eq. 4.3 is performed over the original model parameters θ.

Algorithm 2 Meta-training

Require: X, Y: training pairs

Require: B: batch size

Require: α, β: learning rates

Output: θ: learned parameters

Initialize the network with pre-trained weights θ
while not done **do**
  Sample a training batch $\{X_n, Y_n\}_{n=1}^{B}$
  Generate downsampled $X_{n\downarrow}$
  for each example **do**
    Evaluate loss: $\nabla_\theta L(F_\theta(X_{n\downarrow}), X_n)$
    Compute adapted parameters θ_n:
    $\theta_n \leftarrow \theta - \alpha \nabla_\theta L(F_\theta(X_{n\downarrow}), X_n)$
  end for
  Evaluate the main upsampling task using the adapted parameters and update:
  $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{n=1}^{B} L(F_{\theta_n}(X_n), Y_n)$
end while
return θ

In the outer update of meta-training, we optimize the meta-objective in Eq. 4.3 by performing gradient update as follows:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{n=1}^{B} L(F_{\theta_n}(X_n), Y_n) \tag{4.4}$$

where β is the meta-learning rate.

Algorithm 3 Meta-testing

Require: X: sparse point cloud

Require: α: learning rate

Output: Y: dense point cloud

1: Initialize the network with meta-trained weights θ

2: Generate downsampled $X_{\downarrow}$

3: Evaluate loss: $\nabla_{\theta} L(F_{\theta}(X_{\downarrow}), X)$

4: Compute adapted parameters:

5: $\theta' \leftarrow \theta - \alpha \nabla_{\theta} L(F_{\theta}(X_{\downarrow}), X)$

6: Generate upsampled point cloud Y $= F_{\theta'}(X)$

7: **return** Y

4.2.3 Meta-Testing

At test-time, we downsample the input point cloud X to a sparser version $X_{\downarrow}$. Then, we fine-tune the model parameters by performing a small number of gradient updates using the point cloud pairs $(X_{\downarrow}, X)$. This update is completely self-supervised and exploits the internal features of the input point cloud X.

$$\theta' \leftarrow \theta - \alpha \nabla_{\theta} L(F_{\theta}(X_{\downarrow}), X) \tag{4.5}$$

Finally, the adapted model θ' is used to perform the main upsampling task and generates the densified point cloud $F_{\theta'}(X)$. The inference procedure is summarized in Algorithm 3.

4.3 Experiments

We first describe the experimental setup in Sec. 4.3.1. We then present our experiment results and comparison in Sec. 4.3.2. We also perform extensive ablation studies in Sec. 4.3.3.

4.3.1 Experimental Setup

Datasets and Evaluation Metrics. Following [2], we train our method on the ShapeNet dataset [9] and evaluate the performance of the networks on the ShapeNet dataset [9] (1024 test samples) and the 8iVFB dataset [20] (1200 test samples). We adopt two widely used upsampling evaluation metrics, namely Chamfer distance (CD) and Peak signal-to-noise ratio (PSNR), to measure the quality of the upsampled point cloud compared to the ground truth dense point cloud. CD sums the distances between the nearest neighbors correspondences of the upsampled point cloud and ground truth point cloud. CD is defined as:

$$CD_{(Y,G)} = \sum_{a \in Y} \min_{b \in G} |a - b|^2 + \sum_{b \in G} \min_{a \in Y} |a - b|^2 \tag{4.6}$$

where Y is the upsampled point cloud and G is the ground truth point cloud. PSNR measures the ratio between a normalization factor and point-to-point MSE as defined in [2], which is calculated from the upsampled point cloud Y to the ground truth G as well as in the opposite direction. PSNR is defined as follows:

$$PSNR = min(PSNR_{(Y,G)}, PSNR_{(G,Y)}) \tag{4.7}$$

$$PSNR_{(A,B)} = 10 \log_{10}(\frac{p_s^2}{d_{MSE}(A, B)}) \tag{4.8}$$

where p_s is the normalization factor and d_{MSE} is the average mean squared error between points in one point cloud and their nearest neighbors correspondences in the other point cloud.

Implementation Details. We use the official implementations of PU-GCN [56], Dis-PU [46] and PU-Dense [2] as the backbones of our approach. We first conduct supervised training on the backbones to obtain the initial pre-trained weights. During meta-training, we perform 5 gradient updates in the inner loop as described in Algorithm 2. We set the batch size to 8 and the learning rates α and β to 10^{-5} and 10^{-6}, respectively. We optimize the networks using the Adam optimizer with a learning rate of 10^{-4} and an exponentially decayed factor of 0.99. All experiments are conducted on a single NVIDIA TitanX GPU.

4.3.2 Results and Comparisons

We compare the proposed method with five state-of-the-art upsampling methods: MPU [85], PU-GAN [45], PU-GCN [56], Dis-PU [46], and PU-Dense [2]. We apply our framework with several different backbone networks, including PU-GCN [56], Dis-PU [46], and PU-Dense [2]. The comparison is shown in Table 4.1. All methods are trained on the ShapeNet dataset [9] using the same experimental setup for a fair comparison. As shown in Table 4.1, our method achieves a significant performance improvement on three backbones [56, 46, 2] across all the evaluation metrics with a good margin. More importantly, our method with PU-Dense [2] as the backbone outperforms all other state-of-the-art methods. Notably, we observe that the performance improvement on the 8iVFB dataset [20] is more significant than the ShapeNet

Table 4.1: Quantitative comparison of our method with existing state-of-the-art methods on ShapeNet [9] and 8iVFB [20] datasets. We show the 8x upsampling results based on CD (10^{-2}) and PSNR (dB) evaluation metrics. ↑ (↓) means smaller (larger) values correspond to better performance.

	ShapeNet [9]		8iVFB [20]	
	CD ↓	PSNR ↑	CD ↓	PSNR ↑
MPU [85]	149.20	65.37	105.43	66.83
PU-GAN [45]	174.58	64.88	117.66	66.19
PU-GCN [56]	65.81	69.59	63.71	69.78
Ours + PU-GCN	**50.49**	**71.62**	**41.86**	**72.18**
Dis-PU [46]	55.62	70.23	51.68	70.59
Ours + Dis-PU	**48.25**	**71.96**	**34.65**	**72.53**
PU-Dense [2]	30.52	73.11	33.18	72.57
Ours + PU-Dense	**26.44**	**73.38**	**23.80**	**73.64**

dataset [9]. This demonstrates the effectiveness of our method in boosting the generalization capability of models to unseen test data by enabling the networks to utilize the internal features of point clouds at test time. Besides quantitative results, we present upsampling qualitative comparisons in Figure 4.2. In most cases, the upsampled point clouds of our method are more uniform, less noisy, and preserve edge sharpness.

4.3.3 Ablation Studies

We perform ablation studies to further analyze our proposed method.

Robustness to Noise. To validate the robustness to noise, we add Gaussian noise of varying noise levels to the input point clouds. We use the model trained on ShapeNet [9] for evaluation and report our results in Table 4.2. As the noise level increases,

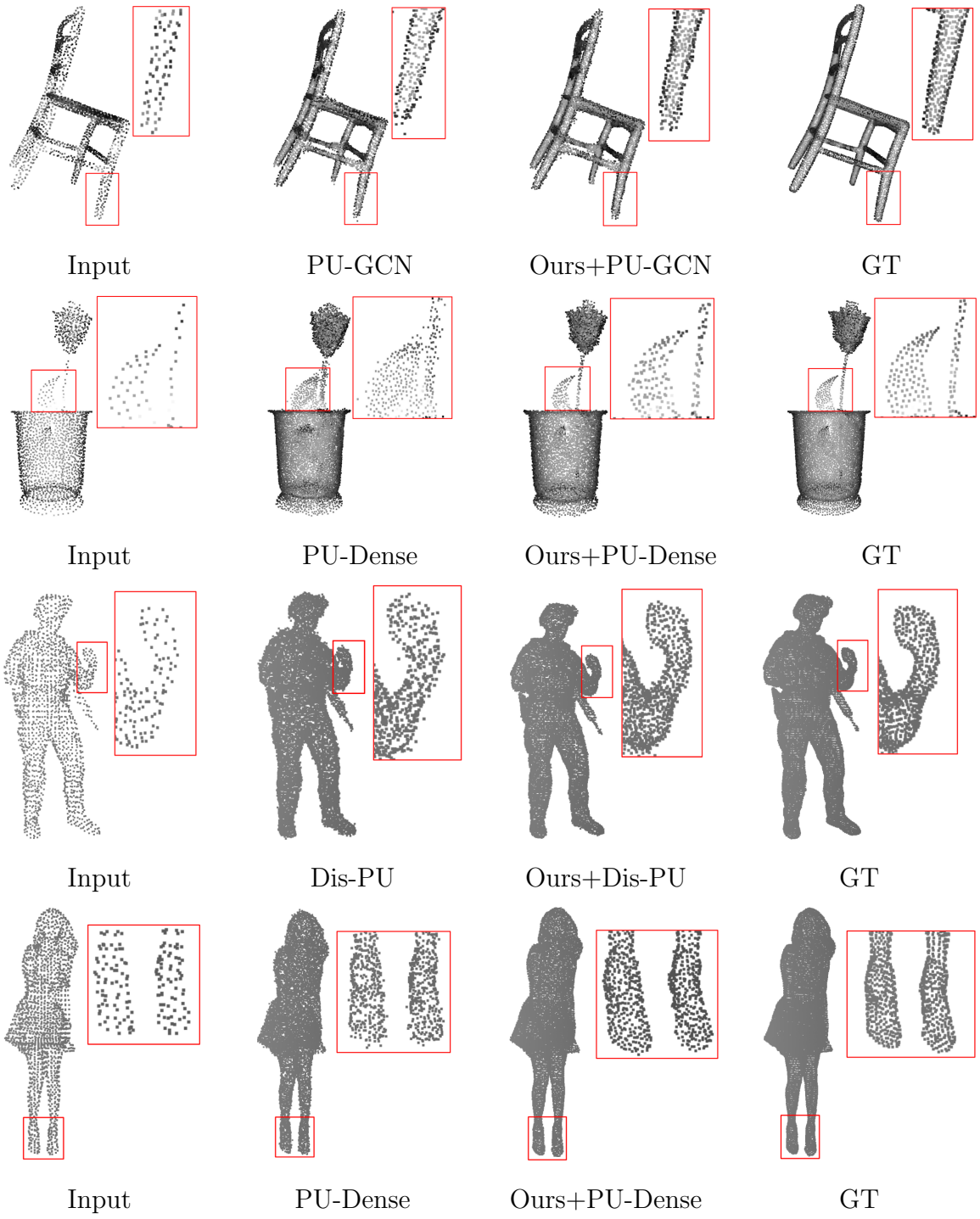


Figure 4.2: Qualitative results of point cloud upsampling on ShapeNet [9](first two rows) and 8iVFB [20](last two rows). We compare the 4x upsampled results with the three baselines: PU-GCN [56], Dis-PU [46], and PU-Dense [2].

Table 4.2: Robustness to upsampling noisy point clouds results with different noise levels on the ShapeNet dataset [9]. We use an upsampling ratio of r=8 and compare different methods using the CD (10^{-2}) evaluation metric.

	0%	0.5%	1%	2%
PU-GCN [56]	65.81	76.46	84.73	159.62
Ours + PU-GCN	**50.49**	**56.32**	**63.61**	**124.79**
Dis-PU [46]	55.62	65.24	71.19	136.87
Ours + Dis-PU	**48.25**	**52.58**	**59.35**	**117.52**
PU-Dense [2]	30.52	35.71	37.55	74.67
Ours + PU-Dense	**26.44**	**29.10**	**33.96**	**62.41**

Table 4.3: Quantitative comparisons with baselines on the ShapeNet dataset [9] with varying upsampling scale ratios.

	4x		16x	
	CD ↓	PSNR ↑	CD ↓	PSNR ↑
PU-GCN [56]	48.15	70.90	121.65	66.27
Ours + PU-GCN	**31.74**	**72.87**	**102.84**	**67.11**
Dis-PU [46]	36.23	72.19	92.88	67.46
Ours + Dis-PU	**25.61**	**73.28**	**84.31**	**69.01**
PU-Dense [2]	18.82	75.24	69.48	70.32
Ours + PU-Dense	**15.33**	**75.86**	**62.19**	**70.73**

we can observe that the performance of all approaches drops. But our approach still outperforms all other methods under each noise level by a significant margin. This demonstrates the robustness of our approach to noise.

Varying Upsampling Ratios. We further investigate the robustness of our framework across different upsampling ratios. We have conducted experiments with upsampling scale ratios r = 4, 8, and 16. Table 4.1 reports the results with upsampling

Table 4.4: Ablation studies on different components of our framework components, including meta-learning and test-time adaptation.

	CD ↓	PSNR ↑
PU-GCN [56]	65.81	69.59
PU-GCN + TTA (w/o meta)	61.74	69.83
Ours + PU-GCN	**50.49**	**71.62**
Dis-PU [46]	55.62	70.23
Dis-PU + TTA (w/o meta)	54.52	70.40
Ours + Dis-PU	**48.25**	**71.96**
PU-Dense [2]	30.52	73.11
PU-Dense + TTA (w/o meta)	28.33	73.18
Ours + PU-Dense	**26.44**	**73.38**

Table 4.5: Ablation studies on the number of gradient updates ($N = 1, 3, 5$).

	CD ↓	PSNR ↑
Dis-PU [46]	55.62	70.23
Ours + Dis-PU (N=1)	53.41	70.86
Ours + Dis-PU (N=3)	50.71	71.89
Ours + Dis-PU (N=5)	**48.25**	**71.96**

scale ratio of 8x on ShapeNet [9] and 8iVFB [20] datasets. Table 4.3 shows the quantitative comparisons on the ShapeNet [9] dataset under upsampling scales of 4x and 16x. We can observe that our method effectively improves the performance of all backbones across different upsampling ratios.

Framework Components. To study the relative contributions of various components in the proposed framework, we conduct additional ablation experiments. We first investigate the effect of test-time adaptation on improving performance. In this experiment, we do not apply our meta-learning approach. Instead, we use the pre-

trained parameters resulting from supervised training. At test time, the input point cloud is downsampled and the model is fine-tuned using the input and the downsampled point clouds. As shown in Table 4.4, the performance of all the backbones has already been improved. This demonstrates the effectiveness of naive test-time adaptation in utilizing the internal features of point clouds, even without meta-learning. When applying our meta-training approach in Algorithm 2, the performance has been further improved. This demonstrates that both TTA and meta-training contribute to the final performance improvement.

Number of Gradient Updates. In this study, we investigate the impact of the number of gradient updates N in the inner loop of Algorithm 2. We use N = 1, 3, and 5 during the meta-training. Table 4.5 shows the evaluation results of our method trained with a different number of gradient updates. Overall, we observe that a large number of gradient updates enables the model to better capture the internal features of test point clouds and thus improve the performance. Note that, we use the same number of gradient updates during training and testing. As shown in Table 4.5, we obtain the best performance results with five gradient updates.

Chapter 5

Conclusion

In this thesis, we have introduced novel test-time adaptation frameworks for point cloud registration and point cloud upsampling. In previous work, the model is typically trained on an external supervised dataset and fixed during evaluation on unseen test data. This approach fails to exploit the useful internal information of the test point clouds. In contrast, our frameworks are designed to efficiently adapt the model parameters for each test instance at inference time to boost the model performance. First, we have addressed the point cloud registration problem. We have proposed three self-supervised auxiliary tasks to learn complementary features from test instances. In addition, we have used a meta-auxiliary learning paradigm to train the primary and auxiliary tasks, so that the adapted model using auxiliary tasks improves the performance of the primary registration task. Second, we have tackled the problem of point cloud upsampling. In this problem, we introduce a novel approach that utilizes both internal and external features of point clouds. Our proposed method employs meta learning to allow fast adaptation of model parameters at test time

using only the input sparse point cloud. Extensive experiments demonstrate the capability of our proposed approaches in improving the performance and outperforming state-of-the-art models.

Bibliography

[1] Idan Achituve, Haggai Maron, and Gal Chechik. Self-supervised learning for domain adaptation on point clouds. *IEEE Winter Conference on Applications of Computer Vision*, 2021.

[2] Anique Akhtar, Zhu Li, Geert Van der Auwera, Li Li, and Jianle Chen. Pu-dense: Sparse tensor-based point cloud geometry upsampling. *IEEE Transactions on Image Processing*, 2022.

[3] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 2003.

[4] Xuyang Bai, Zixin Luo, Lei Zhou, Hongkai Chen, Lei Li, Zeyu Hu, Hongbo Fu, and Chiew-Lan Tai. Pointdsc: Robust point cloud registration using deep spatial consistency. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[5] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[6] Daniel Barath and Jiří Matas. Graph-cut ransac. *IEEE Conference on Computer*

Vision and Pattern Recognition, 2018.

[7] Mark Billinghurst, Adrian J. Clark, and Gun A. Lee. A survey of augmented reality. *Foundations and Trends in Human-Computer Interaction*, 2015.

[8] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[9] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv:1512.03012*, 2015.

[10] Hui Chen and Bir Bhanu. 3d free-form object recognition in range images using local surface patches. *IEEE International Conference on Pattern Recognition*, 2004.

[11] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington. 3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. *IEEE Signal Processing Magazine*, 2020.

[12] Zhi Chen, Kun Sun, Fan Yang, and Wenbing Tao. Sc2-pcr: A second order spatial compatibility for efficient and robust point cloud registration. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[13] Zhixiang Chi, Yang Wang, Yuanhao Yu, and Jingshan Tang. Test-time fast adaptation for dynamic scene deblurring via meta-auxiliary learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[14] Myungsub Choi, Janghoon Choi, Sungyong Baik, Tae Hyun Kim, and Ky-

oung Mu Lee. Scene-adaptive video frame interpolation via meta-learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[15] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[16] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[17] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. *IEEE International Conference on Computer Vision*, 2019.

[18] Ondřej Chum, Jiri Matas, and Josef Kittler. Locally optimized ransac. *Symposium of the German Association for Pattern Recognition*, 2003.

[19] Eduardo R. Corral-Soto, Amir Nabatchian, Martin Gerdzhev, and Liu Bingbing. Lidar few-shot domain adaptation via integrated cyclegan and 3d object detector with joint learning delay. *IEEE International Conference on Robotics and Automation*, 2021.

[20] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A. Chou. 8i voxelized full bodies - a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, Geneva*, 2017.

[21] Guangyao Ding, Meiying Zhang, E Li, and Qi Hao. Jst: Joint self-training for unsupervised domain adaptation on 2d&3d object detection. *IEEE International Conference on Robotics and Automation*, 2022.

[22] Hehe Fan, Xiaojun Chang, Wanyue Zhang, Yi Cheng, Ying Sun, and Mohan S. Kankanhalli. Self-supervised global-local structure modeling for point cloud do-

main adaptation with reliable voted pseudo labels. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[23] Chelsea Finn, P. Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning*, 2017.

[24] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *International Conference on Learning Representation*, 2018.

[25] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[26] Andrea Frome, Daniel F. Huber, Ravi Krishna Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. *European Conference on Computer Vision*, 2004.

[27] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 2013.

[28] Zan Gojcic, Caifa Zhou, Jan Dirk Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[29] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *International Conference on Learning Representation*, 2018.

[30] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H.

Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 2020.

[31] Yu Hao and Yi Fang. 3d meta-segmentation neural network. *arXiv:2110.04297*, 2021.

[32] Chao Huang, Zhangjie Cao, Yunbo Wang, Jianmin Wang, and Mingsheng Long. Metasets: Meta-learning on point sets for generalizable representations. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[33] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics*, 2009.

[34] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. Edge-aware point set resampling. *ACM Transactions on Graphics*, 2013.

[35] Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[36] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. *IEEE International Conference on Computer Vision*, 2021.

[37] Peng Jiang and Srikanth Saripalli. Lidarnet: A boundary-aware domain adaptation model for point cloud semantic segmentation. *IEEE International Conference on Robotics and Automation*, 2021.

[38] Andrew Edie Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1999.

[39] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[40] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. *International Conference on Machine Learning*, 2015.

[41] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 2015.

[42] Rainer Kümmerle, Giorgio Grisetti, Hauke Malte Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. *IEEE International Conference on Robotics and Automation*, 2011.

[43] Junha Lee, Seungwook Kim, Minsu Cho, and Jaesik Park. Deep hough voting for robust global registration. *IEEE International Conference on Computer Vision*, 2021.

[44] Jiaxin Li and Gim Hee Lee. Usip: Unsupervised stable interest point detection from 3d point clouds. *IEEE International Conference on Computer Vision*, 2019.

[45] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: a point cloud upsampling adversarial network. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[46] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Point cloud upsampling via disentangled refinement. *IEEE Conference on Computer Vision and*

Pattern Recognition, 2021.

[47] Zhenyu Li, Zehui Chen, Ang Li, Liangji Fang, Qinhong Jiang, Xianming Liu, and Junjun Jiang. Unsupervised domain adaptation for monocular 3d object detection via self-training. *European Conference on Computer Vision*, 2022.

[48] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics*, 2007.

[49] Shikun Liu, Andrew Davison, and Edward Johns. Selfsupervised generalization with meta auxiliary learning. *Advances in Neural Information Processing Systems*, 2019.

[50] Kaiyue Lu, Nick Barnes, Saeed Anwar, and Liang Zheng. From depth what can you see? depth completion via auxiliary image reconstruction. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[51] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *International Conference on Learning Representation*, 2018.

[52] G. Dias Pais, Pedro Miraldo, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C. Nascimento, and Rama Chellappa. 3dregnet: A deep neural network for 3d point registration. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[53] Seobin Park, Jinsu Yoo, Donghyeon Cho, and Jiwon Kim andTae Hyun Kim. Fast adaptation to super-resolution networks via meta-learning. *European Conference on Computer Vision*, 2020.

[54] Omid Poursaeed, Tianxing Jiang, Han Qiao, Nayun Xu, and Vladimir G Kim.

Self-supervised learning of point clouds via orientation estimation. *IEEE International Conference on 3D Vision*, 2020.

[55] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 2017.

[56] Guocheng Qian, Abdulellah Abualshour, G. Li, Ali K. Thabet, and Bernard Ghanem. Pu-gcn: Point cloud upsampling using graph convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[57] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. Pugeo-net: A geometry-centric network for 3d point cloud upsampling. *European Conference on Computer Vision*, 2020.

[58] Can Qin, Haoxuan You, Lichen Wang, C-C Jay Kuo, and Yun Fu. Pointdan: A multi-scale 3d domain adaption network for point cloud representation. *Advances in Neural Information Processing Systems*, 2019.

[59] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kaiping Xu. Geometric transformer for fast and robust point cloud registration. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[60] Siwen Quan and Jiaqi Yang. Compatibility-guided sampling consensus for 3-d point cloud registration. *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[61] Sachin Ravi and H. Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representation*, 2016.

[62] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional

neural adaptive processes. *Advances in Neural Information Processing Systems*, 2019.

[63] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. *IEEE International Conference on Robotics and Automation*, 2009.

[64] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Strong-weak distribution alignment for adaptive object detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[65] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[66] Khaled Saleh, Ahmed Abobakr, Mohammed Attia, Julie Iskander, Darius Nahavandi, and Mohammed Hossny. Domain adaptation for vehicle detection from bird's eye view lidar point cloud data. *IEEE International Conference on Computer Vision*, 2019.

[67] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. *International Conference on Machine Learning*, 2016.

[68] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. *Advances in Neural Information Processing Systems*, 2019.

[69] Ruwen Schnabel, Roland Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 2007.

[70] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-

shot learning. *Advances in Neural Information Processing Systems*, 2017.

[71] Jae Woong Soh, Sunwoo Cho, and Nam Ik Cho. Meta-transfer learning for zero-shot super-resolution. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[72] Weiwei Sun, Wei Jiang, Eduard Trulls, Andrea Tagliasacchi, and Kwang Moo Yi. Acne: Attentive context normalization for robust permutation-equivariant learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[73] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. *International Conference on Machine Learning*, 2020.

[74] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[75] Federico Tombari, Samuele Salti, and Luigi di Stefano. Unique shape context for 3d data description. *3D Object Retrieval*, 2010.

[76] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, 2016.

[77] Yunnan Wang and Jianxun Li. Bilateral knowledge distillation for unsupervised domain adaptation of semantic segmentation. *IEEE International Conference on Intelligent Robots and Systems*, 2022.

[78] Ze Wang, Sihao Ding, Ying Li, Minming Zhao, Sohini Roychowdhury, Andreas Wallin, Guillermo Sapiro, and Qiang Qiu. Range adaptation for 3d object detection in lidar. *IEEE International Conference on Computer Vision Workshop*,

2019.

[79] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. *IEEE International Conference on Robotics and Automation*, 2019.

[80] Heng Yang, J. Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 2021.

[81] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d: Self-training for unsupervised domain adaptation on 3d object detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[82] Shuquan Ye, Dongdong Chen, Songfang Han, Ziyu Wan, and Jing Liao. Meta-pu: An arbitrary-scale upsampling network for point cloud. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[83] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. *European Conference on Computer Vision*, 2018.

[84] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal V. Fua. Learning to find good correspondences. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[85] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3d point set upsampling. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[86] Hao Yu, Fu Li, Mahdi Saleh, Benjamin Busam, and Slobodan Ilic. Cofinet: Reliable coarse-to-fine correspondences for robust point cloud registration. *Advances in Neural Information Processing Systems*, 2021.

[87] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. *European Conference on Computer Vision*, 2018.

[88] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

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

[90] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. *IEEE International Conference on Computer Vision*, 2019.

[91] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. *Advances in Neural Information Processing Systems*, 2018.

[92] Qian Zhong, Ling Chen, and Yuntao Qian. Few-shot learning for remote sensing image retrieval with maml. *IEEE International Conference on Image Processing*, 2020.

[93] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. *European Conference on Computer Vision*, 2016.

[94] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *IEEE Interna-*

tional Conference on Computer Vision, 2017.