Virtual-Real Object Registration and Shape Completion for Stable and Accurate Augmented Reality Technology

LONG XI

Supervisor: Professor Wen Tang, Professor Tao Xue (External)

Department of Creative Technology
Bournemouth University

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

May 2023

Copyright Statement

Abstract

Augmented Reality (AR) technology integrates virtual objects with the real-world scene and has been widely used in many applications. 3D point cloud registration is one of the main processes to correctly align virtual 3D objects with real-world scenes in AR. The higher quality of the underlying 3D point clouds with fewer missing and noisy points, the more accurate the 3D point cloud registration.

The goal of this thesis is to develop algorithms, computational frameworks and methods to improve the quality of 3D point clouds and in order to further increase the accuracy of 3D point cloud registration for AR applications. To achieve this goal, firstly, a computational framework is developed for recovering dense and high-quality 3D point clouds from mono-endoscopic images captured by mono-endoscopic sensors. This computational framework contains a monocular depth learning network to generate the 3D point clouds from monocular images and a 3D point cloud completion network to recover the missing data from the generated 3D point clouds. The experimental results show that this computational framework can generate dense 3D point clouds of real endoscopic images and recover high-quality 3D point clouds from incomplete point clouds with 60% missing points. Secondly, in order to improve the quality of 3D point clouds from real-world objects, a learning-based neural network (TreeNet) has been proposed. The experimental results show that TreeNet outperforms five state-of-the-art learning-based methods and also shows good generalization on unknown data. TreeNet is also evaluated in the proposed computational framework for endoscopic scenes as an application, which proves the effectiveness of TreeNet for medical data. Thirdly, in order to improve the accuracy of rigid 3D point cloud registration, an unsupervised learning-based network (Iterative BTreeNet) has been proposed. Iterative BTreeNet has been compared with three traditional and six state-of-the-art learning-based methods and outperforms these methods on partial and noisy point clouds without training them. Iterative BTreeNet also exhibits remarkable

generalization to unseen large and dense scenes that are never trained. Finally, a learning-based network (Deform3DNet) has been proposed for non-rigid 3D point cloud registration. Deform3DNet shows how deep learning is used successfully in solving the non-rigid registration and correspondence challenges end-to-end with non-rigid 3D point clouds. The experimental results demonstrate improvement in the quality of the non-rigid registration and correspondence by comparing Deform3DNet with seven state-of-the-art non-rigid 3D point cloud registration and correspondence methods across large deformations, partiality and topological noise.

Contents

List of figures

List of tables

Acknowledgments

This work would not have been possible without the support of many people. In the beginning, I would like to express my sincere gratitude to my supervisor, Prof. Wen Tang, who was always willing to discuss with me and encouraged me in all the time of my academic research and daily life. Her brilliant supervision, patience, enthusiasm and invaluable suggestions enriched this work and made me become more confident in academic research. I appreciate all the fruitful discussions we had that contributes lots of ideas to this thesis.

Apart from my Supervisor, I won't forget to express my gratitude to Prof. Tao Xue, for giving encouragement and sharing insightful suggestions. I would also like to thank Prof. TaoRuan Wan at the University of Bradford for his encouragement and support all through my work. Their endless guidance is hard to forget throughout my life.

I am grateful to Yan Zhao and Qinghong Gao for their collaboration and encouragement, making me more knowledgeable about academic research. I would like to thank Daniel Green, Weilai Xu, and Rantilini Samaratunga who worked with me in the same laboratory during these years. We discussed a lot about new technologies, life in the UK, music and so forth, which would become memorable and valuable memories throughout my life. I would also like to thank the musicians who created fantastic melodies and heal my soul when life brings me down. Nothing else can heal me as music does.

I must express my gratitude to my parents who give me my life and always love me. I love them all my life.

Declaration

I, LONG XI, hereby declare that this thesis represents my own work which has been done after registration for the degree of P.h.D at Bournemouth University and has not been submitted to any other institute for any award or qualifications. I also confirm that this work fully acknowledges the contributions from the work of others.

LONG XI
May 2023

Nomenclature

Acronyms / Abbreviations

1D Conv 1D convolution

AE Auto Encoder

AR Augmented Reality

BCPD Bayesian Coherent Point Drift

BM Block Matching

BN Batch Norm

BP Backward propagation

BTreeNet Binary Tree Network

CD Chamfer Distance

CPD Coherent Point Drift

DCP Deep Closest Point

DeepGMR Deep Gaussian Mixture Registration

DGCNN Deep Graph Convolutional Neural Network

DO Discriminative Optimization

EMD Earth Mover's Distance

FC Fully Connected

FM Functional Maps

FMNet Function Maps Network

FMR Feature-metric Registration

GAN Generative Adversarial Net

GMM Gaussian Mixture Model

IBTreeNet Iterative Binary Tree Network

ICP Iterative Closest Point

KNN K Nearest Neighbour

LK Lucas & Kanade

MAE Mean Absolute Errors

MIE Mean Isotropic Rotation and Translation Errors

NDT Normal Distribution Transformation

Non-rigid ICP Non-rigid Iterative Closest Point

PCN Point Cloud Network

PRST Planar Reflective Symmetry Transform

RGM Robust Graph Matching

RPM Robust Point Matching

SfM Structure from Motion

SGBM Semi-global Block Matching

SLAM Simultaneous Localization and Mapping

SURFMNet Spectral Unsupervised FMNet

SVD Singular Value Decomposition

TLS Terrestrial Laser Scanners

TPS Thin Plate Spline

Chapter 1

Introduction

1.1 Background

Augmented Reality (AR) technology combines the real world with virtual information to enhance the human perception of the world. AR has been widely used in many applications, for example, education [2] [3], minimally invasive surgery [4] [5] [6], robot path planning [7] [8] and digital games [9] [10] [11]. Azuma [12] defines AR as systems that have three characteristics, including a combination of virtual objects and real 3D scenes, interactive in real-time and 3D registration. AR should register or correct the alignment of the virtual 3D objects with real 3D scenes [13] [14] [15]. Therefore, a real-world 3D scene is required and needed to be aligned with the virtual 3D object for more accurate augmented information in the AR system. The higher the quality of the underlying real-world 3D data is, the more accurate the AR system becomes.

3D data, captured by various sensor technologies, such as LiDAR sensors, RGB-D scanners and depth cameras, are usually formed as raw 3D point clouds. The higher the quality of the underlying real-world 3D point cloud scene is, the more accurate the AR system becomes [4]. However, one of the limitations of these sensors is that the captured 3D point clouds suffer from large missing data due to complicated occlusions, unreliable measurements, limited viewing angles and the resolution of various sensors in dealing with texture-less regions (e.g. water, glasses, sky, surfaces of human organs).

3D point clouds can be also reconstructed from depth estimation algorithms that generate 3D depth information from images. Traditional depth estima-

tion algorithms, block matching method[3] (BM) and semi-global block matching (SGBM) [16], use stereo image pairs as input to search for matched pixels in stereo pairs, which results in lower-quality of 3D point clouds. The structure for motion (SFM) systems estimate detailed depth maps and produce dense points [17], which rely on powerful commodity GPU processors for real-time performance and stereo visions. Recent advances in supervised monocular depth estimation predict the depth from a single image [18] [19], which requires ground truth depth. State-of-the-art unsupervised learning-based methods explore easier-to-obtain binocular stereo footage without the need for ground truth depth [20] [21]. However, the major limitations of these methods are the occlusions of objects, the change of brightness of texture-less surfaces, and the surface smoothness for feature extractions.

To deal with the limitations of various sensors and depth estimation algorithms and achieve high-quality 3D point clouds, generating a complete 3D point cloud (i.e. 3D point cloud completion) from a captured or reconstructed incomplete point cloud is an essential task for a wide range of 3D vision applications from AR system [4] [22] and robotics [23] [7] to navigation and scene understanding [24] [8], to minimally invasive surgery [25]. In this thesis, a novel computational framework for recovering dense and high-quality 3D point clouds from single monocular endoscopic images in minimally invasive surgery is first proposed. The proposed computational framework combines two main modules. One is for monocular depth estimation, and the other is for 3D point cloud completion. The depth estimation module generates depth information from monocular images captured by an endoscope during minimally invasive surgery. The depth maps are then reconstructed into 3D point clouds. The 3D point cloud completion module repairs defects of 3D point clouds. By using this framework, five large medical *in-vivo* databases of 3D point clouds are generated from public Laparoscopic/Endoscopic video datasets [26] [27], and two synthetic 3D medical datasets are also created. These datasets are made publicly available for researchers. This framework is the first attempt of its kind applied to 3D point cloud completion in endoscopic scenes during minimally invasive surgery.

3D point cloud completion refers to the process that repairs the flow of the 3D data by filling holes and incomplete parts of 3D point clouds. Research on

[3]https://opencv.org

3D point cloud completion can be categorized into three classes of approaches including geometry-based, data-driven based and learning-based methods.

Geometry-based approaches [28] [29] [30] [31] complete 3D shapes by using the symmetric information from the partial input. Data-driven-based methods [32] [33] retrieve suitable example patches or parts from the shape databases and warp the retrieved models to conform with the partial input data. The major drawbacks of geometry-based and data-driven-based methods are: (i) Not all objects are symmetric and the partial object might not show symmetric features; (ii) Data-driven-based methods are time-consuming and rely on the assumption that the database must include a very similar shape.

Previous learning-based methods for 3D point cloud completion [34] [35] [36] convert 3D datasets into structured 3D voxel grids and use voxelized representations for network training, which consumes high computer memories and reduces output qualities. After the first learning-based 3D point cloud processing network PointNet [1] has been proposed, the first neural network for 3D point cloud completion [37] is proposed based on the PointNet [1] for a single class (e.g. car objects). Recent state-of-the-art learning-based methods, FoldingNet [38], PCN [39], TopNet [40] and Disp3d [41], propose the PointNet-based [1] encoder and their own decoders to regenerate the entire 3D point cloud from real-world partial 3D point clouds (i.e. cars, tables, chairs and etc.). PMPNet [42] translates each point in the partial input to the missing areas instead of directly regenerating the complete 3D point clouds. However, there are two shared problems with these methods. (i) These methods are sensitive to the geometric forms and shapes of training datasets of 3D point clouds. Therefore, they cannot handle multi-classes effectively, even a single class that contains mostly different shapes. Training a class-invariant model with multiple classes of training datasets is difficult with these methods. When the number of classes increases in the training dataset, these methods produce low-quality 3D point cloud outputs that lack structural and spatial details, such as sharp edges and topology changes; (ii) These methods lose original structural and spatial details in the final output due to the fact that they regenerate the entire 3D point cloud and do not separate the reconstructed partial input from the missing points in the final output.

Based on the two problems of these state-of-the-art learning-based methods for 3D point cloud completion, in this thesis, a novel deep learning-based network

by devising hierarchical tree-based decoders is proposed to build a learning-based network-TreeNet for solving the two problems. TreeNet combines two sub-networks. One (TreeNet-multiclass) is for multi-class training and the other (TreeNet-binary) is for missing points generation with original structure-preserving. More specifically, TreeNet-multiclass assigns each class of a 3D point cloud completion task to a specific sub-tree of the root node in the tree. TreeNet-binary splits features of the root node in the tree to a binary tree structure where the left leaf node reconstructs the partial input and the right leaf node generates points in missing areas. Following FoldingNet [38], PCN [39], TopNet [40] and Disp3d [41], the TreeNet is still trained and tested on real-world 3D objects (i.e. cars, tables, chairs and etc.). In addition, the proposed TreeNet is also evaluated in the proposed computational framework in endoscopic scenes as an additional application. The TreeNet focuses on generating the missing areas of the partial endoscopic 3D point clouds, whereas the state-of-the-art learning-based methods, FoldingNet [38], PCN [39], TopNet [40] and Disp3d [41], lose original structural and spatial details in the final output by regenerating the entire 3D point cloud and do not separate the reconstructed partial input from the missing points in the final output, which is unacceptable during the minimally invasive surgery.

3D point cloud registration refers to estimating matching rotations and translations for rigid and non-rigid 3D point clouds. For rigid 3D point clouds (i.e. computer, table and chair) registration, a transformation matrix (rotation and translation) is estimated and applied to all points in one 3D point cloud to align with the other. For non-rigid 3D point clouds (i.e. human, cat and dog) registration, each point in one 3D point cloud needs to be transformed by an independent transformation matrix to align to its corresponding point in the other.

Traditional rigid 3D point cloud registration methods, Iterative Closest Point (ICP) [43], Normal Distribution Transformation (NDT) [44] and Coherent Point Drift (CPD) [45], consider rigid 3D registration as an optimization problem. However, the major drawbacks of these methods are that they are sensitive to the initialization of 3D point clouds and can be computationally expensive and time-consuming.

Recent emerged state-of-the-art learning-based approaches for rigid 3D point cloud registration, PointNetLK [46], Deep Closest Point (DCP) [47], Robust Point Matching (RPM) [48], Robust Graph Matching (RGM) [49], Feature-metric

Registration (FMR) [50] and Deep Gaussian Mixture Registration (DeepGMR) [51], propose the PointNet-based [1] encoders and their own decoders to estimate the transformation matrix to align one 3D point cloud to the other. However, the state-of-the-art learning-based methods [46] [47] [48] [50] [51] [49] have four drawbacks. (i) These methods learn the rotation and translation features together and generate the rotation and the translation in one matrix. As a result, the learning of rotation features and translation features interfere with each other, which leads to lower precision of registration results. (ii) PointNetLK [46], DCP [47], RPM [48], DeepGMR [51] and RGM [49] need ground-truth transformation matrix or point-to-point correspondences to supervise the training process. (iii) PointNetLK [46], DCP [47], RPM [48], DeepGMR [51] and RGM [49] suffer on partial 3D point clouds without training in this scenario, which shows the poor generalization ability of these networks. (iv) PointNetLK [46], DCP [47], RPM [48], FMR [50] and RGM [49] often perform poorly in dealing with large and dense scenes and shapes that are not trained, resulting in misalignment between two 3D point clouds.

Based on the four drawbacks of these state-of-the-art learning-based methods for rigid 3D point cloud registration, in this thesis, a novel unsupervised deep learning-based network - Iterative Binary Tree Network (IBTreeNet) is proposed to improve the registration accuracy for large and dense 3D point clouds. IBTreeNet avoids the interference between the feature extraction of rotation and translation and does not need the ground-truth transformation matrix as supervision. IB-TreeNet learns features for the rotation separately from the translation and avoids the interference between the estimations of rotation and translation in one single matrix and iteratively improves the registration accuracy for large and dense 3D point clouds. The registration results of the IBTreeNet have been compared with three traditional methods and six state-of-the-art learning-based methods. The comparison experiments are evaluated on testing datasets, including clear data, partially visible data, data with Gaussian noise and data with large rotations. Most importantly, the comparison experiments are also tested on large and dense unseen scenes and shapes that are not trained to evaluate the generalization and robustness of each method. IBTreeNet outperforms state-of-the-art learning-based and traditional methods on partial and noisy point clouds without training them

in such scenarios and exhibits remarkable generalization and robustness to unseen large and dense scenes that are never trained.

Non-rigid 3D point clouds registration is a fundamental challenge in computer vision and computer graphics, with applications in shape analysis [52] [53] deformation transfer [54], 3D reconstruction [25] [55] [56], and 3D object tracking [57, 58]. Although previous non-learning based methods [59] [45] [60] have addressed non-rigid 3D point clouds registration, there are several shared problems: (i) These methods heavily rely on the initial poses; (ii) The transformation of the points is constrained by the adjacent points. As a result, these methods cannot find the optimal alignment for 3D point clouds with large and multiple deformations; (iii) These methods can be computationally expensive and time-consuming.

Without taking into account 3D point cloud registration, many state-of-the-art methods [61] [62] [63] [64] [65] have been proposed to find correspondence between non-rigid 3D shapes, which are both non-learning and learning-based approaches. Non-learning-based methods ZoomOut [61] and Fast Sinkhorn Filter [62] focus on structured 3D meshes instead of unstructured 3D point clouds and use functional maps to estimate the point-wise correspondences. However, Non-learning based methods are time-consuming and limited when handling large and multiple deformations between 3D shapes. Learning-based methods FM-Net [63] and SURFMNet [64] are also based on functional maps [66] that require pre-computed feature descriptors of 3D meshes (i.e. SHOT descriptor [67]) as the input of neural networks and apply ground-truth geodesic distances as supervision. CorrNet3D [65] is the first deep neural network that takes raw 3D point clouds as input and considers the learning of correspondence between 3D point clouds as the deformation-like 3D reconstruction. However, there are two major drawbacks to these methods: (i) These methods are limited when handling large and multiple deformations between 3D shapes; (ii) These methods do not provide non-rigid 3D point cloud registration after finding the point-wise correspondences.

To achieve high-performance non-rigid 3D point cloud registration, for the first time, an end-to-end deep learning-based network (Deform3DNet) is proposed for non-rigid 3D point cloud registration. In Deform3DNet, a point-to-point transformation module is proposed, which applies *as rigid as possible* for the non-rigid transformation to generate a transformation matrix for each point in a point-to-point transformation layer. A non-rigid registration loss is proposed,

which is differentiable and minimizes the Frobenius norm between the transformed 3D point cloud and the target. A novel structure-preserving loss is proposed by maximizing the similarity of grouped points between the transformed and the target 3D point clouds, keeping the internal structure in the transformed 3D point cloud.

1.2 Hypothesis and Research Question

This thesis hypothesizes that a more accurate AR system can be obtained from the improved 3D point cloud registration. This thesis also hypothesizes that generating high-quality 3D point clouds by recovering missing data can improve the 3D point cloud registration accuracy and further improve the quality of AR systems. Therefore, the research questions for this thesis are (i) What factors have influenced the 3D point cloud registration accuracy in the state-of-the-art algorithms, and how to deal with it and achieve more accurate 3D point cloud registration? (ii) How can high-quality and complete 3D point clouds be generated from partial observations and how can the 3D point cloud completion algorithm focus on missing points generation? (iii) How can 3D point cloud completion and registration algorithms be applied for improving the accuracy of AR systems?

1.3 Research Contributions

This thesis develops a computational framework, deep learning-based neural networks for 3D point cloud completion, rigid 3D point cloud registration and non-rigid deformable 3D point cloud registration. The AR applications are finally developed based on the proposed completion and registration networks to achieve accurate 3D point cloud registration.

More specifically, the main contributions of this thesis are:

- **Recovering Dense 3D Point Clouds from Single Endoscopic Image** *(Chapter 3)*

  A novel computational framework is proposed to recover high-quality 3D point clouds from single endoscopic images by combining two deep-learning neural networks. One is for monocular depth learning, and the other is

for 3D point cloud completion. Five large medical *in-vivo* databases of 3D point clouds are generated from public Laparoscopic/Endoscopic video datasets [26] [27], and two synthetic 3D medical datasets are also created. 3D point clouds are extracted from every frame of the video datasets. The datasets are publicly available at[1].

- **TreeNet: Structural Preserving for Multi-class 3D Point Cloud Completion** *(Chapter 4)*

  A novel TreeNet-multiclass decoder is proposed for multi-class 3D point cloud completion. The model is evaluated on 50 classes of training datasets(i.e. cars, tables, chairs and etc.), whereas the majority of the state-of-the-art methods only use 8 classes. A novel TreeNet-binary decoder is proposed, which focuses on generating points in missing areas and fully preserving the original partial input 3D point cloud. A novel TreeNet decoder is proposed, which combines the advantages of the TreeNet-multiclass and the TreeNet-binary for 3D point cloud completion. Three novel forward and backward propagation methods are proposed to train TreeNet-multiclass, TreeNet-binary and TreeNet decoders, respectively. TreeNet-multiclass, TreeNet-binary and TreeNet exhibit strong generalization to unknown classes that are never trained. In addition, the proposed TreeNet is also evaluated in the proposed computational framework in endoscopic scenes as an additional application. Experimental results prove the effectiveness of the TreeNet on endoscopic scenes in minimally invasive surgery.

- **Iterative BTreeNet: Unsupervised Learning for Large and Dense 3D Point Cloud Registration** *(Chapter 5)*

  A BTreeNet with a novel forward propagation based on the hierarchical binary tree is proposed to align two 3D point clouds, which learns features for the rotation separately from the translation and estimates the rotation matrix separately from the translation matrix. A novel Iterative BTreeNet (IBTreeNet) is proposed to continuously improve the registration accuracy, which iteratively rotates and translates the source 3D point cloud to the target. The Chamfer Distance and the Earth Mover's Distance are adopted

[1]The datasets are publicly available to researchers at https://github.com/LONG-XI/Endoscopic-3D-Point-Clouds-Datasets/

as the loss function for unsupervised learning of 3D point cloud registration. BTreeNet and IBTreeNet are tolerant to partial overlap, noise and large scenes without training them in such scenarios. Iterative BTreeNet also exhibits remarkable generalization to different unseen large and dense scenes that are never trained.

- **Deform3DNet: A Unified Deep Learning Network for Non-rigid Deformable 3D Point Cloud Registration and Correspondence** *(Chapter 6)*

  An end-to-end learning-based network for non-rigid 3D point cloud registration is proposed, which also leads to finding the point-to-point correspondence. A novel non-rigid registration loss for deformable 3D point clouds and a novel structure preserve loss to keep the internal structure for transformed 3D point clouds. Experimental results illustrate the significant improvement of the Deform3DNet over the state-of-the-art methods on large and multiple deformations, non-rigid partiality and topological noise (Gaussian noise).

- **Augmented Reality Application** *(Chapter 7)*

  The proposed neural networks for 3D point cloud completion and registration are applied to achieve stable and accurate 3D point cloud registration in AR applications. Experimental results illustrate the effectiveness of the proposed completion and registration networks for increasing the accuracy of registration between virtual 3D objects and real-world scenes.

1.4 Thesis Outline

- **Chapter 1** introduces the research background of AR, 3D point cloud completion and registration, the hypothesis of this thesis, the main contributions, the thesis outline and the list of publications.

- **Chapter 2** reviews related works related to 3D point cloud completion, rigid 3D point cloud registration and non-rigid 3D point cloud registration and correspondence.

- **Chapter 3** introduces the proposed computational framework by recovering high-quality 3D endoscopic point clouds from single endoscopic images. The

3D point cloud completion algorithm in this framework regenerates the whole 3D point cloud, which destroys the original partial 3D point cloud in the final output and is not acceptable for medical scenes. Therefore, in Chapter 4, a novel algorithm has been proposed for generating points only in missing areas.

- **Chapter 4** introduces the proposed TreeNet-binary, TreeNet-multiclass and TreeNet for 3D point cloud completion. TreeNet-binary focuses on missing points generation and TreeNet-multiclass is for multi-class training. TreeNet combines the advantages of TreeNet-binary and TreeNet-multiclass.
- **Chapter 5** deals with two shared problems of the state-of-the-art 3D point cloud registration methods and introduces the proposed BTreeNet and IBTreeNet for more accurate rigid 3D point cloud registration.
- **Chapter 6** introduces the proposed Deform3DNet for non-rigid deformable 3D point cloud registration and correspondence.
- **Chapter 7** presents the AR experiments, illustrating how the proposed 3D point cloud completion and registration algorithms improve the accuracy of AR systems.
- **Chapter 8** concludes the thesis and discusses future research directions.

1.5 List of Publications

Accepted Journal Papers

- **Long Xi**, Wen Tang, TaoRuan Wan. TreeNet: Structure Preserving Multi-class 3D Point Cloud Completion. Pattern Recognition, 139 (2023): 109476.
- **Long Xi**, Wen Tang, Tao Xue, TaoRuan Wan. Iterative BTreeNet: Unsupervised Learning for Large and Dense 3D Point Cloud Registration. Neurocomputing 506 (2022):336-354.

- **Long Xi**, Yan Zhao, Long Chen, QingHong Gao, Wen Tang, TaoRuan Wan, Tao Xue. Recovering dense 3D point clouds from single endoscopic image. Computer Methods and Programs in Biomedicine 205 (2021):106077.
- QingHong Gao, Yan Zhao, **Long Xi**, Wen Tang, TaoRuan Wan. Break and Splice: A Statistical Method for Non-rigid Point Cloud Registration, Computer Graphics Forum, 2023.
- Yan Zhao, Wen Tang, Jun Feng, TaoRuan Wan, **Long Xi**. General Discriminative Optimization for point cloud Registration. Computers & Graphics, 102 (2022): 521-532.
- Yan Zhao, Wen Tang, Jun Feng, TaoRuan Wan, **Long Xi**. Reweighted Discriminative Optimization for Least-squares Problems with Point Cloud Registration. Neurocomputing, 464 (2021): 48-71.

Papers Currently Under Review

- **Long Xi**, Wen Tang, TaoRuan Wan. Deform3DNet: A Unified Deep Learning Network for Non-rigid Deformable 3D Point Cloud Registration and Correspondence. (submitted to Computer Graphics Forum)
- Yan Zhao, **Long Xi**, Wen Tang, Jun Feng, TaoRuan Wan. SGRTmreg: A Learning Based Optimization Framework for Multiple Point Clouds Registration. (submitted to Pattern Recognition)

Chapter 2

Related Works

This chapter reviews traditional and state-of-the-art approaches for 3D completion, rigid 3D point cloud registration and non-rigid 3D point cloud registration and correspondence. 3D completion contains 3D mesh completion, volumetric 3D completion and 3D point cloud completion, aiming to generate high-quality and complete 3D data from partial observations. 3D point clouds are the raw 3D data captured by various 3D sensors, such as LiDAR sensors, RGB-D scanners and depth cameras. Therefore, this thesis focuses on 3D point cloud completion. Rigid 3D point cloud registration finds the best matching transformation matrix to align one 3D point cloud to the other. Non-rigid 3D point cloud registration and correspondence find point-to-point correspondences and align them between two non-rigid and deformable 3D point clouds.

2.1 3D Completion

Research on 3D completion can be categorized into three classes of approaches: geometry-based, data-driven based and learning-based methods.

Geometry-based approaches complete 3D meshes by using geometric cues from the partial input. Thrun et al. [28] propose a symmetry-searching algorithm to identify probable symmetries and apply the symmetry information to extend the partial 3D shape. Podolak et al. [29] describe a planar reflective symmetry transform (PRST) algorithm that captures all possible symmetry planes, and then finds the greater symmetries among them. Several extensions of symmetries detecting algorithms [30] [31] have been proposed. They extract several partial symmetries of the 3D mesh object. However, not all objects are symmetric and the partial object might not show symmetric features.

Data-driven based method [32] for 3D mesh completion retrieves suitable example patches or parts from the shape databases and warps the retrieved models to conform with the partial input data. In this method, the proposed non-rigid alignment algorithm aligns the retrieved model with the input data. In addition, Sung et al. [33] have combined the symmetry-based method and the data-driven-based method together for improving the performance of 3D shape completion. However, this technique is time-consuming and relies on the assumption that the database must include a very similar shape.

This thesis focuses on learning-based methods that can be further categorized according to the forms of the input (i.e. 3D voxel grids or 3D point clouds).

2.1.1 Volumetric 3D Completion

Currently, one major promising progress for the 3D completion task is utilizing 3D learning-based neural networks that are successful at

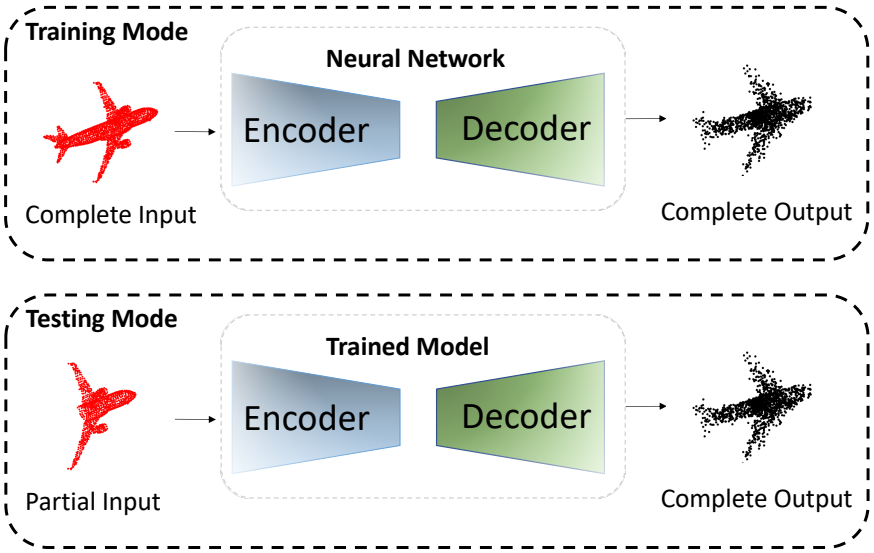


Fig. 2.1 One of the structures of deep learning methods for 3D point cloud completion.

learning 3D data representations and features automatically, reducing the incompleteness caused by designing features manually. 3D learning-based architecture largely depends on the representation of the 3D data, such as volumetric voxel grids or 3D point clouds. Since convolutional neural networks can process structured and ordered 3D datasets more effectively than unstructured datasets, most previous 3D learning-based methods [34] [35] [36] used voxelized representations for 3D shape completions. However, voxelization causes high computational cost when the resolution increases dramatically, and low-resolution results in low-quality output [68] [69].

2.1.2 3D Point Cloud Completion

Figures 2.1 and 2.2 show the structures of deep learning methods for the 3D point cloud completion task with two different training strategies. Training with the complete 3D point clouds [37] [25], as

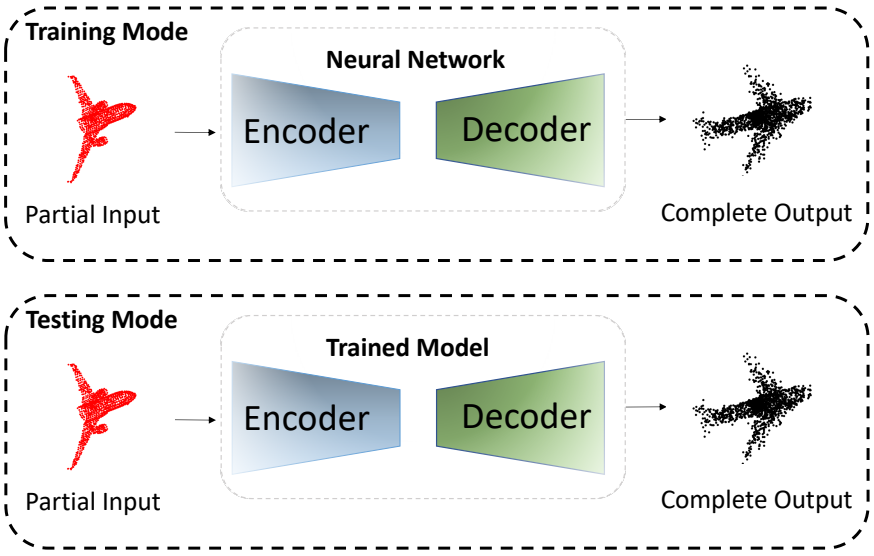


Fig. 2.2 One of the structures of deep learning methods for 3D point cloud completion.

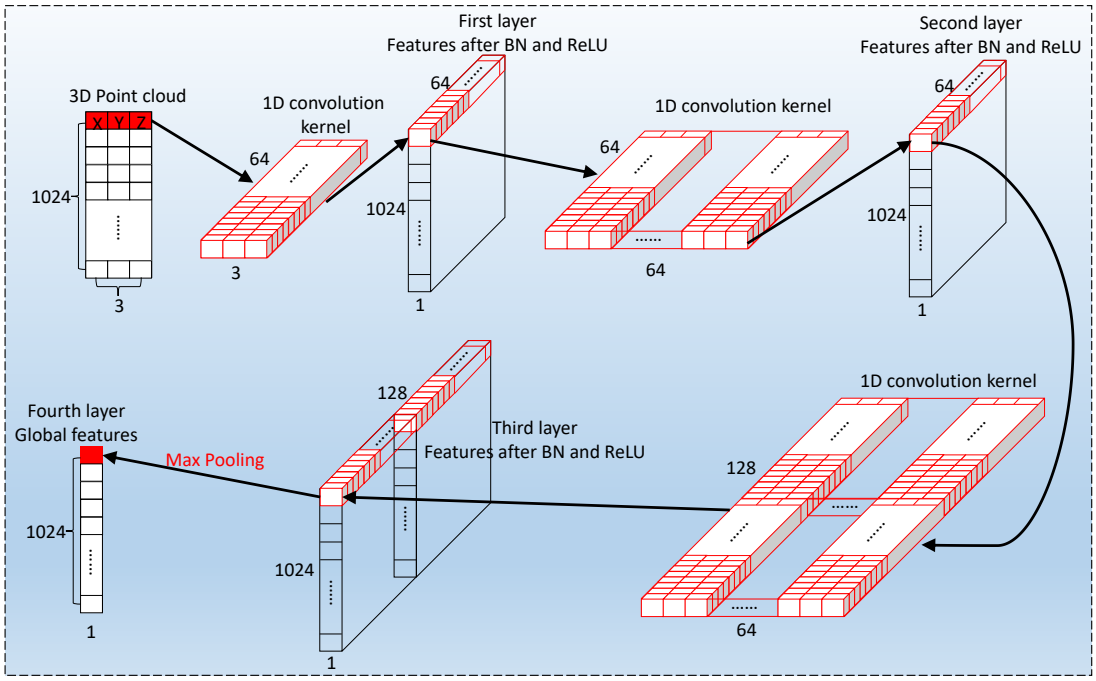


Fig. 2.3 Illustration of PointNet [1] feature extraction encoder. Four layers are illustrated and displayed. The arrows lead to how the features are extracted from a point. Every point has the same operation for feature extraction.

shown in Figure 2.1, aims to extract 3D point cloud representations that can be further used for generating the complete 3D point cloud from the partial input during the testing. Training with the partial 3D point clouds [40] [38] [39] [42] [41], as shown in Figure 2.2, aims to find the global features that correlate partial 3D point clouds with their corresponding complete 3D point clouds using a neural network. The encoders in state-of-the-art methods [25] [37] [40] [38] [39] are based on PointNet [1], and they also propose their own decoders for 3D point cloud completion task.

PointNet [1] is the first deep neural network that applies raw 3D point cloud for 3D point cloud processing without voxelization. The basic idea of PointNet [1] is to learn a spatial encoding of each point and then aggregate all individual point features to the global features of a 3D point cloud, as shown in Figure 2.3. Specifically, PointNet applies a 1D convolution, a batch norm (BN) and a ReLU as a group for a feature extraction module. The 1D convolution, defined in Equation 2.1, aims to generate point features individually. After several feature extraction modules, a max pooling layer is used to generate the global features of a 3D point cloud. Because, max pooling is a symmetric function, which aims to generate global features that are invariant to any permutation of a 3D point cloud. PointNet [1] has been proven to be useful for tasks including 3D point cloud classification [1] [70] [71] [72], object recognition [73] [74], object detection [75] [76], object segmentation [1] [70] [77] [78] [79], object reconstruction [80] [81], completion [25] [37] [40] [38] [39] [42] [41] and registration [82] [46] [47] [48] [50] [51] [49].

$$Conv1D = \sum_{i=1}^{B} \sum_{j=1}^{N} Kernal \cdot input_{ij} \tag{2.1}$$

where $\cdot$ is the dot product operation, i indicates the ith data or features in a batch B, j indicates the jth row of data or features, N indicates there are N points in a 3D point cloud, $Kernal$ is the 1D convolution kernel as shown in Figure 2.3. The size of one sliding window for a $Kernal$ is identical to the size of each row of a 3D point cloud or a feature matrix from the previous layer, and the number of sliding windows is identical to the size of each row of a feature matrix in the next layer.

Achlioptas et al. [37] introduce an Auto Encoder [83] and a Generative Adversarial Net (GAN) [84] to learn 3D point cloud representations by focusing on a single class of 3D point cloud completion task. FoldingNet [38] proposes a folding operation in the decoder that deforms a 2D grid into a 3D point cloud and evaluates the different layers of the folding operations to tune the model. Point Cloud Network (PCN) [39] evaluates the different number of PointNet layers and fully connected layers in the encoder and decoder to achieve the best performance. PCN also uses a folding operation to generate higher-resolution 3D point clouds from the coarse 3D point clouds in the final stage of the decoder. TopNet [40] first evaluates the encoders in PointNet [1], PointNet++ [70] and PCN [39] and finally chooses the encoder from PCN [39]. TopNet [40] then proposes a decoder following tree structure and evaluates the number of 1D convolution layers with different feature sizes in the decoder to achieve the highest accuracy in the 8 classes of datasets.

Specifically, the root node in TopNet [40] is the global feature of the input data. The output of each leaf node represents *a single point* in a 3D point cloud, and all leaf nodes consist of a complete 3D point cloud. Therefore, features of a partial input in the tree root pass through all nodes to regenerate an entire 3D point cloud. The architecture of the TopNet [40] is not designed for multi-class 3D point cloud completion, and it loses the structural and spatial details of partial inputs. PMPNet [42] applies PointNet++ [70] encoder and generates translation matrices for each point in the partial input in the decoder, which translates the incomplete input to the nearest occluded regions. PMPNet [42] also analyses the different recurrent units and the different searching radii in the proposed recurrent path aggregation module to tune the decoder. Disp3d [41] proposes a down-sampling operation, a neighbour pooling and an up-sampling operation to regenerate the complete 3D point clouds.

2.2 Rigid 3D Point Cloud Registration

Research on rigid 3D point cloud registration can be categorized into two classes of approaches: traditional or deep learning-based methods. Traditional rigid 3D point cloud registration methods consider the registration an optimization problem, applying the least square regression or maximizing the likelihood of a probability density function. Deep learning-based methods are successful at learning rigid 3D point cloud representations and features by estimating the rigid transformation in the alignment. This thesis focuses on deep learning-based methods for rigid 3D point cloud registration.

2.2.1 Traditional Rigid 3D Point Cloud Registration Methods

Iterative closest point (ICP) [43] and its variants [85] [86] are well-known traditional methods for rigid 3D point cloud registration by finding point cloud correspondences and solving a least-squares problem to update the alignment. Normal Distribution Transformation (NDT) [44] uses the statistical models of 3D point clouds in the alignment. The key element in NDT [44] is its representation of the 3D point clouds. Instead of using each individual point in the 3D point cloud, NDT [44] converts the 3D point clouds into voxel grids. The grids are represented by a combination of normal distributions, describing the probability of finding a point at a certain position. NDT [44] uses the representation of normal distributions to apply standard numerical optimization methods for registration. Coherent Point Drift (CPD) [45] considers the alignment of two 3D point clouds a probability density estimation problem, where one 3D point cloud represents the Gaussian Mixture Model (GMM) centroids that need to align the other 3D point cloud. CPD [45] moves the GMM centroids coherently as a group to the other 3D point cloud by maximizing the likelihood. However, ICP-based methods [43] [85] [86], NDT [44] and CPD [45] are time-consuming and prone to local minima when two 3D point clouds whose initial positions are far from aligned.

Discriminative Optimization (DO) [87] and its variant Reweighted Discriminative Optimization [88] are the supervised sequential update methods that learn the update steps for solving the least-squares problem to obtain the transformation matrix. The learning

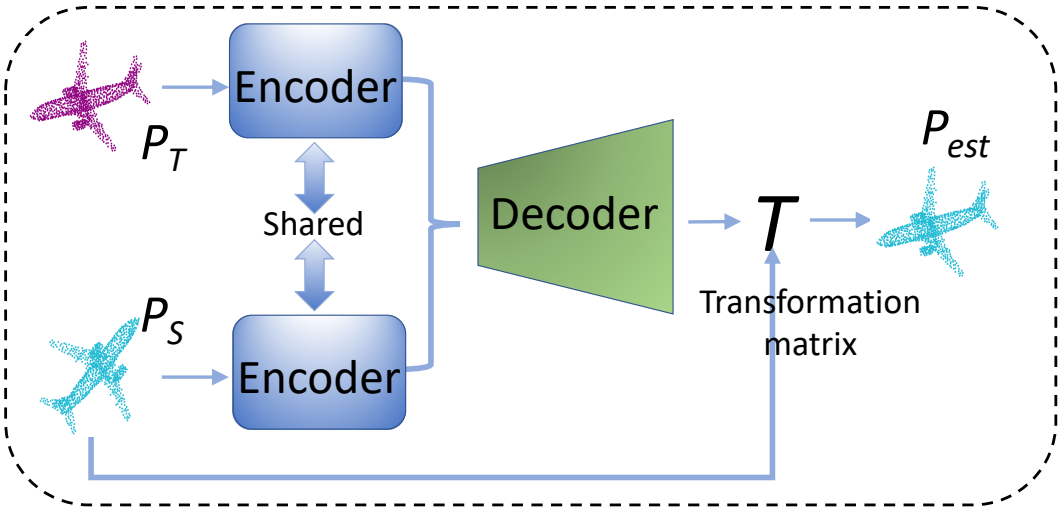


Fig. 2.4 The structure of deep learning methods for rigid 3D point cloud registration.

processes of optimization methods [87] [88] focus on the sequence of update maps for each individual 3D point cloud and need to be retrained on each individual data, whereas deep learning-based methods [46] [47] [48] [50] [51] [49] learn the generalized features from a large 3D point cloud dataset, and the trained features can be used to unseen 3D point clouds that are not trained.

2.2.2 Deep Learning-based Methods on Rigid 3D Point Cloud Registration

Figure 2.4 shows the structure of deep learning methods for rigid 3D point cloud registration. A shared encoder is used to extract the global features of 3D point cloud pairs. The decoder aims to generate a transformation matrix (i.e. rotation and translation) that is transformed from one 3D point cloud to the other.

Recently proposed PointNetLK [46] is a pioneer in the task of rigid 3D point cloud registration. PointNetLK [46] combines a

deep learning method PointNet [1] as an encoder and a traditional registration method Lucas-Kanade algorithm [89] at the end of the decoder to achieve features automatically and minimize the distances between the global feature descriptors in the alignment. DCP [47] utilizes DGCNN [72] and an attention module [90] to extract features of two 3D point clouds and replaces the Lucas-Kanade [89] algorithm in PointNetLK [46] with a proposed differentiable Singular Value Decomposition (SVD) module to reduce feature dimension. The SVD module in DCP [47] estimates a transformation matrix with the size of 7, where the first three output values represent the translation matrix and the last four values represent the rotation quaternion. RPM [48] uses raw 3D point clouds and normals as input for the DGCNN-based [72] feature extraction module to estimate point correspondences between two 3D point clouds. Similarly to DCP [47], the weighted SVD module at the end of the network estimates a transformation matrix with the size of 7 from the point correspondences. RGM [49] transforms 3D point clouds into graphs and learns point and graph features via a graph feature extractor to calculate the point correspondences. Similarly to DCP [47] and RPM [48], the transformation matrix in RGM [49] is also estimated from a differentiable SVD. FMR [50] uses the chamfer distance [91] as a loss function for unsupervised learning and proposes a feature-metric projection error as a decoder for updating the transformation parameters during each iteration. DeepGMR [51] proposes a network that extracts pose-invariant correspondences between 3D point clouds and Gaussian Mixture Model (GMM) parameters. Two differentiable compute blocks are proposed in the decoder of DeepGMR [51] to recover the optimal transformation from matched GMM parameters,

which achieves favourable performance on 3D point clouds with point-to-point correspondences and large transformations, respectively.

However, these state-of-the-art learning-based methods [46] [47] [48] [50] [51] [49] learn the rotation and translation features together and generate the rotation and the translation in one matrix. As a result, the learning of rotation features and translation features interfere with each other, which leads to lower precision of registration results. Moreover, these methods use the ground-truth transformation or point correspondence matrix as supervision. This thesis avoids the interference between the feature extraction of rotation and translation and does not need the ground-truth transformation matrix for supervision.

2.3 Non-rigid 3D Point Cloud Registration and Correspondence

Traditional non-rigid 3D point cloud registration methods [92] [93] [59] [45] [60] consider the non-rigid registration an optimization problem. The state-of-the-art learning-based methods [63] [64] [65] find point-wise correspondences between non-rigid deformable 3D point clouds, but do not consider the non-rigid 3D point cloud registration. The state-of-the-art learning-based methods [46] [47] [48] [50] [51] [49] for rigid 3D point cloud registration generates a transformation matrix for a whole 3D point cloud, which cannot be used for non-rigid 3D point cloud registration.

2.3.1 Non-rigid 3D Point Cloud Registration

Earlier approaches [92] [93] on non-rigid point cloud registration have used Thin Plate Spline (TPS) [94] to handle deformation and deterministic annealing for soft-assignment. TPS parameters estimation solves a maximum likelihood estimation problem to find the optimal controlling points and displacement vector for warping the point set. These methods have been applied to 2D point clouds with a small number of points.

Non-rigid Iterative Closest Point (Non-rigid ICP) [59] extends Iterative Closest Point (ICP) [95] method to compute non-rigid deformations. While retaining the convergence properties of ICP, Non-rigid ICP uses a locally affine regularisation that assigns an affine transformation to each point, minimising the difference of transformation matrices between the adjacent points. Least-squares problem is solved to update the alignment for the Non-rigid ICP. However, Non-rigid ICP can not deal with large and multiple deformations and heavily relies on the initial poses of the two non-rigid 3D point clouds for registration.

Coherent Point Drift (CPD) algorithm [45] uses a probabilistic method for both rigid and non-rigid 3D point cloud registration. CPD treats the alignment of two 3D point clouds as a probability density estimation problem. For non-rigid 3D point cloud registration, the CPD defines the transformation as the initial position plus a displacement function v. It uses the motion coherence theory [96] [97] to enforce the smoothness for v between the point clouds. The CPD finds the functional form of v using the calculus of variation and solves the minimization problem of the negative log-likelihood.

Whereas Bayesian Coherent Point Drift (BCPD) [60] formulates CPD in a Bayesian setting for both rigid and non-rigid 3D point cloud registration, BCPD replaces the motion coherence theory [96] [97] with Bayesian inference. The major difference between BCPD and CPD is that BCPD defines motion coherence using a prior distribution instead of the regularization term. Both CPD and BCPD are time-consuming, prohibiting 3D point clouds with large and multiple deformations.

2.3.2 Non-rigid 3D Shape and 3D Point Clouds Correspondence

Non-rigid 3D shape correspondence deals with structured data such as 3D meshes and finds the point-wise correspondence between deformable 3D shapes. Finding 3D shape correspondences can be done without 3D registration. Below, the state-of-the-art non-learning [61] [62] and learning [63] [64] [65] based shape correspondence methods are reviewed.

Functional maps (FM) [66] are used for 3D shape correspondence [98] [99], which performs spectral analysis on 3D meshes to construct a functional map and solve a least-squares problem to convert the functional map to point-wise correspondence. Constraints on functional maps are used to promote continuity of the point-wise correspondence [100] and incorporate orientation information [101] into functional maps. These methods are computationally expensive and unstable with the dimensionality increase of the spectral embedding. To obtain an accurate correspondence on high dimensional details, ZoomOut [61] recovers a lower resolution to a higher resolution map

through an iterative spectral up-sampling scheme. To obtain an accurate, smooth and bijective point-wise correspondence with acceptable time and memory complexity, [62] proposes Fast Sinkhorn Filters with functional maps promoting bijective point-to-point correspondence. The well-known Sinkhorn algorithm [102] [103] is used to compute the optimal transport distance matrix between functions in a common metric space. The distance matrix is iteratively subject to a matrix scaling procedure leading to a regularized transport plan that is then converted to a correspondence map. However, Both ZoomOut [61] and Fast Sinkhorn Filters [62] are time-consuming and require manually made Laplacian descriptors for functional maps.

To learn the optimal descriptor functions from non-rigid 3D shapes, a supervised Function Maps Network (FMNet) [63] has been proposed. FMNet takes pre-computed SHOT [67] descriptor from two shapes as input and extracts features F and G from the two SHOT descriptors by using shared residual layers [104]. The extracted features F and G are projected onto the Laplacian eigenbases Φ and Ψ to produce the spectral representations $\hat{F}$ and $\hat{G}$ for functional maps. FMNet predicts a matrix C encoding the correspondence between the two shapes by minimizing the F norm between $C\hat{F}$ and $\hat{G}$. The matrix C is then converted to the spatial correspondence P by using C, Φ and Ψ. Finally, a soft error loss is proposed by minimizing the F norm between P and ground-truth geodesic distances. To avoid using the ground-truth geodesic distances, Spectral Unsupervised FMNet (SURFMNet) [64] enforces bijective properties on the map by minimizing the F norm between $C\hat{F}$ and $\hat{G}$ and minimizing the F norm between $C\hat{G}$ and $\hat{F}$. CorrNet3D [65] takes two unstructured 3D point clouds as input instead of structured

3D meshes and consists of a feature extraction module followed by a deformation-like reconstruction module, which reconstructs the two 3D point clouds in the output, respectively. The global features between the two modules are considered the correspondence matrix for point-wise correspondence. However, FMNet and SUPRFMNet take the pre-computed SHOT [67] descriptors as input, which limits the applicability of such methods on unstructured datasets such as 3D point clouds. In addition, FMNet, SUPRFMNet and CorrNet3D are all limited in the large and multiple deformations between two shapes in the form of either structured 3D meshes or unstructured 3D point clouds.

2.4 Summary

In this chapter, the detailed illustrations, advantages and limitations of traditional and state-of-the-art 3D point cloud completion, rigid 3D point cloud registration and non-rigid 3D point cloud registration and correspondence methods are discussed. Therefore, the research objectives for this PhD project are developing algorithms, computational frameworks and methods to address the limitations of these methods, improve the quality of 3D point clouds and further increase the accuracy of 3D point cloud registration in AR applications.

Chapter 3

Recovering Dense 3D Point Clouds from Single Endoscopic Image

In this chapter, a novel computational framework has been proposed to recover dense 3D point clouds from single endoscopic images using two deep-learning neural networks. One is for monocular depth learning, and the other is for 3D point cloud completion to recover the missing data from the initially generated point clouds. The experimental results indicate that the 3D reconstruction method outperforms the state-of-the-art learning-based method and non-learning-based stereo 3D reconstruction algorithms on the synthetic medical datasets. 3D point cloud completion results also show a better performance. Even if the missing rate reaches 60%, the quality of the 3D point cloud completion result is still high.

3.1 Introduction

Augmented reality (AR) information can help surgeons overcome the limited field of view and the lack of depth information during minimally invasive surgery. The higher the quality of the underlying 3D point cloud is, the more accurate the augmented information becomes [4]. During endoscopic surgery, endoscopes are used to visualize organ surfaces in the body and the data acquired is the so-called endoscopic images. Constructing 3D point data from the endoscopic image is challenging due to occlusions of instruments, the change of brightness of organ surfaces, and the surface smoothness for feature extractions [105] [5]. Processing an extensive amount of endoscopic image sequences in real-time is a high computational cost, making it difficult to generate high-quality 3D point clouds [17] [6].

Missing data or information from the initially recovered point cloud is common, and it is a shared problem in many applications that rely on high-quality 3D point clouds, for example, AR information augmentation, robotic manipulation [7] and scene understanding [8]. 3D point cloud completion refers to a process that repairs data flaws by filling holes and parts of the dataset. To the best knowledge, no prior work has been reported on monocular endoscopic 3D point cloud completion, and the vast majority of point cloud completion methods have been focused on objects, for which specialized 3D shapes (e.g. aircraft, furniture) are learned or manually designed. Large medical *in-vivo* databases of 3D point clouds of real endoscopic scenes are scarcely publicly available. The availability of such databases will significantly assist in research innovations. Seven new

medical datasets are generated and made freely available to research communities.

In this chapter, a novel deep learning-based computational framework is proposed for recovering 3D point clouds from single monocular endoscopic images. An unsupervised learning-based network mono-depth is used to generate depth information from monocular images. Given a single mono endoscopic image, the network is capable of depicting a depth map. The depth map is then used to recover a dense 3D point cloud. A generative Endo-AE network based on an auto-encoder is trained to repair defects of the dense point cloud by generating the best representation from incomplete data. The performance of the proposed framework is evaluated against state-of-the-art learning-based methods. The results are also compared with non-learning-based stereo 3D reconstruction algorithms. The proposed methods outperform both the state-of-the-art learning-based and non-learning-based methods for 3D point cloud reconstruction. The Endo-AE model for point cloud completion can generate high-quality, dense 3D endoscopic point clouds from incomplete point clouds with holes. The proposed framework is able to recover complete 3D point clouds with the missing rate of information up to 60%. Five large medical *in-vivo* databases of 3D point clouds of real endoscopic scenes have been generated and two synthetic 3D medical datasets are created. These datasets have been made publicly available for researchers free of charge.

3.2 Previous Work

The proposed framework is closely related to two categories of prior works: 1) Monocular Depth Estimation and 2) 3D Point Cloud Completion.

3D Monocular Depth Estimation: Depth estimation is an integral part of 3D point cloud reconstruction. The state-of-the-art camera tracking and reconstruction systems (structure for motion systems) that estimate detailed depth maps with textures at selected keyframes can produce dense surface maps with millions of points [17]. Some of these systems rely on powerful commodity GPU processors for real-time performance and stereo visions. On the other hand, monocular Simultaneous Localization and Mapping (SLAM) systems that operate with limited processing resources only generate and track sparse feature-based models [106] [6].

Recent advances in monocular depth estimation have shown results of predicting the depth from a single image [18] [19], which can be used for understanding the shape of a scene from a single image, a fundamental problem in machine vision. These methods pose the monocular depth estimation as a learning problem by training models offline [107] [18] [108]. Among these methods, supervised learning [18] [19] needs to train models on large collections of ground truth. Novel unsupervised learning methods explore easier-to-obtain binocular stereo footage without the need for explicit depth data during the training [20] [21]. In this work, since the ground truth of the depth information is unavailable for monocular endoscope scenes, this chapter builds on previous unsupervised learning framework [21] to develop a monocular depth estimation for 3D point

cloud reconstruction from single endoscopic images. The novelty of this approach is a fully differential patch-based cost function and the Zero-Mean Normalized Cross-Correlation is proposed that takes multi-scale patches as a matching strategy. This approach significantly increases the accuracy and robustness of depth learning. However, this method has only been tested with non-medical public datasets. The method is further extended to extract the dense 3D endoscopic point cloud based on the estimated depth and introduce a colour extraction method onto a reconstructed 3D point cloud from a single endoscopic image.

3D Point Cloud Completion: Real endoscopic 3D point clouds present incomplete data (e.g., missing data, holes), due to limited field of view and occlusions during minimally invasive surgery where surgical instruments interact with the organs, as well as the illumination variations caused by the endoscopic light, tissue haemorrhaging and or surgical smoke [105]. Hence, the task of filling missing holes and information for reconstructed 3D point clouds are cast as the task of 3D point cloud completion.

Traditional geometry-based approaches use geometric clues to complete 3D meshes from a partial input [30], while data-driven-based methods rely on the assumption that the database must include a very similar shape [33]. Recently emerged deep learning-based methods [37] [39] [38] [40] have achieved superior performance operating on 3D point clouds through generative models based on Auto Encoder (AE) [83] [37] [109] [40] and Generative Adversarial Net (GAN) [84].

An optimization method has been proposed to select the best seed for the latent GAN to improve the performance for point

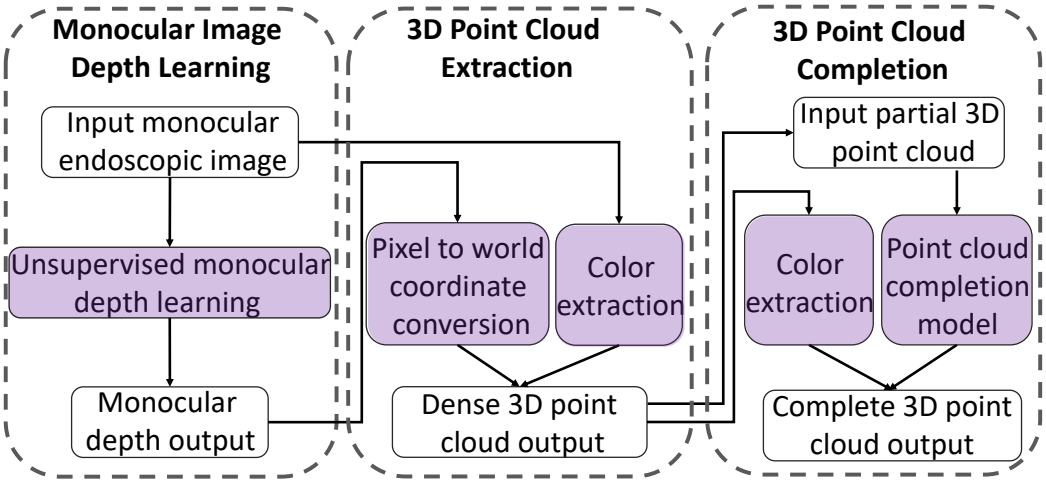


Fig. 3.1 The proposed computational framework consists of three modules: Monocular Image Depth Learning, 3D Point Cloud Extraction, and 3D Point Cloud Completion.

cloud completion [109]. Structural point cloud decoder [40] can only generate sparse 3D point clouds since the decoder consists of most 1D convolution layers. Each 1D convolution layer needs to generate a recovered point cloud, which limits the number of points to be processed. Achlioptas et al. [37] proposed an auto-encoder architecture for 3D point cloud processing. However, this method focuses on 3D point cloud representation learning and has only been tested with non-medical public datasets. this thesis applies the auto-encoder architecture to recover dense endoscopic 3D point clouds, and the proposed approach is the first attempt of its kind applied to endoscopic 3D point cloud completion.

3.3 Overview of the Framework

Figure 3.1 illustrates the entire computational framework with three modules: Monocular Image Depth Learning, 3D Point Extraction and 3D Point Cloud Completion.

Monocular Image Depth Learning Module: In this module, an unsupervised learning-based network mono-depth is developed. Public Laparoscopic/Endoscopic stereo video datasets are used for network training. The unsupervised depth learning method treats the monocular depth estimation as error minimization in image synthesis. During the training, the depth is estimated from the left image of stereo pairs. The depth is then converted into a disparity map to synthesize the right image of stereo pairs. The loss function is used to minimize the error between the reconstructed right image and the original right image. Once trained, the depth information is generated from monocular endoscopic images in the depth learning module.

3D Point Cloud Extraction Module: In the 3D point cloud extraction module, the depth estimated from the depth learning module is converted into a dense 3D point cloud. A coordinate conversion method is used to transform the pixel coordinates into 3D world coordinates. To obtain the colour information, colour attributes of the corresponding input monocular endoscopic image are extracted and applied to the 3D point cloud.

The effectiveness of the proposed 3D point cloud reconstruction framework is evaluated by comparing mono-depth with a state-of-the-art learning-based method [20], as well as with two non-learning-

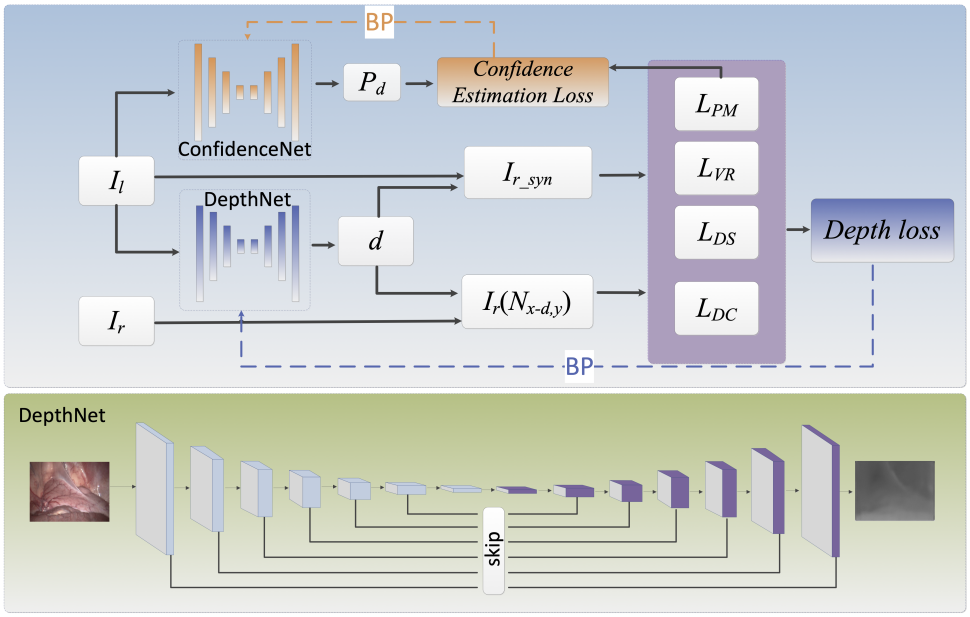


Fig. 3.2 Mono-depth network for estimating depth from monocular images; It consumes a single image as input and consists of 14 layers of an encoder and 14 layers of a decoder. The input is encoded by 7 Conv layers with stride 2, and each layer is followed by a Conv layer with stride 1. The decoder consists of 7 deConv layers with stride 2, and each layer is followed by a Conv layer with stride 1.

based stereo image reconstruction methods [16]. The detailed evaluation is described in section 3.7.1.1.

3D Point Cloud Completion Module: In the 3D point cloud completion module, a generative Endo-AE network based on an auto-encoder is performed for the task of 3D endoscopic point cloud completion. The 3D point clouds generated in the 3D point cloud extraction module are split into the training data and the testing data. The auto-encoder, as an unsupervised network, uses the training data itself as the ground truth. During the training, the input of the network is complete 3D point clouds without any missing data, as shown in Figure 3.5. The network learns global features of training datasets through an encoder and converts global features

into an original 3D point cloud through a decoder. During the testing mode, by randomly deleting consecutive points in the testing data, Endo-AE can generate a complete 3D point cloud from the partial 3D point cloud input, as shown in Figure 3.6. The colour attributes are also extracted from the corresponding 3D point cloud in the testing data.

3.4 Methods

3.4.1 Unsupervised Monocular Depth Learning

Building on the unsupervised mono-depth network [21], the per-pixel depth is estimated from single image input. The mono-depth network is based on a VGG-like fully convolutional neural network architecture [110], as shown in Figure 3.2.

During the training, the single left images I_l of stereo pairs are used as the input data for the DepthNet model to synthesize per-pixel depth D. The depth D is transformed into a disparity map $d = \frac{b \times f}{D}$, where b and f are the camera baseline and focal distance, respectively. The disparity maps d are then used to reconstruct the right views of the stereo pairs I_{r_syn} and the sampling of patches from right views $I_r(N_{x-d,y})$. Finally, the fully differential loss function L_{total} is applied to train the mono-depth network. L_{total}, as illustrated in Equation 3.5, consists of a Patch Matching Loss L_{PM}, a View Reconstruction Loss L_{VR}, a Disparity Smoothness Loss L_{DS}, and a Disparity Consistency Loss L_{DC}. In addition, another parallel ConfidenceNet, as shown in Figure 3.2, is trained by using the L_{PM} to evaluate the performance of the monocular depth estimation. The

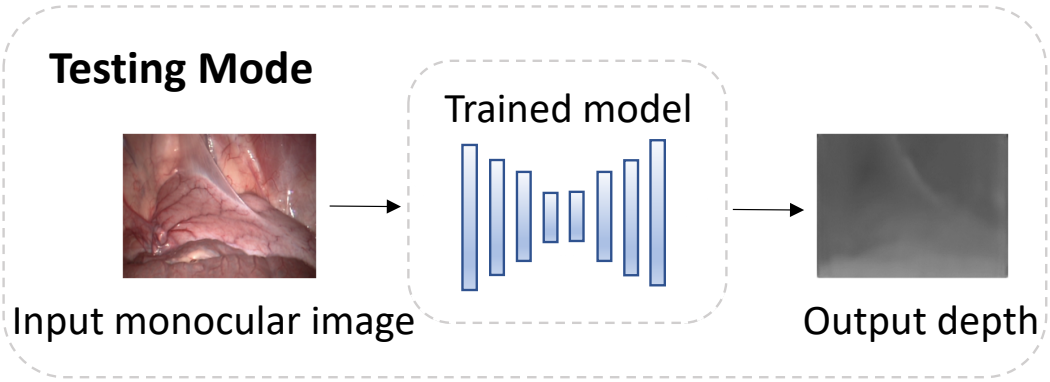


Fig. 3.3 Mono-depth network testing mode: The trained model only consumes monocular images as input and outputs corresponding depth images.

ConfidenceNet produces a confidence map that gives a real-time assessment of the reliability of the predicted depth.

During the testing, the trained mono-depth model does not need the original right image of stereo pairs to calculate the loss anymore. Thus, the trained model can generate per-pixel depth only from the monocular image as shown in Figure 3.3.

Loss Function: Following the loss functions in unsupervised depth learning networks [21] [20], the mono-depth is trained using the combination of L_{PM}, L_{VR}, L_{DS} and L_{DC}.

L_{PM} is proposed to maximize the similarities between patches in the left input image $I_l(N_{x,y})$ and shifted patches $I_r(N_{x-d,y})$ in the reconstructed right image by using the Zero-Mean Normalized Cross-Correlation, as defined in Equation 3.1.

$$ZNCC = \frac{\sum_{i,j \in N_{x,y}} (I_l(i,j) - \overline{I}_l(N_{x,y})) \cdot (I_r(i-d,j) - \overline{I}_r(N_{x-d,y}))}{\sqrt{\sum_{i,j \in N_{x,y}} (I_l(i,j) - \overline{I}_l(N_{x,y}))^2 \cdot \sum_{i,j \in N_{x,y}} (I_r(i-d,j) - \overline{I}_r(N_{x-d,y}))^2}} \quad (3.1)$$

where $\overline{I}(N_{x,y}) = \frac{1}{n} \sum_{x,y \in N_{x,y}}$, $I(x,y)$ is the mean intensity of the patch $N_{x,y}$ centered at the coordinate (x,y).

L_{VR}, as defined in Equation 3.2, minimizes the differences between the original right input image $I_r(x, y)$ and its reconstruction $\widehat{I}_r(x, y)$ using the $L1$ norm.

$$L_{VR} = \sum_{xy} \left| I_r(x, y) - \widehat{I}_r(x, y) \right| \tag{3.2}$$

L_{DS}, as defined in Equation 3.3, regularizes the mono-depth network to produce more smooth depth by calculating the sum of the $L1$ norm of disparity gradients along x and y directions.

$$L_{DS} = \frac{1}{XY} \sum_{x,y} \left| \frac{\partial d(x, y)}{\partial x} \right| + \left| \frac{\partial d(x, y)}{\partial y} \right| \tag{3.3}$$

L_{DC}, as defined in Equation 3.4, attempts to make the left-view disparity map $d_l(x, y)$ to be equal to the reconstructed right-view disparity map $d_r(x - d_l(x, y), y)$ using the $L1$ norm.

$$L_{DC} = \frac{1}{XY} \sum_{x,y} |d_l(x, y) - d_r(x - d_l(x, y), y)| \tag{3.4}$$

The final loss function L_{total} is defined by Equation 3.5.

$$L_{total} = \omega_p L_{PM} + \omega_v L_{VR} + \omega_d L_{DS} + \omega_c L_{DC} \tag{3.5}$$

where ω is the corresponding weight to balance the effect of gradients of the backpropagation.

Since the mono-depth network generates the depth information from single images and is trained without using ground-truth depth in L_{PM}, L_{VR}, L_{DS} and L_{DC} loss functions, this network is defined as an unsupervised network for depth learning.

(a) Left Image (b) Depth Image (c) 3D Point Cloud

Fig. 3.4 3D Point Cloud Reconstruction: (a) Left image; (b) Generated depth image from (a); (c) Reconstructed 3D Point Cloud with colour attributes from (a) and (b).

3.4.2 3D Point Cloud Extraction

3D point cloud extraction is the second module of the proposed framework shown in Figure 3.1. A 3D point cloud is extracted from the generated depth D, as described in section 3.4.1. A coordinate conversion method from the pixel coordinates to the world coordinates is applied to 3D point cloud extraction. Based on the generated depth D, 3D point clouds can be extracted using Equation 3.6.

$$\begin{aligned} x_w &= (u - u_0) * D/f_x \\ y_w &= (v - v_0) * D/f_y \\ z_w &= D \end{aligned} \tag{3.6}$$

where (x_w, y_w, z_w) is the coordinates of a point in the world coordinate system and (u, v) is each pixel in the depth D. (u_0, v_0) are the centre coordinates of the depth D in pixel coordinate system. f_x and f_y are the focal lengths of the left and right cameras.

While reconstructing a 3D point cloud from the generated depth D, the colour attributes of each pixel are extracted from the corresponding left image and assigned to each point in the 3D point cloud, as shown in Figure 3.4.

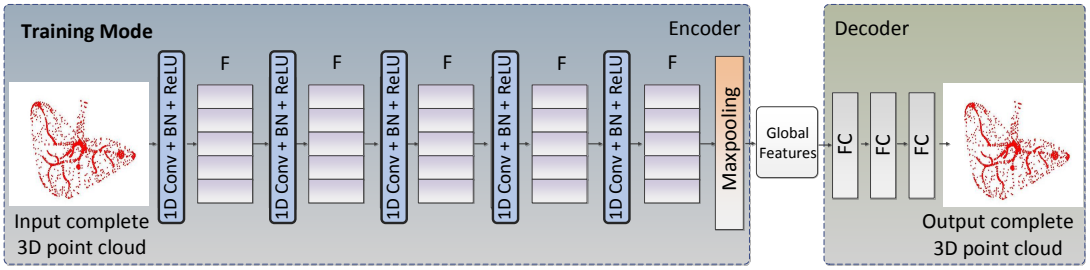


Fig. 3.5 Endo-AE completion network training mode: The input in the training mode is a complete 3D point cloud without missing holes. The encoder is from PointNet. F represents features extracted by each feature extraction layer. The decoder consists of 3 fully connected layers (FC). The first two FC layers consist of the activation function ReLU.

3.4.3 3D Point Cloud Completion

For the point cloud completion task, an Endo-AE network is trained based on Auto-encoder (AE) that includes an encoder and a decoder to generate complete 3D point clouds from partial 3D point clouds with missing data. Since the Endo-AE is an unsupervised network, the ground truth is the input training 3D point cloud itself. The encoder of the Endo-AE network is based on PointNet [1], a state-of-the-art deep learning method on 3D point cloud classification. PointNet combines point-wise multi-layer perceptions with a symmetric aggregation function that is invariant to permutation, which is essential for effective feature learning on 3D point clouds. The main differences between PointNet [1] and the Endo-AE network are the loss function, the ground truth and the output of the two networks. PointNet focuses on 3D point cloud classification, and the loss function of the PointNet classification network is softmax, which can be considered a multi-classes classifier. Every 3D point cloud has a label for classification, and each label is the ground truth for PointNet. Thus, the PointNet classification network outputs

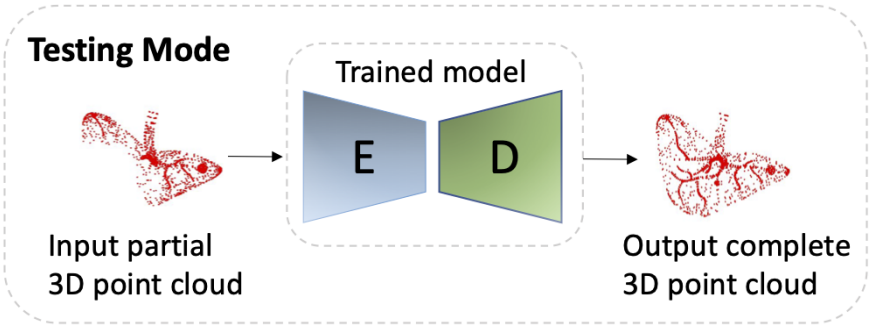


Fig. 3.6 Endo-AE completion network testing mode: The trained model consumes a 3D point cloud with the missing data and outputs a complete 3D point cloud.

the label of an input 3D point cloud. Whereas the loss function of the Endo-AE network is the chamfer distance, as illustrated in Equation 3.7, which minimizes the distance between the input and the output of 3D point clouds. The ground truth of the Endo-AE network is the input training 3D point cloud itself, and the output is the complete 3D point cloud.

During the training, the input of the Endo-AE network is the complete 3D point cloud without missing data, and the output 3D point cloud is the reconstruction of the input. The input and output 3D point clouds in training mode are shown in Figure 3.5. A 3D point cloud with N points is represented as a $N \times 3$ matrix, and each row of the matrix is the 3D coordinates of a point defined as $P_i = (x, y, z)$. The encoder compresses an input 3D point cloud of N points into a k dimensional feature vector $v \epsilon \mathbb{R}^k$. Specifically, the combination of 1D convolution (Conv), batch norm (BN) and ReLU layers is used to transform each point P_i into a point feature vector F. A point-wise max pooling is placed after all feature extraction modules, ensuring the global features are invariant to any permutations of a 3D point cloud and producing a k-dimensional feature vector. The decoder

aims to generate the reconstruction of the input 3D point cloud based on the learned k-dimensional feature vector by using three fully connected layers. The chamfer distance [91] is used as the loss function to maximize the similarity between the output 3D point cloud and the ground truth. Thus, the learned global features can represent the 3D point cloud for the point cloud completion task.

During the testing, the input of the trained completion model is partial 3D point clouds, and the trained model can output complete 3D point clouds based on learned global features extracted from the encoder. The input and output 3D point clouds in testing mode are shown in Figure 3.6.

3.5 Implementation Details

The experiments are conducted in two stages. A mono-depth network is trained to predict depth for 3D point cloud reconstruction. The 3D point cloud completion is then achieved based on reconstructed point clouds with a trained Endo-AE network. The unsupervised mono-depth network and Endo-AE network are trained on an Nvidia Titan X GPU with 12G memory and a CPU with 32G memory. The implementation details for the two networks are explained as follows:

Hyper Parameters: The mono-depth network applies the same hyper-parameters following the state-of-the-art depth learning networks [21] [20]. In terms of training the mono-depth network, all input images are resized to 512×256 with a batch size of four. Adam optimizer with an initial learning rate of 0.0001 and 50 epochs are used for the training process. The weights defined in the total loss are $\omega_p = 0.5$, $\omega_v = 1$, $\omega_d = 0.1$ and $\omega_c = 1$, respectively. In addition,

6 skip connections are implemented, preserving intermediate information during training to ensure the high quality of per-pixel depth estimation. The first four kernel sizes of the encoder are 7, 7, 5, and 5, followed by ten kernel sizes of 3. The kernel size of the decoder in each layer is the reverse order in the encoder.

The Endo-AE network applies the same hyper-parameters following the state-of-the-art 3D point cloud processing networks [111] [37]. The encoder of the Endo-AE network consists of five layers of shared 1D convolution with 64, 128, 128, 256 and 128 filters, respectively. The decoder consists of three fully connected layers with 256, 256 and 4096×3 filters, respectively. Adam optimizer is also used with an initial learning rate of 0.0005, a batch size of 50 and 500 epochs. The Endo-AE is trained with the input size of $M_1 \times 3$ and the ground truth size of $M_2 \times 3$, generating the output point cloud with the size of $M_3 \times 3$, where M_1, M_2 and M_3 can be any number. In this experiments, $M_1 = M_2 = M_3 = 4096$.

Data Augmentation: To increase the robustness of the monodepth network and prevent over-fitting, images are randomly flipped and the brightness and colour of images are changed. During the Endo-AE training, 3D point clouds are augmented by applying a random rotation matrix.

3.6 Evaluation Metrics

The 3D point cloud reconstruction and completion methods are evaluated with minimum matching distance (MMD). Minimum matching distance (MMD) calculates the average distance in the matching between two 3D point clouds. MMD_CD is based on the chamfer

distance [91] (Equation 3.7), and MMD_EMD is about the Earth Mover's distance (EMD) [112] (Equation 3.8).

$$\begin{aligned} CD(S_{out}, S_{gt}) = & \frac{1}{S_{out}} \sum_{p \epsilon S_{out}} \underset{q \epsilon S_{gt}}{min} ||p-q||_2 \\ & + \frac{1}{S_{gt}} \sum_{q \epsilon S_{gt}} \underset{p \epsilon S_{out}}{min} ||q-p||_2 \end{aligned} \tag{3.7}$$

The chamfer distance calculates the average nearest point distance between S_{out} and S_{gt} by finding the closest neighbour with $O(nlogn)$ complexity. In addition, S_{out} and S_{gt} can be 3D point clouds with different sizes.

$$EMD(S_{pred}, S_{gt}) = \underset{\phi}{min} \sum_{p \epsilon S_{pred}} \|p - \phi(p)\|_2 \tag{3.8}$$

where $\phi : S_{pred} \rightarrow S_{gt}$ is bijection. The EMD distance minimizes the distance between S_{pred} and S_{gt} with $O(n^2)$ complexity. Note that EMD requires the same sizes of S_{pred} and S_{gt}.

A major difference between MMD_CD and MMD_EMD is that calculating MMD_EMD is too expensive with $O(n^2)$ complexity and takes more time than calculating MMD_CD with $O(nlogn)$ complexity. Another major difference between them is that MMD_CD can calculate the average distance between two point clouds with different sizes, whereas calculating MMD_EMD requires the two 3D point clouds to have the same sizes.

3.7 Results and Discussions

3.7.1 3D Point Cloud Reconstruction

In this section, the 3D point cloud reconstruction method is first compared with a state-of-the-art learning-based method Godar et al. [20] and two non-learning-based stereo image reconstruction methods [16]. Secondly, 3D endoscopic point cloud datasets are generated based on the proposed 3D point cloud reconstruction method.

3.7.1.1 Comparison experiments

Evaluation Datasets: There are some endoscopic datasets generated by previous researcher projects, such as EndoVis 2019 Sub-challenge dataset[1], Laparoscopic Image to Image Translation Dataset [113] and EndoAbS dataset [114]. EndoVis 2019 Sub-challenge dataset may not be publicly downloadable. The laparoscopic Image to Image Translation Dataset includes simulated monocular images and the corresponding depth. However, this dataset does not provide camera parameters and ground truth 3D point clouds or ground-truth stereo correspondences that are required to train the mono-depth learning network.

The EndoAbs dataset [114] is captured from soft phantoms of abdominal organs with the aim of representing the real surgical scenario as closely as possible. It consists of 120 images of the kidney, liver and spleen, captured under 3 different endoscopic lighting conditions by varying 3 different low light intensities, which poses the challenge for evaluating the 3D reconstruction methods. 20

[1] https://endovissub2019-scared.grand-challenge.org/

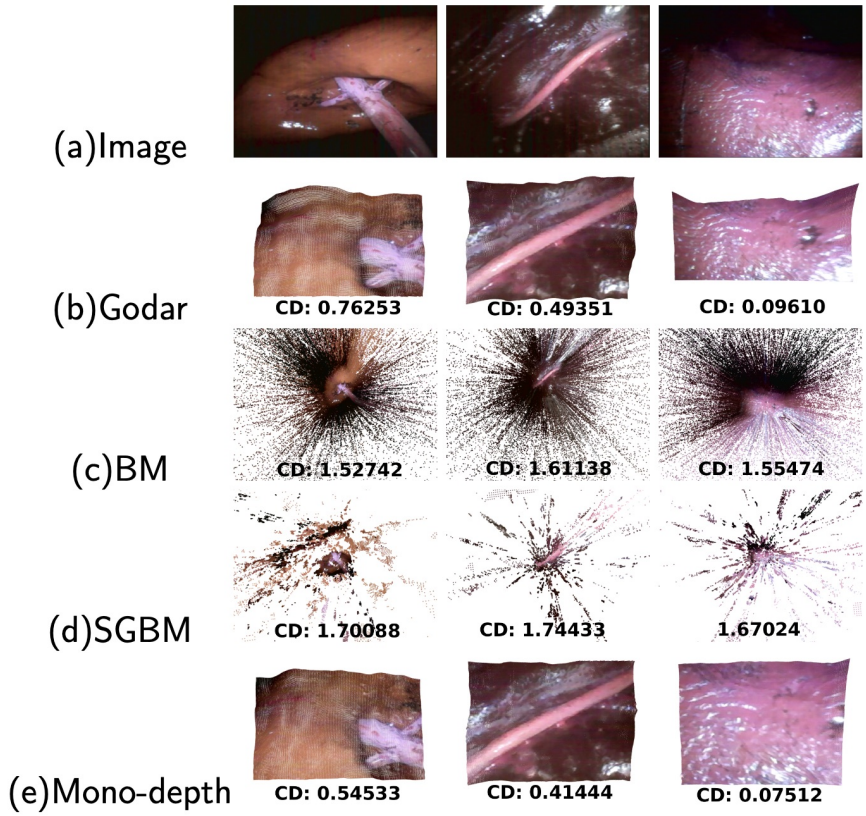


Fig. 3.7 Quantitative comparison on EndoAbS: (a) Images of kidney, liver and spleen in EndoAbS dataset; (b) Results of learning-based Godar method; (c) Results of traditional block matching algorithm from OpenCV; (d) Results of traditional semi-global block matching algorithm; (e) Results of mono-depth.

ground truth 3D point clouds are also captured by the laser scanner. To minimize the error during the evaluation, the ground truth 3D point clouds are manually transformed (translated and rotated) in the EndoAbS dataset to match the views of the images.

A new synthetic medical dataset is also created for the network training by capturing stereo images and the corresponding ground truth 3D point clouds under the same view using a 3D computer modelling software[2]. Specifically, the synthetic dataset contains 200 stereo images and their corresponding depth images from the 3D

[2]https://www.autodesk.com/products/maya/

liver and heart models. All images and corresponding depth are captured by randomly rotating and translating the 3D liver and heart models in the scene. The light in this dataset shines evenly in all directions from the location of the light and every object in the scene is illuminated by the light. Finally, the ground truth 3D point clouds are extracted from 200 left frames and 200 corresponding depth images using Equation 3.6.

Comparison with Learning-Based Method: The Godar [20] and the mono-depth networks are trained on the publicly available Laparoscopic/Endoscopic video datasets [26] [27]. The performance of these two methods on the EndoAbS dataset is first compared. The reconstructed 3D point clouds are shown in (b) and (e) in Figure 3.7. The chamfer distance (CD) is calculated between reconstructed 3D point clouds and ground truth 3D point clouds. Although the EndoAbS dataset is captured under low light conditions, the CD results on the kidney, liver and spleen for the mono-depth are 0.54533mm, 0.41444mm and 0.07512mm, showing better performance than Godar's with CD results of 0.76253mm, 0.49351mm and 0.09610mm.

The proposed 3D reconstruction method is also compared with Godar's on the synthetic medical dataset, as shown in (b) and (e) in Figure 3.8. The CD results in Figure 3.8 show that the proposed 3D reconstruction method also outperforms Godar's method on the synthetic medical dataset. In addition, the average CD is also calculated on the created synthetic medical dataset for the proposed 3D reconstruction method (0.01514mm), which outperforms Godar's method (0.01902mm).

Comparison with Stereo Image Reconstruction Methods: To assess the effectiveness of learning-based methods with

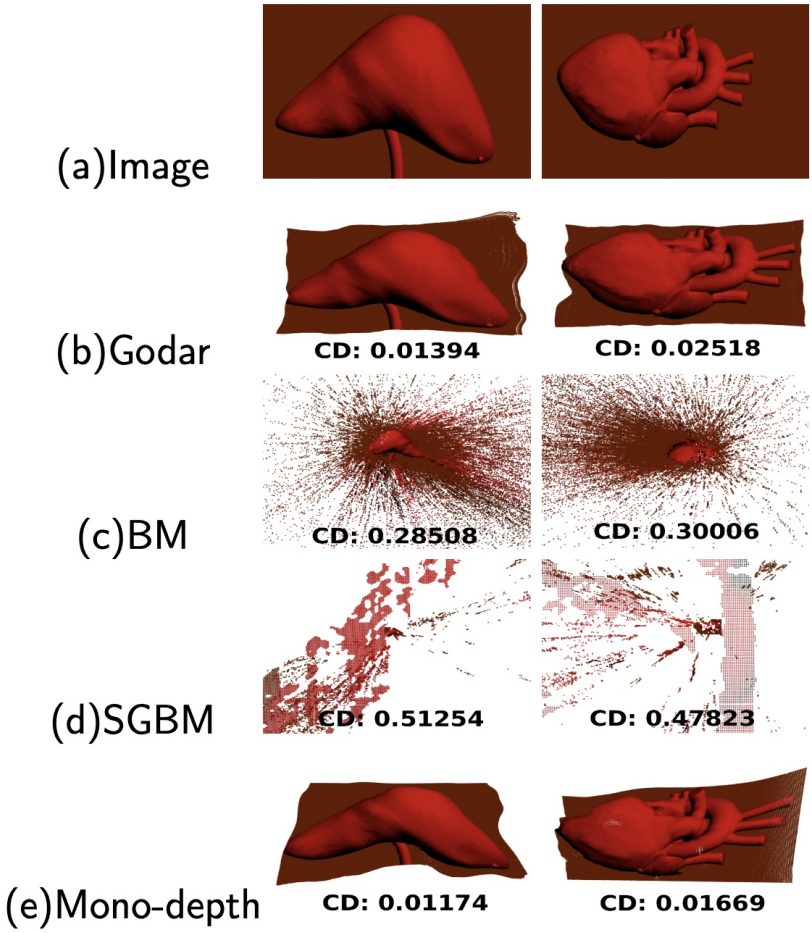


Fig. 3.8 Quantitative comparison on the synthetic medical dataset: (a) Images of liver and heart; (b) Results of learning-based Godar method; (c) Results of traditional block matching algorithm from OpenCV; (d) Results of traditional semi-global block matching algorithm; (e) Results of mono-depth.

non-learning-based stereo-image reconstruction methods, the proposed method is compared with non-learning-based block matching method[3] (BM) and semi-global block matching (SGBM) [16]. BM and SGBM directly use low-level image feature to search for matched pixels in the left and right images of stereo pairs. As a result, the quality of the generated 3D point cloud is usually poor, as shown in (c) and (d) in Figures 3.7 and 3.8. The CD results show that the proposed mono-depth outperforms BM and SGBM on both the EndoAbS dataset and the synthetic medical dataset. In addition, the average CD result on the synthetic medical dataset for mono-depth is 0.01514mm, which shows better performance than BM and SGBM with 0.29784mm and 0.49407mm.

3.7.1.2 Generated Endoscopic Datasets

3D endoscopic point cloud datasets are generated based on depth information generated from publicly available Laparoscopic/Endoscopic video datasets [26] [27] using the proposed 3D point cloud reconstruction method.

Five stereo endoscopic videos from the datasets [26] [27] are chosen to generate depth information, including Abdomen Wall, Uterine Horn, Liver, Nephrectomy scene 1 and Nephrectomy scene 2, respectively. The stereo videos are divided into 35,000 left frames and 35,000 right frames. Around 10,000 Nephrectomy left frames and 10,000 Nephrectomy right frames are randomly selected as the input to train the mono-depth network. Once the model has been trained, the mono-depth network can generate the per-pixel depth from the monocular image. Approximately 35,000 depth images

[3]https://opencv.org

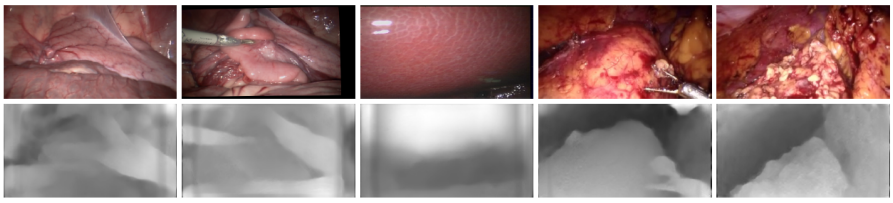


Fig. 3.9 Estimated depth from five different endoscopic image datasets, which are Abdomen Wall, Uterine Horn, Liver, Nephrectomy scene 1 and Nephrectomy scene 2, from left to right.

from 35,000 left frames are generated, as shown in Figure 3.9. The five left frames in Figure 3.9 are randomly selected from Abdomen Wall, Uterine Horn, Liver, Nephrectomy scene 1 and Nephrectomy scene 2, respectively.

Based on the generated depth images, approximately 35,000 *in-vivo* 3D point clouds are generated by using Equation 3.6. The generated datasets consist of five endoscopic point cloud categories, including Abdomen Wall, Uterine Horn, Liver, Nephrectomy scene 1 and Nephrectomy scene 2. The datasets are made publicly available for researchers, which can be used for learning-based methods as training datasets and evaluating 3D reconstruction methods. Each dense 3D point cloud contains approximately 100,000 points on average. Each category is divided into six parts and the first frame of each part is displayed, as shown in Figure 3.10. The points are extracted from every pixel of the input image, thus, points in a reconstructed 3D point cloud are evenly distributed. However, the points in the dark or black area are considered as outliers and have been removed when extracting 3D point clouds. Therefore, the points are sparse in the dark or black area, as shown in the 7th and 8th rows in Figure 3.10. The margins of endoscopic images, i.e., the

black margins in I_2, are due to the movement of the instrument that is not horizontally and vertically positioned. Thus, the margins are removed when extracting 3D point clouds. The datasets contain five classes of *in-vivo* datasets, including approximately 37,000 3D point clouds in total.

3.7.2 3D Point Cloud Completion

3.7.2.1 Evaluation of Completion Performance

For the 3D point cloud completion task, five class-specific Endo-AE networks are trained separately with five classes of endoscopic point cloud datasets generated in Section 3.7.1.2. A two-classes Endo-AE network is also trained with two synthetic 3D models, as mentioned in Section 3.7.1.2.

For each class, 90% of 3D point clouds are randomly selected as the training data and the remaining 10% as the testing data. To evaluate the trained Endo-AE model on partial 3D point clouds, the remaining 10% of testing data is used to create partial 3D point clouds with different missing rates. First, a point is randomly selected from each testing 3D point cloud with the size of $N \times 3$, where N is the total number of points in a 3D point cloud. Second, the nearest $N * delete\_rate$ points around that selected point are deleted to create partial 3D point clouds with different missing rates, where $delete\_rate$ is the rate of deletion, i.e., 0.2, 0.4, 0.7, etc. Third, each partial 3D point cloud is randomly sub-sampled to 4096 points. Finally, for each class of testing datasets, 7 groups of partial 3D point clouds testing data with various missing rates of [20%, 30%, 40%, 50%, 60%, 70%, 80%] are generated. The examples of partial

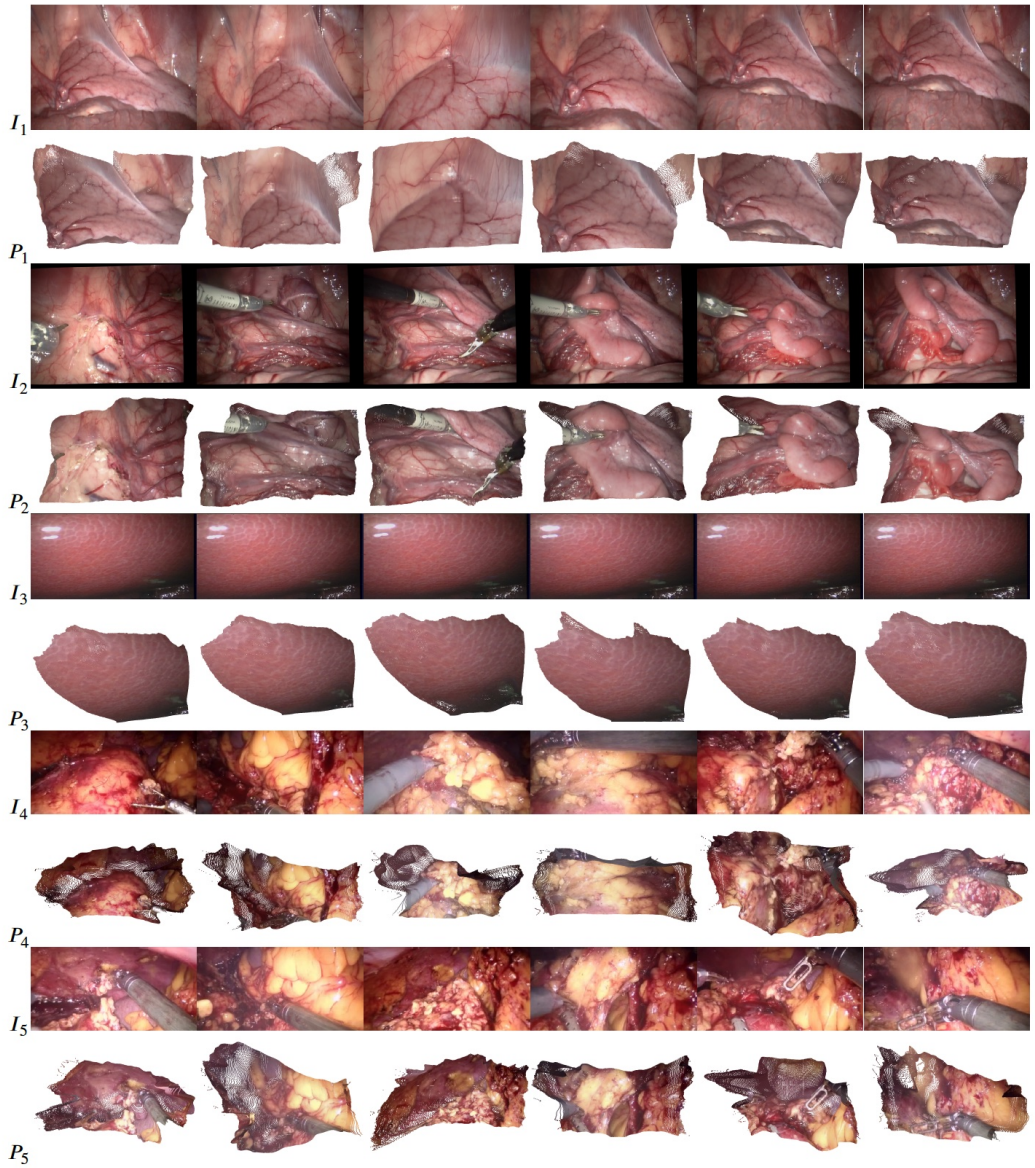


Fig. 3.10 Results of 3D point cloud reconstructions of public Laparoscopic/Endoscopic video: The 3D point cloud is extracted from every frame of the video. I_i represents Images, and P_i is the corresponding extracted 3D point clouds related to I_i. Note that I_4 and I_5 are from the same video stream, but the scene changes from the middle of the video. Thus, it has been divided into two separate datasets.

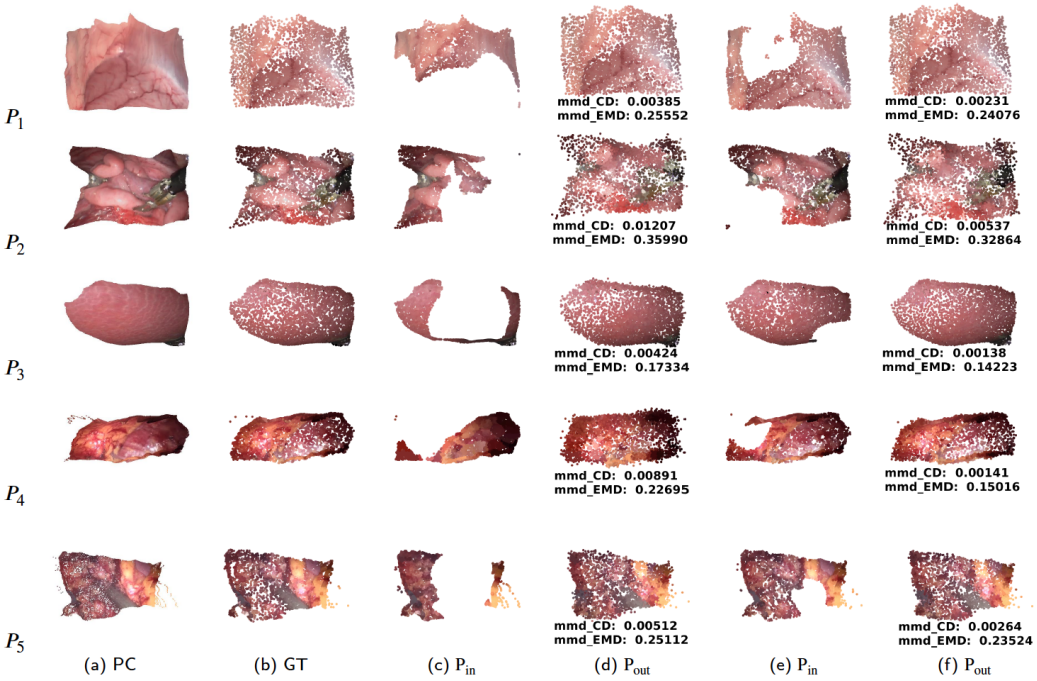


Fig. 3.11 3D point cloud completion of endoscopic datasets: (a) Original 3D point clouds of five endoscopic datasets (around 100,000 points per point cloud); (b) Ground-truth point clouds (4096 points per point cloud) generated from (a); (c) Point cloud inputs with 60% of missing data generated from (a); (d) Completion results; (e) Point cloud inputs with 20% of missing data generated from (a); (f) Completion results.

3D point clouds with 60% and 20% missing data are shown in (c) and (e) in Figure 3.11.

The visualizations of 3D point cloud completion results with 60% and 20% of missing data for five class-specific Endo-AE networks are shown in Figure 3.11. All completion results of MMD_CD and MMD_EMD in (f) are smaller than that in (d), indicating that the less missing input data, the more accurate the recovered 3D point clouds. P_1, P_2, P_3, P_4 and P_5 show the effectiveness of the Endo-AE with the 20% and 60% of missing rates.

The completion result is unstable in the dataset of P_2 when the missing rate reaches 60%. The unstable recovery is due to the large

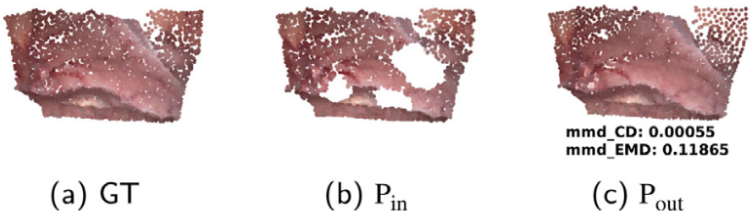


Fig. 3.12 Result of Endo-AE on partial point cloud with multiple missing areas: (a) A ground truth of Abdomen Wall; (b) A point cloud input with multiple missing areas and 20% of missing rate; (c) Completion result of Endo-AE.

deformation of the organ in the video, which is caused by surgical instruments. Thus, it is difficult for a neural network to extract common global features for all different deformation degrees. In addition, an example of a 3D completion result on a partial point cloud with multiple missing areas and a 20% of the missing rate is visualized, as shown in Figure 3.12, which indicates that the Endo-AE can process various partial 3D point clouds with multiple missing areas. Figures 3.11 and 3.12 show that the Endo-AE model can deal with the partial input 3D point cloud with various missing areas and rates.

The K-nearest neighbour (KNN) is calculated to find the corresponding points between the ground-truth 3D point cloud and our generated 3D point cloud, where K is 1. After finding the nearest neighbour between them, the colour of each point is extracted from the ground truth and duplicated to the corresponding point in the generated 3D point cloud, as shown in Figures 3.11 and 3.12.

The two-classes Endo-AE network completion results with 50% of missing regions on simulated 3D heart and liver models are shown in Figure 3.13, which presents superior completion results.

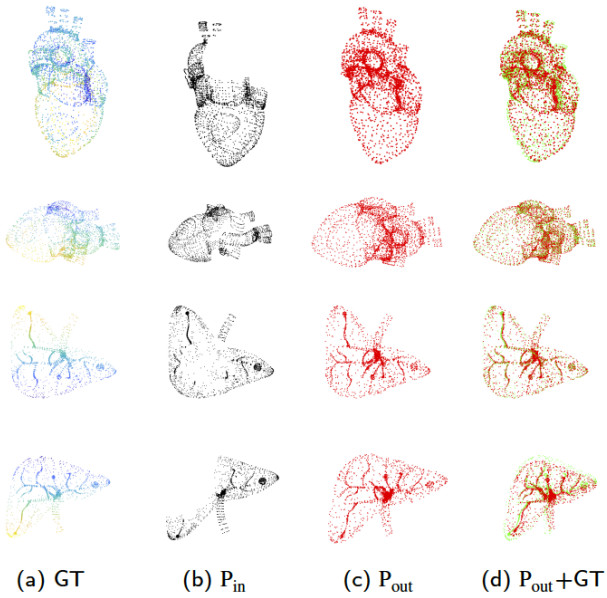


Fig. 3.13 3D point cloud completion of synthetic datasets: (a) Ground-truth; (b) Inputs with 50% of missing data; (c) Completion results; (d) Overlay of ground-truth point clouds and completion results.

Figure 3.14 shows the average values of each evaluation metric calculated between the generated point cloud and the ground truth on 7 groups of partial 3D point clouds testing data. MMD_CD and MMD_EMD show that the error of missing data repairing increases gradually as more regions are occluded. The average MMD_CD on seven datasets is 0.00236mm when the missing rate of input data is 20%. Even if the missing rate reaches 60%, the quality of the Endo-AE completion result is still good with the average MMD_CD 0.00804mm. The results of MMD_EMD on Nephrectomy scene 1 (D4) show better performance when the missing rate is 40% than 20% and 30%. The reason is that some partial inputs in testing sets are similar to each other due to the missing data of incomplete point

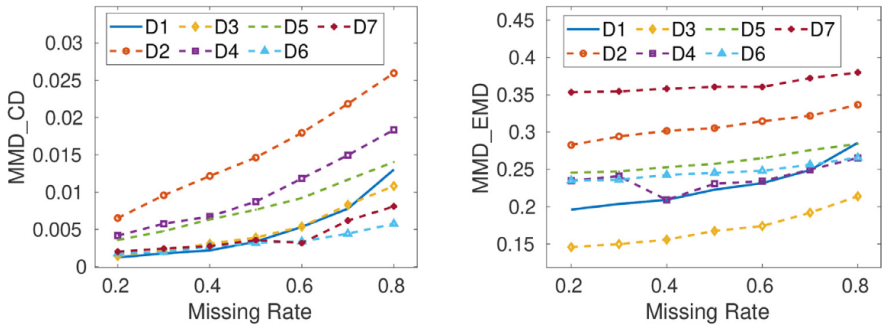


Fig. 3.14 The quantitative completion results with different missing rates: D1 to D7 represent Abdomen Wall, Uterine Horn, Liver, Nephrectomy1, Nephrectomy2, Simulated heart and Simulated liver, respectively. The missing rates for evaluation are 20%, 30%, 40%, 50%, 60%, 70% and 80%.

clouds randomly removing consecutive points from original point clouds.

3.7.2.2 Comparison with GANs

The generative auto-encoder-based Endo-AE is compared with another generative network, generative adversarial network (GAN), by training a raw GAN and a latent-space GAN (l-GAN) [37], respectively.

Raw GAN operates directly on 3D point clouds, and the generator of the raw GAN consists of five fully connected layers with ReLU, producing a 4096×3 point cloud. The architecture of the discriminator is identical to the encoder of the Endo-AE with leaky ReLUs and without any batch normalization, and a sigmoid activation function is placed after the discriminator.

A raw GAN is trained on a single class of Liver and one of the 3D completion results is shown in Figure 3.15. The MMD_CD and MMD_EMD of the proposed Endo-AE are 0.00138 mm and 0.14223 mm, showing better performance than raw GAN with MMD_CD

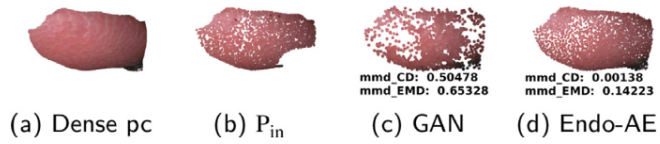


Fig. 3.15 Results of GAN and Endo-AE with single-class training: (a) An original 3D dense point cloud of Liver; (b) A point cloud input with 20% of missing data; (c) Output of GAN; (d) Completion result of Endo-AE.

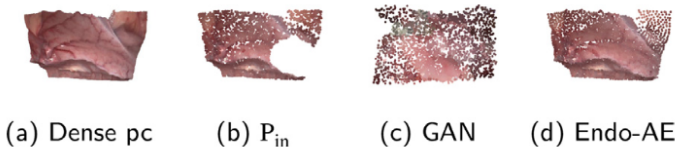


Fig. 3.16 Results of GAN and Endo-AE with multi-classes training: (a) An original 3D dense point cloud of Abdomen Wall; (b) A point cloud input with 20% of missing data; (c) Output of GAN; (d) Completion result of Endo-AE.

0.50478 mm and MMD_EMD 0.65328 mm. In addition, the Endo-AE also outperforms raw GAN on single-class training when the missing rate of testing data reaches 30%, 40%, 50%, 60%, 70% and 80% based on the average values of MMD_CD and MMD_EMD.

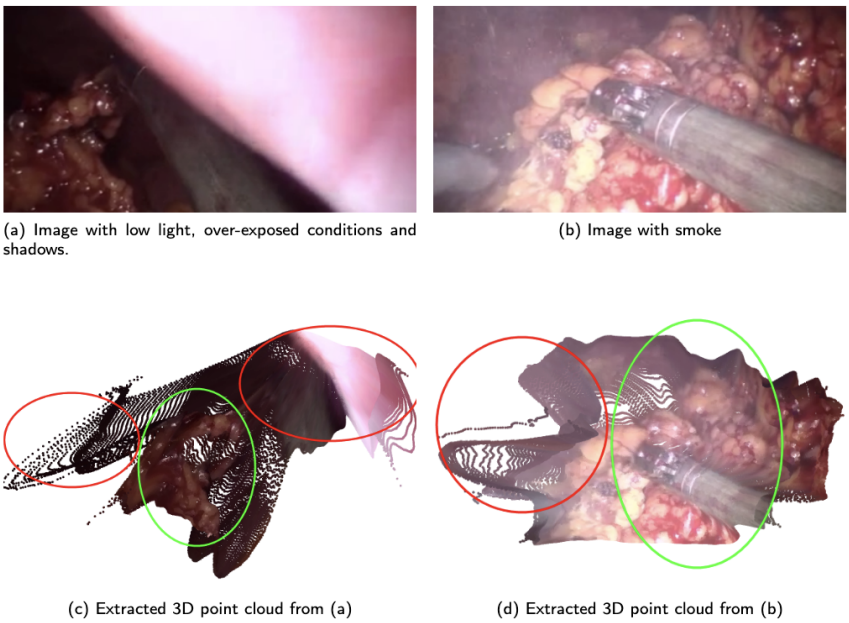
Raw GAN is not able to generate a specific 3D point cloud with respect to the input data for multi-classes training, which means the output of raw GAN can be similar to any data in multi-classes datasets. Figure 3.16 shows the output of raw GAN and Endo-AE, respectively. The input data (b) is a 3D point cloud of the Abdomen Wall. Endo-AE can generate the complete 3D point cloud (d) according to the input (b), but the output of raw GAN (c) is similar to the data in Uterine Horn. An l-GAN is also trained, passing data through a pre-trained Endo-AE, the same problem with raw GAN occurs. Thus, although raw GAN and l-GAN can produce

complete 3D point clouds, it is not reliable for endoscopic 3D point cloud completion.

In addition, even if raw GAN and l-GAN are trained with a single class of endoscopic 3D point clouds, the same problem still occurs. One major reason is that GAN randomly generates data based on the whole training set and does not find the best representation of the corresponding partial input 3D point cloud. Another reason is the specialist in endoscopic data. Because the *in-vivo* endoscopic data is varied and deformable, differences can be large even between the closest frames in the same dataset.

3.7.3 Discussion and Limitation

As shown in Figure 3.17, although the proposed mono-depth network outperforms the state-of-the-art learning-based method (Godar) and non-learning-based stereo 3D reconstruction algorithms (BM and SGBM), the quality of the extracted 3D point cloud is still affected by low light, dark, over-exposed conditions, shadows and smoke. The low light condition, darkness and shadows cause inconsistent and wrong depth information, which is because the network considers the low light and dark areas to be far away from the endoscope. The over-exposed condition causes deformation in the extracted 3D point cloud, as shown in the right red box in Figure 3.17 (c). One possible reason is that the over-exposed areas are texture-less and are difficult to calculate the disparities. The smoke also affects the estimation of depth information, the network considers a large amount of smoke as part of the scene and estimates the depth of the smoke, as shown



(a) Image with low light, over-exposed conditions and shadows.

(b) Image with smoke

(c) Extracted 3D point cloud from (a)

(d) Extracted 3D point cloud from (b)

Fig. 3.17 Effects of low light, over-exposed conditions, shadows and smoke. Red boxes show inconsistent and wrong depth and green boxes show a better quality of depth estimation.

in the red box in Figure 3.17 (d). The network also tolerates a small amount of smoke, as shown in the green box in Figure 3.17 (d).

Figure 3.14 shows the high-quality completion results of the Endo-AE model on seven datasets in terms of various missing rates. Figures 3.15 and 3.16 also illustrate that the Endo-AE outperforms GANs. The limitation of the proposed Endo-AE network is that the output of the complete 3D point cloud is regenerated rather than repaired by increasing the number of points in missing areas.

In the proposed framework, the mono-depth and Endo-AE networks are trained separately. Because the 3D point cloud completion network should be trained using ground-truth 3D point clouds for supervision, and there are no ground-truth 3D point clouds for the public laparoscopic/endoscopic stereo video datasets. Therefore, the depth learning network is first trained to generate the 3D endoscopic

point clouds. These generated 3D endoscopic point clouds are considered as ground-truth 3D point clouds for training the 3D point cloud completion network. Theoretically, the mono-depth and the Endo-AE networks can be trained together once the ground-truth endoscopic 3D point clouds are obtained. In the future, these two networks will be trained together when the ground-truth endoscopic 3D point clouds are obtained.

3.8 Conclusion

A novel framework to recover dense 3D point clouds from single endoscopic images has been proposed. The framework includes an unsupervised mono-depth network that generates the depth from a single endoscopic image. Based on the mono-depth learning network, a dense 3D point cloud can be extracted from an endoscopic image. The seven groups of *in-vivo* 3D endoscopic point cloud datasets have been created and made publicly available to researchers. A generative Endo-AE network is then trained to complete 3D point clouds with various degrees of missing data. Experimental results show the capability of the proposed computational framework for producing dense 3D endoscopic point cloud datasets and its effectiveness in repairing defects of real endoscopic point cloud datasets and synthetic medical models.

In the next chapter, a 3D point cloud completion network is proposed to generate points only in missing areas instead of regenerating all points in the output.

Chapter 4

TreeNet: Structural Preserving for Multi-class 3D Point Cloud Completion

In this chapter, a novel deep learning network (TreeNet) is proposed for 3D point cloud completion. The objectives of this chapter are to focus on multi-class training and missing points generation with original structure-preserving. Experimental results show that TreeNet outperforms five state-of-the-art learning-based methods on the public 3D real-world object dataset (i.e. cars, tables) and exhibits good generalization to unknown classes that are not trained. The proposed TreeNet is also evaluated in the previously proposed computational framework (Chapter 3) on endoscopic scenes, which shows the effectiveness of TreeNet for medical data.

4.1 Introduction

3D point clouds, captured by various sensor technologies such as laser and RGB-D scanners and depth cameras, suffer from large missing

data due to complicated occlusions, unreliable measurements, limited viewing angles and the resolution of various sensors in dealing with texture-less regions of the scene. Therefore, generating a complete 3D point cloud (i.e. 3D point cloud completion) from a captured incomplete point cloud is an essential task for a wide range of 3D vision applications [4] [24] [8] [25]. The 3D point cloud completion networks [38] [39] [40] [42] [41] have achieved the state-of-the-art completion results. However, there are two shared problems with these methods. First, these methods perform poorly on multi-classes, even a single class that contains mostly different shapes. These methods produce low-quality 3D point clouds on multi-classes training data and lose structural and spatial details, such as sharp edges and topology changes. Second, these methods lose original structural and spatial details in the final output due to the fact that they regenerate the entire 3D point cloud and do not separate the reconstructed partial input from the missing points in the final output. Here, an important observation has been made that *the partial input should be fully preserved rather than regenerated.* Based on this original idea, a novel deep learning-based network is proposed by devising hierarchical tree-based decoders to build a learning network - TreeNet for solving the two problems.

TreeNet has two networks in tree structures: (i) TreeNet-multiclass focuses on multi-class training with a specific class of the point cloud completion task on each sub-tree to improve the quality of the 3D point cloud output; (ii) TreeNet-binary focuses on generating points in missing areas and fully preserving the original partial input. TreeNet-multiclass and TreeNet-binary are both network decoders and can be trained independently. The TreeNet decoder is the

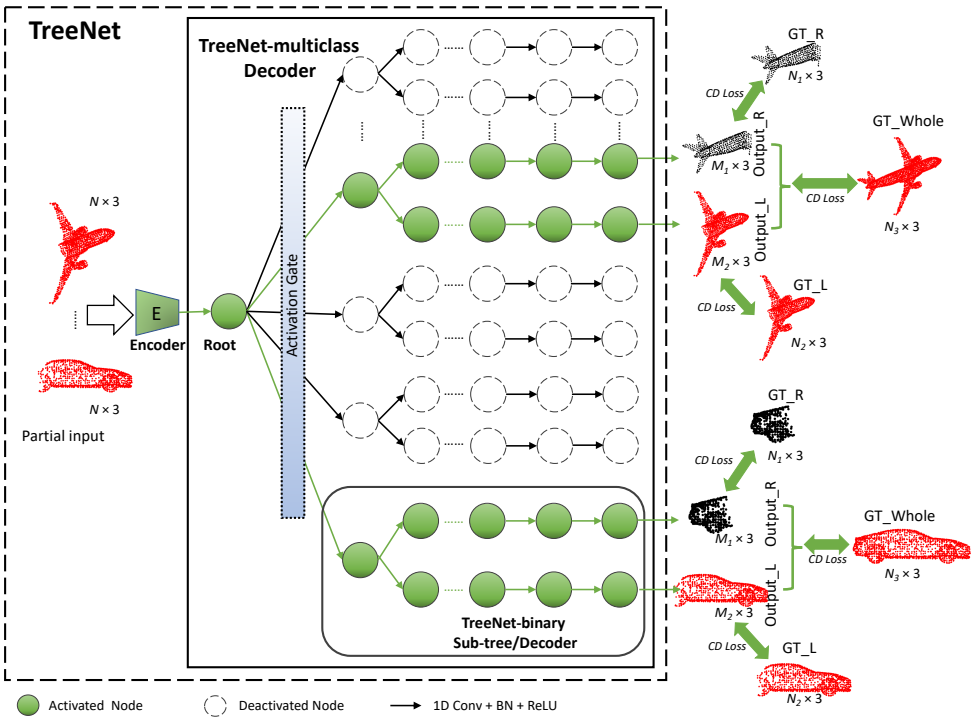


Fig. 4.1 TreeNet architecture. TreeNet combines TreeNet-multiclass and TreeNet-binary.

combination of the TreeNet-multiclass and TreeNet-binary, which is trained with an encoder from existing methods (i.e. PointNet encoder).

Three novel forward propagation methods are proposed to train the TreeNet-multiclass, TreeNet-binary and TreeNet, respectively. First, to train the TreeNet-multiclass for multi-class 3D point cloud completion, an activation gate is proposed for the standard forward propagation to activate and deactivate sub-trees of the root node by assigning each class of a 3D completion task to its corresponding activated sub-tree. Unlike the standard backward propagation, the gradient of the loss function in TreeNet-multiclass is only calculated on the activated sub-trees during each batch of the training. Second,

the forward propagation for TreeNet-binary splits features of the root node to a binary tree structure where the left leaf node reconstructs the partial input and the right leaf node generates points in missing areas. Third, TreeNet is trained using the combined forward propagation methods of TreeNet-multiclass and TreeNet-binary. The proposed TreeNet-multiclass, TreeNet-binary and TreeNet are compared with five state-of-the-art learning-based methods on fifty classes of the public Shapenet dataset [115] and unknown classes [115], which shows that the proposed models provide significant improvements in the overall quality and exhibit strong generalization to unknown classes that are not trained.

4.2 Methodology

Given a partial 3D point cloud input with N points where each point is defined as $P_i = (x, y, z)$, the novel tree-based decoder-TreeNet generates M missing points. TreeNet decoder, as shown in Figure 4.1, contains TreeNet-multiclass and TreeNet-binary. TreeNet-multiclass (Section 4.2.1) is for multi-class 3D point cloud completion, as shown in Figure 4.2. TreeNet-binary (Section 4.2.2) generates points in missing areas and preserves the original partial input, as shown in Figure 4.4. Currently, TreeNet uses TreeNet-binary as the sub-tree of the root node in TreeNet-multiclass and achieves the final structure of TreeNet. However, this does not limit the TreeNet-multiclass and TreeNet-binary to be used as individual decoders. TreeNet-multiclass, TreeNet-binary and TreeNet are trained using their own forward propagation methods (Section 4.2.3) and a combined chamfer distance [91] as loss (Section 4.2.4) and within an auto-encoder

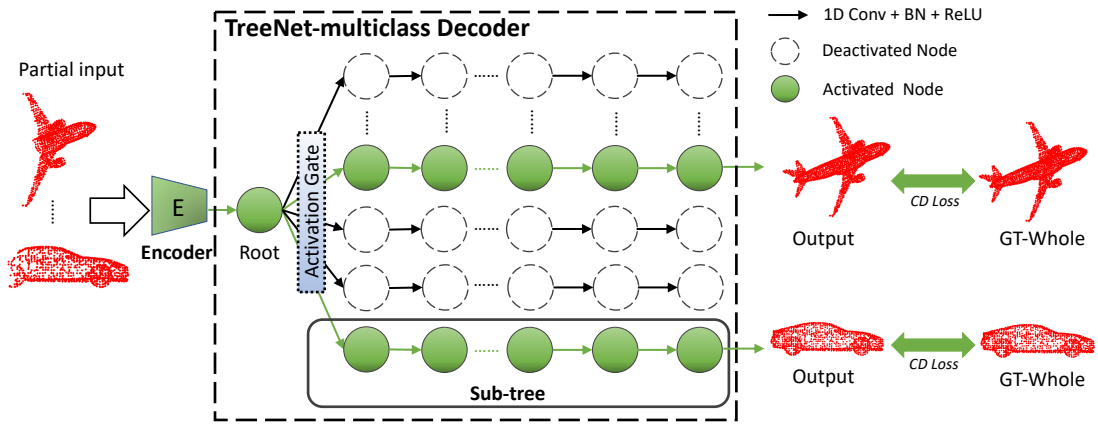


Fig. 4.2 TreeNet-multiclass architecture. Each sub-tree is designed to focus on a specific class of 3D point cloud completion tasks during the training. Once trained, these sub-trees can be used for unknown classes. (Refer to Section 4.3.5 for more details about unknown classes.)

framework that includes a PointNet [1] based encoder as a first stage.

4.2.1 Multi-Class Point Cloud Completion

To train a class-invariant model for multi-class 3D point cloud completion, TreeNet-multiclass assigns each class of the 3D completion task to a specific sub-tree of its root node, where each of the sub-tree is identical to others and designed to focus on generating 3D point clouds from a specific class. To this end, the number of branches to the root node is identical to the number of classes in the training data. The architecture of the TreeNet-multiclass is shown in Figure 4.2. The root node is a feature vector of partial inputs generated from a PointNet-based encoder. The 1D convolution (Conv), batch norm (BN) and ReLU layers are used for generating features at each level of the tree. The root node is connected with D sub-trees that correspond to D classes of data. To assign D classes of 3D

completion tasks to D corresponding sub-trees of the root, an activation gate (Equation 4.2) is proposed for the forward propagation to activate and deactivate the sub-trees in the decoder, as illustrated in Section 4.2.3.1. Each node contains the features extracted from its connected node in the previous layer. The feature only passes through its corresponding activated sub-tree, and other sub-trees will be temporarily deactivated. The deactivated sub-trees will be activated again when the corresponding features of input data pass through. Therefore, these sub-trees do not share weights with each other and focus on each class of the 3D completion task. Each leaf node in each sub-tree represents a complete 3D point cloud. TreeNet-multiclass is trained by using the proposed novel forward and backward method described in Section 4.2.3 and the chamfer distance loss as shown in Equation 4.5.

Assuming there are T 3D point clouds in the training set, including D classes of datasets. The B training point clouds in each batch of the training are randomly selected from the training set, and the whole training set is trained with E epochs. Assuming there are Q different classes of data contained in each batch and Q varies in each batch. Thus, Q different sub-trees are activated at the same time for each batch of training and $D - Q$ different sub-trees are deactivated. Because Q varies in each batch of training, there are $\frac{T}{B} \times E$ different situations of the forward propagation during the training, and the values in the activation gate are automatically updated $\frac{T}{B} \times E$ times based on the class of the randomly selected data. Two possible situations of forward propagation are shown for training during two different batches of training in Figure 4.3. The second sub-tree from the left to the right is deactivated in (a) but

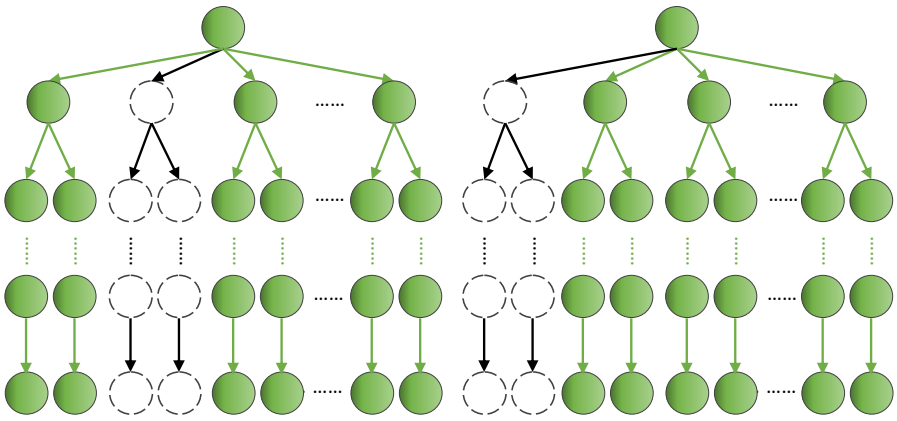


Fig. 4.3 Forward propagation. The forward propagation only operates on the activated nodes in each batch of training. (a) One possible situation during a batch of training. (b) Another possible situation during another batch of training.

activated in (b). The first sub-tree from the left is deactivated in (b) but activated in (a).

Once trained, these sub-trees can be used for unknown classes. A partial 3D point cloud from an unknown class can be passed through the encoder to all sub-trees in the decoder. The activation gate is disabled for unknown classes. Thus, there are D different completion outputs from D sub-trees. Refer to Section 4.3.5 for more details about unknown classes.

Since the output of the TreeNet-multiclass decoder is a complete 3D point cloud by regenerating all points in the output, in the next section, the tree-based decoder is further designed to generate points in missing areas and preserve the original partial input structure (Section 4.2.2).

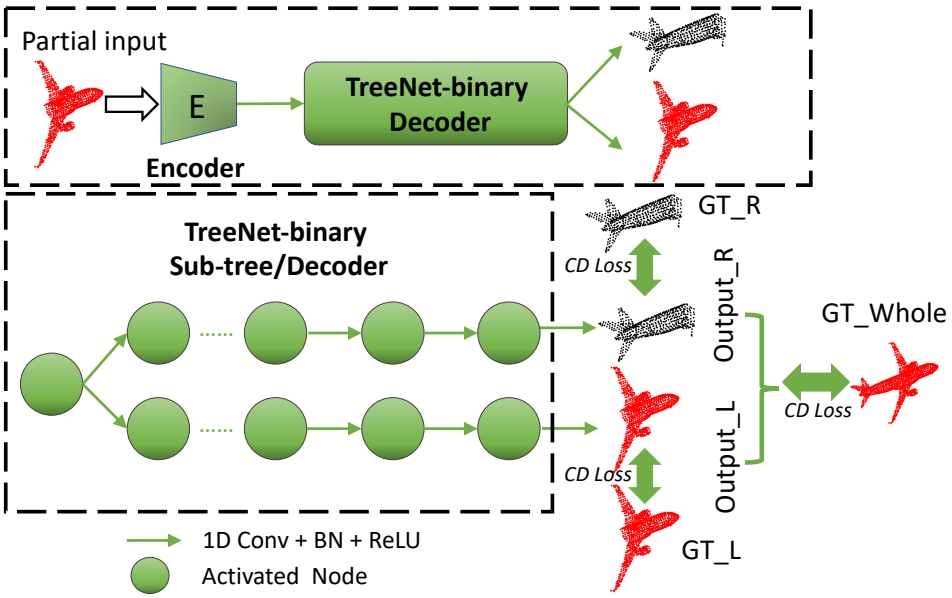


Fig. 4.4 TreeNet-binary architecture. The output 3D point cloud is divided into two segments (Output_L and Output_R). Output_L is the reconstruction of the partial input, and Output_R represents points in missing areas. Output_L and Output_R are used to calculate the combined loss during training.

4.2.2 Missing Points Generation with Original Structure Preserving

To generate points in missing areas and fully preserve the structure of the original partial input, a novel TreeNet-binary decoder is further proposed to follow a binary tree structure. The architecture of the TreeNet-binary is shown in Figure 4.4. During the training, the left leaf node produces the reconstruction of the partial input, and the right leaf node generates points in missing areas. To make the output global shape similar to the ground truth, the outputs from the left and the right leaf nodes are used to calculate the combined loss for the network training, as shown in Equation 4.6 in Section 4.2.4. Once trained, the left sub-tree is disabled and only the right sub-tree is used. The final output is the combination of the output of the

right leaf node and the partial input. Therefore, the original partial input is fully preserved.

Most importantly, there are two uses for the proposed TreeNet-binary. First, the TreeNet-binary can be considered as an individual decoder and trained with the novel forward propagation, as illustrated in Equation 4.3 in Section 4.2.3.1. Second, the TreeNet-binary is used as the sub-tree of the root node in TreeNet-multiclass and the final structure of TreeNet is achieved.

4.2.3 Forward and Backward Propagation in Tree-Based Decoder

In this section, the forward and backward propagation methods are illustrated for training TreeNet-multiclass, TreeNet-binary and TreeNet, respectively.

4.2.3.1 Forward Propagation

Based on the design of the TreeNet-multiclass, an activation gate is proposed to activate and deactivate sub-trees of the root node for novel forward propagation. The features pass through all neurons in standard forward propagation (Equation 4.1), whereas the features only pass through their corresponding activated sub-tree in the proposed forward propagation for TreeNet-multiclass (Equation 4.2).

$$z_i^{(l+1)} = \boldsymbol{w}_i^{(l+1)} \boldsymbol{y}_i^{(l)} + \boldsymbol{b}_i^{(l+1)}; \qquad \boldsymbol{y}_i^{(l+1)} = f(z_i^{(l+1)}) \tag{4.1}$$

where i indexes the hidden neuron in each layer and l indexes the hidden layer. $\boldsymbol{w}_i^{(l+1)}$ and $\boldsymbol{b}_i^{(l+1)}$ denote the i-th weight and bias at layer $l+1$. $\boldsymbol{y}_i^{(l)}$ is the i-th features of inputs at the layer l. $z_i^{(l+1)}$ is

the i-th feature of inputs at the layer $l+1$. $f(\cdot)$ is any activation function, e.g. Tanh.

Given D classes of training data, the number of sub-trees in TreeNet-multiclass is D. Let $d \in \{0, 1, ..., D\}$ and defines the d-th sub-tree from the left to the right. Assuming the feature size of the root node is reshaped to $[B, D, m]$, where B is the batch size and $[D, m]$ is the size of a feature vector F from a partial input 3D point cloud. Thus, there are B feature vectors in the root node preparing to pass through the tree. The purpose is to assign B feature vectors to their corresponding sub-trees during each batch of training. An activation gate (Equation 4.2) activates and deactivates sub-trees of the root node, and the size of the activation gate is $[B, D, m]$ which is as same as the feature size of the root node. Thus, there are D inner gates in the activation gate, where each inner gate g_d is a vector with the size of $m \times 1$ and all values in each g_d are the same, either all 0 or all 1. Each inner gate corresponds to a sub-tree and decides whether the corresponding sub-tree is activated or deactivated. For the sub-trees with the corresponding features F coming through, all values in g_d become 1, and sub-trees without the corresponding F values in g_d become 0. *Activation_Gate* is an element-wise product with B feature vectors in the root node to activate and deactivate sub-trees. Thus, the forward propagation for TreeNet-multiclass is defined by Equation 4.2.

$$
\begin{aligned}
&Activation\_Gate = [g_1, g_2, g_3, ..., g_D] \\
&\tilde{\boldsymbol{y}}_{subtree\_Root} = Activation\_Gate * \boldsymbol{y}_{Root} \\
&z_i^{d^{(l+1)}} = \boldsymbol{w}_i^{d^{(l+1)}} \tilde{\boldsymbol{y}}_{subtree\_Root^i}^{d^{(l)}} + \boldsymbol{b}_i^{d^{(l+1)}} \quad (d \in \{0, 1, ..., D\}) \\
&\boldsymbol{y}_i^{d^{(l+1)}} = f(z_i^{d^{(l+1)}})
\end{aligned}
\tag{4.2}
$$

where $*$ denotes an element-wise product. $\boldsymbol{y}_{Root}$ denotes the features in the root node of the tree. $\tilde{\boldsymbol{y}}_{subtree\_Root}$ denotes features in the root of each sub-tree. d defines the d-th sub-tree from the left to the right. l indexes the hidden layer and i indexes the hidden neuron in each layer. $\tilde{\boldsymbol{y}}^{d(l)}_{subtree\_Root^i}$ denotes the i-th activated or deactivate neuron in d-th sub-tree at the layer l. $\boldsymbol{w}_i^{d(l+1)}$ and $\boldsymbol{b}_i^{d(l+1)}$ denote the i-th weight and bias in d-th sub-tree at layer $l+1$. $z_i^{d(l+1)}$ denotes the i-th feature vector in d-th sub-tree at the layer $l+1$. $\boldsymbol{y}_i^{d(l+1)}$ denotes the i-th feature vector in d-th sub-tree at the layer $l+1$. $f(\cdot)$ is a Tanh activation function.

The forward propagation for TreeNet-binary, as defined by Equation 4.3, is proposed to train TreeNet-binary.

$$
\begin{aligned}
&\boldsymbol{y}_{RST} = \boldsymbol{y}_{Root_{idx}} \quad (idx = 0, ..., m-1) \\
&\boldsymbol{y}_{LST} = \boldsymbol{y}_{Root_{idx}} \quad (idx = m, ..., 2m) \\
&z^{(l+1)}_{RST^{(i)}} = \boldsymbol{w}^{(l+1)}_{RST^{(i)}} \boldsymbol{y}^{(l)}_{RST^{(i)}} + \boldsymbol{b}^{(l+1)}_{RST^{(i)}}; \qquad \boldsymbol{y}^{(l+1)}_{RST^{(i)}} = f(z^{(l+1)}_{RST^{(i)}}) \\
&z^{(l+1)}_{LST^{(i)}} = \boldsymbol{w}^{(l+1)}_{LST^{(i)}} \boldsymbol{y}^{(l)}_{LST^{(i)}} + \boldsymbol{b}^{(l+1)}_{LST^{(i)}}; \qquad \boldsymbol{y}^{(l+1)}_{LST^{(i)}} = f(z^{(l+1)}_{LST^{(i)}})
\end{aligned}
\tag{4.3}
$$

where $\boldsymbol{y}_{Root_{idx}}$ denotes the feature vector in the root node of the tree. $\boldsymbol{y}_{RST}$ and $\boldsymbol{y}_{LST}$ denote features for the right and left sub-trees, respectively. m is the feature size of the left and right sub-trees. $\boldsymbol{y}^{(l)}_{RST^{(i)}}$ and $\boldsymbol{y}^{(l)}_{LST^{(i)}}$ denote the i-th neuron in the right and left sub-trees at the layer l, respectively. $\boldsymbol{w}^{(l+1)}_{RST^{(i)}}$ and $\boldsymbol{b}^{(l+1)}_{RST^{(i)}}$ denote the i-th weight and bias for right sub-tree at layer $l+1$. $\boldsymbol{w}^{(l+1)}_{LST^{(i)}}$ and $\boldsymbol{b}^{(l+1)}_{LST^{(i)}}$ denote the i-th weight and bias for left sub-tree at layer $l+1$. $z^{(l+1)}_{RST^{(i)}}$ and $z^{(l+1)}_{LST^{(i)}}$ denote the i-th feature vector in right and left sub-trees at layer $l+1$, respectively. $\boldsymbol{y}^{(l+1)}_{RST^{(i)}}$ and $\boldsymbol{y}^{(l+1)}_{LST^{(i)}}$ denote the i-th feature

vector after an activation function $f(\cdot)$ in right and left sub-trees at layer $l+1$, respectively.

The TreeNet combines TreeNet-multiclass and TreeNet-binary and is trained using the combined forward propagation of TreeNet-multiclass and TreeNet-binary, as defined by Equation 4.4.

$$
\begin{aligned}
&Activation\_Gate = [g_1, g_2, g_3, ..., g_D];\\
&\tilde{\boldsymbol{y}}_{subtree\_Root} = Activation\_Gate * \boldsymbol{y}_{Root};\\
&\tilde{\boldsymbol{y}}_{RST}^{d^{(l)}} = \tilde{\boldsymbol{y}}_{subtree\_Root_{idx}}^{d^{(l)}} \ \ (d \in \{0, 1, ..., D\})\\
&(idx = 2d \times m, ..., (2d+1) \times m - 1);\\
&\tilde{\boldsymbol{y}}_{LST}^{d^{(l)}} = \tilde{\boldsymbol{y}}_{subtree\_Root_{idx}}^{d^{(l)}} \ \ (d \in \{0, 1, ..., D\})\\
&(idx = (2d+1) \times m, ..., (d+1) \times 2m - 1);\\
&z_{RST^{(i)}}^{d^{(l+1)}} = \boldsymbol{w}_{RST^{(i)}}^{d^{(l+1)}} \boldsymbol{y}_{RST^{(i)}}^{d^{(l)}} + \boldsymbol{b}_{RST^{(i)}}^{d^{(l+1)}}; \qquad \boldsymbol{y}_{RST^{(i)}}^{d^{(l+1)}} = f(z_{RST^{(i)}}^{d^{(l+1)}});\\
&z_{LST^{(i)}}^{d^{(l+1)}} = \boldsymbol{w}_{LST^{(i)}}^{d^{(l+1)}} \boldsymbol{y}_{LST^{(i)}}^{d^{(l)}} + \boldsymbol{b}_{LST^{(i)}}^{d^{(l+1)}}; \qquad \boldsymbol{y}_{LST^{(i)}}^{d^{(l+1)}} = f(z_{LST^{(i)}}^{d^{(l+1)}});
\end{aligned}
\tag{4.4}
$$

where $\tilde{\boldsymbol{y}}_{RST}^{d^{(l)}}$ and $\tilde{\boldsymbol{y}}_{LST}^{d^{(l)}}$ denote the d-th right and left sub-trees at layer l, respectively. m is the feature size of each sub-tree root node and is also the size of each inner gate g_d. $\boldsymbol{w}_{RST^{(i)}}^{d^{(l+1)}}$ and $\boldsymbol{b}_{RST^{(i)}}^{d^{(l+1)}}$ are the i-th weight and bias in d-th right sub-tree at layer $l+1$. $\boldsymbol{w}_{LST^{(i)}}^{d^{(l+1)}}$ and $\boldsymbol{b}_{LST^{(i)}}^{d^{(l+1)}}$ are the i-th weight and bias in d-th left sub-tree at layer $l+1$. $\boldsymbol{y}_{RST^{(i)}}^{d^{(l+1)}}$ and $\boldsymbol{y}_{LST^{(i)}}^{d^{(l+1)}}$ denote the i-th feature vector after an activation function $f(\cdot)$ in d-th right and left sub-trees at layer $l+1$, respectively.

4.2.3.2 Backward Propagation

Backward propagation (BP) is the major learning procedure that repeatedly adjusts the weights and biases of the connections in the

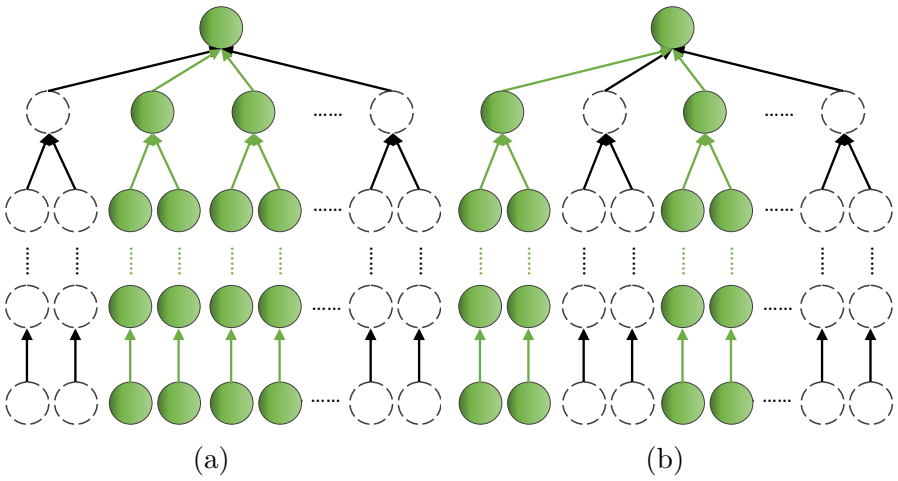


Fig. 4.5 Backward propagation. The backward propagation only operates on the activated nodes in each batch of training. (a) One possible situation during a batch of training. (b) Another possible situation during another batch of training.

network to minimize a measure of the difference between the output and the ground truth. Since the values of nodes in deactivated sub-trees are zero, the gradient of the loss function in the backward propagation is only calculated on the activated sub-trees during each batch of training. Whereas the existing learning-based methods for 3D point cloud completion [37] [38] [39] [40] [42] [41] need all neurons in the decoder to participate in backward propagation during training. Therefore, TreeNet is more efficient and effective in training multi-classes of data. Similarly to forward propagation, two possible situations of backward propagation are shown during two different batches of training in Figure 4.5. The first sub-tree of the root node from the left to the right is deactivated in (a) but activated in (b). The second sub-tree of the root node from the left to the right is activated in (a) but deactivated in (b). The last sub-trees to the right are all deactivated in both (a) and (b).

4.2.4 Loss Function

The loss function for 3D point cloud completion measures the difference between the output 3D point cloud S_{output} and the ground truth point cloud S_{gt}. The loss is defined to be invariant to any permutation of 3D point clouds in both S_{output} and S_{gt}. Similarly to the state-of-the-art methods [38] [39] [40] [42] [41], the chamfer distance (CD) [91] is also used in the loss function.

$$\begin{aligned} CD(S_{output}, S_{gt}) = & \frac{1}{S_{output}} \sum_{x \epsilon S_{output}} \underset{y \epsilon S_{gt}}{min} ||x-y||_2 \\ & + \frac{1}{S_{gt}} \sum_{y \epsilon S_{gt}} \underset{s \epsilon S_{output}}{min} ||y-x||_2 \end{aligned} \tag{4.5}$$

The chamfer distance calculates the average nearest point distance between S_{output} and S_{gt} by finding the closest neighbour with $O(nlogn)$ complexity. In addition, S_{output} and S_{gt} can be the different sizes of 3D point clouds.

The loss function for TreeNet-multiclass is CD, as illustrated in Equation 4.5. Based on the chamfer distance, the final loss functions for TreeNet-binary and TreeNet are the same, as defined in Equation 4.6.

$$\begin{aligned} Loss_1 &= CD(S_{Output\_L}, S_{GT\_L}) \\ Loss_2 &= CD(S_{Output\_R}, S_{GT\_R}) \\ Loss_3 &= CD(S_{Output\_L\&R}, S_{GT\_Whole}) \\ Loss &= (\lambda_1 \cdot Loss_1 + \lambda_2 \cdot Loss_2) + Loss_3 \end{aligned} \tag{4.6}$$

where $Output\_L$ is the output of the left leaf node, and GT_L is the ground truth corresponding to the $Output\_L$. $Output\_R$ is

the output of the right leaf node, and GT_R is the ground truth corresponding to the $Output\_R$. $Output\_L\&R$ is the combination of the $Output\_L$ and $Output\_R$, and GT_Whole is the ground truth corresponding to the $Output\_L\&R$. λ is the corresponding weight to balance the effect of gradients of the backward propagation.

4.3 Experiments

The tree-based decoders are trained within an auto-encoder framework that includes a PointNet-based encoder [39]. An Nvidia Geforce 2080Ti GPU with 12G memory is used for network training. Three-stage experiments are conducted. First, a set of experiments are conducted to decide the final design of the TreeNet-multiclass, TreeNet-binary and TreeNet, including the effectiveness of the existing point cloud encoders, depth of the trees and values of λ in the loss function (Section 4.3.3). Second, the TreeNet-multiclass, TreeNet-binary and TreeNet are compared with the state-of-the-art methods [38] [39] [40] [42] [41] on testing datasets from trained classes (Section 4.3.4). Third, the generalization ability of each network is evaluated on unknown classes that are never trained (Section 4.3.5). Similar to the state-of-the-art methods [38] [39] [40] [42] [41], the chamfer distance (Equation 4.5) is used as the evaluation metric to compare the output 3D point clouds with the corresponding ground truth.

4.3.1 Implementation Details

The TreeNet-multiclass, TreeNet-binary and TreeNet are trained for 600 epochs with a batch size of 32, a learning rate of 0.005, and an

adagrad optimizer. TreeNet-binary and TreeNet have $L = 7$ levels for generating the output 3D point cloud with the size of 2048×3, and TreeNet-multiclass has $L = 3$ levels. The weights defined in the total loss for TreeNet-binary and TreeNet are $\lambda_1 = 0.2$ and $\lambda_2 = 0.8$. The level of the tree and the weights in the loss function are illustrated from the ablation studies in Section 4.3.3.2.

For TreeNet, the feature size for the tree root is 1024, and the feature size for each sub-tree root is 2048. The filter sizes for the left sub-tree in each level are $[1024, 1536, 2048, 2560, 1024 \times 3]$. The filter sizes for the right sub-tree in each level are also $[1024, 1536, 2048, 2560, 1024 \times 3]$. The class label of each partial input 3D point cloud is automatically saved and decides the values in $Activation\_Gate^{(l)}$ (Equations 4.2 and 4.4) during the training.

As shown in Figure 4.1, TreeNet is trained with the input size of $N \times 3$ and three ground truth sizes of $N_1 \times 3$, $N_2 \times 3$ and $N_3 \times 3$, generating the output 3D point cloud with the sizes of $M_1 \times 3$ and $M_2 \times 3$. Once trained, the size of the final output 3D point cloud is $(N + M_1) \times 3$. Note that N, N_1, N_2, N_3, M_1 and M_2 can be any number. In the experiment, $N = 1024$, $N_1 = 512$, $N_2 = 1024$, $N_3 = 2048$, $M_1 = 1024$ and $M_2 = 1024$.

4.3.2 Datasets

Training and Testing Datasets. For a fair comparison, the ShapeNetCore [115] dataset is used as the training and testing datasets following FoldingNet [38], PCN [39], TopNet [40], PMP-Net [42] and Disp3d [41]. ShapeNetCore [115] is composed of 55 object classes with more than 50,000 3D point clouds, where each

3D point cloud contains 2048 points. The ShapeNetCore [115] 3D point cloud dataset is a subset of the full ShapeNet [115] 3D mesh dataset. The 2048 points in each 3D point cloud have been randomly down-sampled from the 3D meshes, thus, points are randomly distributed in each 3D surface. Each class of objects are categorized by global shapes and geometric clues from human experts. Each class contains 3D objects (e.g. laptops, cars, buses, pillows, lamps, sofas, tables, caps, mugs, vessels and etc.).

For the ShapeNetCore dataset of the 8 classes (29,774 point clouds), the data used for training is 97%, and the data used for testing is 3%. This percentage of dataset splitting used is followed by the recent state-of-the-art methods PCN [39], TopNet [40], PMP-Net [42] and Disp3d [41]. This dataset includes airplanes, cabinets, cars, chairs, lamps, sofas, tables and vessels.

In addition to the 8 classes, the further 42 classes in ShapeNetCore are randomly selected and combined with the 8 classes of ShapeNet-Core. This dataset contains 50 classes (51,188 point clouds). The data used for training is 90%, and the data used for testing is 10%. The testing dataset is not used for training and only for testing. This dataset includes airplanes, lamps, mugs, bowls, caps, laptops, buses, pillows, etc.

Since all 3D point clouds in ShapeNetCore are complete 3D point clouds, the partial 3D point clouds are created from all training and testing sets. During each epoch of training, the partial 3D point clouds are created from the training set with missing rates from 20% to 50%. During the testing, the partial inputs are created from the testing set with missing rates from 20% to 50%. The removed areas are selected from k nearest points around a randomly selected point.

Table 4.1 Encoder analysis: Quantitative comparison of the TreeNet against previous works using different encoders. E represents the encoder, and D is the TreeNet decoder. The encoders in TopNet and PCN are identical. The chamfer distance is reported multiplied by (10^3).

Methods	CD
PointNet [1](E) + TreeNet(D)	1.002
PointNet++ [70](E) + TreeNet(D)	7.284
AE [37](E) + TreeNet(D)	1.267
PCN [39]/TopNet [40](E) + TreeNet(D)	**0.817**

Testing Dataset from Unknown Classes. To evaluate the robustness and generalization ability of a network, the remaining five classes of the ShapeNetCore dataset are used as the unknown classes for the evaluation, including cameras, baskets, stoves, towers and printers. These five unknown classes consist of 843 3D point clouds that are not trained and strange to all networks. The testing partial point clouds are created with 50% of the missing rate.

4.3.3 Ablation Studies

In this section, the results of the ablation studies are presented to analyse the effectiveness of four state-of-the-art PointNet-based encoders for feature extraction modules in methods [1] [70] [37] [39] [40] and the tree-based decoders. Following PCN [39], TopNet [40], PMP-Net [42] and Disp3d [41], the model training and testing are based on the same 8 classes of the ShapeNetCore datasets.

Table 4.2 Quantitative comparison between different levels of the proposed networks tested on 8 classes of testing data. The chamfer distance is reported multiplied by (10^3).

Level	3	4	5	6	7	8
TreeNet-multiclass	**1.31**	1.37	1.38	1.37	1.47	1.45
TreeNet-bianry	1.999	1.098	0.990	0.989	**0.919**	0.957
TreeNet	1.065	0.853	0.853	0.853	**0.817**	0.828

4.3.3.1 Encoder Analysis

The effectiveness of five state-of-the-art PointNet-based encoders are analysed in PointNet [1], PointNet++ [70], AE [37], PCN [39] and TopNet [40].

In this experiment, to select the most effective encoder, the same TreeNet decoder is used as described in Section 4.3.3.2 but only change the encoder. The results of this analysis are reported in Table 4.1. As can be seen, the PCN's encoder shows a better performance than others. Thus, PCN's encoder is chosen as the final encoder. Note that compared across these methods using the same encoder, the TreeNet model outperforms the state-of-the-art methods (see Section 4.3.4.3).

4.3.3.2 Tree-Based Decoders Analysis

Tree-based decoders analysis is based on choosing the number of tree levels L in TreeNet-multiclass, TreeNet-binary and TreeNet, and weights λ_1 and λ_2 in Equation 4.6.

The depths of trees are one of the important design parameters. The number of tree levels L in TreeNet-multiclass, TreeNet-binary and TreeNet is chosen for an output point cloud size 2048×3 by

Table 4.3 Quantitative comparison between different weights for the final loss on 8 classes of ShapeNet. The chamfer distance is reported multiplied by (10^3).

λ_1, λ_2	TreeNet-binary	TreeNet
$\lambda_1 = 0.1$, $\lambda_2 = 0.9$	0.980	0.866
$\lambda_1 = 0.2$, $\lambda_2 = 0.8$	**0.919**	**0.817**
$\lambda_1 = 0.3$, $\lambda_2 = 0.7$	0.943	0.824
$\lambda_1 = 0.4$, $\lambda_2 = 0.6$	0.930	0.858
$\lambda_1 = 0.5$, $\lambda_2 = 0.5$	0.976	0.870
$\lambda_1 = 0.6$, $\lambda_2 = 0.4$	0.922	0.838
$\lambda_1 = 0.7$, $\lambda_2 = 0.3$	0.959	0.895
$\lambda_1 = 0.8$, $\lambda_2 = 0.2$	0.946	0.884
$\lambda_1 = 0.9$, $\lambda_2 = 0.1$	0.952	0.896

varying L in $3, 4, 5, 6, 7, 8$, respectively. The results on 8 classes of the ShapeNet dataset are reported in Table 4.2. Based on the results, 3 levels in TreeNet-multiclass and 7 levels in TreeNet-binary and TreeNet are used. As shown in the table, one notices that after a certain level, the increase of the tree depth does not necessarily increase the quality of the output, which is due to the fact that 3 levels in TreeNet-multiclass and 7 levels in TreeNet-binary and TreeNet already sufficiently include the majority of features in the output of 2048 points in this experiment. The results for different weights λ_1 and λ_2 are reported in Table 4.3. $\lambda_1 = 0.2$ and $\lambda_2 = 0.8$ are set based on these results. In all experiments, regardless of the value for L and λ above, the TreeNet-multiclass, TreeNet-binary and TreeNet outperform the state-of-the-art methods (see Table 4.4 in Section 4.3.4.3).

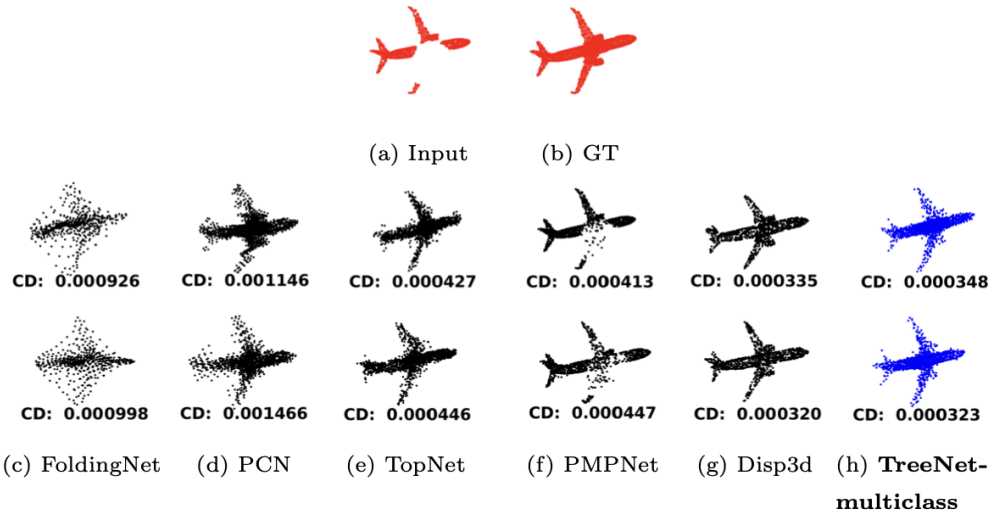


Fig. 4.6 Qualitative completion results from the same partial input based on 8 and 50 classes of training data. The second and third rows show the completion results of models trained on 8 and 50 classes of data, respectively.

4.3.4 Evaluation of Tree-Based Decoder

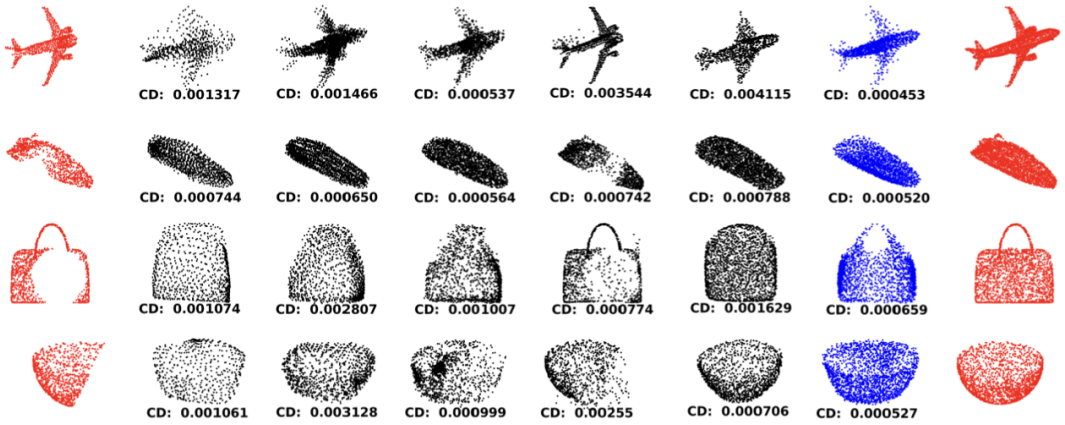
In this section, the effectiveness of TreeNet-multiclass, TreeNet-binary and TreeNet trained on 8 and 50 classes of training data is analysed by comparing them with FoldingNet [38], PCN [39], TopNet [40], PMPNet [42] and Disp3d [41].

4.3.4.1 Effectiveness of TreeNet-multiclass

To analyse the effectiveness of the proposed TreeNet-multiclass for multi-class 3D point cloud completion, all networks including FoldingNet [38], PCN [39], TopNet [40], PMPNet [42], Disp3d [41] and the TreeNet-multiclass are trained on the same 8 and 50 classes of training datasets, respectively. All networks are evaluated on the 8 and 50 classes of testing datasets (Section 4.3.2), respectively.

Figure 4.6 shows the results of each trained model from the same partial input. The second row in Figure 4.6 shows the results of each method based on 8 classes of training data, and the third row shows the results based on 50 classes of training data. The chamfer distance has been calculated between each result and ground truth and shown at the bottom of each figure. Based on the chamfer distances, the quality of output 3D point clouds from FoldingNet [38], PCN [39], TopNet [40] and PMPNet [42] decreases when the number of classes increases from 8 to 50 in the training data, whereas the TreeNet-multiclass model has shown stable results when the number of classes increases.

The average chamfer distances of the TreeNet-multiclass against the state-of-the-art methods tested on 8 and 50 classes of the testing data are shown in Tables 4.4, 4.5 and 4.6 with qualitative results in Figure 4.7. The TreeNet-multiclass outperforms FoldingNet [38], PCN [39], TopNet [40], PMPNet [42] and Disp3d [41] across all 8 and 50 classes of testing data. As shown in Figure 4.7, Disp3d [41] generates a car from a partial vessel as an input, which shows the poor ability on multi-classes training. The multi-classes training for the TreeNet-multiclass solves this problem and distinguishes features between each class. Most noticeably, the result of TreeNet-multiclass has shown a 5.08% improvement trained on 8 classes of data over the next best method TopNet [40], and a 10.19% improvement trained on 50 classes over TopNet [40], demonstrating the capability of TreeNet-multiclass in handling multi-class 3D point cloud completion tasks effectively.



(a) Input (b) FoldingNet (c) PCN (d) TopNet (e) PMPNet (f) Disp3d (g) **TreeNet-multiclass** (h) GT

Fig. 4.7 Qualitative completion results of models trained on 8 (first and second row) and 50 (third and fourth row) classes of data, respectively.

Table 4.4 Quantitative comparison of the proposed approach against previous works tested on 8 and 50 classes of testing data. The chamfer distances are reported multiplied by (10^3). Bold denotes the top three performing measures.

Methods	8 Classes	50 Classes
FoldingNet [38]	1.862	1.242
PCN [39]	1.946	1.251
TopNet [40]	1.378	1.079
PMPNet [42]	1.911	1.081
Disp3d [41]	1.880	1.649
TreeNet-multiclass	**1.308**	**0.969**
TreeNet-binary	**0.926**	**0.757**
TreeNet	**0.823**	**0.596**

Table 4.5 Evaluations of every class in 8 classes of testing datasets. The chamfer distances reported are multiplied by (10^3). Bold denotes the top three performing measures.

Methods	plane	cabinet	car	chair	lamp	sofa	table	vessel
FoldingNet [38]	1.56	1.99	1.15	2.87	2.39	1.78	1.95	1.21
PCN [39]	1.33	2.11	1.18	2.94	2.43	1.95	2.06	1.57
TopNet [40]	0.89	1.57	1.02	1.99	**1.70**	1.48	**1.35**	1.03
PMPNet [42]	1.18	2.65	1.53	2.46	2.27	1.75	2.50	0.95
Disp3d [41]	0.95	1.51	0.92	2.73	3.73	1.46	2.42	1.32
TreeNet-multiclass	**0.75**	**1.40**	**0.91**	**1.97**	1.75	**1.38**	1.48	**0.82**
TreeNet-binary	**0.53**	**1.17**	**0.60**	**1.30**	**1.37**	**0.82**	**1.00**	**0.62**
TreeNet	**0.44**	**0.96**	**0.53**	**1.12**	**1.29**	**0.74**	**0.98**	**0.53**

Table 4.6 Evaluations on the 50 classes of testing datasets. Nine classes of results are selected and displayed. The chamfer distance is reported multiplied by (10^3). Bold denotes the top three performing measures.

Methods	table	bench	bus	laptop	pistol	pot	monitor	bed	mug
FoldingNet [38]	1.98	1.42	0.60	1.40	1.21	1.44	1.16	1.98	1.85
PCN [39]	1.96	1.14	0.63	1.01	1.07	1.49	1.15	2.23	1.92
TopNet [40]	1.49	0.92	0.63	0.86	0.96	1.58	0.98	1.99	1.59
PMPNet [42]	2.46	0.83	0.77	0.92	**0.80**	**0.97**	**0.72**	**1.02**	1.17
Disp3d [41]	5.05	1.33	**0.49**	**0.61**	0.89	1.99	1.53	3.05	1.46
TreeNet-multiclass	**1.44**	**0.81**	0.57	0.62	0.84	1.50	0.88	**1.98**	**1.01**
TreeNet-binary	**1.02**	**0.69**	**0.37**	**0.59**	**0.55**	**0.93**	**0.62**	2.11	**1.09**
TreeNet	**0.87**	**0.48**	**0.30**	**0.38**	**0.48**	**0.87**	**0.50**	**1.15**	**0.78**

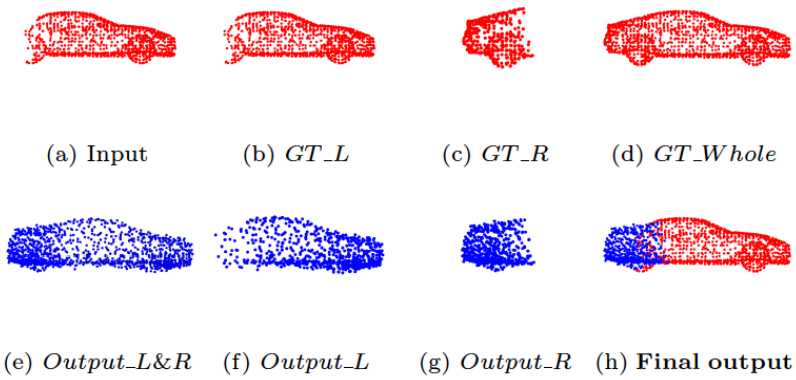
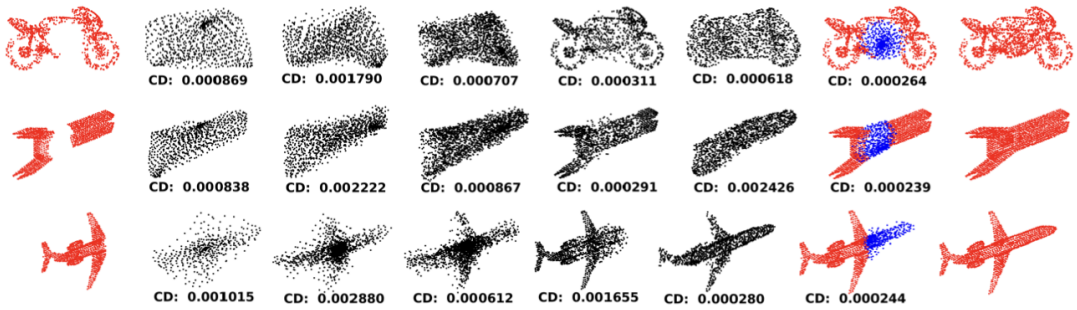


Fig. 4.8 Illustration of TreeNet-binary output.

4.3.4.2 Effectiveness of TreeNet-binary

To analyse the effectiveness of the proposed TreeNet-binary, all networks are trained on the same 8 and 50 classes of training datasets and evaluated on the same 8 and 50 classes of testing datasets (Section 4.3.2), respectively. Figure 4.8 illustrates the final output (h) of the TreeNet-binary which is the combination of the partial input (a) and the output of the right leaf node (g), which proves that TreeNet-binary is able to generate points in missing areas and fully preserve the original partial input.

The average chamfer distances are shown in Tables 4.4, 4.5 and 4.6 with qualitative completion results in Figure 4.9. As can be seen that TreeNet-binary also outperforms FoldingNet [38], PCN [39], TopNet [40], PMPNet [42] and Disp3d [41] across 8 and 50 classes of testing data, respectively. The result of TreeNet-binary has shown a 32.80% improvement trained on 8 classes of data over the next best method TopNet [40] and a 29.84% improvement trained on 50 classes of data over TopNet [40].



(a) Input (b) FoldingNet (c) PCN (d) TopNet (e) PMPNet (f) Disp3d (g) **TreeNet-binary** (h) GT

Fig. 4.9 Qualitative completion results of methods trained on 50 classes of data.

4.3.4.3 Effectiveness of TreeNet

To assess the effectiveness of TreeNet, the TreeNet is not only compared with FoldingNet [38], PCN [39], TopNet [40], PMPNet [42] and Disp3d [41] but also with TreeNet-multiclass and TreeNet-binary.

The average chamfer distances of the TreeNet against the state-of-the-art methods on 8 and 50 classes of the testing data are shown in Tables 4.4, 4.5 and 4.6, with qualitative results shown in Figure 4.10. TreeNet combines the advantages of TreeNet-multiclass and TreeNet-binary and outperforms TreeNet-multiclass and TreeNet-binary on 8 and 50 classes of testing data. In addition, TreeNet significantly outperforms FoldingNet [38], PCN [39], TopNet [40] , PMPNet [42] and Disp3d [41] across 8 and 50 classes of testing data. The result of TreeNet has shown a 40.28% improvement on 8 classes over the next best method TopNet [40], and a 44.76% improvement on 50 classes over TopNet [40]. As shown in Figure 4.10, FoldingNet [38], PCN [39] and TopNet [40] have failed to recover structural and spatial details such as sharp edges of a bench (2nd column), a laptop

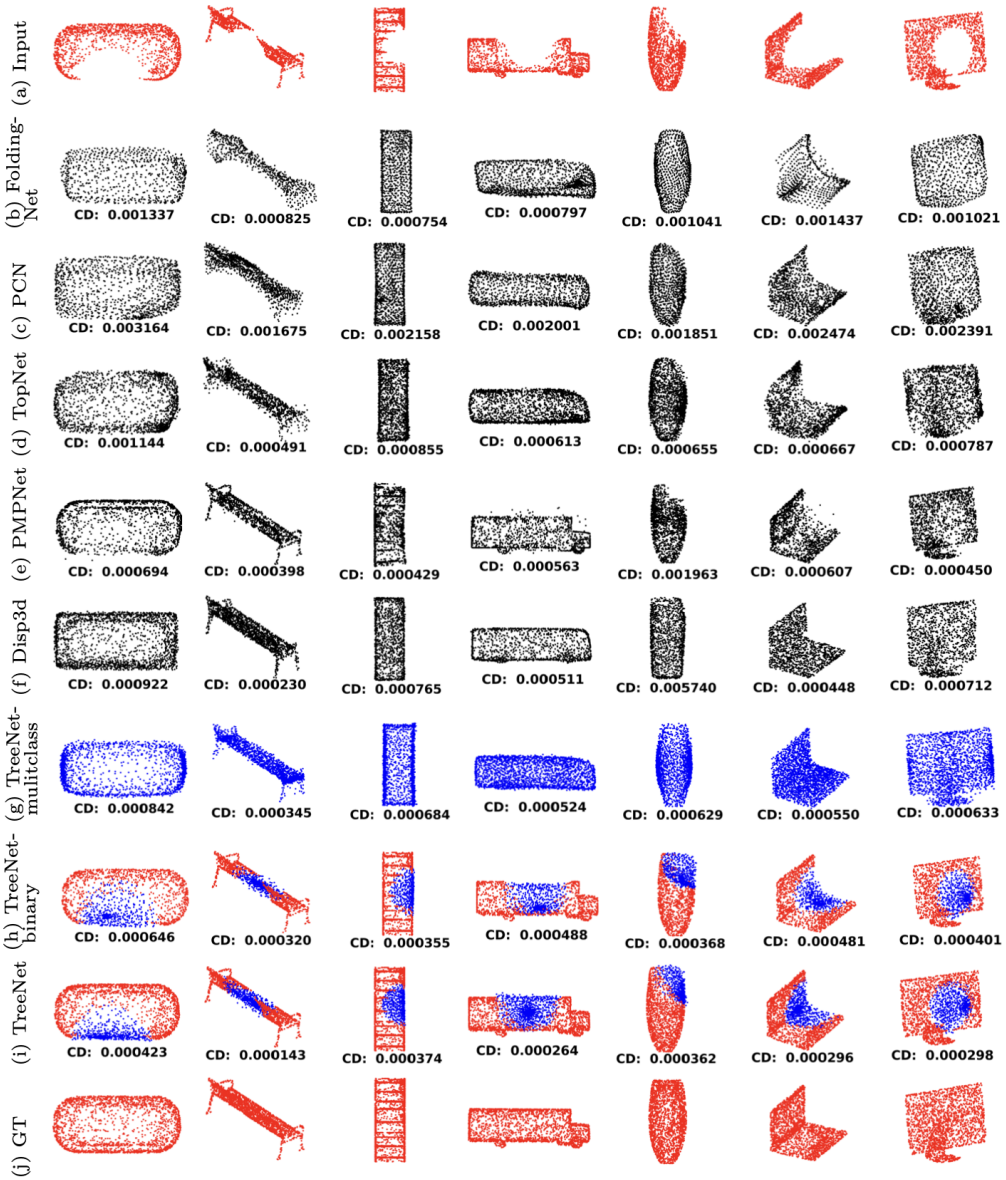


Fig. 4.10 Qualitative completion results of models trained on 50 classes of data.

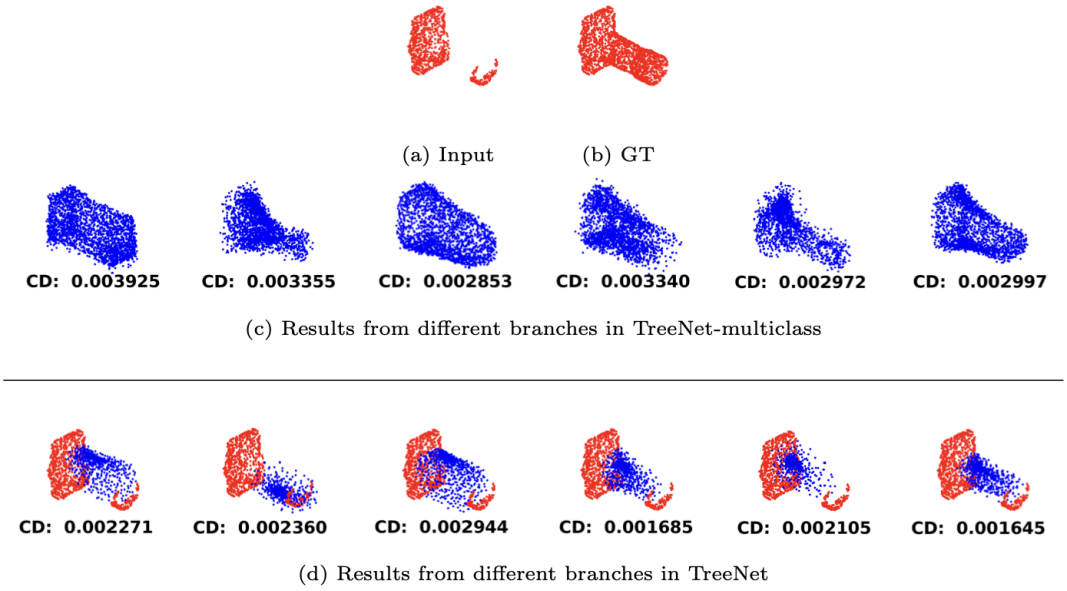


Fig. 4.11 Completion results from different branches in the tree for data in unknown classes.

(6th column) and a monitor (7th column) and topology change of a jar (5th column). Disp3d [41] confuses features between classes to some extent and generates a pot from a partial jar (5th column). In contrast, TreeNet has successfully generated these details and never confuses features between classes. Most importantly, FoldingNet [38], PCN [39], TopNet [40] and Disp3d [41] all lose the structural and spatial details of the original partial input. On the contrary, TreeNet-binary and TreeNet generate points in missing areas and preserve the partial input.

4.3.5 Generalization on Unknown Classes

Several comparison experiments have been extensively conducted to evaluate the generalization capability of the proposed networks. All learned models including FoldingNet [38], PCN [39], TopNet [40],

PMPNet [42], Disp3d [41], TreeNet-multiclass, TreeNet-binary and TreeNet are trained on the same 50 classes of training data and directly tested on the unknown classes (Section 4.3.2) that have never been trained. These shape datasets in unknown classes are different from the training datasets, which poses a significant challenge to the generalization of all learning-based methods.

A partial 3D point cloud from an unknown class can be passed through the encoder to all sub-trees in the decoder. The root nodes in TreeNet-multiclass and TreeNet contain common features extracted from the encoder, and each sub-tree in the decoder uses these common features to generate a complete 3D point cloud. As shown in Figure 4.11, a partial 3D shape from an unknown class is passed through the encoder to all sub-trees in the decoder and the generalization abilities from different sub-trees in TreeNet-multiclass and TreeNet are shown, respectively. The final output is the best completion result with the lowest chamfer distance between each output and the ground truth.

The performance of each method on unknown classes is reported in Table 4.7 with qualitative completion results of each method in Figure 4.12. The TreeNet ranks first and achieves the best completion results on all five unknown classes, which shows the strong generalization and robustness evaluated on unknown classes and also proves that the sub-trees can share common features among different classes. Refer to Section 4.4 for more detailed discussions and illustrations about unknown data.

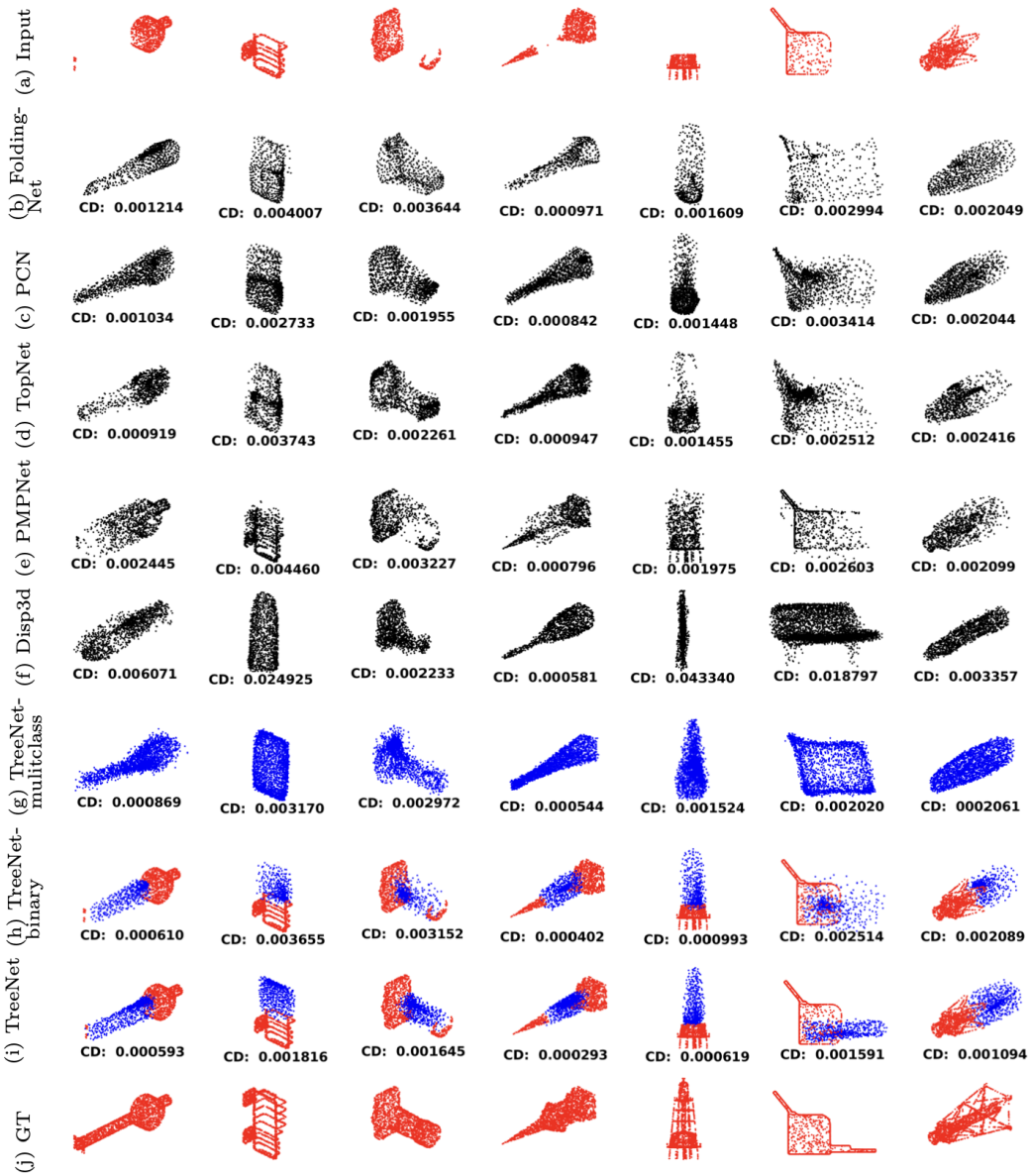


Fig. 4.12 Qualitative completion results of models on unknown classes with unknown shapes.

Table 4.7 Evaluations on unknown classes. The chamfer distance is reported multiplied by (10^3). Bold denotes the top three performing measures.

Methods	camera	basket	stove	tower	printer	*Avg.*
FoldingNet [38]	2.05	1.62	1.59	1.42	1.99	1.73
PCN [39]	2.03	1.64	1.49	1.34	2.02	1.70
TopNet [40]	2.09	1.67	1.49	1.34	1.81	1.68
PMPNet [42]	**1.73**	2.34	1.75	1.63	**1.47**	1.78
Disp3d [41]	5.58	2.71	2.76	5.47	3.93	4.09
TreeNet-multiclass	1.88	**1.33**	**1.20**	**1.13**	1.57	**1.42**
TreeNet-binary	**1.73**	**1.11**	**1.34**	**1.02**	**1.39**	**1.32**
TreeNet	**0.98**	**0.85**	**0.77**	**0.62**	**0.90**	**0.82**

Table 4.8 Evaluations on computational time for each completion when the input number of points increases from 1,024 to 16,384. This evaluation has been tested on 8 classes of testing datasets with 800 3D point clouds and the average computational time for each completion has been calculated and displayed.

Methods	Computational Time (s)			
	1,024	4,096	8,192	16,384
FoldingNet [38]	0.006644	0.008724	0.010075	0.013188
PCN [39]	0.005407	0.007779	0.010128	0.012389
TopNet [40]	0.008054	0.009938	0.011651	0.014097
PMPNet [42]	0.021583	0.033593	0.049685	0.080932
Disp3d [41]	0.053951	0.065234	0.103138	0.206863
TreeNet-multiclass	**0.005357**	**0.007574**	**0.009813**	**0.012687**
TreeNet-binary	**0.007568**	**0.009755**	**0.011250**	**0.015332**
TreeNet	**0.007901**	**0.009818**	**0.012404**	**0.015540**

4.4 Discussion and Limitation

In this section, the performance of computational time, completion results on unknown datasets and the limitations of the proposed TreeNet are discussed.

To evaluate the computational time when the input number of points increases, all networks are tested on 8 classes of the testing dataset and the average computational time for each completion is shown in Table 4.8. Disp3d takes the longest computational time, while the computational times for the other networks are close to each other. Most importantly, the computational time linearly increases from 1,024 points to 16,384 points for all networks.

To show the common features that can be shared among sub-trees, six types of car models are tested using a single sub-tree in the decoders of TreeNet-multiclass and TreeNet. The results show that the proposed networks can handle the six types of 3D car models with better performance than the other five state-of-the-art networks (FoldingNet [38], PCN [39], TopNet [40], PMPNet [42] and Disp3d [41]), as shown in Figure 4.13. This experiment indicates that a single sub-tree can share common features for different types of 3D models in one class. Also shown in the experimental results on unknown classes, the sub-trees can share common features among different classes, as shown in Figures 4.12 and 4.11.

For datasets from unknown classes with unknown shapes, all networks have certain generalizations to generate complete shapes under such challenging conditions. Although the TreeNet ranks first and achieves the best completion results on all five unknown classes, which shows the strong generalization and robustness evaluated on

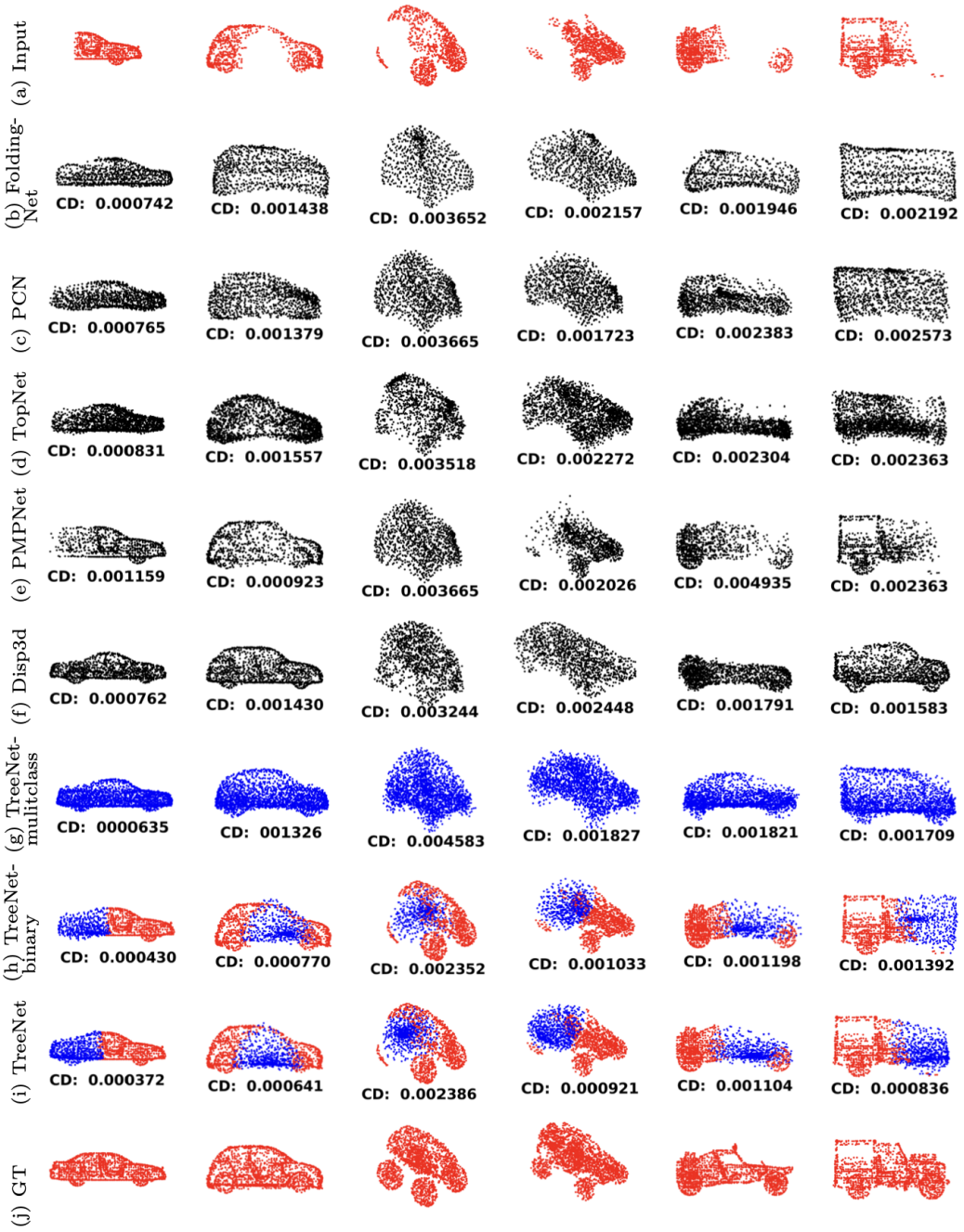


Fig. 4.13 Qualitative completion results tested on a trained class (cars) with unknown shapes.

Table 4.9 The number of network parameters in each method. '-' means the number of network parameters does not increase when the number of training classes increases.

Methods	Network Parameters (trained on 8 classes)	Network Parameters (trained on 50 classes)
FoldingNet [38]	2,402,054	-
PCN [39]	5,286,659	-
TopNet [40]	9,965,117	-
PMPNet [42]	5,435,163	-
Disp3d [41]	100,318,976	-
TreeNet-multiclass	15,515,904	59,599,104
TreeNet-binary	22,869,248	-
TreeNet	30,216,448	74,299,648

unknown classes, all networks fail to generate detailed information in missing areas, as shown in Figure 4.12 and Table 4.7. The missing points generated from the TreeNet are randomly distributed in the missing area, which loses finely detailed information. One possible reason could be that the current loss function measures geometrical features based on the global shape information, which lacks local feature information, thus, resulting in insufficient features for fine details locally. This limitation will be addressed in future work.

One limitation of the TreeNet-multiclass and TreeNet is that the model size is linearly correlated with the number of training classes, as shown in Table 4.9. However, the number of parameters is still much smaller than that of Disp3d [41]. It is worth noting that the number of parameters in the TreeNet-binary does not increase when the number of training classes increases, which is similar to the state-of-the-art networks (FoldingNet [38], PCN [39], TopNet [40],

PMPNet [42] and Disp3d [41]). Although the number of parameters of the TreeNet-multiclass and TreeNet is around 3 (8 classes) to 10 (50 classes) times bigger than that of FoldingNet [38], PCN [39], TopNet [40] and PMPNet [42], the result of TreeNet is 40.28% better on 8 classes than the next best method TopNet [40], and 44.76% better on 50 classes than TopNet [40]. This limitation will be addressed in the feature work.

4.5 Evaluations on Endoscopic Data

In this section, TreeNet-multiclass, TreeNet-binary and TreeNet are evaluated in the proposed computational framework (Chapter 3) for endoscopic scenes as an additional application. TreeNet-multiclass, TreeNet-binary and TreeNet are trained and tested on five endoscopic 3D point cloud datasets generated in Section 3.7.1.2.

For each group of endoscopic 3D point cloud datasets, 90% of 3D point clouds are randomly selected as the training data and the remaining 10% as the testing data. To evaluate the trained TreeNet-multiclass, TreeNet-binary and TreeNet models on partial 3D point clouds, the remaining 10% of testing data is used to create partial 3D point clouds with different missing rates. First, a point from each testing 3D point cloud with the size of $N \times 3$ is randomly selected, where N is the total number of points in a 3D point cloud. Second, the nearest $N * delete\_rate$ points around that selected point are deleted to create partial 3D point clouds with different missing rates, where $delete\_rate$ is the rate of deletion, i.e., 0.3, 0.5, etc. Third, each partial 3D point cloud is randomly sub-sampled to 4096 points. Finally, for each class of testing datasets, six groups

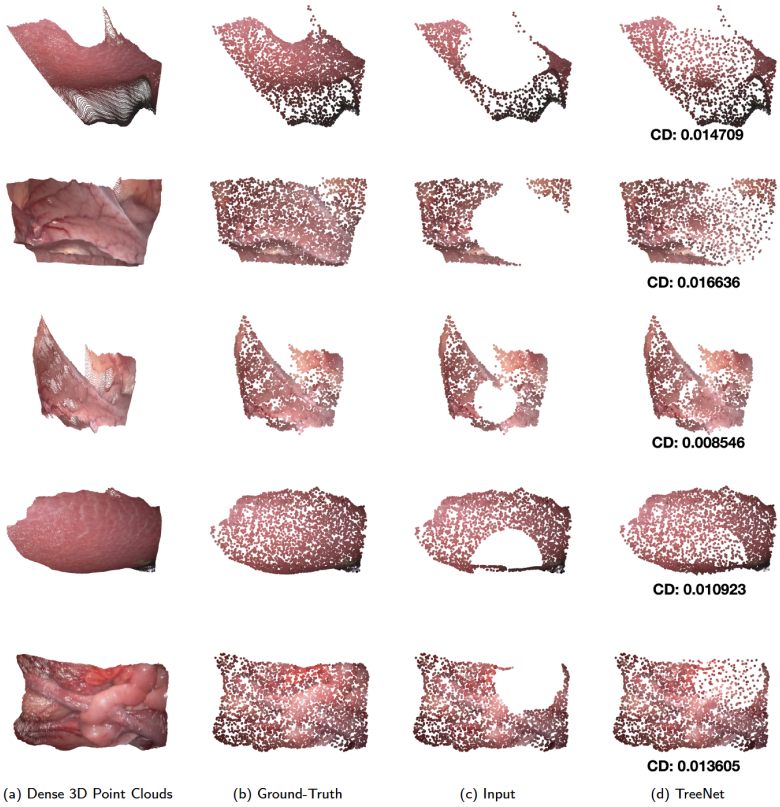


Fig. 4.14 3D point cloud completion of endoscopic datasets: (a) Original 3D point clouds in endoscopic datasets; (b) Ground-truth; (c) Partial 3D Point cloud; (d) Completion results from TreeNet.

of partial 3D point clouds testing data are generated with various missing rates of [25%, 30%, 35%, 40%, 45%, 50%]. The examples of partial 3D point clouds with different missing rates are shown in Figure 4.14.

The average chamfer distances of TreeNet-multiclass, TreeNet-binary and TreeNet are shown in Table 4.10, with qualitative results in Figure 4.14, which indicates that the less missing data, the more accurate the completion results. TreeNet outperforms TreeNet-multiclass and TreeNet-binary on partial 3D point cloud datasets with missing rate ranges from 25% to 50%. The K-nearest neighbour

Table 4.10 Quantitative comparison of the TreeNet-multiclass, TreeNet-binary and TreeNet on five endoscopic 3D point cloud testing datasets.

Missing Rates	TreeNet-multiclass	TreeNet-binary	TreeNet
25%	0.038077	0.019125	**0.017576**
30%	0.038229	0.020286	**0.018452**
35%	0.038606	0.022137	**0.020070**
40%	0.039473	0.024176	**0.021887**
45%	0.040067	0.026772	**0.024355**
50%	0.041460	0.030560	**0.027160**

(KNN) is calculated to find the corresponding points between the ground-truth 3D point cloud and the generated 3D point cloud, where K is 1. After finding the nearest neighbour between them, the colour of each point is extracted from the ground truth and duplicated to the corresponding point in the generated 3D point cloud, as shown in Figure 4.14. This experiment shows the effectiveness of TreeNet for medical data.

4.6 Conclusion

In this chapter, TreeNet-multiclass, TreeNet-binary and TreeNet have been proposed for 3D point cloud completion. The proposed networks can produce high-quality 3D point clouds on multi-class datasets for point cloud completion, and these novel network structures outperform the state-of-the-art learning-based methods in terms of the quality of the 3D point cloud completion results on trained and unknown classes. TreeNet-multiclass is for multi-class training and assigns a specific class of the 3D completion task to each

sub-tree, while TreeNet-binary preserves the original partial input and generates points in missing areas. three forward propagation methods are proposed to train TreeNet-multiclass, TreeNet-binary and TreeNet separately. The proposed models achieve high-quality completion results and show remarkable generalization and robustness to unknown classes that are not trained. The proposed networks also show effectiveness in endoscopic scenes.

Chapter 5

Iterative BTreeNet: Unsupervised Learning for Large and Dense 3D Point Cloud Registration

In this chapter, a novel unsupervised deep learning network - Iterative Binary Tree Network (IBTreeNet) is proposed to continuously improve the registration accuracy for large and dense 3D point clouds, which learns features for the rotation separately from the translation and avoids the interference between the estimations of rotation and translation in one single matrix. Experimental results show that BTreeNet and IBTreeNet outperform six state-of-the-art learning-based and three traditional methods on clean, partial and noisy point clouds and also exhibit remarkable generalization and robustness to unseen large and dense scenes that have been never trained.

5.1 Introduction

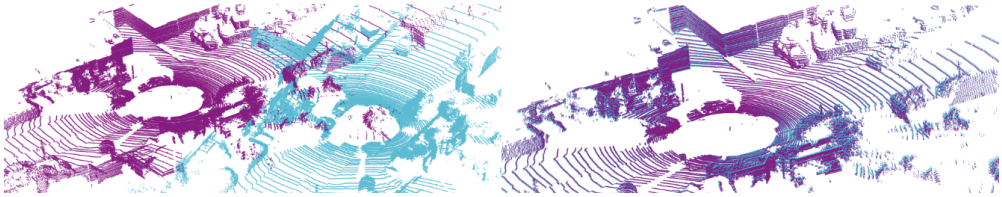
Point cloud rigid registration is a task that aligns two point clouds, captured by various sensor technologies (i.e. laser and RGB-D scanners and depth cameras), by estimating the rigid transformation between them. Point cloud registration is a well-known problem and has been used in many computer vision applications, for example, 3D reconstruction [116] [117] [25] and localization [118] [58] [119].

Traditional methods [43] [44] [45] have considered 3D point cloud registration as an optimization problem and they are sensitive to the initialization of 3D point clouds and can be computationally expensive and time-consuming. Deep learning-based approaches learn features from the neural networks and estimate the transformation matrix for the alignment. However, the state-of-the-art learning-based methods, PointNetLK [46], DCP [47], RPM [48], FMR [50], DeepGMR [51] and RGM [49] share three common problems. First, the ground-truth transformation matrix or point-to-point correspondences are used to supervise the training process in PointNetLK [46], DCP [47], RPM [48], DeepGMR [51] and RGM [49]. Second, these methods [46] [47] [48] [51] [49] perform poorly and show the poor generalization abilities on partial 3D point clouds without training in this scenario (Section 5.3.4). Third, these methods [46] [47] [48] [50] [49] often perform poorly in dealing with large and dense scenes and shapes that are not trained, resulting in poor generalization abilities (see Section 5.3.6).

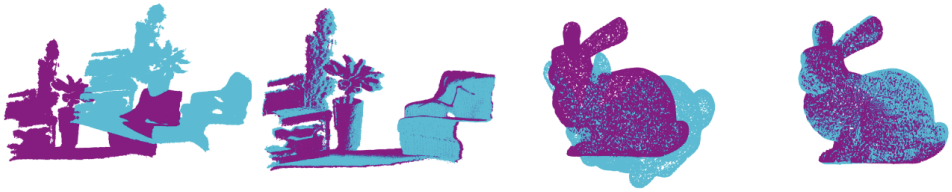
In this chapter, a novel unsupervised deep learning network - *Binary Tree Network (BTreeNet)* is proposed. The BTreeNet consists of a novel forward propagation, which learns features for the

rotation separately from the translation and avoids the interference between the estimations of rotation and translation in one single matrix. Specifically, the root node of the tree is a global feature vector generated from a PointNet-based encoder [1], and the BTreeNet follows a binary tree structure that has a left sub-tree and a right sub-tree. The left sub-tree learns features for rotation and the right sub-tree learns features for translation. The left leaf node estimates the rotation matrix and the right leaf node estimates the translation matrix. To continuously improve the registration accuracy between two 3D point clouds, an Iterative Binary Tree Network (IBTreeNet) is then proposed, which iteratively rotates and translates the registration results of BTreeNet to the target 3D point cloud through the reuse of the trained IBTreeNet model. Note that IBTreeNet has an identical architecture to BTreeNet, but it is trained based on the registration results of a trained BTreeNet model. The objective of IBTreeNet is to extract features of the rotated and translated 3D point cloud from the BTreeNet for the next alignment iteration. Once trained, IBTreeNet can be used repeatedly and iteratively to improve the registration accuracy between two 3D point clouds. The chamfer distance and the Earth Mover's Distance are adopted as the loss function for unsupervised learning. The proposed IBTreeNet exhibits remarkable generalization and robustness to unseen large scenes and shapes that are never trained, as shown in Figure 5.1. This generalization and robustness performance can be attributed to the proposed forward propagation that avoids the interference between the feature extraction of rotation and translation.

The registration results of the BTreeNet and IBTreeNet have been compared with traditional methods [43] [44] [45] and state-



(a) Input unseen outdoor scenes (b) IBTreeNet (iter. 8)



(c) Input unseen indoor scenes (d) IBTreeNet (iter. 7) (e) Input unseen shapes (f) IBTreeNet (iter. 5)

Fig. 5.1 3D point cloud registration on the unseen KITTI, 3DMatch and Stanford Bunny datasets that are not trained. IBTreeNet can iteratively align the source 3D point clouds to the template, even though it was not trained on these scenes and shapes. Refer to Section 5.3.6 for more details.

of-the-art learning-based methods [46] [47] [48] [50] [51] [49]. The comparison experiments are evaluated on testing datasets, including clear data (Section 5.3.3), partially visible data (Section 5.3.4), data with Gaussian noise (Section 5.3.5) and data with large rotations (Section 5.3.7). Moreover, the comparison experiments are also tested on unseen scenes and shapes that are not trained to evaluate the generalization and robustness of each method (Section 5.3.6).

5.2 Methods

Let P_T and P_S define the target and the source 3D point clouds, respectively, where each point in a 3D point cloud is defined as $P_i = (x, y, z)$. The purpose of the proposed method is to estimate the rigid transformation that best rotates and translates P_S to P_T. The

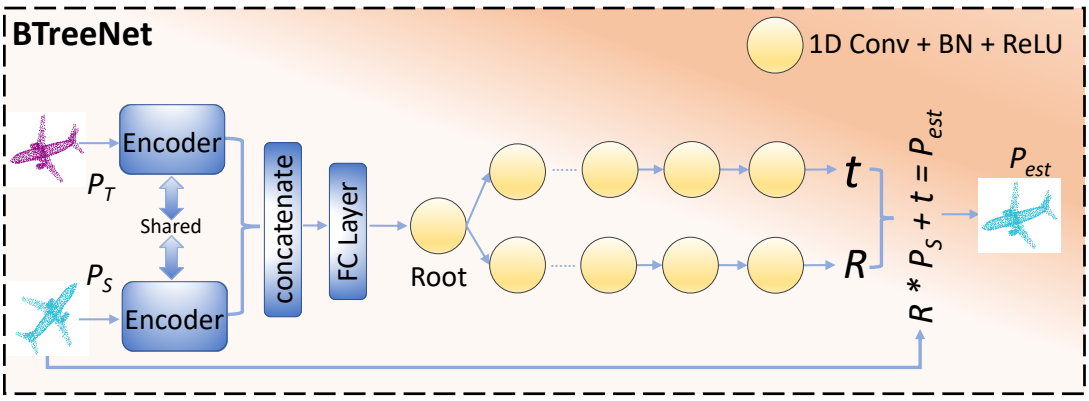


Fig. 5.2 BTreeNet architecture. The source and the target 3D point clouds are given as input through a shared PointNet-based encoder. The global features from the encoder are concatenated and provided as input to a fully connected layer to achieve the root node of the binary tree. The binary tree learns features for rotation separately from the translation through several group layers of 1D Conv, BN and ReLU to generate the rotation matrix separately from the translation matrix to align two 3D point clouds.

rigid transformation includes a rotation matrix R and a translation matrix t, where $R \in SO(3)$ and $t \in \mathbb{R}^3$. BTreeNet (Section 5.2.1), as shown in Figure 5.2, is used to estimate R and t that align P_T and P_S. An iterative BTreeNet (IBTreeNet) (Section 5.2.2), as shown in Figures 5.3 and 5.4, is to iteratively improve the registration accuracy between P_T and P_S. The BTreeNet and IBTreeNet are trained using a chamfer distance and an Earth Mover's Distance as loss function (Section 5.2.3) for unsupervised learning.

5.2.1 BTreeNet

The novelty of the BTreeNet, as shown in Figure 5.2, compared to the latest state-of-the-art learning-based methods [46] [47] [48] [50] [51] [49] is the hierarchical binary tree-based forward propagation that leans features for the rotation separately from the translation and estimates the rotation matrix separately from the translation ma-

trix. Specifically, in these state-of-the-art learning-based methods [46] [47] [48] [50] [51] [49], the combination of rotation and translation in the output layer makes the network deal with two tasks (e.g. estimating the rotation and the translation) simultaneously. Thus, the two tasks interfere with each other on feature learning based on all different initial rotations along any arbitrary axes and any random translations. As a result, these methods converge to a local optimum, which leads to lower precision of registration results.

Following the definition of state-of-the-art methods [46] [47] [48] [50] [51] [49] for 3D point cloud registration, let $P_T \in \mathbb{R}^{N\times 3}$ and $P_S \in \mathbb{R}^{N\times 3}$ are two rigid 3D point clouds that need to be aligned. Both P_T and P_S are given as input to a shared PointNet-based encoder that consists of several group layers of 1D Conv, BN and ReLU and a symmetric max-pooling function at the end to extract global features. The combination of 1D Conv, BN and ReLU extract point-wise features $f_T \in \mathbb{R}^{N\times d}$ and $f_S \in \mathbb{R}^{N\times d}$, where d is the dimension for point-wise features. Weights and biases are shared in feature extraction models for P_T and P_S. The symmetric max-pooling function aggregates point-wise features f_T and f_S to extract the global features $F_T \in \mathbb{R}^{1\times d}$ and $F_S \in \mathbb{R}^{1\times d}$, as defined in Equation 5.1.

$$F = \underset{p_i \in P}{MAX} \{h(p_1), ..., h(p_n)\} \, (i = 1, ..., n) \tag{5.1}$$

where F represents either F_T or F_S. p_i is a point in either P_T or P_S. h is the combination of 1D Conv, BN and ReLU layers. The MAX represents the max-pooling that returns a new vector of the element-wise maximum and guarantees that the input 3D point clouds are invariant to any permutations.

The state-of-the-art learning-based methods [46] [47] [48] [50] [51] [49] adopt the standard forward propagation, as defined in Equation 5.2, to learn the features of rotation and translation from the global features F in the same neuron at each layer. The final output in these methods is a transformation matrix with a size of 7, where the first three output values represent the translation matrix and the last four values represent the rotation quaternion.

$$z_i^{(l+1)} = \boldsymbol{w}_i^{(l+1)} y_i^{(l)} + \boldsymbol{b}_i^{(l+1)}; \qquad y_i^{(l+1)} = f(z_i^{(l+1)}) \tag{5.2}$$

where l indexes the hidden layer and i indexes the hidden neuron in each layer. $y_i^{(l)}$ is the i-th features of P_T and P_S at the layer l. $\boldsymbol{w}_i^{(l+1)}$ and $\boldsymbol{b}_i^{(l+1)}$ denote the i-th weight and bias at layer $l+1$. $z_i^{(l+1)}$ denotes the i-th feature vector of inputs at the layer $l+1$. $f(\cdot)$ is an activation function ReLU.

Unlike PointNetLK [46], DCP [47], RPM [48], FMR [50], DeepGMR [51] and RGM [49], the proposed novel forward propagation, as defined in Equation 5.3, aims to learn features for the rotation separately from the translation and avoids the interference of feature extraction between them. Thus, a hierarchical binary tree-based structure is proposed for the novel forward propagation. The global features of $F_T \in \mathbb{R}^{1\times d}$ and $F_S \in \mathbb{R}^{1\times d}$ are concatenated and given as an input to a fully connected layer to achieve the root node $F_{root} \in \mathbb{R}^{2\times d}$ of the binary tree. The binary tree has two sub-trees,

including left and right sub-trees.

$$
\begin{aligned}
&F_{lst} = F_{root_{idx}}(idx = 0, ..., d); \quad F_{rst} = F_{root_{idx}}(idx = d, ..., 2d) \\
&z_{i\_lst}^{(l+1)} = \boldsymbol{w}_{i\_lst}^{(l+1)} F_{lst}^{(l)} + \boldsymbol{b}_{i\_lst}^{(l+1)}; \quad y_{i\_lst}^{(l+1)} = f(z_{i\_lst}^{(l+1)}) \\
&z_{i\_rst}^{(l+1)} = \boldsymbol{w}_{i\_rst}^{(l+1)} F_{rst}^{(l)} + \boldsymbol{b}_{i\_rst}^{(l+1)}; \quad y_{i\_rst}^{(l+1)} = f(z_{i\_rst}^{(l+1)})
\end{aligned} \tag{5.3}
$$

where $F_{lst} \in \mathbb{R}^{1\times d}$ and $F_{rst} \in \mathbb{R}^{1\times d}$ represent the features are given as input to the left sub-tree and right sub-tree, respectively. l indexes the hidden layer of the binary tree and i indexes the hidden neuron in each layer. $\boldsymbol{w}_{i\_lst}^{(l+1)}$ and $\boldsymbol{b}_{i\_lst}^{(l+1)}$ denote the i-th weight and bias for left sub-tree at layer $l+1$, and $\boldsymbol{w}_{i\_rst}^{(l+1)}$ and $\boldsymbol{b}_{i\_rst}^{(l+1)}$ denote the i-th weight and bias for right sub-tree at layer $l+1$. $z_{i\_lst}^{(l+1)}$ denotes the i-th feature vector of inputs for left sub-tree at the layer $l+1$, and $z_{i\_rst}^{(l+1)}$ denotes the i-th feature vector of inputs for right sub-tree at the layer $l+1$. $f(\cdot)$ is any activation function, e.g. ReLu. $y_{i\_lst}^{(l+1)}$ denotes the i-th feature vector of left sub-tree outputs at the layer $l+1$, and $y_{i\_rst}^{(l+1)}$ denotes the i-th feature vector of right sub-tree outputs at the layer $l+1$.

The final output in left sub-tree $y_{i\_lst}^{(l+1)}$ is a rotation quaternion q with the size of 1×4, and the final output in right sub-tree $y_{i\_rst}^{(l+1)}$ is a translation matrix t with the size of 1×3. The rotation quaternion q, defined by Equation 5.4, is transformed into a rotation matrix R with the size of 3×3, as defined in Equation 5.5. Finally, the P_S is rotated and translated using $P_{est} = R * P_S + t$.

$$
\begin{aligned}
&q = [q_0, q_1, q_2, q_3]^T \\
&q_0 = cos\frac{\theta}{2}; q_1 = n_x sin\frac{\theta}{2}; q_2 = n_y sin\frac{\theta}{2}; q_3 = n_z sin\frac{\theta}{2} \\
&n = [n_x, n_y, n_z]^T
\end{aligned} \tag{5.4}
$$

where θ is the rotation angle (i.e. 45 degrees) and n is the axis of rotation.

$$R = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix} \tag{5.5}$$

5.2.2 Iterative BTreeNet

Although three random Euler angles and translations on each axis are sampled and applied to source point clouds during the training, networks still cannot extract the generalized and effective features for every pose of point cloud pairs. As shown in Figure 5.10, eight iterations have been applied for all networks to reuse their trained models repeatedly for further alignments. These networks cannot achieve accurate registration in the following iterations once it fails to align two 3D point clouds in the first iteration, which indicates that the trained models cannot extract the generalized and effective features of P_T and P_{est} for the next alignment. Refer to Section 5.3.6.1 for more details. Thus, further alignments are needed to improve registration accuracy.

To extract effective features of P_T and P_{est} for further alignment and continuously improve the registration accuracy in the following iterations, an iterative BTreeNet (IBTreeNet) is trained based on the result of BTreeNet, as shown in Figure 5.3. The architecture of IBTreeNet is the same as BTreeNet, but they are trained under different conditions. The differences between BTreeNet and IBTreeNet can be divided into two parts, including the training and testing processes. During the training, IBTreeNet is trained

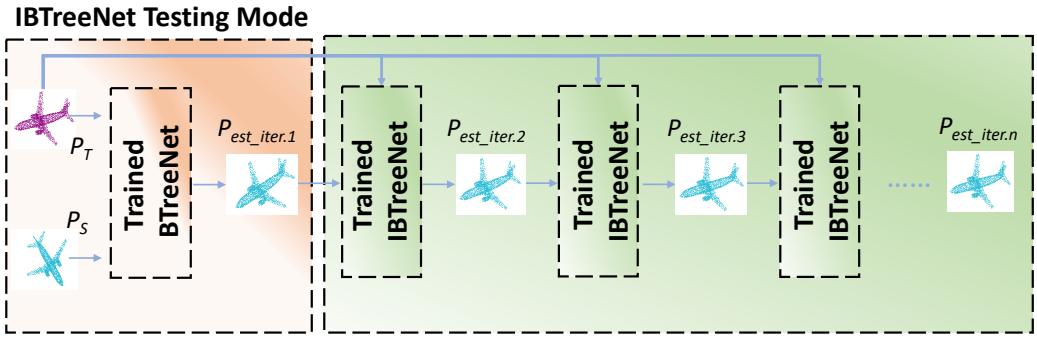


Fig. 5.3 Iterative BTreeNet training mode. The BTreeNet needs to be completely trained before training IBTreeNet. The input 3D point clouds are the P_{est} from the trained BTreeNet and the P_T. The architecture of IBTreeNet is identical to BTreeNet.

based on the registration results from a pre-trained BTreeNet model. Thus BTreeNet needs to be trained in advance before training IBTreeNet. As shown in Figure 5.3, the first transformation for P_S from the trained BTreeNet model gives a transformed 3D point cloud $P_{est\_iter.1}$. The transformed point cloud $P_{est\_iter.1}$ and the target 3D point cloud P_T are given as input to IBTreeNet that learns features of P_T and $P_{est\_iter.1}$ for the next alignment. Thus, two iterations are adopted in the training process. During the testing, the trained BTreeNet is used for the first iteration, and the trained IBTreeNet can be used repeatedly and iteratively to improve the registration accuracy between two 3D point clouds, as shown in Figure 5.4. Thus, infinite iterations can be adopted during the testing. In practice, four iterations are used for data similar to the training data (Section 5.3.3), and ten iterations are used for unseen datasets that are not trained. (Section 5.3.6).

The objective of IBTreeNet is to extract effective features of the rotated and translated 3D point cloud from the BTreeNet for the next alignment iteration. Once trained, IBTreeNet can be

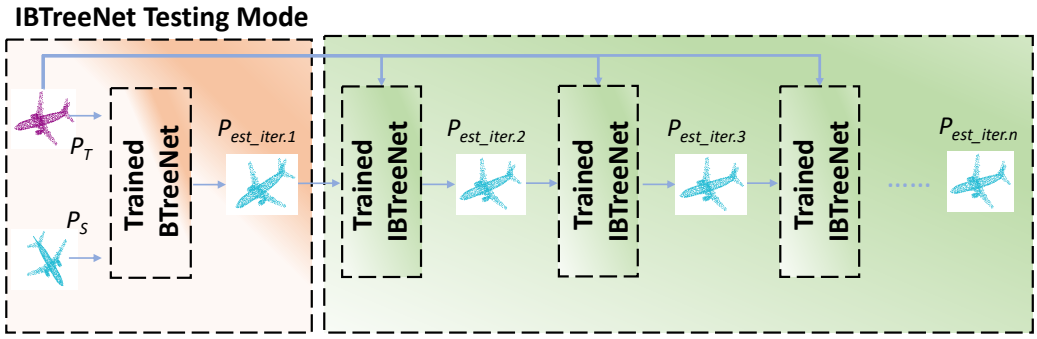


Fig. 5.4 Iterative BTreeNet testing mode. Once BTreeNet and IBTreeNet have been trained, the trained IBTreeNet model can be repeatedly used to iteratively rotates and translates P_S to P_T.

used repeatedly and iteratively to improve the registration accuracy between two 3D point clouds. The IBTreeNet exhibits remarkable generalization and robustness to unseen large scenes and shapes that are never trained (Section 5.3.6). This generalization and robustness performance can be attributed to the proposed forward propagation that avoids the interference between the feature extraction of rotation and translation.

5.2.3 Loss Function

The loss function for 3D point cloud registration measures the difference between the target 3D point cloud P_T and the transformed 3D point cloud P_{est}. The loss is defined to be invariant to any permutation of 3D point clouds in both P_T and P_{est}. Therefore, the chamfer distance [91] is used as the loss function.

The chamfer distance calculates the average nearest point distance between P_T and P_{est} by finding the closest neighbour with $O(nlogn)$ complexity. In addition, P_T and P_{est} can be at the different sizes of 3D point clouds.

The Earth Mover's distance [112] is adopted as the second term in the loss function. The EMD finds a bijection ϕ and minimizes the distance between corresponding points based on ϕ with $O(n^2)$ complexity.

The final loss function, as defined in Equation 5.6, consists of two terms, CD and EMD. Note that both CD and EMD only require the 3D point clouds as input for unsupervised learning.

$$Loss(P_T, P_{est}) = CD(P_T, P_{est}) + EMD(P_T, P_{est}) \tag{5.6}$$

5.2.4 Implementation Details

BTreeNet and IBTreeNet are trained for 300 epochs with a batch size of 32, a learning rate of 0.005, and an adagrad optimizer. The filter sizes for the PointNet-based encoder are $[64, 64, 64, 128, 256, 512]$. The features extracted from P_T and P_S are concatenated and given as input to a fully connected layer to generate the root of the binary tree with the size of $[1 \times 1024]$. The filter sizes for the left sub-tree in each level are $[512, 256, 128, 64, 4]$. The filter sizes for the right sub-tree in each level are $[512, 256, 128, 64, 3]$. The level of the tree, the weights for the loss function and the feature extraction modules are illustrated from the ablation studies in Section 5.3.8. BTreeNet and IBTreeNet are trained with the input size of $N \times 3$, where N can be any number and $N = 1024$ is set during the training. Once trained, the size of the input 3D point cloud is not constrained to 1024. For example, 1024, 20480 and 121210 points have been used when evaluated on the testing dataset (Sections 5.3.3 and 5.3.6).

5.3 Evaluation of Registration Performance

An Nvidia Geforce 2080Ti GPU with 12G memory is used for network training. BTreeNet, IBTreeNet and the state-of-the-art learning-based methods [46] [47] [48] [50] [51] [49] are trained on ModelNet40 training dataset. Based on these trained models, comparison experiments are conducted to compare the BTreeNet and IBTreeNet with the traditional registration methods [43] [44] [45] and state-of-the-art learning-based methods [46] [47] [48] [50] [51] [49]. First, comparison experiments on ModelNet40 clean testing datasets (Section 5.3.3) are conducted. Second, the generalization of these methods is evaluated on partial and noisy point clouds without training them in such scenarios (Sections 5.3.4 and 5.3.5). Third, the generalization and robustness ability of all learning-based methods are evaluated on large and dense unseen KITTI [120], Whu-TLS [121] [122] [123], 3DMatch [124] and Stanford 3D datasets [125] that are not trained (Section 5.3.6). Fourth, comparison experiments are conducted on 3D point clouds with large initial rotations (Section 5.3.7). Finally, the ablation studies are conducted to evaluate the effectiveness of each component in the BTreeNet (Section 5.3.8).

5.3.1 Datasets

ModelNet40 Clean Training Data. For a fair comparison, the ModelNet40 [126] dataset is used as the training dataset following PointNetLK [46], DCP [47], RPM [48], FMR [50], DeepGMR [51] and RGM [49]. ModelNet40 [126] is composed of 12,311 meshed CAD object models from 40 object categories (e.g. bookshelf, chair, desk, etc.). It contains official train/test splits (9,843 for training

and 2,468 for testing). The 2,048 points are uniformly sampled from the CAD 3D mesh surfaces, and then further processed by moving to the origin and scaling into a unit sphere. All networks in the experiments are trained on the official train split in ModelNet40 [126]. Specifically, 2,048 points are sampled as the source point cloud P_S from each object model in the train split and normalize P_S into a unit sphere. The source point clouds P_S are randomly rotated and translated to obtain the target point clouds P_T. For the rotation and translation applied, three Euler angles are randomly sampled in the range of $[0, 45]$ degrees and translations in the range of $[-0.5, 0.5]$ on each axis during the training. Finally, the point orders are shuffled in P_S and P_T, respectively. During the training, all point cloud pairs are clean data without noise and missing data. Thus, exact point-to-point correspondences exist between P_S and P_T in training data.

Testing Data. To evaluate the robustness and generalization ability of a network, the training and the testing sets should be in different scenarios. Thus, all networks are trained only on ModelNet40 Clean Training Data and test them on clean, partial, noisy and unseen point clouds, respectively. The testing data contains seven different groups of 3D point cloud datasets, including *ModelNet40 Clean* (Section 5.3.3), *ModelNet40 Partial* (Section 5.3.4), *ModelNet40 Noise* (Section 5.3.5), *Unseen KITTI* (Section 5.3.6.1), *Unseen Whu-TLS* (Section 5.3.6.2), *Unseen 3DMacth* (Section 5.3.6.3) and *Unseen Shapes* (Section 5.3.6.4).

The ModelNet40 training datasets are sparse 3D objects, whereas the unseen scene testing datasets are large and dense and captured from real-world outdoor and indoor scenes, which contain huge dif-

ferent scales and point distributions from the ModelNet40 training datasets. *Unseen KITTI* is the testing dataset from KITTI odometry LiDAR 3D point cloud datasets. This dataset is large-scale and outdoor scenes from different cities, which are captured with realistic sensor noises. The points are dense for objects and scenes near the LiDAR sensor and the points are very sparse for faraway objects. Thus, the point distribution varies by the distance between objects and the LiDAR sensor. *Unseen Whu-TLS* is also a large-scale, outdoor and real-world scene captured from different 3D scanner systems with different sensor noise and variations in the point density, field of view, clutter and occlusion, which contains similar point distributions because they are all captured by LiDAR sensors. *Unseen 3DMacth* is an RGBD-reconstruction indoor dataset, the points are evenly distributed in the 3D surface, which contains the different point densities and distribution compared with *Unseen KITTI* and *Unseen Whu-TLS*. *Unseen Shapes* dataset consists of the Stanford 3D scanning models from different 3D scanners. The 20,480 points are randomly down-sampled from the 3D meshes. The shapes are totally different from the ModelNet40 Clean Training Data. These four unseen datasets are not trained and have been used to evaluate the generalization ability of each network.

5.3.2 Evaluation Metrics

The registration by computing six evaluation metrics is evaluated. First, the chamfer distance is adopted, to measure how close the two point clouds are brought to each other. Second, the transformation matrix is further evaluated through the Frobenius norm based on

the Special Euclidean Group for the overall registration errors, as defined in Equation 5.7.

$$F(\mathbf{T}) = \|\mathbf{T}_{gt} - \mathbf{T}_{est}\|_F \,; \quad \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{5.7}$$

where $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$.

Finally, the mean isotropic rotation and translation errors (MIE) proposed in RPM [48] and the mean absolute errors (MAE) of rotation and translation proposed in DCP [47] are used, as defined in Equation 5.8.

$$\begin{aligned} &MIE(\mathbf{R}) = \angle(\mathbf{R}_{gt}^{-1}\mathbf{R}_{est}); \quad MIE(\mathbf{t}) = \|\mathbf{t}_{gt} - \mathbf{t}_{est}\|_2 \\ &MAE(\mathbf{R}) = \frac{1}{m}\sum_{i=1}^{m} |\angle(\mathbf{R}_{gt} - \mathbf{R}_{est})| \,; \\ &MAE(\mathbf{t}) = \frac{1}{m}\sum_{i=1}^{m} |\mathbf{t}_{gt} - \mathbf{t}_{est}| \end{aligned} \tag{5.8}$$

where $\mathbf{R}_{gt}$ and $\mathbf{t}_{gt}$ denote the ground-truth rotation and translation, and $\mathbf{R}_{est}$ and $\mathbf{t}_{est}$ represent the estimated rotation and translation. $\angle(\cdot)$ denotes the angle of the rotation matrix in degrees.

5.3.3 ModelNet40 Clean Test

In this experiment, the registration performance is evaluated on *ModelNet40 Clean* testing dataset that contains 2,468 clean point cloud pairs. *ModelNet40 Clean* testing dataset is generated from the official test split in ModelNet40 [126] that includes 2,468 clean point clouds. 1,024 points are randomly sampled and normalized into a unit sphere. To obtain target point clouds during the testing, the random rotations are in the range of $[0, 45]$ degrees and translations

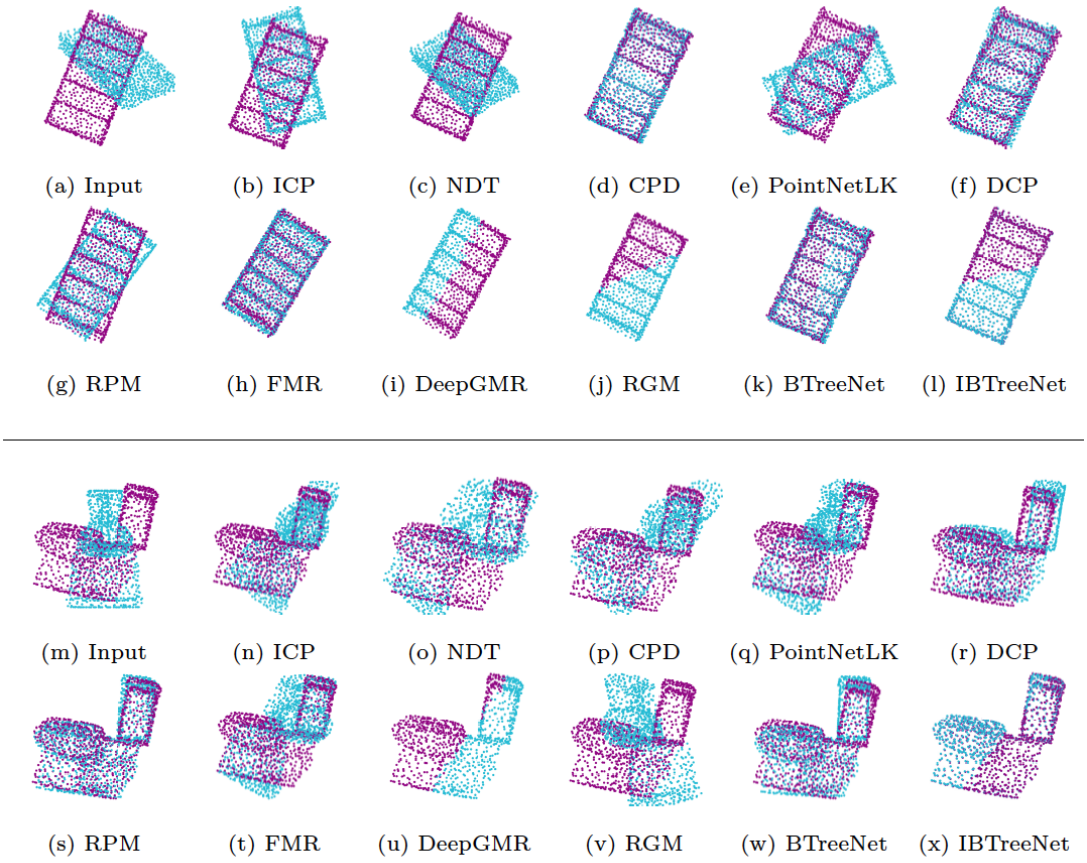


Fig. 5.5 Registration results on two clean 3D point clouds. Each 3D point cloud contains 1,024 points.

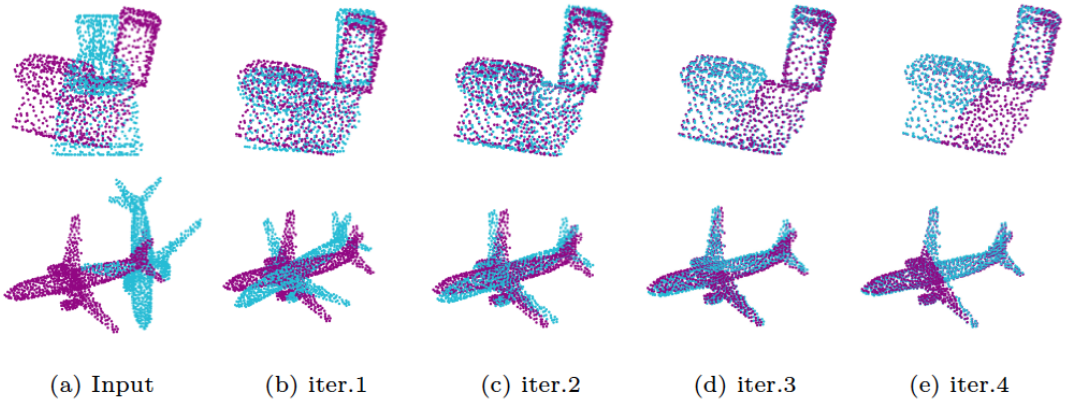


Fig. 5.6 IBTeeNet illustration. (a) Input 3D point clouds. (b) - (e) Iteration 1 to 4.

Table 5.1 Evaluations on the *ModelNet40 Clean* testing dataset with $[0, 45]$ degrees of initial rotations. Bold denotes the top four performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.053823	0.408176	15.926107	0.095590	8.106933	0.048669
NDT	0.060876	0.520965	21.684867	0.043825	10.500771	0.022000
CPD	0.015127	0.090626	3.749195	*0.002207*	1.784223	*0.001123*
PointNetLK	0.016350	0.235991	10.447890	0.013864	5.581651	0.006875
DCP	0.016620	0.071662	2.874782	0.008266	1.469114	0.004150
RPM	**0.000709**	**0.007587**	**0.309407**	**0.002072**	**0.160725**	**0.000947**
FMR	0.014335	0.075151	3.171169	0.002772	1.561003	0.001385
DeepGMR	**0.000104**	**0.003671**	**0.240103**	**0.000024**	**0.115103**	**0.000013**
RGM	$<$ **0.000001**	**0.001464**	**0.057414**	**0.000785**	**0.023415**	**0.000399**
BTreeNet	0.012294	0.065951	2.618611	0.007907	1.379738	0.003988
IBTreeNet	**0.010008**	**0.059417**	**2.346349**	**0.007623**	**1.119935**	**0.003796**

in the range of $[-0.5, 0.5]$ on each axis are applied to the official test split without noise and missing data with exact point-to-point correspondences. All learning-based network models (PointNetLK [46] DCP [47], RPM [48], FMR [50], DeepGMR [51], RGM [49] TreeNet-multiclass, TreeNet-binary and TreeNet) are trained on the same ModelNet40 Clean Training Data.

Figure 5.6 illustrates the iteration process of the IBTreeNet, which shows that the IBTreeNet can iteratively improve the registration accuracy between two 3D point clouds. Note that the first iteration (b) in Figure 5.6 is the result of BTreeNet, and other iterations (c), (d) and (e) are the results of IBTreeNet.

The average registration errors of BTreeNet and IBTreeNet against three traditional methods [43] [44] [45] and six state-of-the-art learning-based methods [46] [47] [48] [50] [51] [49] are shown in

Table 5.1 with qualitative results in Figure 5.5. Lower average errors indicate lower registration errors. Although the proposed networks do not achieve the best registration performance in *ModelNet40 Clean* testing set, BTreeNet and IBTreeNet still outperform three traditional methods [43] [44] [45] and three learning-based network models [46] [47] [50]. DeepGMR [51], RGM [49] and RPM [48] achieve top measures on clean data. However, they perform poorly on partial and noisy 3D point clouds without training them in such scenarios, and detailed information is described in Sections 5.3.4 and 5.3.5.

Furthermore, the number of parameters, supervision method, iterations and computational time are also essential for each network. Since the traditional algorithms [43] [44] [45] do not provide the GPU acceleration, a 32GB memory card is used with an Intel i7-9700 3.00GHz CPU for a fair comparison of computational time. The computational time is collected on two testing datasets with different numbers of points, including *ModelNet40 Clean* (1,024 points) and *Unseen KITTI* (20,480 points) (Section 5.3.6.1) testing datasets. As shown in Table 5.2, PointNetLK [46] DCP [47], RPM [48], DeepGMR [51] and RGM [49] are fully-supervised, which needs ground-truth rotation and translation as the supervision signal while FMR [50] and the proposed BTreeNet and IBTreeNet are unsupervised, thus, can largely reduce the training cost.

For sparse 3D point clouds with 1,024 points, DCP [47], RPM [48] and RGM [49] require a longer computational time on a single instance, respectively. The proposed BTreeNet only takes around 0.1 seconds and ranks first in computational time. For dense 3D point clouds with 20,480 points, CPD [45] and RPM [48] require the longest

Table 5.2 Quantitative comparison on parameters, number of iterations and computational time. OOM means 32GB memory with an Intel i7 CPU or a 12GB NVIDIA GPU out of memory with a forward pass on a single instance.

Methods	Supervision	Network Parameters	Iterations	Computational Time on CPU(s)			
				(1,024 points)		(20,480 points)	
				all iter.	per iter.	all iter.	per iter.
ICP	-	-	20	1.450	0.073	1.436	0.072
NDT	-	-	30	0.230	0.008	1.539	0.051
CPD	-	-	20	0.700	0.035	259.712	12.986
PointNetLK	Yes	151,686	10	0.797	0.080	2.895	0.290
DCP	Yes	5,568,905	1	6.414	6.414	22.200	22.200
RPM	Yes	905,154	5	4.881	0.976	264.246	52.849
FMR	No	3,440,006	10	0.168	0.017	9.193	0.919
DeepGMR	Yes	1,527,440	1	1.532	1.532	2.282	2.282
RGM	Yes	25,000,836	2	2.685	1.343	OOM	OOM
BTreeNet	No	1,483,463	1	0.141	0.141	0.806	0.806
IBTreeNet	No	1,483,463	4	1.114	0.279	4.225	1.056

computational time on dense point clouds and take approximately 4.5 minutes for a single instance. The proposed BTreeNet requires less than 1 second on dense 3D point clouds with 20,480 points, ranking first in computational time. The proposed IBTreeNet requires 1.056 seconds for one iteration and also takes less time than others.

RMG [49] contains the most significant number of network parameters and has the out-of-memory issue on processing dense 3D point clouds, which requires a high computational cost and relies on powerful commodity GPU or CPU processors. DCP-v2 [47] contains the second largest number of network parameters, which requires a high computational cost and has the out-of-memory issue on dense 3D point clouds. Thus, DCP-v1 [47] is used to evaluate dense point clouds with 20,480 points and DCP-v2 [47] on sparse point clouds

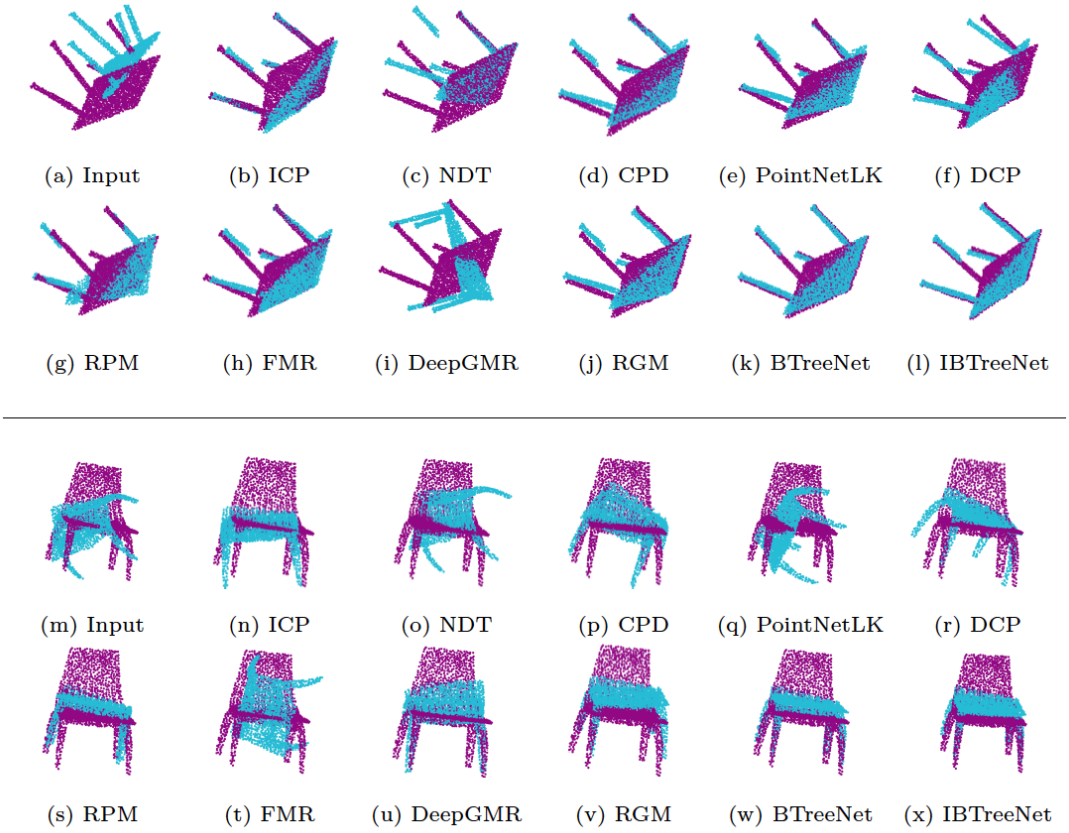


Fig. 5.7 Registration results on partial 3D point clouds. Each 3D point cloud contains 1,024 points.

with 1,024 points. Traditional methods [43] [44] [45] take a maximum of 20 or 30 iterations for 3D point cloud registration whereas learning-based methods PointNetLK [46] and FMR [50] require maximum 10 iterations. RPM [48] and RGM [49] need 5 and 2 iterations during the testing. DCP [47], DeepGMR [51] and the BTreeNet only require 1 iteration (see Table 5.2).

5.3.4 Partial Visibility

To generate partial point clouds that do not fully overlap in extent, the protocol in RPM [48] and RGM [49] are followed and the *Mod-*

Table 5.3 Evaluations on ModelNet40 partial 3D point clouds with 30% missing data. All networks are trained on Modelnet40 Clean Training Data without missing and noisy points. Bold denotes the top-performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.068601	0.559043	21.683043	0.154110	11.064304	0.077339
NDT	0.078709	0.625850	25.139421	0.135835	12.272863	0.065737
CPD	0.064756	0.435410	15.968742	0.162647	8.067676	0.081858
PNLK	0.098572	0.919244	40.113300	0.181374	21.456369	0.091555
DCP	0.102747	1.043511	46.917641	0.001283	25.693749	0.000646
RPM	0.126092	1.284177	60.215692	0.207321	33.902779	0.103731
FMR	0.058666	0.413444	15.552787	0.139811	8.057808	0.069266
DeepGMR	0.149778	2.022872	104.408779	0.202822	60.829105	0.101588
RGM	0.061925	0.380452	12.045048	0.215026	6.335146	0.215026
BTreeNet	**0.027250**	**0.164416**	**6.679313**	**0.000063**	**3.392343**	**0.000029**
IBTreeNet	**0.025889**	**0.150333**	**6.105696**	**0.000049**	**3.037774**	**0.000024**

elNet40 Partial testing dataset is generated from the *ModelNet40 Clean*, which is more realistic and closer to real-world applications. For each source point cloud in the *ModelNet40 Clean*, a random clipping plane passing through the origin is created and shifted to retain 70% of the points. In *ModelNet40 Partial*, the initial rotations and translations are the same as the *ModelNet40 Clean*, but exact point-to-point correspondences have been broken and do not exist.

Most importantly, to evaluate the robustness and generalization ability of each network, all networks are only trained on ModelNet40 Clean Training Data (Section 5.3.1) without missing points and directly tested on *ModelNet40 Partial*. For a fair comparison, following DCP [47], RPM [48] and RGM [49] and the average registration errors are reported on the group with 30% missing rate in Table 5.3 with qualitative results in Figure 5.7. Three traditional

algorithms [43] [44] [45] outperform the four learning-based methods including PointNetLK [46], DCP [47], RPM [48] and DeepGMR [51]. FMR [50] outperforms the traditional algorithms [43] [44] [45] and other learning-based methods [46] [47] [48] [51] [49]. The BTreeNet and IBTreeNet are significantly more accurate than both traditional and learning-based methods, which indicates remarkable generalization on partial 3D point clouds.

The two possible reasons are analysed for the failure registration of the state-of-the-art learning-based methods [46] [47] [51] [48] [49] on partial 3D point clouds. These methods perform poorly on partial 3D point clouds without training them in this scenario, indicating the poor generalization of each network on partial visibility with broken point-to-point correspondences. A trained network model should extract generalized features for different datasets and scenarios. Otherwise, it can only perform well on a specific dataset or a scenario and needs to be retrained under different conditions. Second, these networks apply the ground-truth transformation labels (rotation and translation) as supervision during the training, which forces the network to output a certain transformation based on a certain pattern of input 3D point cloud pairs. Since the broken point-to-point correspondences have changed the pattern of input 3D point cloud pairs during the testing, the trained network models cannot recognize the changed pattern and generate wrong transformations.

FMR [50] and BTreeNet are all trained in an unsupervised manner and find the global registration on input 3D point cloud pairs without ground-truth transformation, which avoids the limitation of the supervised manner and outperforms other learning-based methods on partial 3D point cloud registration.

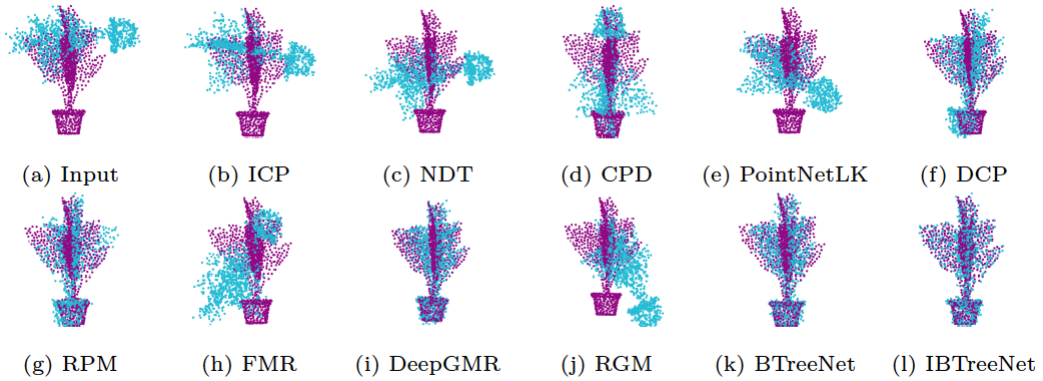


Fig. 5.8 Registration results on 3D point clouds with Gaussian noise. Each 3D point cloud contains 1,024 points.

5.3.5 Gaussian Noise

To evaluate the robustness and generalization ability of each method to noise, the *ModelNet40 Noise* is generated from the *ModelNet40 Clean*. Specifically, noise from Gaussian distribution is sampled for the source 3D point cloud in *ModelNet40 Clean* with 0 mean and a standard deviation of 0.05. In the *ModelNet40 Noise*, the Gaussian noise is added to all source 3D point clouds in the *ModelNet40 Clean*, and these noises destroy the original point-to-point correspondences. The initial rotations and translations are the same as the *ModelNet40 Clean*.

Most importantly, all networks are only trained on ModelNet40 Clean Training Data (Section 5.3.1) without noise and directly tested on *ModelNet40 Noise*. The average registration errors of each method to noise with a standard deviation equal to 0.05 are reported in Table 5.4 with qualitative results shown in Figure 5.8. BTreeNet and IBTreeNet are much more accurate than the state-of-the-art

Table 5.4 Evaluations on Gaussian noise with the standard deviation equals to 0.05. All networks are trained on Modelnet40 Clean Training Data without missing and noisy points. Bold denotes the top-performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.049258	0.413435	16.154294	0.096340	8.241147	0.049071
NDT	0.058998	0.518387	21.541951	0.044982	10.476870	0.022424
CPD	0.016864	0.140918	5.753399	0.006682	2.750101	0.003356
PNLK	0.032994	0.332014	14.352293	0.037295	7.654664	0.018523
DCP	0.050837	0.463393	20.397403	0.001136	10.807378	0.000572
RPM	0.083747	1.079716	51.637915	0.003194	29.808305	0.001602
FMR	0.020769	0.151866	6.166866	0.022762	3.141542	0.011399
DeepGMR	0.067272	0.803346	37.094805	0.006741	22.525591	0.003408
RGM	0.050613	0.586429	29.838333	0.000880	15.885876	0.000443
BTreeNet	**0.013962**	**0.081754**	**3.317700**	**0.000070**	**1.700144**	**0.000034**
IBTreeNet	**0.012318**	**0.069757**	**2.829888**	**0.000060**	**1.389002**	**0.000031**

learning-based methods [46] [47] [48] [50] [51] [49] and traditional methods [43] [44] [45].

Similarly to the results on partial 3D point clouds, DeepGMR [51], RGM [49] and RPM [48] do not provide accurate registration on noisy point clouds if these networks are not trained in this scenario, which indicates the poor robustness and generalization ability of these networks with noise. Traditional method CPD [45] outperforms the state-of-the-art learning-based methods [46] [47] [48] [50] [51] [49] on noisy point clouds, which indicates that CPD [45] can tolerate large noise on point clouds.

Unsupervised learning-based method FMR [50] and IBTreeNet still outperform other supervised learning-based methods [46] [47] [48] [51] [49] on noisy 3D point clouds, which demonstrates the effectiveness of the unsupervised manner on the network generalization

ability. The possible reasons for the failed registration of the state-of-the-art learning-based methods on noisy 3D point clouds are the same as that on partial 3D point clouds (see Section 5.3.4).

5.3.6 Generalization across Unseen Datasets

Several comparison experiments are extensively conducted to evaluate the generalization and robustness capability of the proposed method. All learned models including PointNetLK [46], DCP [47], RPM [48], FMR [50], DeepGMR [51], RGM [49] and ours are trained on the same ModelNet40 Clean Training Data (Section 5.3.1), and directly tested on the completely unseen datasets that are never trained. Note that the ModelNet40 Clean Training Data consists of sparse 3D point clouds (e.g. table, bookshelf and guitar), and the unseen datasets are the large and dense outdoor and indoor scenes captured from various sensors and totally different from the ModelNet40 Clean Training Data. This large domain gap between these datasets poses a significant challenge to the generalization of all learning-based methods.

5.3.6.1 Generalization on Unseen KITTI Scenes

To evaluate the generalization of each network on unseen datasets that are not trained, *Unseen KITTI* testing dataset is generated, including 1,146 KITTI odometry [120] LiDAR 3D point cloud pairs with realistic sensor noise for the evaluation. KITTI odometry [120] is an outdoor dense point cloud dataset acquired by Velodyne-64 3D LiDAR scanners. Every six frames are selected in the sequence of outdoor scans to ensure that the selected 1,460 point clouds can fully

Table 5.5 Evaluations on the unseen KITTI dataset that is not trained. All networks are trained on the Modelnet40 Clean Training Data without missing and noisy points. Bold denotes the top three performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.030695	0.505911	16.626091	0.260118	8.847980	0.130960
NDT	0.030422	0.877822	36.667289	0.102342	16.307743	0.051711
CPD	**0.001883**	**0.086971**	**3.557581**	**0.009284**	**1.241874**	**0.004788**
PNLK	**0.002546**	**0.085499**	**3.672698**	**0.006742**	**2.088179**	**0.003364**
DCP	0.003720	0.181936	3.151685	0.143950	1.624810	0.073790
RPM	0.010139	0.360977	17.587084	0.029334	8.679858	0.014839
FMR	0.008252	0.245331	10.790774	0.031303	7.057602	0.015883
DeepGMR	**0.000039**	**0.000580**	**0.027025**	**0.000036**	**0.011429**	**0.000018**
RGM	0.005711	0.155436	6.258668	0.015307	3.305840	0.007752
BTreeNet	0.006533	0.256744	7.235349	0.144797	3.510368	0.074197
IBTreeNet	0.005452	0.235064	5.982500	0.144624	2.783603	0.074102

represent the scenario and difficulty in this dataset. 20,480 points are sampled from each selected dense 3D point cloud and normalized into a unit sphere. Note that, the number of the down-sampled KITTI 3D point cloud is still 20 times larger than that in the *ModelNet40 Clean* testing dataset. The settings of initial rotations and translations are the same as the *ModelNet40 Clean* testing dataset. Since the target point clouds are rotated and translated from the source point clouds, thus exact point-to-point correspondences exist in *Unseen KITTI*.

In this experiment, all networks are trained on ModelNet40 Clean Training Data (Section 5.3.1) and directly tested on *Unseen KITTI*. RGM [49] requires significant memory and has the out-of-memory issue with a forward pass on a single instance tested on a 32GB memory with an Intel i7-9700 3.00GHz CPU and a 12GB NVIDIA RTX 2080Ti GPU. By this hardware setting, RGM [49] can only

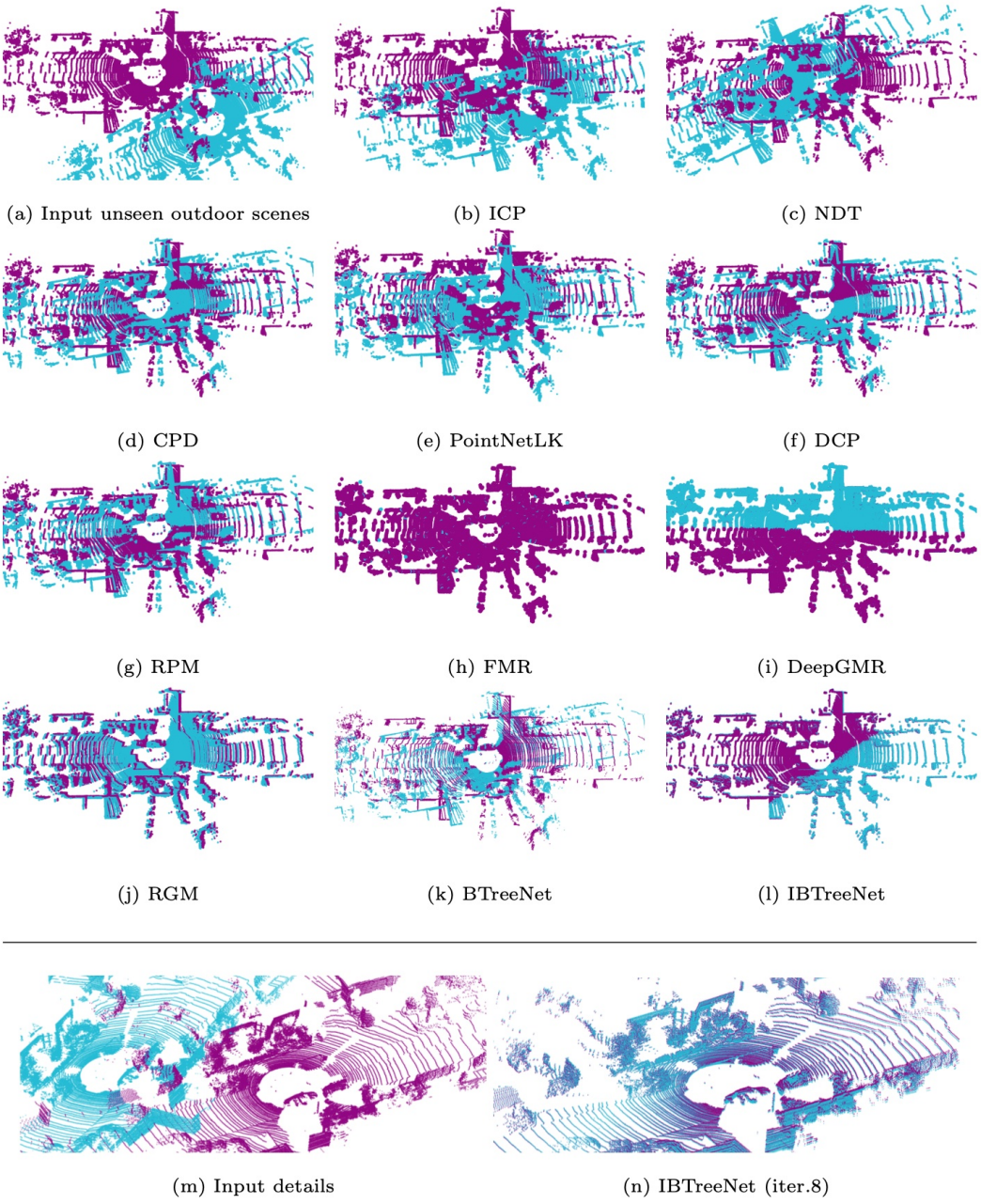


Fig. 5.9 Qualitative registration results on unseen large and dense KITTI LiDAR 3D point clouds. Each 3D point cloud contains 121,210 points for display. (a) Input 3D point clouds. (b) - (d) Registration results of non-learning based methods ICP, NDT and CPD. (e) - (h) Registration results of learning-based methods PointNetLK, DCP, RPM and RGM. (i) Registration results of IBTreeNet. (j) and (k) Details of input and registration results of IBTreeNet.

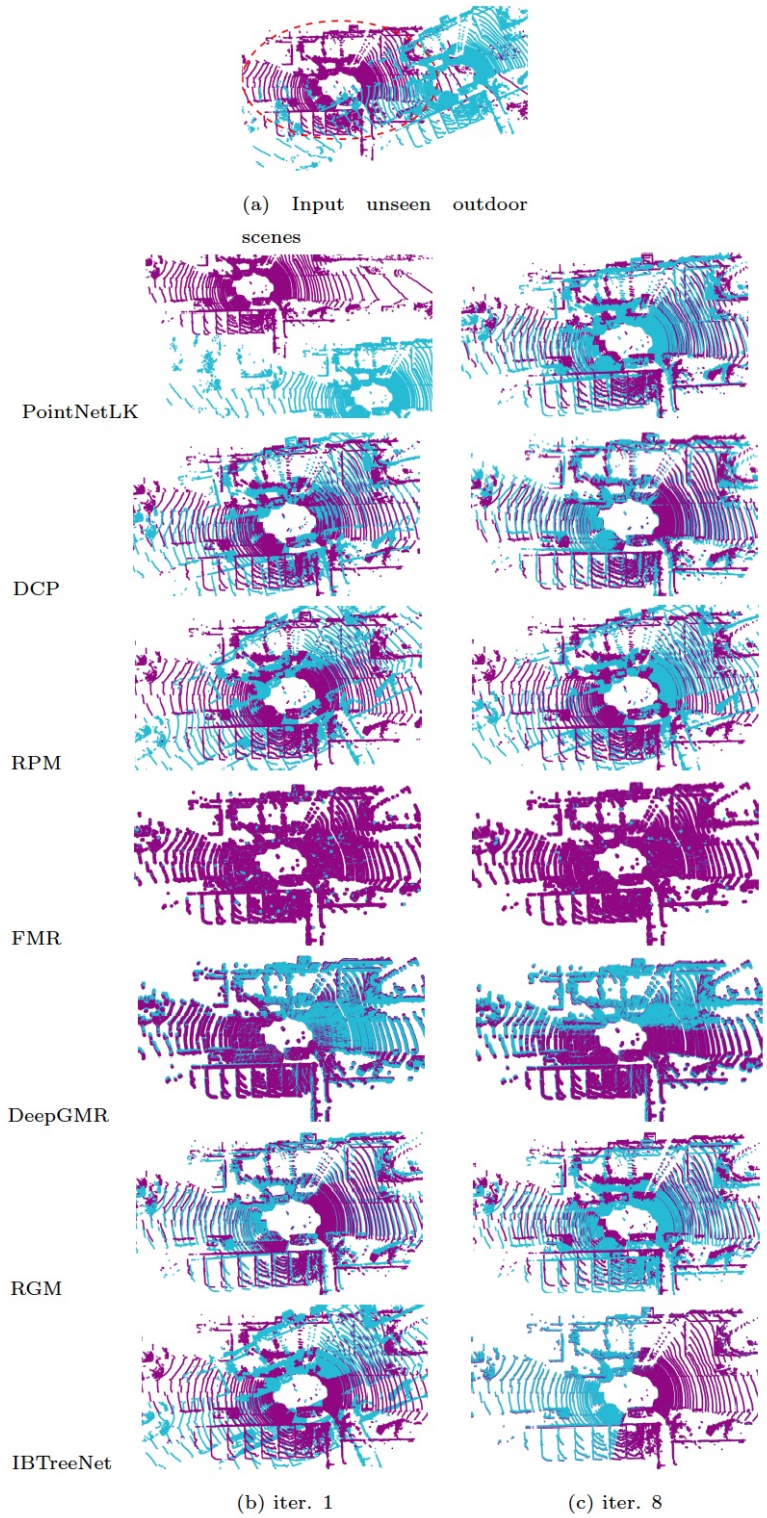


Fig. 5.10 Iteration illustration on unseen KITTI LiDAR 3D point clouds. Each 3D point cloud contains 111,989 points for display.

process approximately 3,078 points. Thus, all point cloud pairs in *Unseen KITTI* are down-sampled to 3,078 points only for RGM [49]. Once the transformation matrix has been obtained from RGM [49], the estimated transformation matrix is applied to the original source point cloud with 20,480 points.

The performance of each method on *Unseen KITTI* is reported in Table 5.5. DeepGMR [51] ranks first in all performing measures and achieves the best registration performance on *Unseen KITTI*. Traditional method CPD [45] outperforms all learning-based methods except DeepGMR [51], which shows the strong generalization of the CPD algorithm. The IBTreeNet outperforms PRM [48], FMR [50], RGM [49], ICP [43] and NDT [44].

Figure 5.9 shows the qualitative registration results of each method on KITTI datasets. Figure 5.10 shows the iteration illustration of each learning-based method. It is worth noting that, the iteration methods proposed in DCP [47], RPM [48] and RGM [49] are limited in improving the registration accuracy, whereas the IBTreeNet iteratively achieves precise registration and shows great generalization and robustness to unseen KITTI LiDAR points.

5.3.6.2 Generalization on Unseen Whu-TLS Scenes

To evaluate the generalization of each network on different unseen datasets that are not trained, *Unseen Whu-TLS* testing dataset is generated, including 1,000 Whu-TLS [121] [122] [123] 3D point cloud pairs with realistic sensor noise for the evaluation. Whu-TLS [121] [122] [123] dataset, captured by terrestrial laser scanners (TLS), is a large-scale 3D point cloud dataset, which includes 10 different 3D point cloud scenes (i.e., subway station, residence, riverbank,

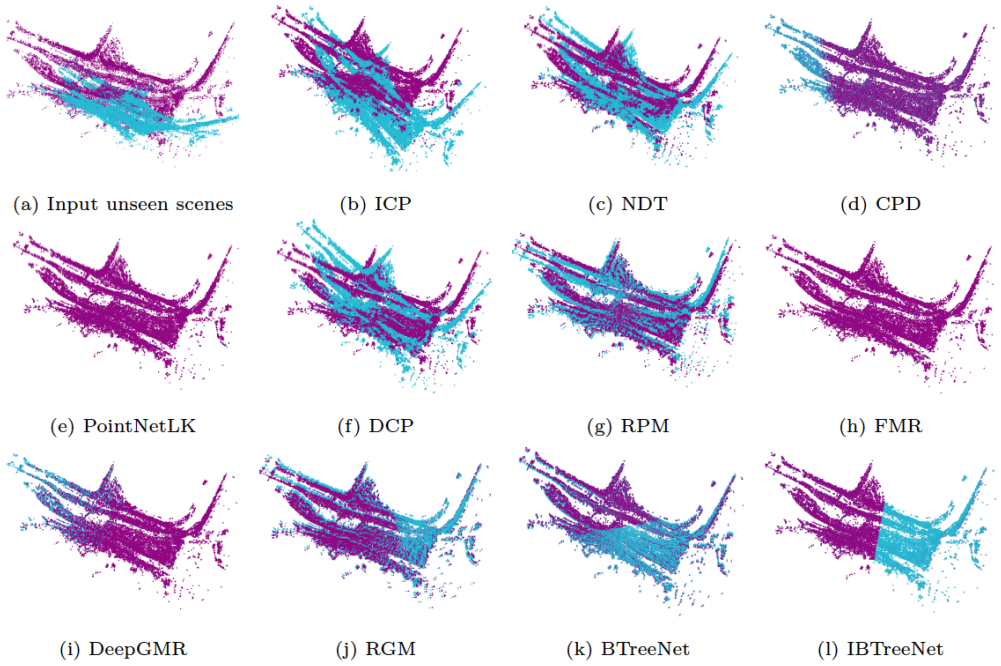


Fig. 5.11 Qualitative registration results on unseen large-scale 3D point clouds. Each 3D point cloud contains 20,480 points.

Table 5.6 Evaluations on the *Unseen Whu-TLS* datasets that are not trained. All networks are trained on Modelnet40 Clean Training Data without missing and noisy points. Bold denotes the top three performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.030002	0.870613	36.191176	0.337518	19.398201	0.170002
NDT	0.048445	0.702454	27.452705	0.165263	13.076433	0.082384
CPD	**0.003510**	**0.057525**	**2.313392**	**0.007674**	**0.888502**	**0.004049**
PNLK	0.009444	0.456522	13.951705	0.278294	7.738035	0.129433
DCP	0.009758	0.351936	9.775516	0.344886	5.153150	0.162643
RPM	0.024656	0.466777	21.402695	0.069478	11.701789	0.034369
FMR	0.022515	0.390340	18.571252	0.062369	11.474063	0.030415
DeepGMR	**0.000128**	**0.022431**	**0.679253**	**0.014806**	**0.399354**	**0.007705**
RGM	0.008434	0.175065	6.893064	0.026624	3.693086	0.013407
BTreeNet	0.012251	0.378394	9.169036	0.277799	4.697491	0.139201
IBTreeNet	**0.008388**	**0.142164**	**6.251942**	**0.277579**	**2.882725**	**0.139073**

etc.) with variations in the point density, clutter and occlusion[1]. The Whu-TLS dataset is challenging because multiple laser scanner systems (i.e., VZ-400, ScanStationC5, Leica HDS6100, etc.) with differences in terms of the measurement range, accuracy and field of view are used to capture the 3D point clouds. For each scene in 10 different scenes, 100 point cloud pairs with the same settings as the *Unseen KITTI* testing dataset are generated.

In this experiment, all networks are still trained on ModelNet40 Clean Training Data (Section 5.3.1) and directly tested on *Unseen Whu-TLS*. As illustrated in Section 5.3.6.1, the same settings for RGM [49] are used.

The performance of each method on *Unseen Whu-TLS* is reported in Table 5.6 with qualitative registration results of each method in Figure 5.11. Learning-based method DeepRGM [51] ranks first in all performing measures, and the traditional method CPD [45] outperforms all learning-based methods except DeepGMR [51], which shows the strong generalization of the DeepRGM [51] and CPD [45] on *Unseen Whu-TLS*. The IBTreeNet outperforms RGM [49], PointNetLK [46], DCP [47], FMR [50], RPM [48], ICP [43] and NDT [44] based on the overall registration errors.

5.3.6.3 Generalization on Unseen Indoor Scenes

The comparison experiments are then conducted on unseen indoor scenes and generate *Unseen 3DMatch* testing dataset, including 433 3DMatch [124] 3D point cloud pairs with the same settings as the *Unseen KITTI* testing dataset. 3DMatch [124] is an RGB-D reconstruction dataset, which contains 433 reconstructed dense 3D

[1]Whu-TLS generates 11 different scenes and 10 scenes are publicly available for researches.

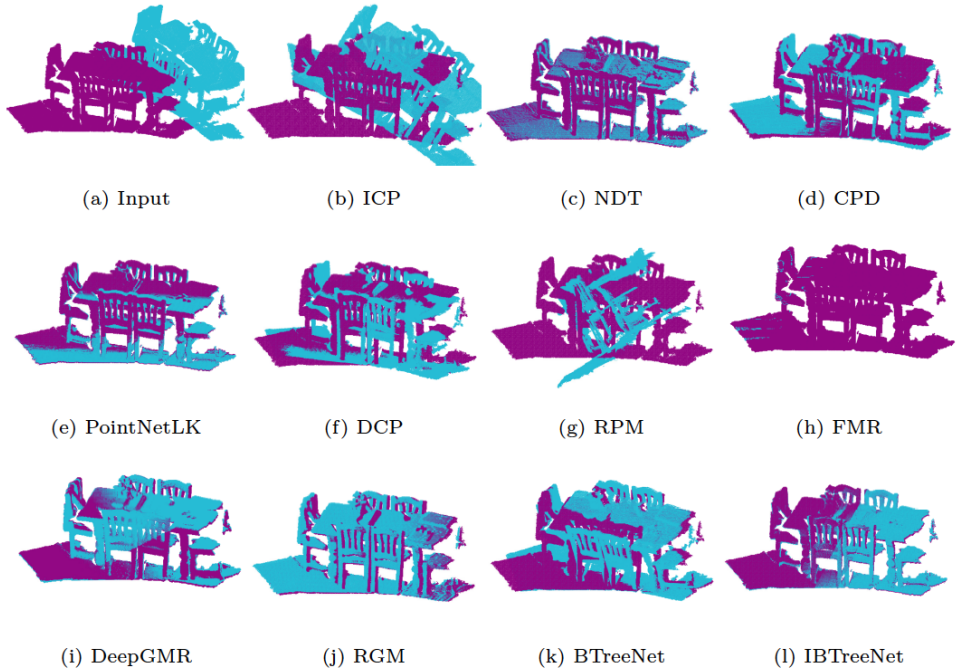


Fig. 5.12 Registration results on unseen 3DMatch datasets.

Table 5.7 Evaluations on the unseen 3DMatch datasets that are not trained. All networks are trained on modelnet40 clean data without missing and noisy points. Bold denotes the top three performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.073672	0.480146	18.846508	0.128525	10.100460	0.064284
NDT	0.079254	0.521475	21.268932	0.109614	10.633771	0.056268
CPD	**0.018102**	**0.112831**	**4.536769**	**0.019583**	**2.186277**	**0.009842**
PNLK	0.029412	0.223548	9.839967	0.030038	5.330842	0.014840
DCP	0.057605	0.499294	14.018398	0.312191	7.619330	0.156427
RPM	0.051456	0.401897	12.399297	0.045833	6.329186	0.023072
FMR	0.031666	0.454459	10.464576	0.036622	5.778973	0.018512
DeepGMR	**0.005068**	**0.017258**	**0.678505**	**0.002990**	**0.360289**	**0.001487**
RGM	0.042781	0.439900	10.011215	0.041061	5.486697	0.020612
BTreeNet	0.039944	0.398068	8.667772	0.310266	4.306098	0.155422
IBTreeNet	**0.028588**	**0.221609**	**6.914699**	**0.310182**	**3.353310**	**0.155365**

point clouds from eight sets of 2D scene images created from the official testing split of the RGB-D reconstruction datasets [124]. The data distribution of the reconstructed 3D point cloud is different from the data captured by the LiDAR and laser scanners. In this experiment, all networks are still trained on ModelNet40 Clean Training Data (Section 5.3.1) and directly tested on *Unseen 3DMatch*. As illustrated in Section 5.3.6.1, the same settings for RGM [49] are still used.

The average registration errors of BTreeNet and IBTreeNet against ICP [43], NDT [44], CPD [45], PointNetLK [46], DCP [47], RPM [48], FMR [50], DeepGMR [51] and RGM [49] on *Unseen 3DMatch* are shown in Table 5.7, with qualitative results shown in Figure 5.12.

Similarly to the results on *Unseen Whu-TLS*, the learning-based method DeepGMR [51] ranks first in all performing measures, and the traditional method CPD [45] outperforms all learning-based methods except DeepGMR [51]. The IBTreeNet outperforms RGM [49], PointNetLK [46], DCP [47], FMR [50], RPM [48], ICP [43] and NDT [44] based on the overall registration errors. By comparison, the IBTreeNet also achieves remarkable generalization to unseen indoor scenes.

Note that for all learning-based methods and traditional methods except CPD [45], the registration performance decreases to some extent on *Unseen 3DMatch* compared to the evaluation on *Unseen KITTI* and *Unseen Whu-TLS*. One possible reason is that the data distribution and density of the reconstructed 3D point clouds from 2D images are different from the LiDAR and laser scanners.

Table 5.8 Evaluations on the *Unseen Shapes* testing datasets that are not trained. All networks are trained on Modelnet40 Clean Training Data without missing and noisy points. Bold denotes the top three performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.064974	0.445631	17.599977	0.103102	9.721335	0.052357
NDT	0.050633	0.410507	16.808854	0.065777	8.472930	0.033677
CPD	**0.014910**	**0.078536**	**3.070854**	**0.017432**	**1.340475**	**0.008933**
PNLK	0.023981	0.232628	10.280659	0.029590	6.199633	0.014813
DCP	0.079755	0.663195	25.690459	0.196156	13.536360	0.095452
RPM	0.084928	0.653797	26.749477	0.073703	13.512754	0.035491
FMR	0.026686	0.253822	11.499332	0.045954	5.902958	0.023032
DeepGMR	**0.000121**	**0.000404**	**0.020372**	**0.000026**	**0.007894**	**0.000013**
RGM	0.049332	0.356896	14.399312	0.045966	7.821818	0.022589
BTreeNet	0.026233	0.267393	6.878716	0.196028	3.622638	0.095358
IBTreeNet	**0.019934**	**0.243259**	**5.151717**	**0.195974**	**2.528907**	**0.095340**

5.3.6.4 Generalization on Unseen Shapes

Additionally, the performance of each method on unseen shapes is evaluated including the Stanford Dragon [125], Bunny [125], Hand [127] and Children[2] that are not trained. An *Unseen Shapes* testing dataset is created, including 1,000 3D point cloud pairs with the same settings as the *Unseen KITTI* testing dataset. All networks are still trained on ModelNet40 Clean Training Data (Section 5.3.1) and directly tested on *Unseen Shapes*. As illustrated in Section 5.3.6.1, the same settings for RGM [49] are still used.

The performance of each method on *Unseen Shapes* is reported in Table 5.8. Similarly to the results on *Unseen Whu-TLS* and *Unseen 3DMatch*, DeepRGM [51] ranks first in all performing mea-

[2]http://visionair.ge.imati.cnr/

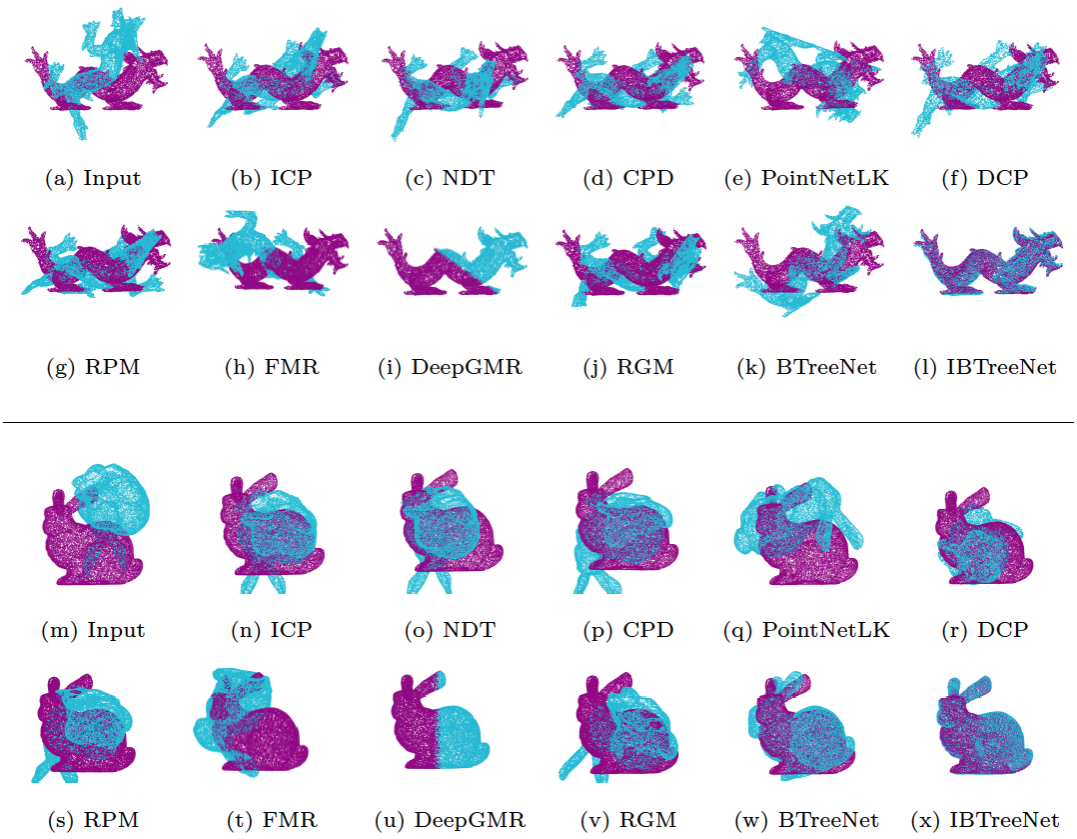


Fig. 5.13 Registration results on unseen shapes datasets. Each 3D point cloud contains 20,480 points.

sures and CPD [45] outperforms all learning-based methods except DeepGMR [51] on *Unseen Shapes*. The learning-based method PointNetLK [46], FMR [50] and the BTreeNet achieve approximately similar registration results based on the overall evaluation metrics. The IBTreeNet outperforms all learning-based methods except DeepGMR [51]. The qualitative registration results are shown in Figure 5.13.

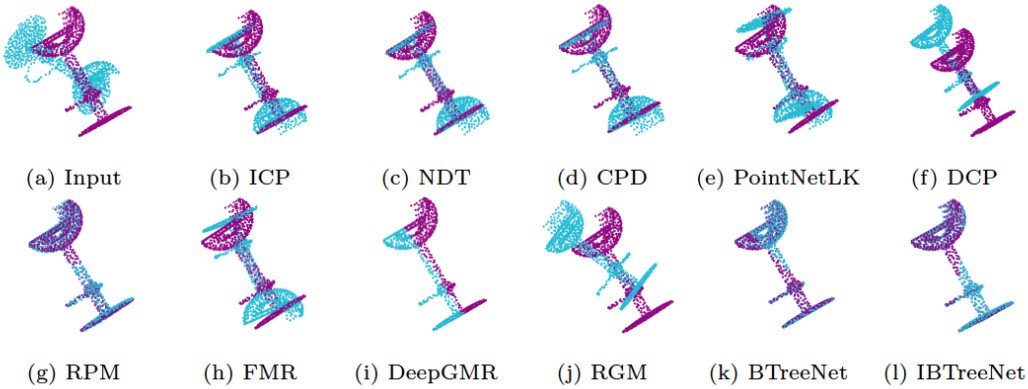


Fig. 5.14 Registration results on 3D point clouds with large rotations.

Table 5.9 Evaluations on 3D point clouds with the initial rotation ranges from 0 to 180 degrees. Bold denotes the top three performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.105312	2.463621	141.334273	0.162676	74.069710	0.079737
NDT	0.143004	2.472329	139.066895	0.113671	76.105349	0.056608
CPD	0.068746	2.386553	143.228976	0.053039	78.755450	0.026269
PNLK	0.126135	2.415207	137.081225	0.173611	79.806911	0.086528
DCP	0.137270	2.418509	134.438871	0.095526	74.231256	0.048041
RPM	**0.055095**	**1.012850**	**59.677372**	**0.010019**	**50.646270**	**0.005000**
FMR	0.090998	2.363750	137.574644	0.127406	77.094416	0.063657
DeepGMR	**0.000010**	**0.000050**	**0.015365**	**0.000002**	**0.001595**	**0.000001**
RGM	0.115444	2.642357	137.574520	0.737000	84.700130	0.365891
BTreeNet	0.067034	1.22952	64.790884	0.008188	57.402523	0.004077
IBTreeNet	**0.040702**	**0.920210**	**55.247189**	**0.007913**	**47.852112**	**0.003916**

5.3.7 Registration with Large Rotations

Large rotation is a challenging task for 3D point cloud registration. In this experiment, all networks are trained on ModelNet40 Clean Training Data (Section 5.3.1) with random rotations from 0 to 180 degrees. All trained models are then tested on *ModelNet40 Clean* testing dataset with random rotations from 0 to 180 degrees. Note that, the random rotations from $[0, 45°]$ to $[0, 180°]$ are only changed in ModelNet40 Clean Training Data (Section 5.3.1) and *ModelNet40 Clean* testing dataset, respectively. Other settings are not changed in these datasets.

As can be seen in Tables 5.9 and 5.1, even if all networks are trained on point clouds with large initial rotations, registration errors of all learning-based methods increase dramatically when the initial rotation ranges from 0 to 180 degrees, except DeepGMR [51]. DeepGMR [51] significantly outperforms other learning-based methods and traditional methods with large rotations and ranks first in all performing measures. DeepGMR [51] proposes a correspondence network and two differentiable computing blocks for solving the problem on large rotations. Although the IBTreeNet suffers on the large rotations, it still outperforms ICP [43], NDT [44], CPD [45], PointNetLK [46], DCP [47], FMR [50] and RGM [49]. The qualitative comparisons are shown in Figure 5.14.

PointNetLK [46], FMR [50], DCP [47] and RGM [49] do not converge during the training whereas RPM [48] and BTreeNet and IBTreeNet converge to the local optima. One possible reason is that more point cloud pairs are needed to be trained under $[0, 180°]$ compared with that under $[0, 45°]$, which increases the variety and

Table 5.10 Ablation studies on the proposed binary tree and loss functions.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
without BT	0.021145	0.152280	7.028391	0.008171	4.090947	0.004103
BT-FC	0.016139	0.087589	3.511191	0.008165	1.771228	0.004099
BT-1DConv	0.012294	0.065951	2.618611	0.007907	1.379738	0.003988
BT-Supervision	0.019359	0.096562	3.885027	0.008143	1.984579	0.004091
CD	0.015577	0.085754	3.434221	0.636930	1.735888	0.004100
CD+EMD	0.012294	0.065951	2.618611	0.007907	1.379738	0.003988

complexity of training point cloud pairs and is challenging to extract the generalized features for the increased varieties.

5.3.8 Ablation Studies

In this section, the results of the ablation studies are presented to analyse the effectiveness of each component in BTreeNet. In particular, all ablated models are trained on the ModelNet40 Clean Training Data (Section 5.3.1) and test them on the *ModelNet40 Clean* testing datasets. The initial rotations in this analysis are in the range of $[0, 45]$ degrees, and the initial translations are in the range of $[-0.5, 0.5]^3$.

The effectiveness of the proposed binary tree-based forward propagation with the standard forward propagation without using a binary tree (without BT) is first analysed, as reported in Table 5.10. BT-FC and BT-1DConv denote the fully connected layers and the combination of 1D Conv, BN and ReLU layers that are adopted in the binary tree, respectively. It is obvious that both BT-FC and BT-1DConv outperform the standard forward propagation without using a binary tree, which shows the effectiveness of the proposed

Table 5.11 Ablation studies on the weighted loss function. λ_1 and λ_2 denote the weights of CD and EMD, respectively.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
$\lambda_1 = 0.1; \lambda_2 = 0.9$	0.014545	0.078982	3.158734	0.008142	1.601008	0.004088
$\lambda_1 = 0.2; \lambda_2 = 0.8$	0.013478	0.074375	2.967007	0.008142	1.490887	0.004089
$\lambda_1 = 0.3; \lambda_2 = 0.7$	0.012810	0.071483	2.847388	0.008142	1.425907	0.004089
$\lambda_1 = 0.4; \lambda_2 = 0.6$	0.013562	0.075923	3.031684	0.008141	1.523975	0.004089
$\lambda_1 = 0.5; \lambda_2 = 0.5$	0.014778	0.080827	3.234577	0.008142	1.626481	0.004089
$\lambda_1 = 0.6; \lambda_2 = 0.4$	0.013858	0.076176	3.043757	0.008141	1.536414	0.004089
$\lambda_1 = 0.7; \lambda_2 = 0.3$	0.014435	0.079602	3.182555	0.008141	1.599607	0.004089
$\lambda_1 = 0.8; \lambda_2 = 0.2$	0.013205	0.073918	2.947582	0.008141	1.483749	0.004090
$\lambda_1 = 0.9; \lambda_2 = 0.1$	0.015058	0.084448	3.383622	0.008141	1.703698	0.004090
$\lambda_1 = 1.0; \lambda_2 = 1.0$	**0.012294**	**0.065951**	**2.618611**	**0.007907**	**1.379738**	**0.003988**

Table 5.12 Ablation studies on the feature extraction modules.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
BT-V1	0.012294	0.065951	2.618611	0.007907	1.379738	0.003988
BT-V2	0.066103	2.822350	174.840489	0.009611	100.515426	0.004796
BT-V3	0.045849	0.310504	12.621608	0.008175	6.370999	0.004105

Table 5.13 Ablation studies on the level of the binary tree.

Methods	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)	F Norm ↓ (**T**)	CD ↓
BT-L2	3.624825	0.008165	1.824710	0.004099	0.090332	0.016149
BT-L3	3.068289	0.008164	1.536206	0.004099	0.076868	0.013486
BT-L4	2.830392	0.008165	1.416922	0.004099	0.071089	0.012618
BT-L5	**2.618611**	**0.007907**	**1.379738**	**0.003988**	**0.065951**	**0.012294**
BT-L6	13.380222	0.008176	7.443230	0.004106	0.286715	0.035227
BT-L7	19.369034	0.008166	8.403775	0.004099	0.406013	0.043825

binary tree. Since BT-1DConv outperforms BT-FC, the combination of 1D Conv, BN and ReLU layers are adopted in the binary tree. The loss function of the CD shows a lower registration accuracy by comparing it with the combination of the CD and the EMD. The BTreeNet is still evaluated in a supervised manner following the supervised loss function in DCP [47], which minimizes the difference between the estimated transformation matrix and the ground-truth transformation matrix and is denoted as BT-Supervision. The combination of the CD and the EMD loss function also outperforms the supervised manner on BTreeNet. Thus, the combination of the CD and the EMD is used as the final loss function. The weighted test of the final loss function is reported in Table 5.11. The best performance is selected in all performing measures with $\lambda_1 = 1.0$ and $\lambda_2 = 1.0$ for CD and EMD, respectively.

The effectiveness of three state-of-the-art feature extraction modules in PointNet [1], PointNet++ [70] and DGCNN [72] are then analysed to select the most effective feature extraction module for the BTreeNet, and the resulting methods are denoted as BT-V1, BT-V2 and BT-V3. As reported in Table 5.12, the feature extraction module in PointNet [1] outperforms others with the lowest average registration errors. PointNet++ [70] is a representative of the class of hierarchical feature extraction networks that aggregate local features before global pooling. PointNet [1] outperforms PointNet++ [70] in the encoder using global pooling. This is because local pooling is less stable than global pooling due to suboptimality in the selection of local neighbourhoods for the whole point cloud. Aggregating local features before global pooling results in unstable global features for the binary tree-based decoder to estimate the optimal rotation

and translation. DGCNN [72] learns local geometric features via constructing the k points for each point and also aggregates local features using local pooling, which is also not suitable for the binary tree-based decoder. In addition, PCN [39] and TopNet [40] achieve a similar conclusion for the problem of local pooling in PointNet++ [70] on the 3D point cloud completion task.

The number of binary tree level L in BTreeNet is examined by varying L in $2, 3, 4, 5, 6, 7$. The results are reported in Table 5.13, which shows 5 levels in BTreeNet have a good trade-off between alignment accuracy and computational performance. As shown in the table, after a certain level the increase of the tree depth does not increase the accuracy of the registration. The reason is that 5 levels in BTreeNet already sufficiently include the majority of features in the alignment in this experiment.

5.4 Discussion and Limitation

Two prominent failure cases have been identified for the proposed BTreeNet and IBTreeNet.

First, the models fail to transform 3D point clouds with large missing data. As shown in Figure 5.15, 50% of the points are missing in the input source 3D point cloud. The partial point clouds should be transformed into the same parts in target point clouds rather than the centre of the target point clouds, as shown in Figure 5.15 (k), (l), (w) and (x). Because the loss function of the combined CD and EMD finds the global registration between two shapes without considering the local to global registration. However, other supervised learning-based methods still meet this problem because they force the network

Table 5.14 Evaluations on 3D point clouds with the large rotations, Gaussian noise and partial visibility together. The initial rotation ranges from 0 to 180 degrees. The standard deviation of Gaussian noise is 0.01. 70% of the points are retained in the source 3D point cloud. All networks are trained on Modelnet40 Clean Training Data with the $[0, 180°]$ rotations without missing and noisy points. All networks are tested on *ModelNet40 Partial & Noise* with random rotations from 0 to 180 degrees. Bold denotes the best-performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.127023	2.478114	140.039088	0.232349	73.982576	0.116232
NDT	0.163471	2.491908	138.536217	0.227821	76.550642	0.111283
CPD	0.097589	2.419856	140.816336	0.187100	80.316289	0.093111
PNLK	0.166495	2.438380	133.998653	0.264212	81.443547	0.131969
DCP	0.143335	2.418145	133.654160	0.096254	74.054247	0.048559
RPM	0.132424	1.384155	66.200785	0.205663	54.264680	0.102969
FMR	0.121134	2.395068	135.549886	0.215384	80.910621	0.106631
DeepGMR	0.140693	1.896580	97.199968	0.196313	72.883297	0.098153
RGM	0.446829	2.645025	136.359483	0.766860	84.548646	0.382923
BTreeNet	0.074901	1.310750	68.201930	0.008189	61.199467	0.004077
IBTreeNet	**0.058597**	**1.211751**	**60.334884**	**0.007321**	**50.298829**	**0.003508**

to output a certain transformation based on a certain pattern of input 3D point cloud pairs by using the supervised manner and still do not consider the local to global registration in their loss function.

Second, the proposed models fail to transform the 3D point cloud with the combination of partial overlap, noise and large initial rotations. To evaluate the robustness of all algorithms on 3D point cloud pairs with the combination of these scenarios, a *ModelNet40 Partial & Noise* testing dataset is generated from *ModelNet40 Clean*. Specifically, for each source 3D point cloud in *ModelNet40 Clean*, noise is firstly sampled from Gaussian distribution with a standard deviation of 0.01 and then a random clipping plane is created passing through

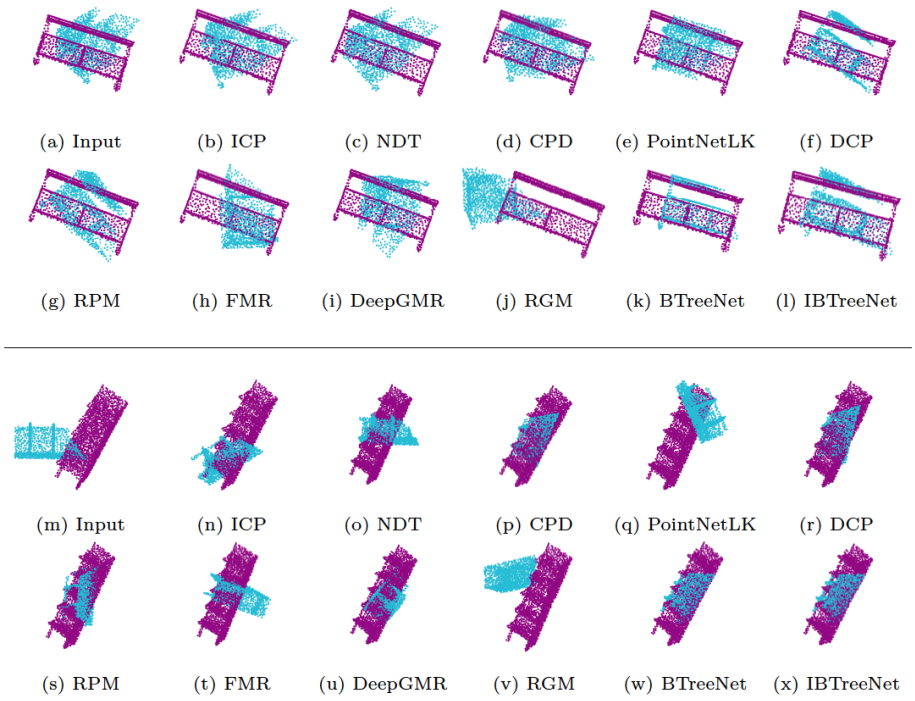


Fig. 5.15 Failure cases on partial point clouds with 50% of missing data. The partial 3D point clouds should be transformed into the same parts in target point clouds rather than the centre of the target point clouds.

the origin and is shifted to retain 70% of the points. All networks are trained on ModelNet40 Clean Training Data (Section 5.3.1) with random rotations from 0 to 180 degrees. All trained models are tested on *ModelNet40 Partial & Noise* with random rotations from 0 to 180 degrees. The performance of each method on *ModelNet40 Partial & Noise* with the large rotation is reported in Table 5.14. Both traditional and learning-based methods fail to achieve accurate registration, which indicates the poor robustness and generalization ability of each network under such challenging conditions.

DeepGMR [51] shows the best performance on 3D point clouds with point-to-point correspondences (Sections 5.3.3, 5.3.6 and 5.3.7), however, it shows poor robustness and generalization ability on

partial visibility and noise with broken point-to-point correspondences, as reported in Tables 5.3 and 5.4. Thus, DeepGMR [51] fails under the combination of partial overlap, noise and large rotations. Although the BTreeNet and IBTreeNet still do not achieve accurate registration in this challenging condition, they achieve the best-performing measures based on the overall registration errors. The possible reasons: (i) BTreeNet and IBTreeNet can tolerate noise and partial overlap to some extent without training them in these scenarios (Sections 5.3.4 and 5.3.5); (ii) Although BTreeNet and IBTreeNet suffer on the large rotations, they still outperform ICP [43], NDT [44], CPD [45], PointNetLK [46], DCP [47], FMR [50] and RGM [49] (Section 5.3.7).

5.5 Conclusion

A novel unsupervised deep learning network – BTreeNet has been proposed for 3D point cloud registration. BTreeNet consists of a hierarchical binary tree-based forward propagation that learns features for the rotation separately from the translation and avoids the interference between the estimations of rotation and translation in one single matrix. Based on the BTreeNet, IBTreeNet is proposed to iteratively rotate and translate the source 3D point cloud to the target. With an identical network architecture to BTreeNet, the IBTreeNet is trained based on the registration result of a trained BTreeNet model. Once trained, the IBTreeNet model can be reused and iteratively improve registration accuracy. The proposed IBTreeNet achieves precise registration and shows remarkable generalization

and robustness to unseen outdoor and indoor scenes that are not trained.

Chapter 6

Deform3DNet: A Unified Deep Learning Network for Non-rigid Deformable 3D Point Cloud Registration and Correspondence

In this chapter, an end-to-end learning-based network (Deform3DNet) for non-rigid 3D point cloud registration is proposed, which also leads to finding the point-to-point correspondence. The Deform3DNet learns features from non-rigid 3D point clouds and generates a point-to-point transformation matrix through a point-to-point transformation module that considers the non-rigid registration as rigid as possible. A novel non-rigid registration loss is proposed to guarantee each point in one point cloud can be transformed to its corresponding point in the other, which leads to finding the point-to-point correspondence. A structure preservation loss is proposed to keep the internal structure for the transformed 3D point cloud. Experimental results show the improvement in the quality of the registration and correspondence by comparing Deform3DNet with state-of-the-art

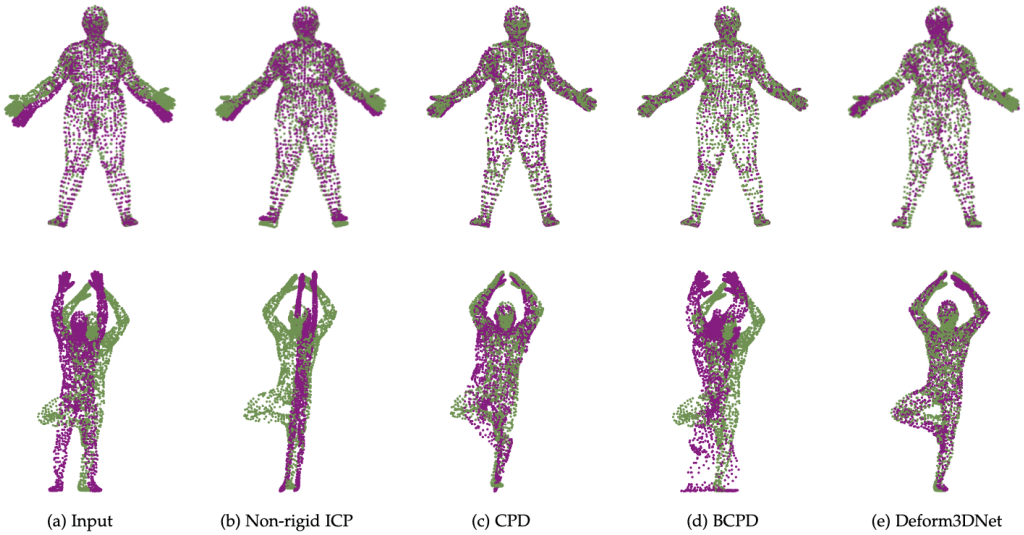


Fig. 6.1 Registration on two non-rigid 3D point clouds with small and large deformations. (a) Input non-rigid 3D point clouds with small (above) and large (bottom) deformations. (b) (c) (d) Registration results of Non-rigid ICP, CPD and BCPD. (e) Registration results of Deform3DNet.

non-rigid 3D point cloud registration and correspondence methods across large deformations, partiality and topological noise.

6.1 Introduction

Non-rigid 3D point clouds registration and correspondence are fundamental challenges in computer vision and computer graphics, with applications in shape analysis [52] [128], deformation transfer [54], 3D reconstruction [129] [25], and 3D object tracking [130].

Previous non-learning based methods [59] [45] [60] for non-rigid 3D point clouds registration and correspondence heavily depend on the initial poses and the transformation of the points is constrained by the adjacent points, which results in a misalignment between two non-rigid 3D point clouds with large and multiple deformations. These methods are also computationally expensive and

time-consuming. Figure 6.1 (a) shows non-rigid 3D point clouds with small and large deformations, respectively. Figure 6.1 (b), (c) and (d) show registration results produced by non-rigid 3D registration methods Non-rigid Iterative Closest Point (Non-rigid ICP) [59], Coherent Point Drift (CPD) [45] and Bayesian Coherent Point Drift (BCPD) [60], respectively. It is worth noting that Non-rigid ICP [59] and BCPD [60] perform poorly on large and multiple deformations, for example, the deformations in arms and legs. CPD [45] only successfully aligns the arms without the body and legs, whereas Deform3DNet achieves successful registration with both small and large deformations, as shown in Figure 6.1 (e).

Without considering 3D shape alignment, many state-of-the-art non-learning and learning-based approaches [61] [62] [63] [64] [65] have been proposed to find correspondence between non-rigid 3D shapes. However, these methods are limited when handling large and multiple deformations between 3D shapes. The results of non-rigid shape correspondence (indicated by colour transfer) produced by these methods are shown in Figure 6.2. FMNet [63], SURFMNet [64] and CorrNet3D [65] find the incorrect correspondence between arms, body and legs, as shown in Figure 6.2 (f), (g) and (h). Figure 6.2 (i) indicates that ZoomOut [61] produces the vertically symmetric point-wise correspondence and does not provide the correct correspondence between two 3D meshes, whereas Figure 6.2 (j) shows that Fast Sinkhorn Filter [62] provides the proper correspondence. The point-to-point correspondences of Deform3DNet results are visualized in the form of a 3D point cloud and a mesh, respectively, as shown in Figure 6.2 (k) and (l). The registration results produced by non-rigid 3D registration methods Non-rigid ICP [59], CPD [45]

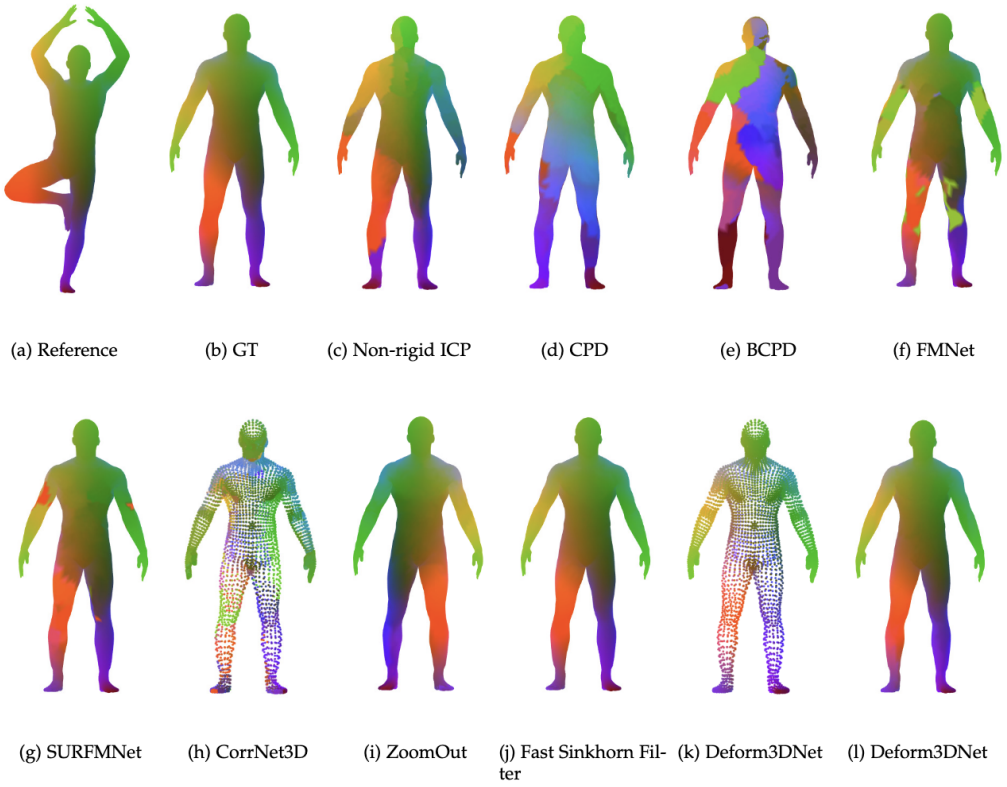


Fig. 6.2 Correspondence between two non-rigid 3D point clouds with large deformations. The point-to-point correspondences are visualized via colour transfer. (a) Reference non-rigid 3D point cloud. (b) Ground-truth correspondence. (c) (d) (e) Results from Non-rigid ICP, CPD and BCPD. (f) (g) (h) Results from learning-based FMNet, SURFMNet and CorrNet3D. (i) (j) Results from non-learning based methods ZoomOut and Fast Sinkhorn Filter. (k) (l) Results of Deform3DNet in the form of 3D point cloud and mesh.

and BCPD [60] are also shown in Figure 6.2 (c), (d) and (e), which indicates that these methods are limited when handling large and multiple deformations. Most importantly, these state-of-the-art learning-based methods [63] [64] [65] do not provide the non-rigid 3D registration after finding the point-wise correspondences.

To achieve high-performance non-rigid 3D point clouds registration and correspondence, two main concerns are observed. (i) The transformation of non-rigid 3D point clouds can be done through a point-to-point rigid transformation (rotation and translation); (ii) If each individual point in one 3D point cloud is transformed to its corresponding point in the other, it will lead directly to the correspondences between the two non-rigid 3D point clouds. Therefore, in this chapter, an end-to-end deep learning-based network (Deform3DNet) for non-rigid 3D point cloud registration is proposed. In Deform3DNet, the combination of 1D Conv, BN and ReLU layers is used to extract point-wise features that are then aggregated to the global features via a max-pooling operator. The global features of two non-rigid 3D point clouds are given as the input to a point-to-point transformation module that applies *as rigid as possible* for the non-rigid transformation to generate a transformation matrix for each point in a point-to-point transformation layer. A novel non-rigid registration loss is proposed to guarantee that the effective features can be learned to generate the point-to-point rigid transformation for optimal correspondence. The non-rigid registration loss is differentiable and minimizes the Frobenius norm between the transformed 3D point cloud and the target. A novel structure preservation loss is then proposed by maximizing the similarity of grouped points between the transformed and the target 3D point

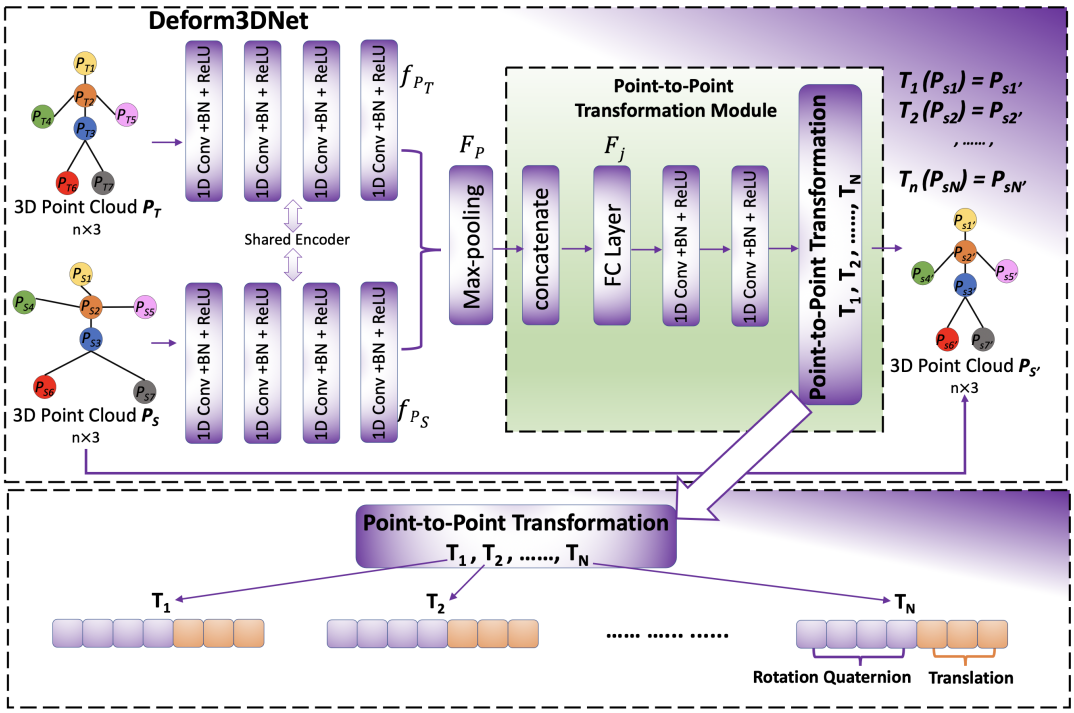


Fig. 6.3 Deform3DNet architecture. The source and the target non-rigid 3D point clouds are given as input to the shared encoder to generate the global features for each 3D point cloud. A point-to-point transformation module is proposed to generate the point-to-point rigid transformation.

clouds, keeping the internal structure in the transformed 3D point cloud.

6.2 Methods

Let P_S and P_T define the source and the target non-rigid 3D point clouds, respectively. A 3D point cloud is represented as a set of 3D points $\{p_i \mid i = 1, 2, ..., N\}$, where each point in P_S and P_T is defined as $p_i = (x, y, z)$. The main purposes of this method are: (i) Considering the non-rigid 3D point cloud registration as rigid as possible and generating a point-to-point rigid transformation matrix (rotation and translation) for each point in P_S to align with

P_T; (ii) Designing a loss function that makes each individual point in P_S to transform to its corresponding point in P_T and finds the correspondences between non-rigid 3D point clouds. Deform3DNet (Section 6.2.1), as shown in Figure 6.3, is proposed to generate the point-to-point rigid transformation. A non-rigid 3D registration loss and a structure preservation loss (Section 6.2.2) are proposed to find the optimal registration for point-to-point correspondences and keep the internal structure for the transformed P_S, respectively.

6.2.1 Deform3DNet

As illustrated in Figure 6.3, two non-rigid 3D point clouds $P_S \in \mathbb{R}^{N\times 3}$ and $P_T \in \mathbb{R}^{N\times 3}$ are given as input and need to be aligned. P_S and P_T pass through the shared encoder that takes each point independently and generates the high-dimensional point-wise features $f_{P_S} \in \mathbb{R}^{N\times d}$ and $f_{P_T} \in \mathbb{R}^{N\times d}$, where d is the dimension for point-wise features. The max-pooling is applied to aggregate point-wise features f_{P_S} and f_{P_T} to extract the global features $F_{P_S} \in \mathbb{R}^{1\times d}$ and $F_{P_T} \in \mathbb{R}^{1\times d}$ by using the Equation 6.1.

$$F_P = \underset{p_i \in P}{MAX} \{h(p_1), ..., h(p_n)\} \, (i = 1, ..., n) \tag{6.1}$$

where p_i is a point in either P_S or P_T. h is a combination of 1D Conv, BN and ReLU layers that aims to extract f_{P_S} and f_{P_T}, and MAX represents the max-pooling operator that returns a new vector of the element-wise maximum.

The features F_{P_S} and F_{P_T} are given as input to the proposed point-to-point transformation module that considers the non-rigid transformation as rigid as possible and generates the point-to-point

rigid transformation matrices. Specifically, F_{P_S} and F_{P_T} are concatenated and passed through a fully connected layer to obtain the global feature vector $F_j \in \mathbb{R}^{1\times 2d}$ $(j = 1, ..., 2d)$ for the non-rigid 3D registration. The global feature vector F_j is decoded from two layers of $h(h(\cdot))$ to generate the point-to-point rigid transformation $T \in \mathbb{R}^{N\times 7}$ form a point-to-point transformation layer by using the Equation 6.2.

$$\begin{aligned} T &= \{h(h(F_1), ..., h(F_j))\}\,(j = 1, ..., 2d) \\ &= \{T_1, ..., T_N\}\,(T_N \in \mathbb{R}^{1\times 7}) \end{aligned} \tag{6.2}$$

where N is the number of points in P_S, 1×7 represents a size of 1×7 transformation matrix for each point in P_S, including a size of 1×4 rotation quaternion q_i $(i = 1, ..., N)$ and a size of 1×3 translation $t_i \in \mathbb{R}^3$.

Each rotation quaternion q_i, defined in Equation 6.3, is transformed into a rotation matrix $R_i \in SO(3)$ $(i = 1, ..., N)$ with the size of 3×3, as defined in Equation 6.4.

$$\begin{aligned} q_i = [q_a, q_b, q_c, q_d]^T;\ q_a = cos\frac{\theta}{2}; q_b = n_x sin\frac{\theta}{2}; \\ q_c = n_y sin\frac{\theta}{2}; q_d = n_z sin\frac{\theta}{2};\ n = [n_x, n_y, n_z]^T \end{aligned} \tag{6.3}$$

where θ is the rotation angle, and n is the axis of rotation.

$$R_i = \begin{bmatrix} 1 - 2q_c^2 - 2q_d^2 & 2q_bq_c + 2q_aq_d & 2q_bq_d - 2q_aq_c \\ 2q_bq_c - 2q_aq_d & 1 - 2q_b^2 - 2q_d^2 & 2q_cq_d + 2q_aq_b \\ 2q_bq_d + 2q_aq_c & 2q_cq_d - 2q_aq_b & 1 - 2q_b^2 - 2q_c^2 \end{bmatrix} \tag{6.4}$$

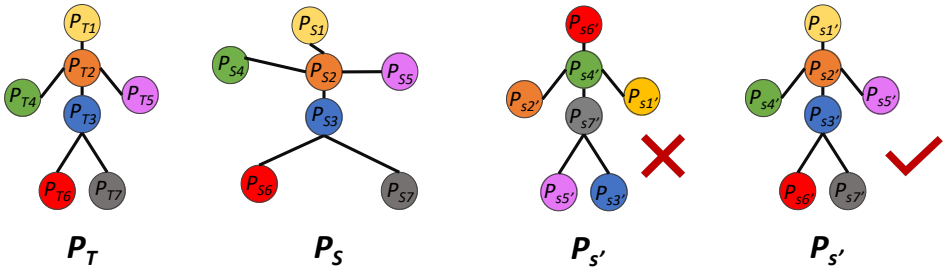


Fig. 6.4 Non-rigid 3D point cloud registration analysis. Each point in P_S needs to be transformed to its correspondence in P_T. The global alignment without finding its correspondence results in incorrect non-rigid 3D point cloud registration.

Finally, each point in the P_S is rotated and translated by Equation 6.5, and the transformed 3D point cloud $P_S^{'}$ is obtained.

$$P_S^{'} = (R_1 p_1 + t_1, ..., R_i p_i + t_i) \ (p_i \in P_S) \tag{6.5}$$

Once the point-to-point transformation matrices are estimated, the point correspondence between P_S and P_T is computed by finding the nearest point between $P_S^{'}$ and P_T using Euclidean distance 6.6.

$$Eucli\_dist(P_S^{'}, P_T) = \underset{p_i \in P_S^{'}}{MIN} (p_i - P_T)^2 \tag{6.6}$$

The index of the point with the minimum Euclidean distance is the estimated correspondence between input P_S and P_T.

6.2.2 Loss Function

The loss function in non-rigid 3D point cloud registration needs to guarantee that each individual point in P_S can be transformed to its corresponding point in P_T, as illustrated in Figure 6.4. To this end, a differentiable $Loss_{deform}$ is proposed, defined in Equation 6.7, to minimize the Frobenius norm between the transformed 3D point

cloud $P_S^{'}$ and target 3D point cloud P_T. The Frobenius norm is the matrix norm, which maximizes the similarity of two matrices of two 3D point clouds. The Frobenius norm loss and the commonly used CD loss for 3D point cloud processing are illustrated and compared in ablation studies in Section 6.3.4.

$$Loss_{deform} = \left\| P_T - P_S^{'} \right\|_F \tag{6.7}$$

Note that, the point-wise correspondences for the Deform3DNet input P_S and P_T are required as the prior knowledge, which guarantees that the output point-to-point rigid transformation matrix T learned by $Loss_{deform}$ can align each point in P_S with its corresponding point in P_T.

To keep the internal structure of the transformed 3D point cloud $P_S^{'}$, a structure preservation loss $Loss_{sp}$ is proposed, as defined in Equation 6.8.

$$\begin{aligned} P_{T\_stru} &= \sum_{i=1}^{N} \sum_{k \in K} (p_k - p_i)^2 \ (p_i \in P_T) \\ P_{S'\_stru} &= \sum_{i=1}^{N} \sum_{k \in K} (p_k - p_i)^2 \ (p_i \in P_{S'}) \\ Loss_{sp} &= \left\| P_{T\_stru} - P_{S'\_stru} \right\|_F \end{aligned} \tag{6.8}$$

where K is the index set of k nearest points of a point p_i.

The k nearest points $\{p_k \mid i = 1, 2, ..., K\}$ from p_i are found in P_T and $P_S^{'}$, respectively. The quadratic sum of the Eucliden distance is calculated between p_k and p_i in both P_T and $P_S^{'}$, which results in P_{T_stru} and $P_{S'\_stru}$, respectively. The structure preservation loss $Loss_{sp}$ is designed to minimize the Frobenius norm between P_{T_stru} and $P_{S'\_stru}$, which maximizes the similarity between the

transformed 3D point cloud $P_S^{'}$ and target 3D point cloud P_T to keep the internal structure.

Finally, the overall loss function for Deform3DNet is defined as Equation 6.9.

$$Loss = Loss_{deform} + Loss_{sp} \tag{6.9}$$

6.3 Experiments

6.3.1 Evaluation Datasets

For non-rigid 3D point cloud registration, four different non-rigid 3D datasets are adopted, including MPI-FAUST [53], DFAUST [52], TOSCA [131] and SHREC16 [132]. These datasets consist of males, females, kids and animals with different poses and also provide point-to-point correspondence between each pose. Both TOSCA and SHREC16 datasets are more challenging since such datasets contain large and multiple deformations between each pose. To train the Deform3DNet, more than 4,000 training pairs and 400 testing pairs are randomly selected, respectively, where each dataset consists of around 1,000 training pairs and 100 testing pairs. Each mesh vertices are randomly down-sampled to 2297 points in a 3D point cloud and the point-to-point correspondence is still kept. The 2297 points in each 3D point cloud are randomly distributed on the 3D surface.

For a fair comparison of non-rigid 3D point cloud correspondence with the state-of-the-art learning-based FMNet [63], SUPERFM-Net [64] and CorrNet3D [65], the same training and testing pairs are adopted in such methods.

6.3.2 Evaluation Matrix

To quantitatively compare different methods on non-rigid 3D point cloud registration, chamfer distance [91] is used to evaluate the registration results. The chamfer distance calculates the average nearest point distance between the target P_T and the transformed $P_S^{'}$, which is commonly used as the evaluation matrix for 3D point cloud processing [48] [49] [50] [41] [40] [91] [25] [133].

The state-of-the-art 3D shape correspondence methods [61] [62] [63] [64] are based on 3D mesh and adopt geodesic distance as the evaluation matrix requiring edges for 3D shape. However, such edges are not available for 3D point clouds. A corresponding evaluation matrix is defined based on 3D point clouds, defined in Equation 6.10, to measure the correspondence error.

$$\begin{aligned} Eucli(P_S, P_T) &= \sum_{x \in P_S, y \in P_T} (x - y)^2 \\ Eucli(P_S^{'}, P_T) &= \sum_{x \in P_S^{'}, y \in P_T} (x - y)^2 \\ Corres &= \left\| Eucli(P_S, P_T) - Eucli(P_S^{'}, P_T) \right\|_F \end{aligned} \tag{6.10}$$

where the sum of the Euclidean distance $Eucli(P_S, P_T)$ between P_S and P_T is calculated as the ground-truth. The sum of the Euclidean distance $Eucli(P_S^{'}, P_T)$ between $P_S^{'}$ and P_T is calculated as the estimated output from each method. The final correspondence error is the Frobenius norm between $Eucli(P_S, P_T)$ and $Eucli(P_S^{'}, P_T)$.

6.3.3 Implementation Details

An Nvidia Geforce 2080Ti GPU with 12G memory is used for network training. The Deform3DNet is trained for 600 epochs with a batch

size of 64, an adam optimizer and a learning rate of 0.001. Two non-rigid 3D point clouds P_T and P_S with 2297 points are given as input for the network during the training. Following the filter sizes in PointNet, the filter sizes for feature extraction in 1D Convolution layer are $[64, 64, 64, 128, 1024]$. Both features with 1024 dimensions from P_T and P_S are concatenated and given as input to a fully connected layer to generate the global features with the size of $[1 \times 2048]$. The detailed comparison between three different feature extraction modules is illustrated in ablation studies in Section 6.3.4. The filter sizes for decoding the global features are $[2048, 2048 \times 2]$. The point-to-point transformation layer generates the point-to-point transformation matrix with the size of $[2297 \times 7]$ from the features with the size of $[2048 \times 2]$.

Table 6.1 Feature extraction analysis. Quantitative comparison of Deform3DNet against state-of-the-art methods using different Feature extraction modules.

Methods	CD	Corres error	Computation Time(s)	Parameters
PointNet	**0.010180**	**0.452496**	**0.199(CPU); 0.027(GPU)**	**78, 625, 999**
PointNet++	0.042001	2.276731	0.907 (CPU); 0.198 (GPU)	788,899,999
DGCNN	0.067470	5.940949	0.549 (CPU); 0.251 (GPU).	79,049,807

6.3.4 Ablation Studies

In this section, the results of the ablation studies are presented to analyse the effectiveness of each component in Deform3DNet.

First, the design of Deform3DNet is validated by analysing the state-of-the-art 3D point cloud feature extraction modules in PointNet [111], PointNet++ [134] and DGCNN [135] to select the most effective feature extraction module. The chamfer distance, corre-

spondence error, computation time and the number of parameters are reported in Table 6.1 based on these feature extraction modules. The feature extraction in PointNet outperforms others with the lowest chamfer distance, correspondence error, and less computation time and parameters. Thus, the feature extraction module in Deform3DNet follows the PointNet structure.

PointNet++ [134] applies several hierarchical feature extraction networks, which aggregate local features before the global pooling. The feature extraction module in PointNet [111] outperforms PointNet++ [134] by using global pooling in this experiment. One possible reason is that local pooling is less stable than global pooling due to suboptimality in the selection of local neighbourhoods for non-rigid 3D point clouds. Aggregating local features before global pooling results in unstable global features for the proposed point-to-point transformation module to estimate the optimal rotations and translations. DGCNN [135] also extracts local features using local pooling, which is also not suitable for Deform3DNet. Moreover, PCN [39] and TopNet [40] achieve a similar conclusion for the problem of local pooling in PointNet++ [134] on the 3D point cloud completion task.

Second, the effectiveness of the proposed $Loss_{deform}$ and $Loss_{sp}$ are evaluated and compared with the commonly-used chamfer distance loss [91] in the 3D point cloud processing networks [50] [91] [133] [39] [40]. In Table 6.2, the performance of Deform3DNet when trained with different loss functions is compared. The results are calculated from the 400 testing pairs in Section 6.3.1. Either registration error or Correspondence error is decreased by adding the proposed $Loss_{sp}$, which shows the effectiveness of the structure preservation loss $Loss_{sp}$.

Fig. 6.5 Comparison with chamfer distance loss. (a) Input non-rigid 3D point clouds. (b) Registration result of Deform3DNet. (C) Registration results by using Deform3DNet with chamfer distance loss. (d) (e) The source and target 3D point clouds with correspondences. Correspondence is visualized by colours mapped from the source 3D point cloud. (f) Visual correspondence results of Deform3DNet. (d) Visual correspondence results by using Deform3DNet with chamfer distance loss.

Table 6.2 Loss analysis. Quantitative comparison of Deform3DNet loss against chamfer distance loss.

Loss	CD	Corres error
$Loss_{deform}$	0.043065	0.461453
$Loss_{deform} + Loss_{sp}$	**0.010180**	**0.452496**
$Loss_{CD}$	0.023257	30.764978

It is worth noting that the chamfer distance loss can be used to align one non-rigid 3D point cloud to the other with lower registration error but with much higher correspondence error, which indicates that chamfer distance loss focuses on the global shape registration and does not find the optimal correspondence between two input non-rigid 3D point clouds. Figure 6.5 (c) and (g) show the good registration results and incorrect correspondences by using chamfer distance loss, whereas Figure 6.5 (b) and (f) show the optimal registration results and correspondences from Deform3DNet loss.

Table 6.3 Evaluations of 3D point cloud registration on the four different testing datasets. The average chamfer distance errors have been calculated to evaluate each method.

Methods	MPI-FAUST	DFAUST	TOSCA	SHREC16
Non-rigid ICP	0.067033	0.066303	0.134136	0.137583
CPD	0.033186	0.025576	0.044377	0.053272
BCPD	0.093096	0.087347	0.230031	0.175747
Deform3DNet	**0.005533**	**0.007326**	**0.016152**	**0.011710**

6.3.5 Evaluation of Registration Performance

To analyse the effectiveness of Deform3DNet on non-rigid 3D point cloud registration, the Deform3DNet with Non-rigid ICP [59], CPD [45] and BCPD [60] on MPI-FAUST [53], DFAUST [52], TOSCA [131] and SHREC16 [132] testing pairs are compared.

Table 6.4 Quantitative comparison of Deform3DNet against previous works on the number of iterations and computation time. The number of points in each 3D point cloud is 2297 in this comparison.

Method	Iterations	Computation Time(s)
Non-rigid ICP	160	93.791 (CPU)
CPD	20	24.124 (CPU)
BCPD	200	61.706 (CPU)
Deform3DNet	**1**	**0.199(*CPU*); 0.027(*GPU*)**

The average chamfer distance error and the correspondence error against Non-rigid ICP, CPD and BCPD are shown in Tables 6.3 and 6.5. The Deform3DNet outperforms Non-rigid ICP, CPD and BCPD on non-rigid 3D point cloud registration and also achieves a significant improvement in finding correspondences between two input 3D point clouds. Figures 6.6 and 6.7 show the qualitative

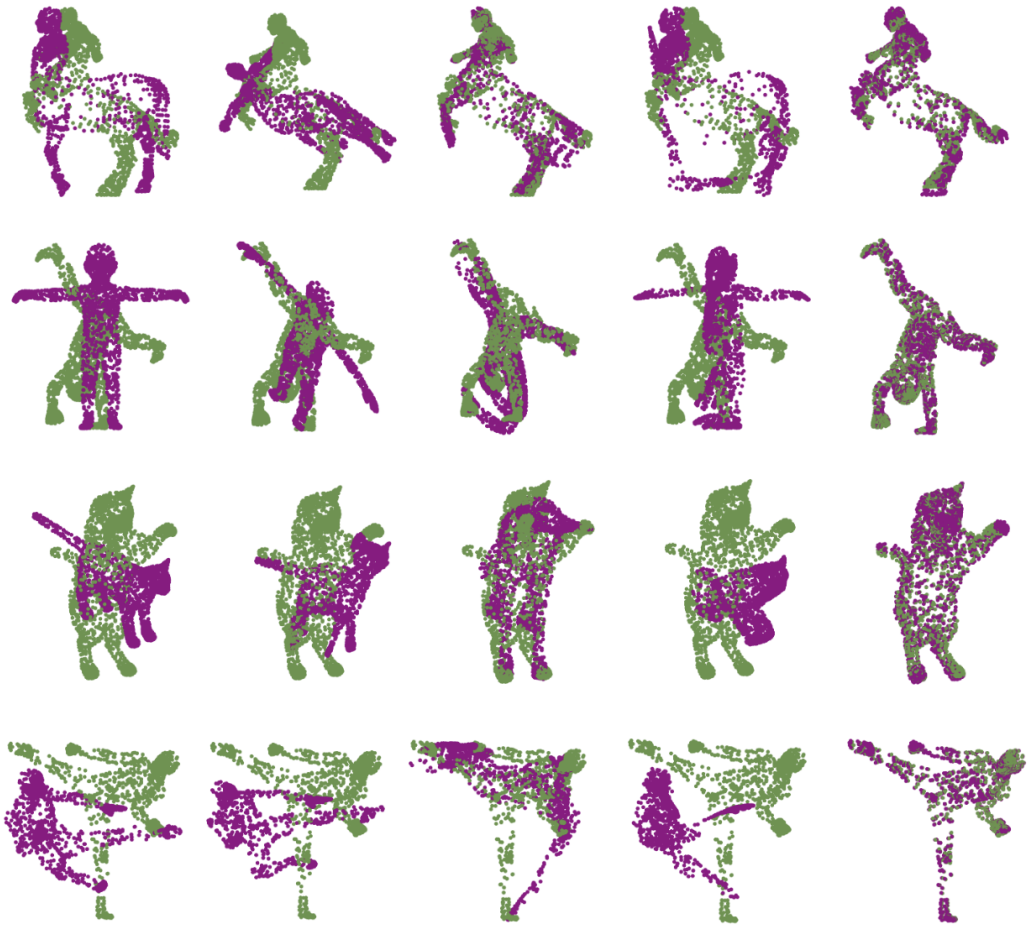


Fig. 6.6 Qualitative non-rigid 3D point cloud registration results of each method. (a) Input non-rigid 3D point clouds. (b) (c) (d) Registration results of Non-rigid ICP, CPD and BCPD. (e) Registration results of Deform3DNet.

Table 6.5 Evaluations of 3D point cloud correspondence on the four different testing datasets. The average correspondence errors have been calculated to evaluate each method.

Methods	MPI-FAUST	DFAUST	TOSCA	SHREC16
Non-rigid ICP	7.662257	6.867315	27.311985	29.354054
CPD	9.373883	3.275189	17.679979	25.350317
BCPD	8.306217	5.423714	22.126240	19.779131
Deform3DNet	**0.152320**	**0.129198**	**0.916890**	**0.611596**

results of each method on non-rigid 3D point cloud registration and correspondence, which indicates that Non-rigid ICP, CPD and BCPD constrain the transformations of points and are extremely sensitive to the large and multiple deformations.

The maximum iterations and computation time for each method are listed in Table 6.4. Deform3DNet requires only one transformation that is significantly lower than Non-rigid ICP, CPD and BCPD. Deform3DNet requires 0.199 seconds on the CPU to align one 3D point cloud to the other, which is significantly less than Non-rigid ICP and BCPD which require around one minute. The GPU acceleration is also provided and the Deform3DNet only takes 0.027 seconds for aligning two non-rigid 3D point clouds, whereas Non-rigid ICP, CPD and BCPD do not provide GPU acceleration.

6.3.6 Evaluation of Correspondence Performance

To analyse the effectiveness of Deform3DNet on non-rigid 3D point cloud correspondence, the Deform3DNet with state-of-the-art non-learning and learning-based methods are compared including ZoomOut

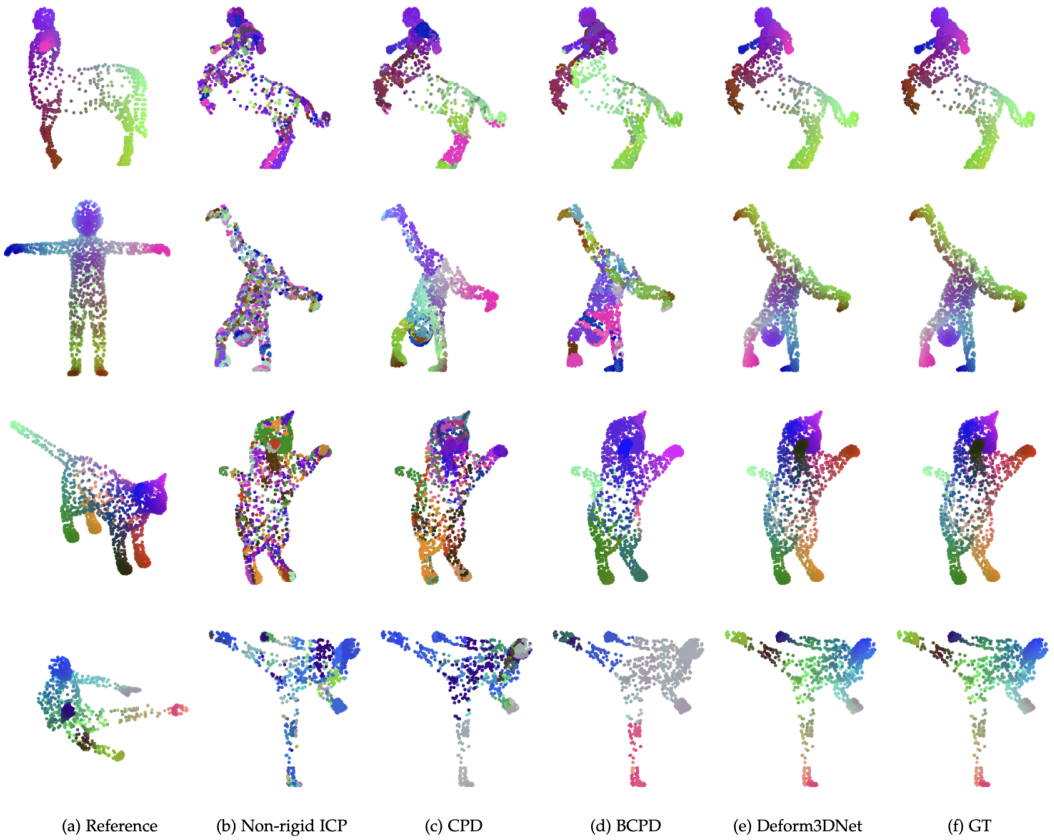


Fig. 6.7 Comparisons on finding correspondences after registration between non-rigid 3D point clouds. Correspondence is visualized by colours mapped from the leftmost Reference 3D point cloud. (a) Reference non-rigid 3D point cloud. (b) (c) (d) Correspondence results of non-rigid ICP, CPD and BCPD. (e) Correspondence results of Deform3DNet. (f) The ground-truth correspondences.

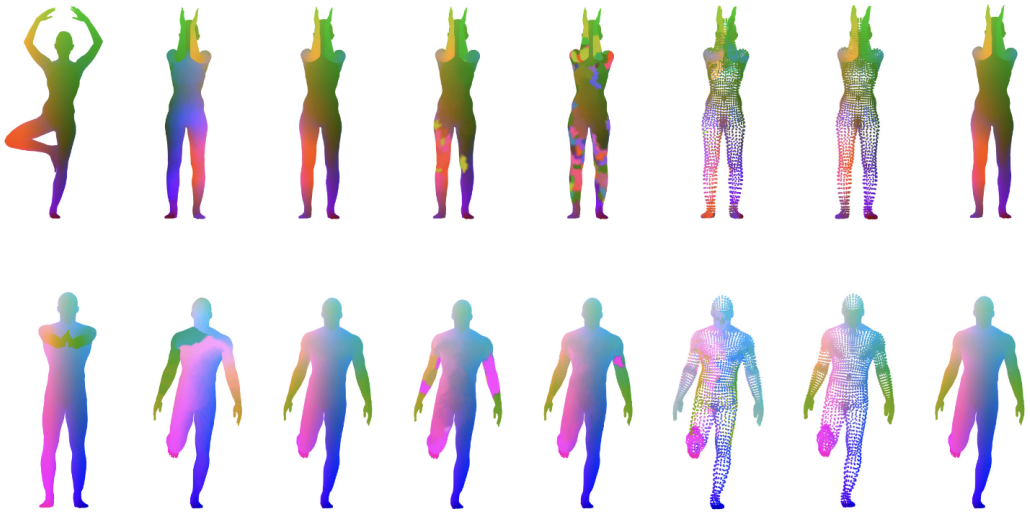


Fig. 6.8 Comparisons on finding correspondences against non-learning and learning-based methods between non-rigid 3D shapes in the form of meshes. Correspondence is visualized by colour mapped from the leftmost Reference 3D point cloud. (a) Reference non-rigid 3D meshes. (b) (c) Correspondence results of non-learning-based methods ZoomOut and Fast Sinkhorn Filter. (d) (e) (f) Correspondence results of learning-based methods FMNet, SURFMNet and CorrNet3D. (g) Correspondence results in the form of 3D point clouds for Deform3DNet. (h) The ground-truth correspondences.

[61], Fast Sinkhorn Filter [62], FMNet [63], SURFMNet [64], and CorrNet3D [65].

Since ZoomOut, Fast Sinkhorn Filter, FMNet, SURFMNet and CorrNet3D only provide the point-to-point correspondence and do not provide the transformation matrix to align non-rigid 3D shapes, the evaluation matrix of the chamfer distance for the registration error cannot be calculated for such methods. 100 training pairs and 100 testing pairs are selected to compare the effectiveness of the learning-based methods FMNet, SURFMNet and CorrNet3D with the Deform3DNet. The average correspondence errors for

training and testing datasets, number of maximum iterations and computation time against these methods are listed in Table 6.6.

The Deform3DNet outperforms ZoomOut, Fast Sinkhorn Filter, FMNet, SURFMNet and CorrNet3D in finding the correspondences either on training or testing pairs with the lowest correspondence error. The non-learning methods ZoomOut and Fast Sinkhorn Filter need around 20 iterations and 50 seconds, whereas the learning-based methods FMNet and SURFMNet require one iteration and approximately 10 seconds. Similarly to learning-based methods, the Deform3DNet also needs one iteration but with less than 1 second for finding correspondences on the CPU. FMNet and SURFMNet also provide GPU acceleration and need around 4.5 seconds per pair. The Deform3DNet takes less computation time than FMNet and SURFMNet on the GPU with 0.097 seconds. CorrNet3D takes the least computation time on GPU at only 0.048 seconds but with higher computation time on CPU at 1.421 seconds. The K-Nearest Neighbour (KNN) algorithm applied in the CorrNet3D requires more computation time on the CPU than that on the GPU. In addition, CorrNet3D shows the largest correspondence error on testing sets, which indicates the poor generalization and robustness of CorrNet3D on datasets that are not trained. Figure 6.8 shows the qualitative results of each method on non-rigid 3D shape correspondence.

It is worth noting that, the learning-based methods FMNet, SURFMNet, CorrNet3D and Deform3DNet achieve a lower performance of finding correspondences on testing datasets compared to that on training datasets. However, the Deform3DNet model shows the optimal robustness on testing sets compared with FMNet, SURFMNet and CorrNet3D models. Fast Sinkhorn Filter shows

Table 6.6 Quantitative comparison of Deform3DNet against state-of-the-art non-learning and learning methods on 3D shape correspondence. The number of vertices in each 3D mesh is 6890 in this comparison.

Methods	Corres error (train sets)	Corres error (test sets)	Iterations	Computation Time(s)
ZoomOut	9.783099	6.587007	25	54.816 (CPU)
Fast Sinkhorn Filter	0.306198	0.295183	20	45.997 (CPU)
FMNet	0.895228	3.539739	1	12.738 (CPU); 4.737 (GPU)
SURFMNet	1.329879	2.137329	1	9.628 (CPU); 4.406 (GPU)
CorrNet3D	1.267763	11.472962	1	1.421 (CPU); 0.048 (GPU)
Deform3DNet	**0.139297**	**0.215360**	**1**	**0.697(*CPU*); 0.097(*GPU*)**

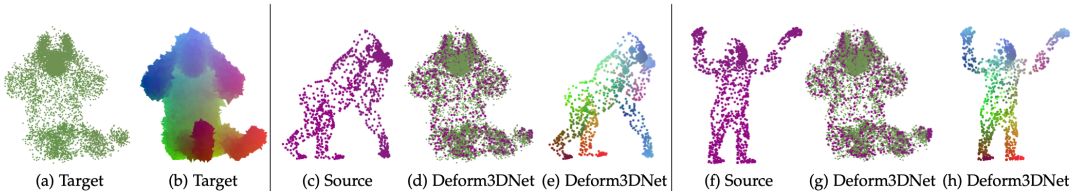


Fig. 6.9 Registration on one of the non-rigid 3D point clouds with Gaussian noise. (a) Target non-rigid 3D point cloud with the standard deviation of Gaussian noise equal to 0.05. (b) Target non-rigid 3D mesh with the standard deviation of Gaussian noise equal to 0.05 and with colour for correspondence visualization. (c) and (f) Source 3D point clouds. (d) and (g) Registration results of Deform3DNet. (e) and (h) Correspondence results of Deform3DNet.

robust performance in finding correspondences, although, it requires longer computation time and around 20 iterations.

6.4 Discussion and Limitations

In this section, the Deform3DNet on topological noise and non-rigid partiality are evaluated and the limitations of the Deform3DNet are analysed.

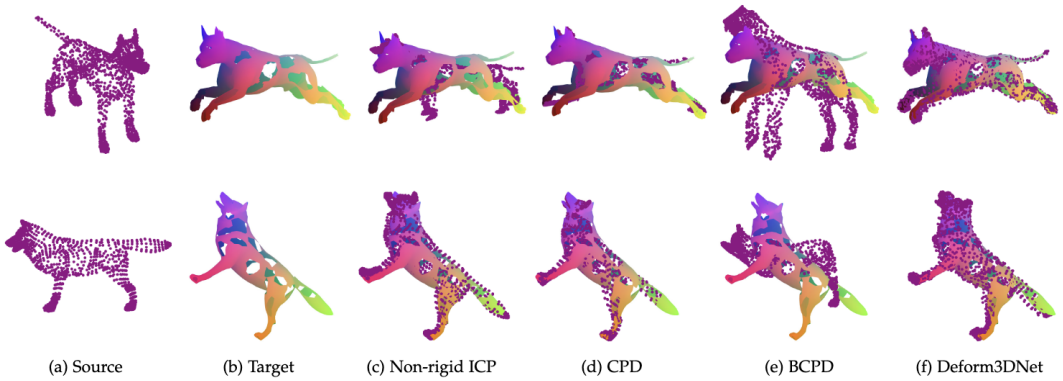


Fig. 6.10 Qualitative non-rigid 3D point cloud registration results of each method with multiple missing holes. (a) Source non-rigid 3D point clouds. (b) Target non-rigid 3D meshes with multiple missing holes. (c) (d) (e) Registration results of Non-rigid ICP, CPD and BCPD. (f) Registration results of Deform3DNet.

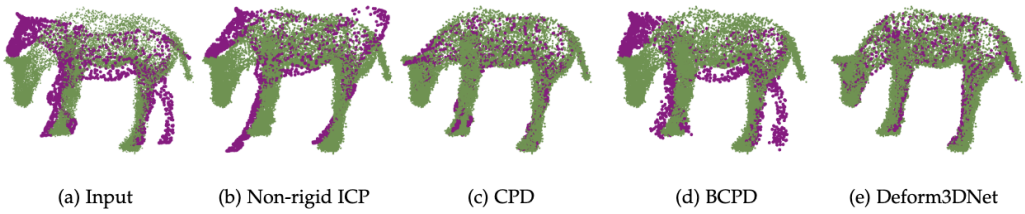


Fig. 6.11 Qualitative non-rigid 3D point cloud registration results of each method with Gaussian noise. (a) Input non-rigid 3D point clouds. (b) (c) (d) Registration results of Non-rigid ICP, CPD and BCPD. (e) Registration results of Deform3DNet.

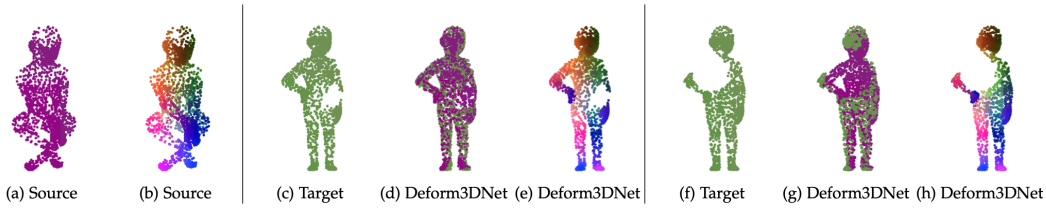


Fig. 6.12 Registration on one of the non-rigid 3D point clouds that contain missing data. (a) Source non-rigid 3D point cloud. (b) Source non-rigid 3D point cloud with colour for correspondence visualization. (c) and (f) Target 3D point clouds with 10% and 20% missing data. (d) and (g) Registration results of Deform3DNet. (e) and (h) Correspondence results of Deform3DNet.

The noise is sampled from Gaussian distribution for non-rigid 3D point clouds with 0 mean and a standard deviation varying in the range of 0.01 to 0.05, and the Gaussian noise is added to all testing pairs. The average chamfer distance errors and correspondence errors are shown in Table 6.7, which illustrates a slight increase of the errors with the standard deviation increases from 0.01 to 0.05 on four different datasets and proves that the Deform3DNet is robust to Gaussian noise. Figure 6.9 shows the registration and correspondence results with the standard deviation equal to 0.05 from TOSCA datasets [131]. Figure 6.11 shows the comparison experiments on Gaussian noise between each method.

The particularly challenging condition of non-rigid 3D point cloud registration and correspondence occurs whenever one of the two 3D point clouds contains missing geometry. To evaluate the robustness of Deform3DNet to non-rigid partially visible data, missing holes are created in the target non-rigid 3D point cloud P_T with the different missing rates ranging from 10% to 30%, and the missing holes are created to all testing pairs. Figure 6.12 shows the registration and correspondence results with 10% and 20% of missing data from SHREC16 datasets [132]. Figure 6.10 shows the comparison experiments on multiple missing holes between each method. The average chamfer distance errors and correspondence errors are shown in Table 6.8. However, the Deform3DNet can only tolerate 10% of missing data, and large registration and correspondence errors occur under large partiality. One of the reasons as being that the network finds the global shape registration and correspondence. Once a large number of points are missing in one shape, those corresponding points in the other shape cannot find their correspondences to form

the whole shape, which results in incorrect transformation and large errors. Another limitation is that Deform3DNet requires the point-wise correspondence of input non-rigid 3D point cloud pairs as prior knowledge.

Table 6.7 Evaluation of non-rigid 3D point cloud registration and correspondence on Gaussian noise.

Gaussian	MPI-FAUST		DFAUST		TOSCA		SHREC16	
Noise	(CD)	(Corres.)	(CD)	(Corres.)	(CD)	(Corres.)	(CD)	(Corres.)
0	0.006524	0.215360	0.007424	0.136900	0.016902	0.932000	0.012941	0.637605
0.01	0.006694	0.218157	0.008343	0.154986	0.016506	0.936492	0.011742	0.612584
0.02	0.009300	0.277870	0.011230	0.204047	0.017575	0.976422	0.012622	0.666853
0.03	0.018521	0.551911	0.014637	0.288147	0.022301	1.583510	0.015210	0.891057
0.04	0.014477	0.549507	0.019520	0.403454	0.028590	1.605392	0.020183	1.422657
0.05	0.020715	0.786765	0.023458	0.510780	0.032119	2.514427	0.021406	1.431405

Table 6.8 Evaluation of non-rigid 3D point cloud registration and correspondence on partially visible data with different missing rates.

Missing	MPI-FAUST		DFAUST		TOSCA		SHREC16	
Rate	(CD)	(Corres.)	(CD)	(Corres.)	(CD)	(Corres.)	(CD)	(Corres.)
0%	0.006524	0.215360	0.007424	0.136900	0.016902	0.932000	0.012941	0.637605
10%	0.039248	2.055594	0.034166	1.140262	0.043884	2.864103	0.037560	2.999121
20%	0.047794	2.294883	0.053138	2.376900	0.086112	7.048545	0.070209	7.607599
30%	0.066892	7.320436	0.060671	3.466895	0.123143	14.941867	0.106812	11.813262

6.5 Conclusion

This chapter has presented a novel end-to-end deep learning network Deform3DNet for non-rigid 3D point cloud registration and correspondence. It shows how deep learning is used successfully in addressing the non-rigid registration and correspondence challenges end-to-end with two non-rigid 3D point clouds. The Deform3DNet requires less time to find the optimal alignment and correspondence for 3D point clouds with large and multiple deformations. To align one non-rigid 3D point cloud to the other, the Deform3DNet considers the process of non-rigid alignment as rigidly as possible to generate the point-to-point rigid transformation, including the point-to-point rotation and translation. To align each point in one 3D point cloud with its corresponding point in the other, a novel non-rigid 3D registration loss function is proposed to learn the features of point-to-point transformation. The structure preservation loss function preserves the internal structure of the transformed 3D point cloud, which further improves the registration results. Once the point-to-point rigid transformation is found, the correspondence is computed by finding the nearest point of each point between transformed and target 3D point clouds. The effectiveness and robustness of the Deform3DNet are demonstrated by comparing it with the state-of-the-art non-learning and learning-based methods on different non-rigid 3D point cloud datasets with challenging settings, such as partially visible data and topological noise.

Chapter 7

Augmented Reality Application

This chapter applies the previously proposed neural networks (TreeNet and Iterative BTreeNet) to achieve stable and accurate 3D point cloud registration in AR. The 3D point clouds are captured by Sony IMX590 TOF 3D LiDAR scanner from iPhone 13 pro-Max. The 3D point cloud completion network (TreeNet) can be used to achieve high-quality 3D point clouds before the rigid 3D point cloud registration network (Iterative BTreeNet). The proposed Deform3DNet requires the point-wise correspondence of two non-rigid 3D point cloud pairs (the captured non-rigid 3D point clouds and the virtual non-rigid 3D object) as prior knowledge and these point-wise correspondences are difficult to obtain from the 3D LiDAR scanner, therefore, the Deform3DNet cannot be directly applied to AR applications.

The AR application is implemented using ARKit[1], which is able to combine the virtual 3D object with the real-world scene. Finally, the AR application is deployed on an iPhone. Note that, the ARKit only provides the technology that combines the virtual 3D object with

[1]https://developer.apple.com/augmented-reality/arkit/

Fig. 7.1 The workflow for AR application with 3D point cloud registration.

the real-world scene without rotating and translating the virtual 3D object to match the same pose of the target object in the real-world scene. Therefore, the proposed networks (TreeNet and Iterative BTreeNet) are used to achieve stable and accurate 3D point cloud registration in AR.

7.1 Virtual 3D Objects in AR Application

AR combines virtual 3D objects with real-world scenes. In this section, four virtual 3D objects are captured and prepared for the AR application, including a 3D chair, a 3D guitar, a 3D bottle and a 3D pillow.

The 3D point clouds of a chair, a guitar a bottle and a pillow are captured from Sony IMX590 TOF 3D LiDAR scanner, as shown in Figure 7.1 (a). Since the LiDAR scanner performs poorly on texture-less regions (e.g. water, transparent bottle, glass and etc.), the captured 3D point cloud bottle is severely distorted and deformed, as shown in Figure 7.1 (third row, first column). Thus, a 3D point cloud bottle from Modelnet40 [126] is selected and used as the virtual 3D object in the AR application, as shown in Figure 7.1 (third row, second columns). The ARKit 1 requires virtual 3D meshes in AR applications for the visualization, thus, the Poisson Surface Reconstruction algorithm [136] is used to reconstruct the 3D object meshes from the captured 3D point clouds, as shown in Figure 7.1 (c).

7.2 AR Implementation using ARKit

This section illustrates how ARKit 1 is used to combine the virtual 3D object with the real-world scene for AR application.

Figure 7.2 (top) shows the AR application using ARKit without 3D point cloud registration. A virtual 3D chair is placed around the target chair in the real-world scene in Figure 7.2 (top). Specifically, the position of the virtual 3D chair is determined by the initial

Fig. 7.2 AR application with and without using 3D point cloud registration.

position of the iPhone. By default, the iPhone is placed in front of the target object at a fixed distance (one meter for the chair) and then the AR application is launched at the initial position. The virtual 3D object chair will be placed one meter away in front of the iPhone camera, as shown in Figure 7.2 (top). The position of the virtual 3D object chair will stay around the same position during the movement of the camera. This is because the ARKit automatically calculates the relative positions between the virtual 3D object chair and the real-world target scene. The real-time AR video is captured from the screenshot of the iPhone and the three images in Figure 7.2 (top) are randomly selected and displayed.

During the movement of the iPhone, the underlying target 3D point cloud scene is also captured from the initial position of the iPhone using ARKit Depth API [2] that can access to the LiDAR scanner of iPhone. Once the underlying target 3D point cloud scene

[2]https://developer.apple.com/documentation/arkit/environmental_analysis/displaying_a_point_cloud_using_scene_depth

is captured, the underlying processes of 3D point cloud completion and registration are used to align the virtual 3D object to the target object in the target scene.

After the underlying 3D point cloud registration, the virtual 3D object chair is rotated and translated to the target object in the target scene. The iPhone camera is still placed at the initial position (one meter away in front of the target chair), and then the AR application is refreshed at the initial position to display the registration results in the AR application, as shown in Figure 7.2 (bottom).

Note that, launching the AR application at a fixed position in front of the target object of the real-world scene can avoid using 3D object detection and tracking algorithms to find the position of the target object in real-world 3D scenes. This thesis does not focus on 3D object detection and tracking algorithms but on object registration. Therefore, the position of the virtual 3D object is assumed as the known information by launching the AR application at the fixed initial position in front of the target object.

7.3 Rigid 3D Point Cloud Registration Results in AR Application

In this section, rigid 3D point cloud registration is applied in the AR application. The rigid 3D point cloud registration method used is the proposed Iterative BTreeNet.

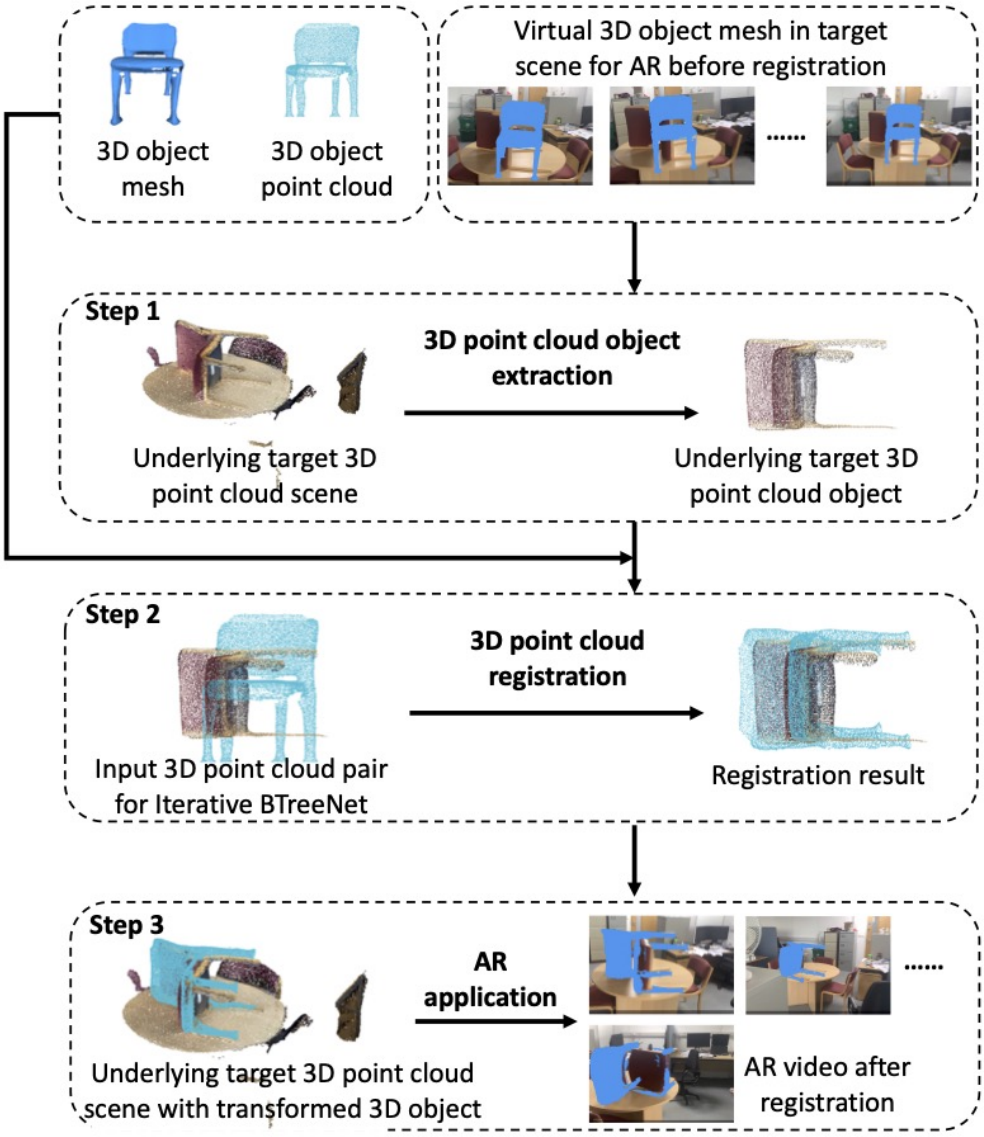


Fig. 7.3 The workflow for AR application with 3D point cloud registration.

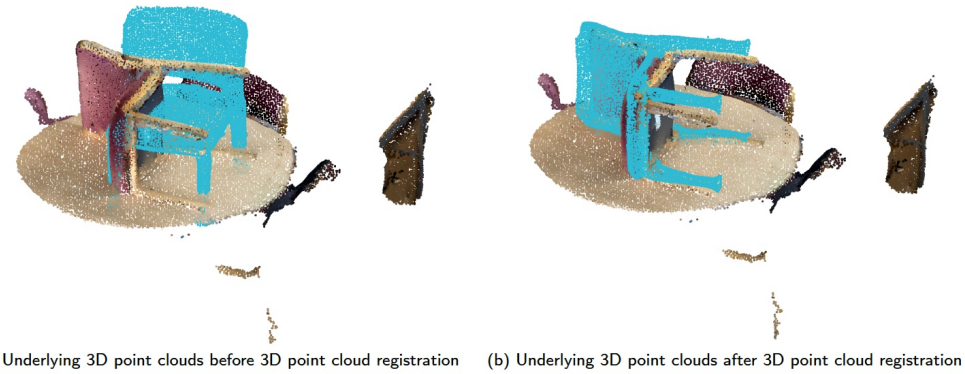
7.3.1 Overview of Registration in AR Application

Figure 7.3 shows the workflow for AR application with 3D point cloud registration. Virtual 3D object meshes and point clouds are captured by a LiDAR scanner and displayed in real-world scenes for the AR application. The virtual 3D point clouds are used for underlying registration, and the virtual 3D meshes are used for AR visualization.

During step one, the underlying target 3D point cloud scenes are captured from a LiDAR scanner. Since this thesis does not consist of 3D point cloud object detection, tracking and segmentation, the target 3D point cloud objects are extracted manually from the underlying target 3D point cloud scenes. This thesis focuses on how stable and accurate 3D point cloud registration is used in AR applications.

During step two, the virtual 3D point cloud and the underlying target 3D point cloud object are given as input for the proposed Iterative BTreeNet that transforms the virtual 3D point cloud to the underlying target 3D point cloud object. Therefore, the virtual 3D point cloud is also aligned with the underlying target 3D point cloud scenes.

During step three, the virtual 3D object mesh is transformed using the transformation matrix estimated from Iterative BTreeNet. Finally, the transformed virtual 3D object mesh is combined with the real-world scene in the AR application using ARKit 1.



(a) Underlying 3D point clouds before 3D point cloud registration (b) Underlying 3D point clouds after 3D point cloud registration

Fig. 7.4 AR application with and without using 3D point cloud registration. The proposed Iterative BTreeNet is used for 3D point cloud registration in AR applications.

7.3.2 Registration Results and AR Application

In this section, four virtual 3D objects are combined and aligned with real-world scenes in an AR application.

Figures 7.4, 7.2 and 7.5 show the AR application before and after the 3D point cloud registration for the chair and guitar, which indicates the stable and accurate 3D point cloud registration of the proposed Iterative BTreeNet. Figures 7.6 and 7.7 show the underlying processing for AR applications.

Figures 7.8 and 7.10 show the AR application before and after the 3D point cloud registration for the bottle and pillow. However, the underlying 3D point cloud bottle is severely distorted and deformed, as shown in Figure 7.9 (b), because of the LiDAR scanner technology, which results in the misalignment for AR application. In addition, the underlying 3D point cloud pillow contains the missing points, as shown in Figure 7.11 (b) and (c), on the area of the top, the left side and the bottom, which also results in the misalignment for AR application.

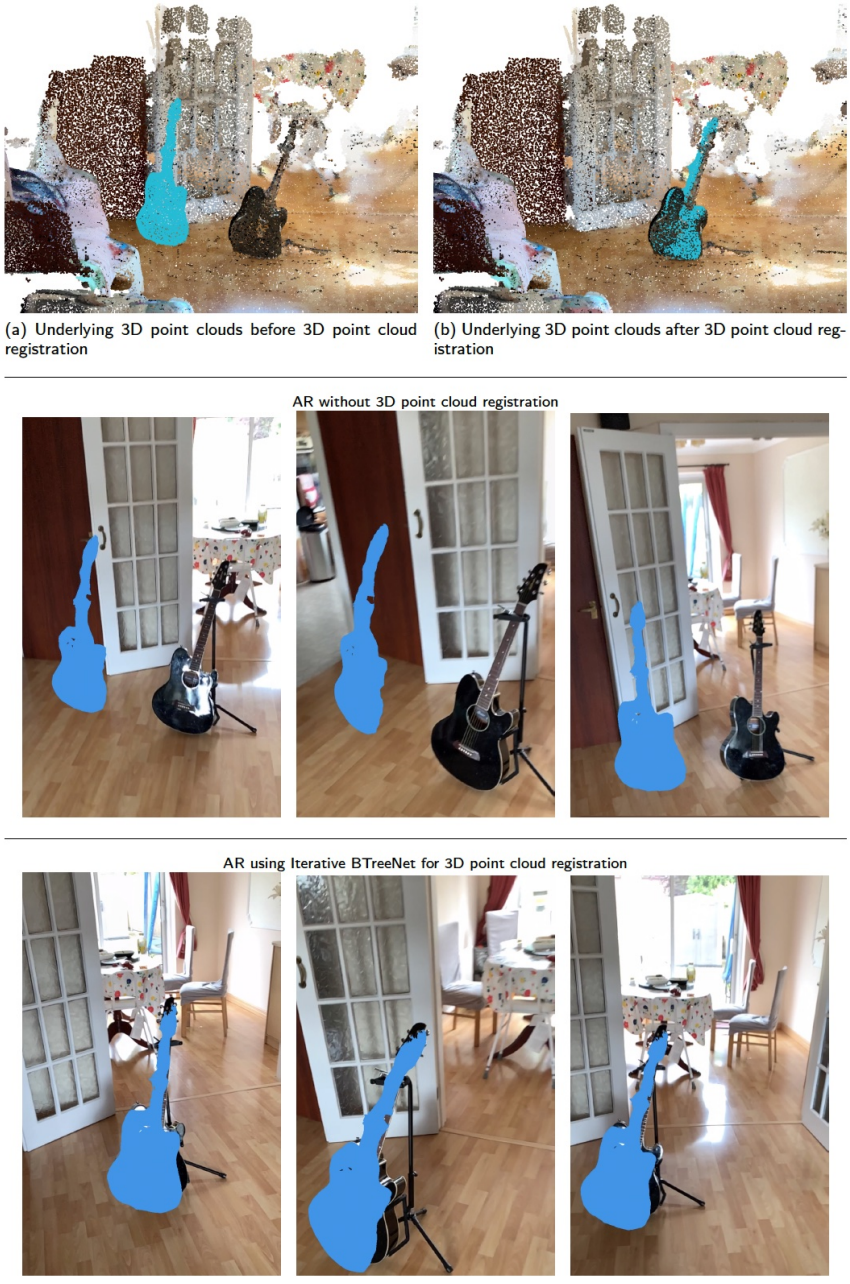
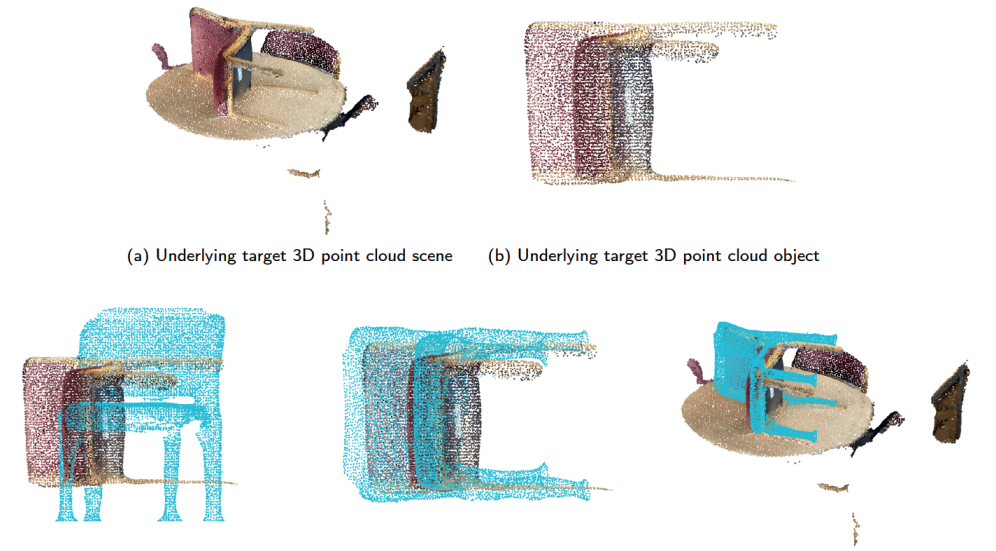


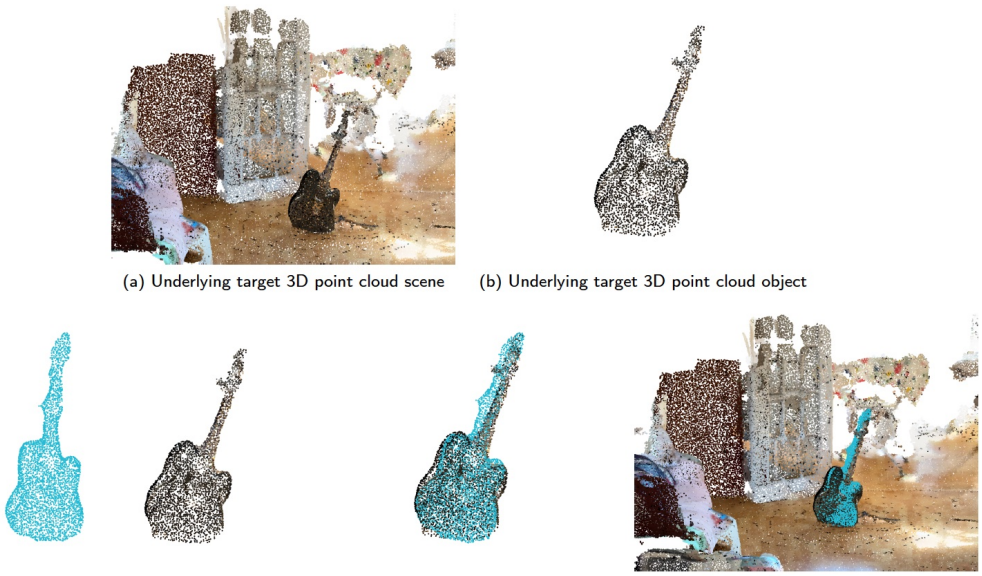
Fig. 7.5 AR application with and without using 3D point cloud registration. The proposed Iterative BTreeNet is used for 3D point cloud registration in AR applications. The images are randomly selected from the real-time AR videos with different views.



(a) Underlying target 3D point cloud scene (b) Underlying target 3D point cloud object

(c) Input for Iterative BTreeNet (d) Output from Iterative BTreeNet (e) Underlying 3D point cloud scene with transformed 3D object

Fig. 7.6 Underlying processing of the chair for AR.



(a) Underlying target 3D point cloud scene (b) Underlying target 3D point cloud object

(c) Input for Iterative BTreeNet (d) Output from Iterative BTreeNet (e) Underlying 3D point cloud scene with transformed 3D object

Fig. 7.7 Underlying processing of the guitar for AR.

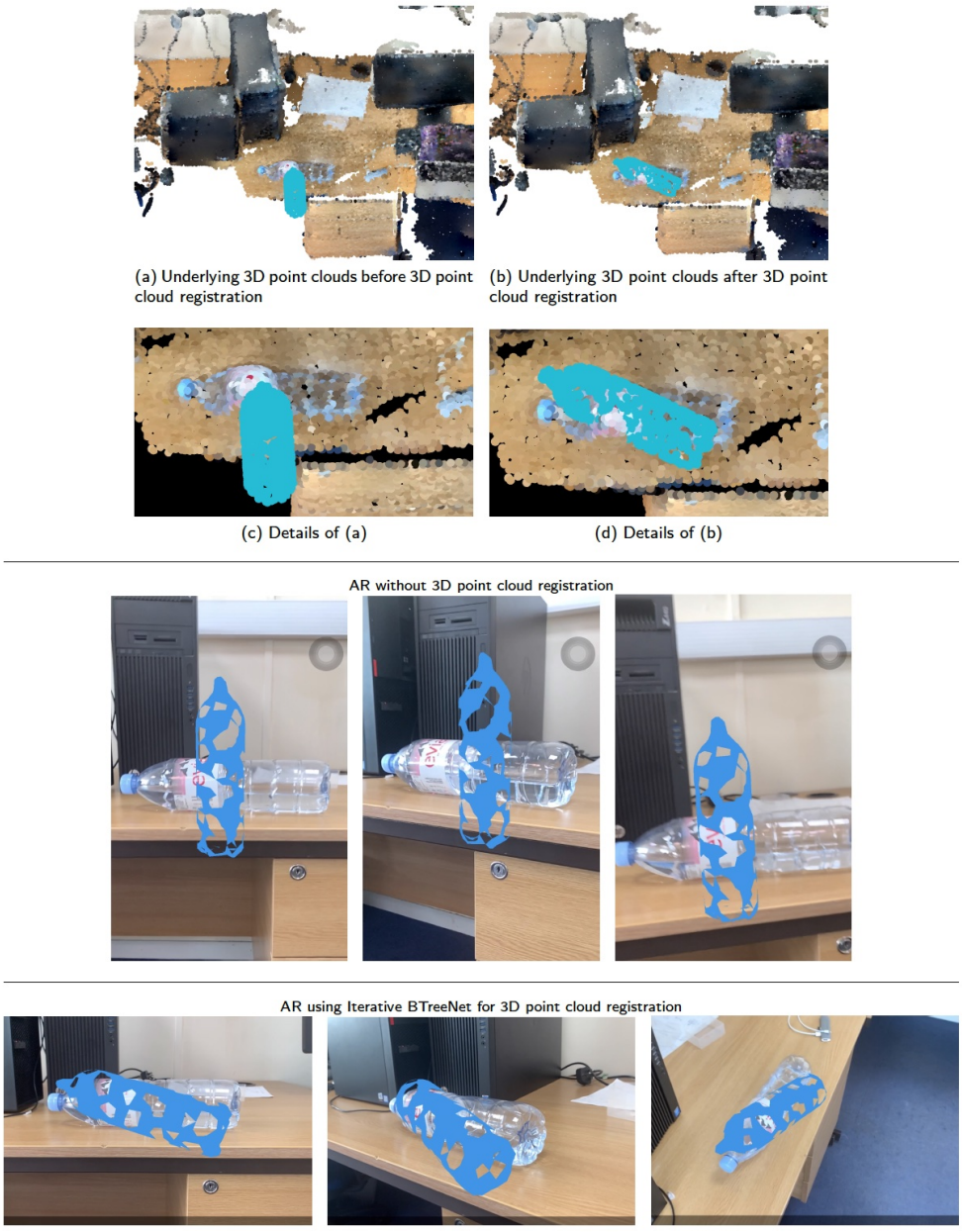


Fig. 7.8 AR application with and without using 3D point cloud registration. The proposed Iterative BTreeNet is used for 3D point cloud registration in AR applications. The images are randomly selected from the real-time AR videos with different views.

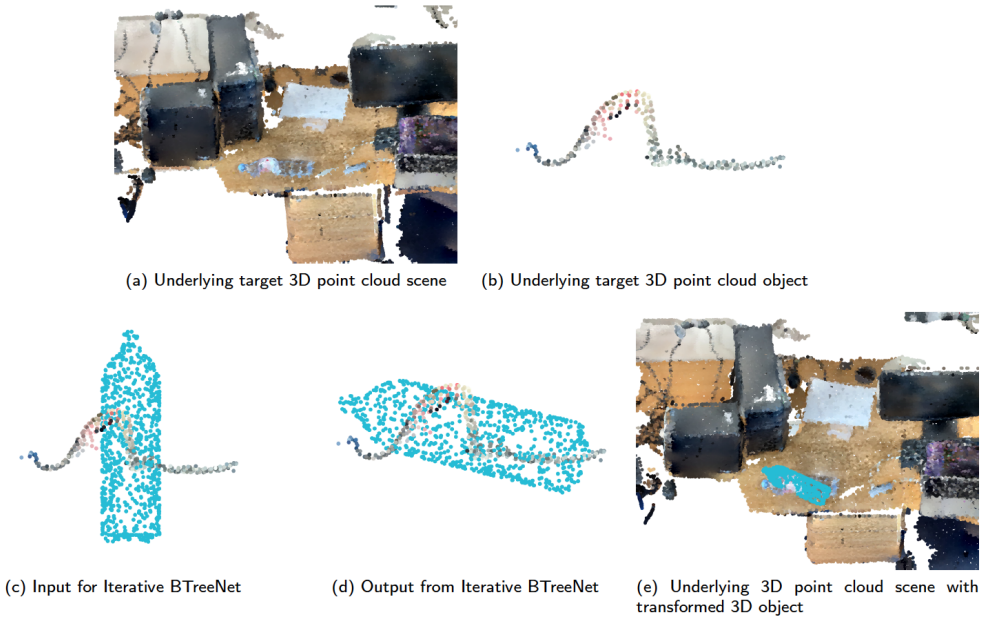


Fig. 7.9 Underlying processing of the bottle for AR.

Based on the problems of the LiDAR scanner and underlying 3D point cloud with large missing data, the 3D point cloud completion is used to achieve the high-quality 3D point clouds before registration, as illustrated in Section 7.4.

7.4 Rigid 3D Point Cloud Completion and Registration Results in AR

In this section, the 3D point cloud completion method is used to achieve high-quality 3D point clouds and improve the accuracy of the 3D point cloud registration in the AR application.

Fig. 7.10 AR application with and without using 3D point cloud registration. The proposed Iterative BTreeNet is used for 3D point cloud registration in AR applications. The images are randomly selected from the real-time AR videos with different views.

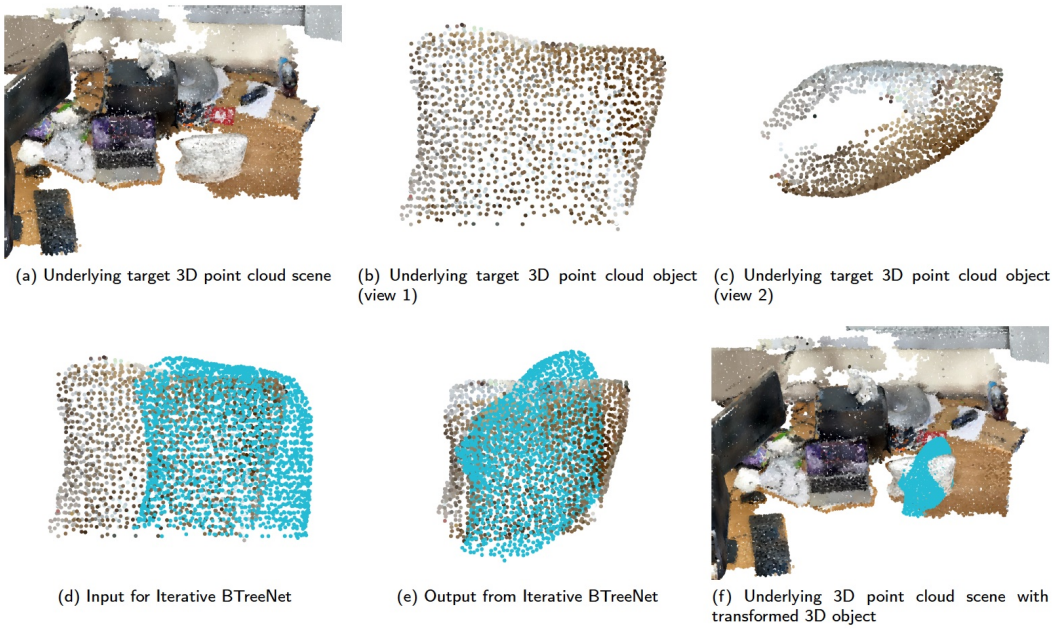


Fig. 7.11 Underlying processing of the pillow for AR.

7.4.1 Overview of Completion and Registration in AR Application

Figure 7.12 shows the workflow of the AR application with 3D point cloud completion and registration.

The step one in Figure 7.12 is similar to that of in Figure 7.3. However, the 3D LiDAR scanner performs poorly on texture-less regions (e.g. a bottle of water), which results in large missing data and severe distortion and deformation. Therefore, the 3D point cloud completion method can be used to achieve a high-quality 3D point cloud.

During step two, the proposed 3D point cloud completion network (TreeNet) is used to generate the high-quality 3D point cloud for improving registration accuracy.

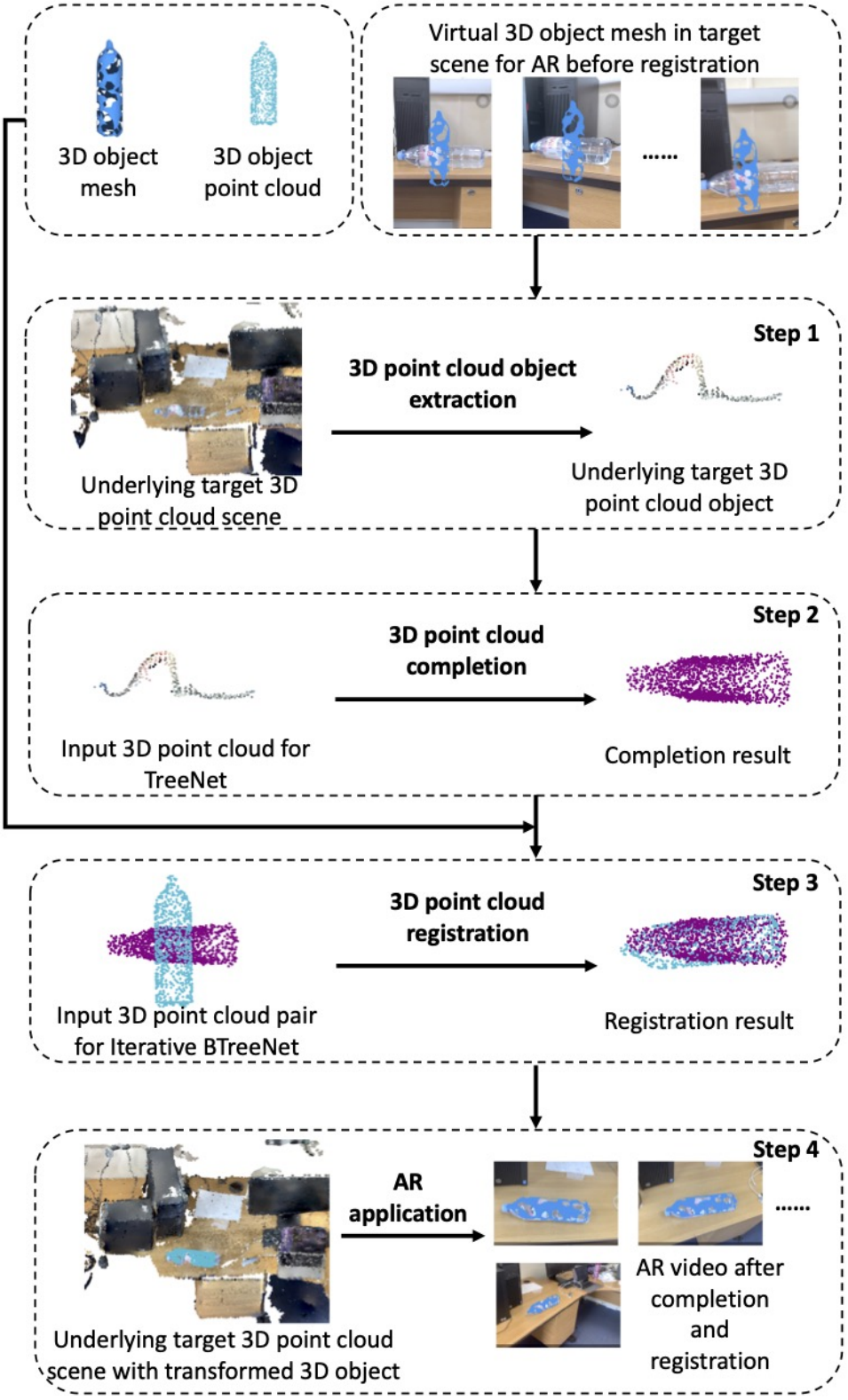


Fig. 7.12 The workflow for AR application with 3D point cloud registration.

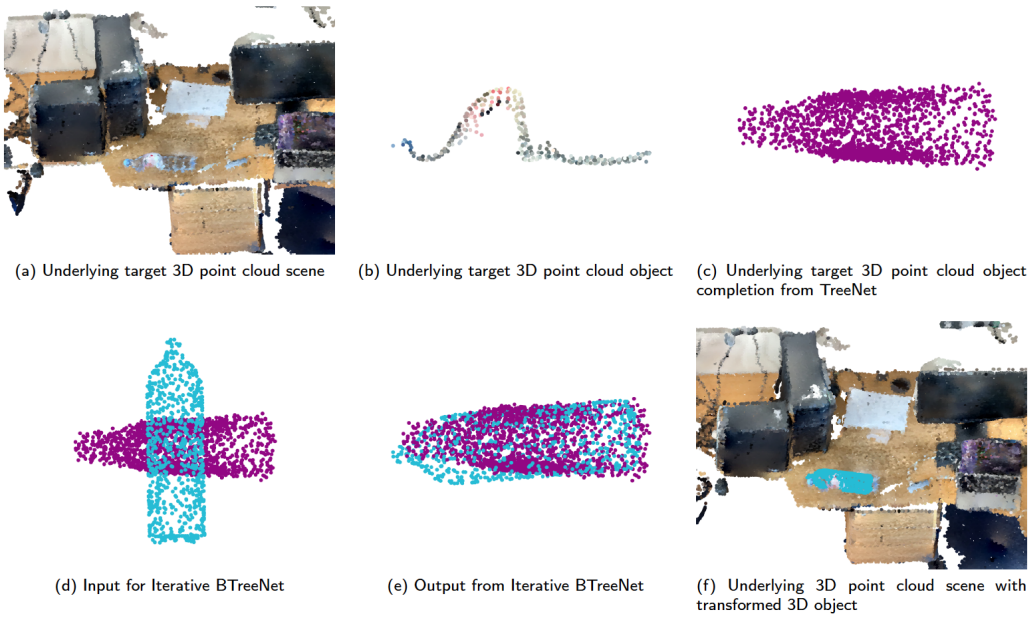


Fig. 7.13 Underlying processing of the bottle for AR.

During step three, the completion result of the target 3D point cloud object and the virtual 3D point cloud is given as input for the proposed Iterative BTreeNet, transforming the virtual 3D point cloud to the target one.

During step four, the virtual 3D object mesh is transformed using the transformation matrix estimated from Iterative BTreeNet. Finally, the transformed virtual 3D object mesh is combined with the real-world scene in the AR application

7.4.2 Completion & Registration Results and AR Application

Based on the misalignment of bottle and pillow in AR application (see Section 7.3), the 3D point cloud completion network (TreeNet) is used to achieve high-quality 3D point cloud bottle and pillow for improving registration accuracy.

Figure 7.13 (b) shows the distorted and deformed 3D point cloud bottle, and Figure 7.13 (c) shows the completion result from the proposed TreeNet that generates the complete 3D point cloud bottle from (b). Figure 7.13 (d) displays the input 3D point cloud pair for the proposed Iterative BTreeNet, and Figure 7.13 (e) displays the registration result from Iterative BTreeNet. Comparing Figure 7.9 with Figure 7.13, the 3D point cloud completion network (TreeNet) generates the complete 3D point cloud bottle and improves the registration accuracy. Figure 7.14 shows the AR application for bottles with the use of TreeNet and Iterative BTreeNet.

Figure 7.15 (b) and (c) show the underlying target 3D point cloud chair in different views, which contains missing points on the area of the top, the left side and the bottom. Figure 7.15 (d) and (e) show the completion results from the proposed TreeNet, which indicates high-quality completion results without large missing regions. Figure 7.15 (f) and (g) show the 3D point cloud registration of Iterative BTreeNet. Figure 7.16 shows the AR application with and without using 3D point cloud completion and registration. Comparing Figure 7.16 and Figure 7.10, the proposed TreeNet for 3D point cloud completion improves the registration accuracy for the stable and accurate AR application.

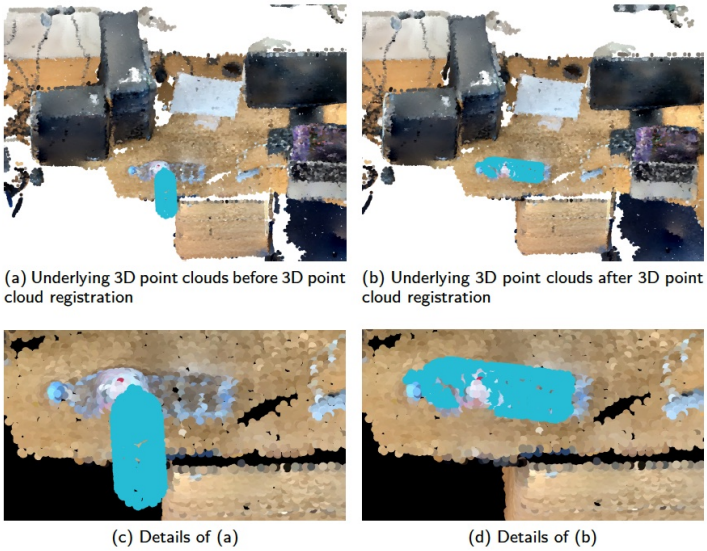
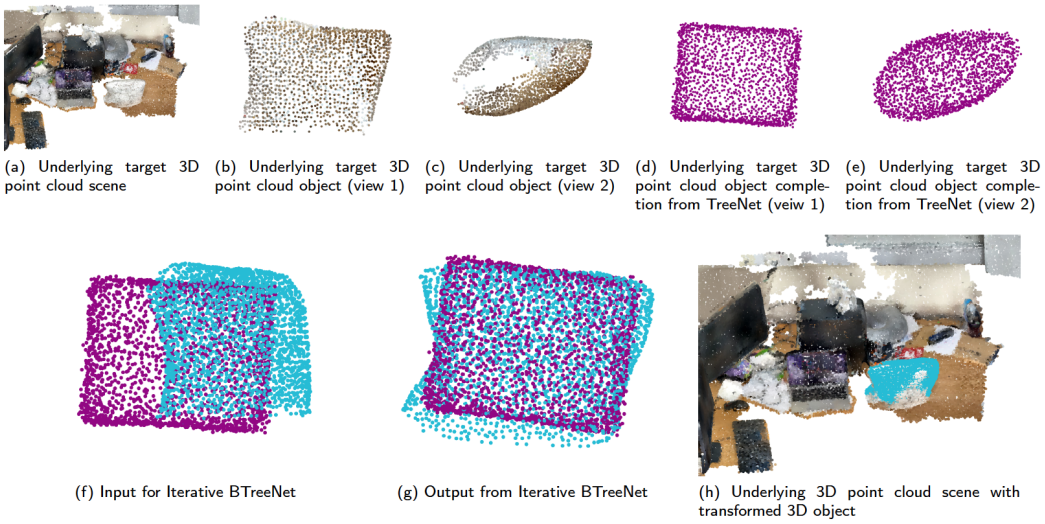


Fig. 7.14 AR application with and without using 3D point cloud registration. The proposed Iterative BTreeNet is used for 3D point cloud registration in AR applications. The images are randomly selected from the real-time AR videos with different views.



(a) Underlying target 3D point cloud scene

(b) Underlying target 3D point cloud object (view 1)

(c) Underlying target 3D point cloud object (view 2)

(d) Underlying target 3D point cloud object completion from TreeNet (veiw 1)

(e) Underlying target 3D point cloud object completion from TreeNet (view 2)

(f) Input for Iterative BTreeNet

(g) Output from Iterative BTreeNet

(h) Underlying 3D point cloud scene with transformed 3D object

Fig. 7.15 Underlying processing of the pillow for AR.

Fig. 7.16 AR application with and without using 3D point cloud registration. The proposed Iterative BTreeNet is used for 3D point cloud registration in AR applications. The images are randomly selected from the real-time AR videos with different views.

Chapter 8

Conclusions and Future Works

8.1 Conclusions

This thesis has mainly presented computational frameworks and algorithms on 3D point cloud completion, rigid 3D point cloud registration and non-rigid 3D point cloud registration, which aims to improve the quality of 3D point clouds and further achieve stable and accurate registration in AR applications.

The proposed computational framework focuses on recovering dense and high-quality 3D point clouds from single monocular images and has been evaluated on endoscopic scenes in minimally invasive surgery. The computational framework combines two main networks including a monocular depth learning network and a 3D point cloud completion network. The framework generates depth information from monocular images and repairs defects of 3D point clouds to achieve high-quality 3D point clouds. Seven large medical databases of 3D point clouds are generated from endoscopic video datasets and have been made publicly available for researchers.

The proposed TreeNet focuses on two main objectives, including multi-class 3D point cloud completion tasks and the generation of points in missing areas with the preservation of the original input structures. For the multi-class 3D point cloud completion task, the TreeNet-multiclass assigns each class of the 3D point cloud completion task to a specific sub-tree of the root node in the tree. For the generation of the missing points with the preservation of the original partial input structures, the TreeNet-binary is designed following a binary tree structure, where the left leaf node produces the reconstruction of the original partial input and the right leaf node generates points in missing areas. Once trained, the final output is the combination of the original partial input and the output of the right leaf node. Experiment results on public 3D real-world object datasets have demonstrated that the proposed TreeNet has achieved high-quality completion results and outperforms the state-of-the-art networks. The TreeNet has been also evaluated on medical data in the proposed computational framework as an additional application, which proves that TreeNet can also be used in endoscopic scenes in minimally invasive sugary.

The proposed BTreeNet and IBTreeNet are unsupervised deep learning networks for rigid 3D point cloud registration. They learn features for the rotation separately from the translation, which avoids the interference between the estimations of rotation and translation in one single matrix. IBTreeNet rotates and translates the source 3D point cloud to the target iteratively to improve registration accuracy. BTreeNet and IBTreeNet outperform state-of-the-art learning-based and traditional methods on partial, noisy, unseen large and dense point clouds, which shows that the proposed methods

exhibit remarkable generalization and robustness to 3D point clouds that are not trained.

The proposed Deform3DNet is a learning-based network for non-rigid 3D point cloud registration and correspondence. Deform3DNet considers the process of non-rigid alignment as rigidly as possible to generate the point-to-point rigid transformation. A novel non-rigid 3D registration loss function is proposed to learn the features of point-to-point transformation, which aims to align each point in the source 3D point cloud with its corresponding point in the target. The structure loss function preserves the internal structure of the transformed 3D point cloud and further improves the registration results. Deform3DNet outperforms the state-of-the-art non-learning and learning-based methods on different non-rigid 3D point cloud datasets.

The developed AR applications show the effectiveness of the proposed 3D point cloud completion and registration networks that can achieve high-quality 3D point clouds and further achieve accurate 3D point cloud registration between virtual 3D objects and real-world scenes in AR.

8.2 Future Works

Although the proposed networks have been evaluated on public datasets with promising results, there are still some issues that need to be solved in the future.

- **Future works on 3D point cloud completion**

The model size for TreeNet in Chapter 4 is gradually increasing when the number of classes increases. In future work, data in each class can be gathered semantically not geometrically, and shapes with similar semantic features pass through their corresponding sub-trees of the root node. Each sub-tree will focus on shapes with similar semantic features. This may largely reduce the number of sub-trees when the number of training classes increases drastically. The loss function for calculating the semantic features can also be considered to generate detailed information in the missing areas.

- **Future works on rigid 3D point cloud registration**

  The registration results in Chapter 5 show that state-of-the-art learning-based methods perform poorly on unseen large and dense scenes that are never trained, whereas Iterative BTreeNet exhibits remarkable generalization and robustness to this situation. However, although Iterative BTreeNet outperforms state-of-the-art learning-based methods and traditional optimization algorithms on the partial 3D point cloud, these methods still result in misalignment under large missing data. Thus, more accurate registration of partial 3D point clouds with large missing rates would be an interesting direction to pursue.

- **Future works on non-rigid 3D point cloud registration and correspondence**

  The registration results in Chapter 6 show that the traditional optimization algorithms and the proposed Deform3DNet are not able to handle non-rigid registration under large partiality. In future work, more accurate registration under large

partiality would be interesting directions to pursue. Another limitation of Deform3DNet is that it requires the point-wise correspondence of input non-rigid 3D point cloud pairs as prior knowledge. However, these point-wise correspondences are difficult to obtain from the 3D sensors (e.g. 3D LiDAR scanner). The Deform3DNet cannot be directly used in the real-world scene unless point-wise correspondences are achieved. Therefore, an unsupervised network for non-rigid 3D point cloud registration without prior knowledge of the point-wise correspondences would be necessary to pursue.

References

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

[2] Sandeep N Kundu, Nawaz Muhammad, and Fraha Sattar. Using the augmented reality sandbox for advanced learning in geoscience education. In *2017 IEEE 6th international conference on teaching, assessment, and learning for engineering (TALE)*, pages 13–17. IEEE, 2017.

[3] DWF Van Krevelen and Ronald Poelman. A survey of augmented reality technologies, applications and limitations. *International journal of virtual reality*, 9(2):1–20, 2010.

[4] Long Chen, Thomas W Day, Wen Tang, and Nigel W John. Recent developments and future challenges in medical mixed reality. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 123–135. IEEE, 2017.

[5] Long Chen, Wen Tang, and Nigel W John. Real-time geometry-aware augmented reality in minimally invasive surgery. *Healthcare technology letters*, 4(5):163–167, 2017.

[6] Long Chen, Wen Tang, Nigel W John, Tao Ruan Wan, and Jian Jun Zhang. Slam-based dense surface reconstruction in monocular minimally invasive surgery and its application to augmented reality. *Computer methods and programs in biomedicine*, 158:135–146, 2018.

[7] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen. Shape completion enabled robotic

grasping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2442–2447. IEEE, 2017.

[8] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2018.

[9] Michael Y Ni, Rex WH Hui, Tom K Li, Anna HM Tam, Lois LY Choy, Kitty KW Ma, Felix Cheung, and Gabriel M Leung. Augmented reality games as a new class of physical activity interventions? the impact of pokémon go use and gaming intensity on physical activity. *Games for health journal*, 8(1):1–6, 2019.

[10] Long Chen, Karl Francis, and Wen Tang. [poster] semantic augmented reality environment with material-aware physical interactions. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pages 135–136. IEEE, 2017.

[11] Long Chen, Wen Tang, Nigel W John, Tao Ruan Wan, and Jian J Zhang. Context-aware mixed reality: A learning-based framework for semantic-level interaction. In *Computer Graphics Forum*, volume 39, pages 484–496. Wiley Online Library, 2020.

[12] Ronald T Azuma. A survey of augmented reality. *Presence: teleoperators & virtual environments*, 6(4):355–385, 1997.

[13] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *IEEE computer graphics and applications*, 21(6):34–47, 2001.

[14] Mark Billinghurst, Adrian Clark, Gun Lee, et al. A survey of augmented reality. *Foundations and Trends® in Human–Computer Interaction*, 8(2-3):73–272, 2015.

[15] Luís Fernando de Souza Cardoso, Flávia Cristina Martins Queiroz Mariano, and Ezequiel Roberto Zorzal. A survey of industrial augmented reality. *Computers & Industrial Engineering*, 139:106159, 2020.

[16] Boykov and Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 26–33 vol.1, 2003.

[17] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[18] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[19] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[20] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[21] Long Chen, Wen Tang, Tao Ruan Wan, and Nigel W John. Self-supervised monocular image depth learning and confidence estimation. *Neurocomputing*, 381:272–281, 2020.

[22] Alberto Martín and Antonio Adán. 3d real-time positioning for autonomous navigation using a nine-point landmark. *Pattern Recognition*, 45(1):578–595, 2012.

[23] Baofu Fang, Gaofei Mei, Xiaohui Yuan, Le Wang, Zaijun Wang, and Junyang Wang. Visual slam for robot navigation in healthcare facility. *Pattern Recognition*, 113:107822, 2021.

[24] Jingfan Fan, Jian Yang, Danni Ai, Likun Xia, Yitian Zhao, Xing Gao, and Yongtian Wang. Convex hull indexed gaussian mixture model (ch-gmm) for 3d point set registration. *Pattern Recognition*, 59:126–141, 2016. Compositional Models and Structured Learning for Visual Recognition.

[25] Long Xi, Yan Zhao, Long Chen, Qing Hong Gao, Wen Tang, Tao Ruan Wan, and Tao Xue. Recovering dense 3d point clouds from single endoscopic image. *Computer Methods and Programs in Biomedicine*, 205:106077, 2021.

[26] S Giannarou, D Stoyanov, D Noonan, G Mylonas, J Clark, M Visentini-Scarzanella, P Mountney, and GZ Yang. Hamlyn centre laparoscopic/endoscopic video datasets, 2012.

[27] Peter Mountney, Danail Stoyanov, and Guang-Zhong Yang. Three-dimensional tissue deformation recovery and tracking. *IEEE Signal Processing Magazine*, 27(4):14–24, 2010.

[28] Sebastian Thrun and Ben Wegbreit. Shape from symmetry. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1824–1831. IEEE, 2005.

[29] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. A planar-reflective symmetry transform for 3d shapes. In *ACM SIGGRAPH 2006 Papers*, pages 549–559. 2006.

[30] Niloy J Mitra, Leonidas J Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (TOG)*, 25(3):560–568, 2006.

[31] Niloy J. Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. *Computer Graphics Forum*, 32(6):1–23, 2013.

[32] Mark Pauly, Niloy J Mitra, Joachim Giesen, Markus H Gross, and Leonidas J Guibas. Example-based 3d scan completion. In *Symposium on Geometry Processing*, number CONF, pages 23–32, 2005.

[33] Minhyuk Sung, Vladimir G Kim, Roland Angst, and Leonidas Guibas. Data-driven structural priors for shape completion. *ACM Transactions on Graphics (TOG)*, 34(6):1–11, 2015.

[34] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017.

[35] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 85–93, 2017.

[36] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer Graphics Forum*, volume 34, pages 435–446. Wiley Online Library, 2015.

[37] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017.

[38] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 06 2018.

[39] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018.

[40] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 383–392, 2019.

[41] Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. Learning local displacements for point cloud completion. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1558–1567, 2022.

[42] Xin Wen, Peng Xiang, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. Pmp-net: Point cloud completion by learning multi-step point moving paths. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7439–7448, 2021.

[43] Paul J. Besl and Neil D. Mckay. A method for registration of 3-d shapes. *Proceedings of Spie the International Society for Optical Engineering*, 14(3):239–256, 1992.

[44] P. Biber and W. Strasser. The normal distributions transform: a new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 3, pages 2743–2748 vol.3, 2003.

[45] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010.

[46] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey. Pointnetlk: Robust and efficient point cloud registration using pointnet. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[47] Y. Wang and J. Solomon. Deep closest point: Learning representations for point cloud registration. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[48] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11821–11830, 2020.

[49] Kexue Fu, Shaolei Liu, Xiaoyuan Luo, and Manning Wang. Robust point cloud registration framework based on deep graph

matching. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8889–8898, 2021.

[50] Xiaoshui Huang, Guofeng Mei, and Jian Zhang. Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11363–11371, June 2020.

[51] Wentao Yuan, Benjamin Eckart, Kihwan Kim, Varun Jampani, Dieter Fox, and Jan Kautz. Deepgmr: Learning latent gaussian mixture models for registration. In *European Conference on Computer Vision (ECCV)*, pages 733–750. Springer, 2020.

[52] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[53] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Piscataway, NJ, USA, June 2014. IEEE.

[54] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, aug 2004.

[55] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[56] David Droeschel and Sven Behnke. Efficient continuous-time slam for 3d lidar-based online mapping. In *IEEE International Conference on Robotics & Automation*, 2018.

[57] Yang Li, Aljaž Božič, Tianwei Zhang, Yanli Ji, Tatsuya Harada, and Matthias Nießner. Learning to optimize non-rigid tracking. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4909–4917, 2020.

[58] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-net: Towards learning based lidar localization for autonomous driving. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[59] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[60] Osamu Hirose. A bayesian formulation of coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2269–2286, 2021.

[61] Simone Melzi, Jing Ren, Emanuele Rodolà, Abhishek Sharma, Peter Wonka, and Maks Ovsjanikov. Zoomout: Spectral upsampling for efficient shape correspondence. *ACM Trans. Graph.*, 38(6), nov 2019.

[62] Gautam Pai, Jing Ren, Simone Melzi, Peter Wonka, and Maks Ovsjanikov. Fast sinkhorn filters: Using matrix scaling for non-rigid shape correspondence with functional maps. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 384–393, 2021.

[63] O. Litany, T. Remez, E. Rodola, A. Bronstein, and M. Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5660–5668, Los Alamitos, CA, USA, oct 2017. IEEE Computer Society.

[64] Jean-Michel Roufosse, Abhishek Sharma, and Maks Ovsjanikov. Unsupervised deep learning for structured shape matching. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1617–1627, 2019.

[65] Yiming Zeng, Yue Qian, Zhiyu Zhu, Junhui Hou, Hui Yuan, and Ying He. Corrnet3d: Unsupervised end-to-end learning of dense correspondence for 3d point clouds. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6048–6057, 2021.

[66] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: A flexible representation of maps between shapes. *ACM Trans. Graph.*, 31(4), jul 2012.

[67] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Computer Vision ECCV 2010*, volume 6313, pages 356–369, 09 2010.

[68] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[69] Zongji Wang and Feng Lu. Voxsegnet: Volumetric cnns for semantic part segmentation of 3d shapes. *IEEE Transactions on Visualization and Computer Graphics*, 26(9):2919–2930, 2020.

[70] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[71] Saifullahi Aminu Bello, Cheng Wang, Naftaly Muriuki Wambugu, and Jibril Muhammad Adam. Ffpointnet: Local and global fused feature for 3d point clouds analysis. *Neurocomputing*, 461:55–62, 2021.

[72] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay Sarma, Michael Bronstein, and Justin Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38:1–12, 01 2019.

[73] Guanyu Zhu, Yong Zhou, Jiaqi Zhao, Rui Yao, and Man Zhang. Point cloud recognition based on lightweight embeddable attention module. *Neurocomputing*, 472:138–148, 2022.

[74] Hongsen Liu, Yang Cong, Chenguang Yang, and Yandong Tang. Efficient 3d object recognition via geometric information preservation. *Pattern Recognition*, 92:135–145, 2019.

[75] Qian Xie, Yu-Kun Lai, Jing Wu, Zhoutao Wang, Yiming Zhang, Kai Xu, and Jun Wang. Vote-based 3d object detection with context modeling and sob-3dnms. *International Journal of Computer Vision*, 129, 06 2021.

[76] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7459–7468, 2021.

[77] J. Li, B. M. Chen, and G. Lee. So-net: Self-organizing network for point cloud analysis. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9397–9406, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[78] Fengda Hao, Rui Song, JiaoJiao Li, Kailang Cao, and Yunsong Li. Cascaded geometric feature modulation network for point cloud processing. *Neurocomputing*, 492:474–487, 2022.

[79] Yiming Cui, Xin Liu, Hongmin Liu, Jiyong Zhang, Alina Zare, and Bin Fan. Geometric attentional dynamic graph convolutional neural networks for point cloud analysis. *Neurocomputing*, 432:300–310, 2021.

[80] Haozhe Xie, Hongxun Yao, Shengping Zhang, Shangchen Zhou, and Wenxiu Sun. Pix2vox++: Multi-scale context-aware 3d object reconstruction from single and multiple images. *International Journal of Computer Vision*, 128, 12 2020.

[81] Bo Yang, Sen Wang, Andrew Markham, and Niki Trigoni. Robust attentional aggregation of deep feature sets for multi-view 3d reconstruction. *International Journal of Computer Vision*, 128, 01 2020.

[82] X. Bai, Z. Luo, L. Zhou, H. Chen, L. Li, Z. Hu, H. Fu, and C. Tai. Pointdsc: Robust point cloud registration using deep spatial consistency. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15854–15864, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society.

[83] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[84] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[85] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse iterative closest point. *Computer Graphics Forum*, 32(5), 2013.

[86] Andrew W Fitzgibbon. Robust registration of 2d and 3d point sets. *Image & Vision Computing*, 21(13–14):1145–1153, 2001.

[87] Jayakorn Vongkulbhisal, Fernando De la Torre, and João P. Costeira. Discriminative optimization: Theory and applications to point cloud registration. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3975–3983, 2017.

[88] Yan Zhao, Wen Tang, Jun Feng, TaoRuan Wan, and Long Xi. Reweighted discriminative optimization for least-squares problems with point cloud registration. *Neurocomputing*, 464:48–71, 2021.

[89] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[90] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017.

[91] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[92] Haili Chui and A. Rangarajan. A new algorithm for non-rigid point matching. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 2, pages 44–51 vol.2, 2000.

[93] Yang Yang, Sim Heng Ong, and Kelvin Weng Chiong Foong. A robust global and local mixture distance based non-rigid point set registration. *Pattern Recogn.*, 48(1):156–173, jan 2015.

[94] F.L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE PAMI)*, 11(6):567–585, 1989.

[95] Paul J. Besl and Neil D. Mckay. A method for registration of 3-d shapes. *Proceedings of Spie the International Society for Optical Engineering*, 14(3):239–256, 1992.

[96] A.L. Yuille and N.M. Grzywacz. The motion coherence theory. In *[1988 Proceedings] Second International Conference on Computer Vision (ICCV)*, pages 344–353, 1988.

[97] Alan L Yuille and Norberto M Grzywacz. A mathematical analysis of the motion coherence theory. *International Journal of Computer Vision (IJCV)*, 3, 1989.

[98] Artiom Kovnatsky, Michael M. Bronstein, Xavier Bresson, and Pierre Vandergheynst. Functional correspondence by matrix completion. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 905–914, 2015.

[99] Yonathan Aflalo, Anastasia Dubrovina, and Ron Kimmel. Spectral generalized multi-dimensional scaling. *International Journal of Computer Vision (IJCV)*, 118, 2016.

[100] Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. Topological function optimization for continuous shape matching. *Computer Graphics Forum*, 37(5):13–25, 2018.

[101] Jing Ren, Adrien Poulenard, Peter Wonka, and Maks Ovsjanikov. Continuous and orientation-preserving correspon-

dences via functional maps. *ACM Trans. Graph.*, 37(6), dec 2018.

[102] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS) - Volume 2*, NIPS'13, page 2292–2300, Red Hook, NY, USA, 2013. Curran Associates Inc.

[103] Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Trans. Graph.*, 34(4), jul 2015.

[104] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[105] Nazim Haouchine, Jeremie Dequidt, Igor Peterlik, Erwan Kerrien, Marie-Odile Berger, and Stéphane Cotin. Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery. In *2013 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 199–208. IEEE, 2013.

[106] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[107] Lubor Ladicky, Jianbo Shi, and Marc Pollefeys. Pulling things out of perspective. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 89–96, 2014.

[108] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2015.

[109] Swaminathan Gurumurthy and Shubham Agrawal. High fidelity semantic shape completion for point clouds using latent

optimization. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1099–1108. IEEE, 2019.

[110] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

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

[112] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[113] Micha Pfeiffer, Isabel Funke, Maria R Robu, Sebastian Bodenstedt, Leon Strenger, Sandy Engelhardt, Tobias Roß, Matthew J Clarkson, Kurinchi Gurusamy, and Brian R and Davidson. Generating large labeled data sets for laparoscopic image processing tasks using unpaired image-to-image translation. 2019.

[114] Veronica Penza, Andrea S Ciullo, Sara Moccia, Leonardo S Mattos, and Elena De Momi. Endoabs dataset: Endoscopic abdominal stereo image dataset for benchmarking 3d stereo reconstruction algorithms. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 14(5):e1926, 2018.

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

[116] Francis Engelmann, Konstantinos Rematas, Bastian Leibe, and Vittorio Ferrari. From points to multi-object 3d reconstruction. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4586–4595, 2021.

[117] Chuanchuan Chen, Dongrui Liu, Changqing Xu, and Trieu-Kien Truong. Genecgan: A conditional generative adversarial network based on genetic tree for point cloud reconstruction. *Neurocomputing*, 462:46–58, 2021.

[118] Xupeng Wang, Mumuxin Cai, Ferdous Sohel, Nan Sang, and Zhengwei Chang. Adversarial point cloud perturbations against 3d object detection in autonomous driving systems. *Neurocomputing*, 466:27–36, 2021.

[119] Keisuke Yoneda, Hossein Tehrani, Takashi Ogawa, Naohisa Hukuyama, and Seiichi Mita. Lidar scan feature for localization with highly precise 3-d map. In *Intelligent Vehicles Symposium*, 2014.

[120] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[121] Zhen Dong, Bisheng Yang, Yuan Liu, Fuxun Liang, Bijun Li, and Yufu Zang. A novel binary shape context for 3d local surface description. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130:431–452, 2017.

[122] Zhen Dong, Bisheng Yang, Fuxun Liang, Ronggang Huang, and Sebastian Scherer. Hierarchical registration of unordered tls point clouds based on binary shape context descriptor. *ISPRS Journal of Photogrammetry and Remote Sensing*, 144:61–79, 2018.

[123] Zhen Dong, Fuxun Liang, Bisheng Yang, Yusheng Xu, Yufu Zang, Jianping Li, Yuan Wang, Wenxia Dai, Hongchao Fan, Juha Hyyppä, and Uwe Stilla. Registration of large-scale terrestrial laser scanner point clouds: A review and benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 163:327–342, 2020.

[124] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *2017*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 199–208, 2017.

[125] Marc Levoy Greg Turk. The stanford 3d scanning repository. In *Stanford University Computer Graphics Laboratory*, 2005.

[126] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 06 2015.

[127] G. Turk and B. Mullins. Large geometric models archive. *Georgia Inst. Technology*, 2008.

[128] Jie Yang, Lin Gao, Qingyang Tan, Huang Yihua, Shihong Xia, and Yu-Kun Lai. Multiscale mesh deformation component analysis with attention-based autoencoders. *IEEE transactions on visualization and computer graphics*, pages 1–1, 2021.

[129] Kangkan Wang, Sida Peng, Xiaowei Zhou, Jian Yang, and Guofeng Zhang. Nerfcap: Human performance capture with dynamic neural radiance fields. *IEEE transactions on visualization and computer graphics*, pages 1–13, 2022.

[130] Yang Li, Aljaž Božič, Tianwei Zhang, Yanli Ji, Tatsuya Harada, and Matthias Nießner. Learning to optimize non-rigid tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4909–4917, 2020.

[131] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, 2009.

[132] Z. Lähner, E. Rodolà, M. M. Bronstein, D. Cremers, O. Burghard, L. Cosmo, A. Dieckmann, R. Klein, and Y. Sahillioglu. Shrec'16: Matching of deformable shapes with topological noise. In *Proc. of Eurographics Workshop on 3D Object Retrieval (3DOR)*, 2016.

[133] Long Xi, Wen Tang, Tao Xue, and TaoRuan Wan. Iterative btreenet: Unsupervised learning for large and dense 3d point cloud registration. *Neurocomputing*, 506:336–354, 2022.

[134] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc.

[135] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay Sarma, Michael Bronstein, and Justin Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38:1–12, 01 2019.

[136] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.