BEN-GURION UNIVERSITY OF THE NEGEV
FACULTY OF ENGINEERING SCIENCES
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DeepUME: Learning the Universal Manifold Embedding for Robust Point Cloud Registration

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE M.Sc DEGREE

By: Natalie Lang

July 18, 2021

Abstract

Registration of point clouds related by rigid transformations is one of the fundamental problems in computer vision. However, a solution to the practical scenario of aligning differently sampled noisy observations of the point clouds is still lacking. We approach registration in this scenario with a fusion of the Universal Manifold Embedding (UME) method and an unsupervised deep neural network. In order to overcome a major obstacle in the learning process under full rotation range, we employ an SO(3)-invariant coordinate system to learn SO(3)-invariant features, later to be utilized by the closed-form geometric UME method for transformation estimation. We evaluate the performance of the proposed method using the standard RMSE metric as well as using two alternative metrics, designed to overcome the ambiguity problem emerging in ModelNet40 dataset, when noisy scenarios are considered. Finally, we show that our hybrid method outperforms state-of-the-art registration methods in various scenarios, and generalizes well to unseen datasets.

Keywords

Rigid Transformation, Registration, Parameter Estimation, 3D Computer Vision, 3D point clouds, 3D feature learning, Neural networks for 3D data analysis

Acknowledgments

This work was supported in part by the Israel Ministry of Innovation, Science and Technology under grant no. 3-16583, and by NSF-BSF Computing and Communication Foundations (CCF) grants, CCF-2016667, CCF-1712788 and BSF-2016667.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The massive development of 3D range sensors [Collis, 1970; Smisek et al., 2013] led to an intense interest in 3D data analysis. As 3D data is commonly acquired in the form of a point cloud, many related applications have been studied in recent years for that data form. In wide range of applications, specifically in medical imaging [Hill et al., 2001], autonomous driving [Bresson et al., 2017] and robotics [Durrant-Whyte and Bailey, 2006], the alignment of 3D objects into a coherent world model is a crucial problem. Point cloud rigid alignment is a deep-rooted problem in computer vision and graphics, and various methods for point cloud registration have been suggested [Pomerleau et al., 2015].

In general, the point clouds to be registered are sampled from a physical object. When two point clouds are sampled at two different poses of an object, it is unlikely that the same set of object points is sampled in both. The difference between the sampling patterns of the object may result in model mismatch in performing the registration, and we therefore refer to it as sampling noise.

Registration of point clouds in the presence of noise has been extensively studied, both by classical methods [Besl and McKay, 1992a; Yang et al., 2015; Zhou et al., 2016; Rusinkiewicz and Levoy, 2001] and by learning based methods [Aoki et al., 2019; Wang and Solomon, 2019a; Yuan et al., 2020; Fu et al., 2021]. In most of these works, the noise is modeled as an Additive White Gaussian Noise (AWGN) on the coordinates. However, this type of model is inadequate for modeling sampling noise. As we show in our experiments, sampling noise affects the registration error differently than additive noise on the coordinates of the point cloud and introduces large registration errors.

It is important to comment that the registration under sampling noise scenarios can be accurately solved when dense point clouds are considered; the denser are the point clouds, the lower the effect sampling noise has on their registration. However, as the size of the point cloud grows, so does the demands of the registration task. Reducing the number of points is crucial in many aspects, such as reduction of power consumption, computational cost and communication load, to name a few. This is often done by processing under-sampled point clouds. Therefore, in many point cloud registration applications, under-sampled point clouds are being used. As such, they are highly sensitive to sampling noise that may severely damage their registration (see experiments chapter). Considering this practical observation, our goal is to solve the under-sampled point cloud registration problem under sampling noise scenarios. In fact, this is the actual situation encountered in parts of a real sample, since actual scanning of 3D data highly depends on the 3D shape of the scanned object and on the relative position of the scanner. Here we are examining the problem and its solution on a specific experimental framework of object point clouds.

More specifically, in this work, we address registration of 3D point clouds in the presence of sampling noise and an additive coordinate noise. Our strategy is to combine the closed-form Universal Manifold Embedding (UME) registration method [Efraim and Francos, 2019], and a learning based framework. The UME nonlinearly maps functions related by geometric transformations of coordinates (rigid, in our case) to matrices that are linearly related by the transformation parameters. In the UME framework, the embedding of the orbit of possible observations on the object to the space of matrices is based on constructing an operator that evaluates a sequence of low-order geometric moments of some function defined on the point clouds to be registered. This representation is therefore more resilient to noise than local operators, as under reasonable noise, the geometric structure of the point cloud is preserved. Since the UME is an operator defined on functions of the coordinates, in order to enable registration, these functions need to be invariant to the transformation. We generate such invariant features using an unsupervised deep neural network architecture, which is based on DCP [Wang and Solomon, 2019a]. The framework is explained in details in Section 5. We train our framework on ModelNet40 [Wu et al., 2015] dataset, and test on both seen datasets (ModelNet40), and two unseen dataset (FAUST [Bogo et al., 2014] and Stanford 3D Scanning Repository [Stanford Scanning Repository, 1994]).

Our main contributions are as follows:

- We address the highly practical, yet less studied problem of point cloud registration in the presence of sampling noise. We point out an ambiguity problem in ModelNet40 dataset that emerges in this scenario and propose alternative error measures that eliminate this ambiguity.

- We integrate the UME registration methodology for the first time into a DNN framework. With the resulting hybrid framework, we show that a successful registration is achievable for the full range of rotation angles and subject to various types of noise, outperforming the compared methods in all evaluated scenarios and metrics.

Chapter 2

Related Work

Point cloud registration has remained a popular research area for many years due to its challenging nature and common presence as an important component of many 3D perception applications. Here we broadly categorize closed-form prior work into local and global techniques, and categorize the emerging learning-based methods into feature-learning methods and end-to-end learning registration.

2.1 Closed-form Registration Methods

2.1.1 Local Registration

Local approaches are often highly efficient and can be highly effective if limited to regimes where transformations are known a priori to be small in magnitude. These refinement algorithms employ a numerical optimization to iteratively minimize an objective function that measures the distance between points in the observation and the assumed correspondence points in the reference model [Bookstein, 1989; Zhang, 1994a], or between points in the observation and the surface of the model [Bookstein, 1989; Pottmann et al., 2004a; Montemerlo et al., 2008].

The Iterative Closest Point algorithm (ICP) [Bookstein, 1989; Zhang, 1994a] is the standard algorithm in this category. ICP iteratively alternates between two phases: point-to-point correspondence and distance minimization. Over the years, many variants of the ICP algorithm have been proposed in attempt to improve the convergence rate, robustness and accuracy of the algorithm. In [Chetverikov et al., 2002] for example, not all points in both models are taken into account, and the number of obtained matches is trimmed according to an assumed overlap between the models undergoing registration. This approach overcomes bias in the registration that is due to the closest point paradigm failure (*i.e.*, when there is a partial overlap the closest point may be in fact very far). Other variants of ICP attempt to improve the formation of correspondences by including additional information such as surface normals, or varying the direction of closest point search *etc.* (see [Rusinkiewicz and Levoy, 2001] for a review of ICP variants). By definition, the ICP, like any other local iterative numerical optimization method ([Pottmann et al., 2004a; Montemerlo et al., 2008; Zabulis et al., 2018; Tam et al., 2012]) requires a good initial alignment, otherwise registration may converge to a local minimum of the objective function. In addition, as presented in [Rusinkiewicz and Levoy, 2001], not all variants are appropriate to all data types, and the algorithm should be matched to the data type for optimal performance.

Another branch of work on local methods concerns probabilistic registration, often via the use of GMMs and the EM algorithm [Dempster et al., 1977]. Traditional examples include EM-ICP [Granger and Pennec, 2002], GMMReg [Jian and Vemuri, 2010], and methods based on the Normal Distributions Transform (NDT [Stoyanov et al., 2012]). More recent examples offer features such as batch registration (JRMPC [Evangelidis and Horaud, 2017]) or robustness to density variance and viewing angle (DARE [Lawin et al., 2018]).

Other recent approaches have focused on efficiency, including filter-based methods [Gao and Tedrake, 2019], GPU-accelerated hierarchical methods [Eckart et al., 2018], Monte Carlo methods [Dhawale et al., 2018] or efficient distribution-matching techniques [Tabib et al., 2018].

2.1.2 Global Registration

Unlike local methods, global methods are invariant to initial conditions and aimed at estimating large deformations, but often at the cost of efficiency. Some approaches exhaustively search SE(3) via branch-and-bound techniques (Go-ICP [Yang et al., 2015], GOGMA [Campbell and Petersson, 2016] and GOSMA [Campbell et al., 2019]). Other approaches use 3D local features matching adaptations of the 2D image processing solutions, such as variants of 3D-SIFT [Darom and Keller, 2012; Maes et al., 2010a] and the Harris key-points detector [Sipiran and Bustos, 2011a].

In 3D, with the absence of a regular sampling grid, artifacts, sampling noise and the challenging nature of salient geometry (edges are not common in 3D as in images, for example), key-points matching is prone to high outlier rates and localization errors. Hence, the alignment estimated by key-points matching usually employs outlier rejection methods such as RANSAC [Fischler and Bolles, 1981] or semi-definite programming [Yang et al., 2020a], and is followed by a refinement stage using local optimization algorithms [Bookstein, 1989; Zhang, 1994a; Pottmann et al., 2004a; Montemerlo et al., 2008].

One notable exception to the general rule that global methods are often inefficient, is Fast Global Registration (FGR) [Zhou et al., 2016], which achieves invariance to initial pose while remaining as fast or faster than many local methods.

Global registration methods are not restricted only to methods based on the extraction and matching of key-points. In [Isola et al., 2011], for example, an initial alignment is found by employing a matched filter in the frequency space of local orientation histograms. In [Ferencz and Shimshoni, 2017a] an initial alignment is found by clustering the orientations of local point cloud descriptors and estimating the relative rotation between clusters.

In [Aiger et al., 2008a], the algorithm searches congruent sets of four co-planar points between point clouds to create point correspondences. In [Bülow and Birk, 2018], a global registration procedure based on Fourier-Mellin transform is derived. It is a three step procedure where an SO(3) Fourier transform implemented using spherical harmonics is employed to estimate the rotation. In the second step using the Mellin transform the scale is estimated, and finally the translation.

A different approach is to approximate the surface using Gaussian Mixture Models (GMM) and perform registration on the GMM models rather than directly on the point clouds, [Seo and Milanfar, 2010; Zhu et al., 2014; Campbell and Petersson, 2016]. The GMM modelling methods tend to be computationally expensive as each point in the point cloud is assumed to be the center of a model component. Another drawback is that the final result depends on the model initialization which is usually random.

Super4PCS [Mellado et al., 2014a] is a global registration solution that incorporates local geometric constraints on the considered key-points. It is a modification of [Aiger et al., 2008a] that yields accurate results without employing ICP-like fine alignment step. It is based on searching for congruent sets of 4 points between the models being processed to create point correspondence. While some of the above mentioned methods may achieve sufficiently accurate results without employing a fine alignment algorithm, the accuracy of the final result is usually improved using an appropriate variant of ICP.

In this work a global closed-form solution that employs the UME representation of the shapes to be registered, is being used and detailed in chapter 4. As a result, an efficient and accurate registration scheme is achieved where no initial alignment is required.

2.1.3 Closed-form solution for registration of point clouds from known correspondences

Closed form solutions for rigid motion registration dates back to the 1980's where solutions using unit quaternions to minimize a least square problem were introduced by [Faugeras and Hebert, 1986] with an alternative formulation published by [Horn, 1987]. An equivalent solution using orthonormal matrices that was later introduced by [Horn et al., 1988b], is being exploited by this work and is summarized below.

[Horn, 1987; Horn et al., 1988b] derived a closed form solution for the problem of recovering the transformation between two sets of corresponding points in different Cartesian coordinate systems *i.e.*

$$\mathbf{r}_{r,i} = s\mathbf{R}\mathbf{r}_{l,i} + \mathbf{r}_0, \ i = 1, ..., n \tag{2.1}$$

where $\{\mathbf{r}_{r,i}\}_{i=1}^{n}$ is the "right hand" (or stationary) set of points corresponding to $\{\mathbf{r}_{l,i}\}_{i=1}^{n}$ the "left hand" (or moving) set of points by scale s, rotation $\mathbf{R}$ and translation $\mathbf{r}_0$. Minimizing the sum of squared errors

$$\sum_{i=1}^{n} ||\mathbf{e}_i||^2 = \sum_{i=1}^{n} ||\mathbf{r}_{r,i} - s\mathbf{R}\mathbf{r}_{l,i} - \mathbf{r}_0||^2 \tag{2.2}$$

Horn shows that the translation $\mathbf{r}_0$ to minimize the sum of squared errors is the difference between the centroid of $\{\mathbf{r}_{r,i}\}_{i=1}^{n}$ and the rotated centroid of $\{\mathbf{r}_{l,i}\}_{i=1}^{n}$, *i.e.*

$$\mathbf{r}_0 = \frac{1}{n}\sum_{i=1}^{n}\mathbf{r}_{r,i} - \frac{1}{n}\mathbf{R}\sum_{i=1}^{n}\mathbf{r}_{r,i}. \tag{2.3}$$

The rotation to minimize the sum of square errors is shown to be the matrix to maximize the trace of $\mathbf{R}^T\mathbf{M}$, where

$$\mathbf{M} = \sum_{i=1}^{n}\mathbf{r}'_{r,i}, \mathbf{r}'^{T}_{l,i} \tag{2.4}$$

and $\mathbf{r'}_{r,i}, \mathbf{r'}_{l,i}$ represents the centered versions of $\mathbf{r}_{r,i}, \mathbf{r}_{l,i}$. $\mathbf{M}$ is shown to have a decomposition $\mathbf{M} = \mathbf{U}\mathbf{S}$ where

$$\mathbf{U} = \mathbf{M}(\mathbf{M}^T\mathbf{M})^{-1/2} \tag{2.5}$$

is a unitary matrix and $\mathbf{S}$ is a positive semi-definite matrix. It is then shown by analysis of eigenvalues of the matrix $\mathbf{M}$ that $\mathbf{R} = \mathbf{U}$ is the matrix minimizing the sum of squared errors.

2.2 Learned Methods for Rigid Motion Registration

In the following we give an introduction to learned-based methods categorized as feature-learning methods and end-to-end learning registration, and analyze their advantages and limitations. We finally summarize the key ideas and review the recent critical development of each category.

2.2.1 Feature learning methods for registration

Unlike the classical optimization-based registration methods, feature learning methods [Zeng et al., 2017; Deng et al., 2018b; Gojcic et al., 2019] use a deep neural network to learn a robust feature correspondence search. Then, the transformation can be estimated using one-step optimization (e.g. SVD or RANSAC) without iterating between correspondence estimation and transformation estimation.

For instance [Zeng et al., 2017] uses AlexNet to learn a 3D feature from an RGB-D dataset and [Deng et al., 2018b] proposes a local PPF (Point-Pair-Features) by using the distribution of neighbour points that serves as the network input for deep feature learning. These methods are using deep learning as a feature extraction tool. By developing sophisticated network architectures or loss functions, they aim to estimate robust correspondences by the learned distinctive feature.

The advantages of this category are two folds:

1. Deep learning-based point feature could provide robust and accurate correspondence searching.

2. The accurate correspondences could lead to accurate registration results by using a simple RANSAC method.

The limitations of this kind of methods are three aspects:

1. They need large training data.

2. The registration performance drops dramatically in unknown scenes if the scenes have a large distribution difference to the training data.

3. They use a separated training process to learn a stand-alone feature extraction network. The learned feature network is to determine point-point matching other than registration.

Learning on point clouds

In the following we review several feature-learning for point cloud registration methods, all aim at design advanced neural networks to extract distinctive features.

PPFNet [Deng et al., 2018b] is a neural network that uses a point pair feature (PPF) [Drost et al., 2010] to pre-process the input point cloud patches for achieving rotation invariance. Then, the point clouds are input into a PointNet [Qi et al., 2017a], to extract a local feature and a global feature that is obtained by applying a max-pooling operation. Both the global and local features are input in an MLP block to generate the final correspondence search feature. Thus, instead of feeding the network with volumetric data, it learns local descriptors on pure geometry while being highly aware of the global context. The limitation is that it requires a large amount of annotation data. To solve this issue, PPF-FoldNet [Deng et al., 2018a] proposes an unsupervised method to remove the annotation requirement constraint. The basic idea is to use PointNet to encode a feature and use a decoder to decode the feature into data, to be the same as the input. The whole network is optimized by using the difference between the input and output using Chamfer loss. Similarly, SiamesePointNet [Zhou et al., 2020] produces the descriptor of interest points by a hierarchical encoder-decoder architecture.

By not requiring manual annotation of matching point cluster, 3DFeat-Net [Yew and Lee, 2018] introduces a weakly-supervised approach that leverages alignment and attention mechanisms to learn feature correspondences from GPS/INS tagged 3D point clouds without explicitly specifying them. More specifically, the network takes a set of triplets containing an anchor, positive and negative point cloud. They train the neural network with the triplet loss by minimizing the difference between the anchor and positive point clouds while maximizing the difference between the anchor and negative point clouds.

Alignment [Groß et al., 2019] focuses on the partially observed object alignment by using a tracking framework, which is trying to estimate the object-centric relative motion. Moreover, this approach uses a neural network that takes the noisy 3D point segments of objects as input to estimate their motion instead of approximating targets with their centre points. [Yang et al., 2020c] utilizes both the colour and spatial geometric information to solve the point cloud registration.

Other DNN framework methods are inspired by the local closed-form ICP registration algorithm. Since the ICP requires hard assignments of closest points, it is sensitive to the initial transformation and noisy/outliers. Therefore, the ICP usually converges to the wrong local minima. RPMNet [Yew and Lee, 2020] introduces a less sensitive to initialization and more robust deep learning-based approach for rigid point cloud registration. This method's network can get a soft assignment of point correspondences and can solve the point cloud partial visibility. The deep closest point (DCP) [Wang and Solomon, 2019a] employs a dynamic graph convolutional neural network (DGCNN [Wang et al., 2019b]) for feature extraction and an attention module to generate a new embedding that considers the relationships between two point clouds. Besides, a singular value decomposition module is used to calculate rotation and translation.

IDAM [Li et al., 2020] incorporates both geometric and distance features into the iterative matching process. Point matching involves computing a similarity score based on the entire concatenated features of the two points of interest. [Yang et al., 2020b] find that more compact and distinctive representations can be achieved by optimizing a neural network (NN) model under the triplet framework that non-linearly fuses local geometric features in Euclidean spaces. The NN model is trained by an improved triplet loss function that fully leverages all pairwise relationships within the triplet. Moreover, they claimed that their fused descriptor is also competitive to deeply learned descriptors from raw data while being more lightweight and rotational invariant.

2.2.2 End-to-end learning-based registration methods

The end-to-end learning-based methods solve the registration problem with an end-to-end neural network. The input of this category is two point clouds, and the output is the transformation matrix to align these two point clouds. The transformation estimation is embedded into the neural network optimization, which is different from the above feature-learning methods, whose focus is point feature learning. The neural network optimization is separated from the transformation estimation.

End-to-end learning methods transform the registration problem into a regression problem. For example, [Yang et al., 2019] tries to learn a feature from the point clouds to be aligned and then regresses the transformation parameters from the feature. [Wang et al., 2019a] proposes a registration network to formulate the correlation between source and target point sets and predict the transformation using the defined correlation. [Elbaz et al., 2017] proposes an auto-encoder registration network for localization, which combines super points extraction and unsupervised feature learning. [Lu et al., 2019] proposes a key-point detection method and estimates the relative pose simultaneously. FMR [Huang et al., 2020] proposes a feature-metric registration method, which converts the registration problem from the previous minimizing point-point projection error to minimizing feature difference. This method is a pioneer work of feature-metric registration by combining deep learning and the conventional Lucas-Kanade optimization method.

The advantages of this category are two folds:

1. The neural network specifically designs and optimizes for registration task.
2. It could leverage both the merits of conventional mathematical theories and deep neural networks.

The limitations of current methods are two aspects:

1. The regression methods regard transformation parameter estimation as a black box, and the distance metric is measured in the coordinate-based Euclidean space, which is sensitive to noise and density difference.
2. The feature-metric registration method does not consider the local structure information, which is crucial for registration.

The end-to-end learning-base registration methods can be divided into two categories:

1. Considering the registration as a regression problem and using the neural network to fit a regression model for the transformation matrix estimation [Wang et al., 2019a; Yang et al., 2019; Deng et al., 2019; Pais et al., 2020].

2. Considering the registration as an end-to-end framework by the combination of neural network and optimization [Huang et al., 2020; Choy et al., 2020].

In the following we review several methods in each of the two categories, all aim to train a deep neural network to directly solve the registration problem.

Registration by regression

[Deng et al., 2019] propose a relativeNet to estimate the pose directly from features. [Lu et al., 2019] propose DeepVCP model to detect key-points based on learned matching probabilities among a group of candidates, which can boost the registration accuracy. [Pais et al., 2020] develop a classification network to identify the inliers/outliers and uses a regression network to estimate the transformation matrix from the inliers.

Registration by optimization and neural network

The main idea of this category is to combine the conventional registration-related optimization theories with deep neural networks to solve the registration problem. PointNetLK [Aoki et al., 2019] uses PointNet [Qi et al., 2017a] to extract global features for two input point clouds and then use an inverse compositional (IC) algorithm to estimate the transformation matrix. By estimating the transformation matrix, the objective is to minimize the feature difference between the two features. For this feature based IC algorithm, the Jacobian estimation is challenging. PointnetLK uses an approximation method through a finite difference gradient computation. This approach allows the application of the computationally efficient inverse compositional

Lucas-Kanade algorithm. [Huang et al., 2020] further improve PointNetLK with an auto-encoder and a point distance loss which can also reduce the dependence on labels. DGR [Choy et al., 2020] proposes a 6-dimensional convolutional network architecture for inlier likelihood prediction and estimate the transformation by a weighted Procrustes module. The recently proposed RGM [Fu et al., 2021] transforms point clouds into graphs to perform deep graph matching, extracting soft deep features correspondence matrix.

In chapter 6 we show that registration performance using [Aoki et al., 2019; Wang and Solomon, 2019a; Fu et al., 2021] deteriorates when the objects are related by large rotations. DeepGMR [Yuan et al., 2020] addresses this problem by extracting pose-invariant correspondences between raw point clouds and Gaussian mixture model (GMM) parameters, and recovers the transformation from the matched mixtures. However, its performance deteriorates in the presence of sampling noise. The above methods show that the combination of conventional optimization methods and recent deep learning strategies obtain better accuracy than previous methods.

Chapter 3

Problem Definition

3.1 Registration Under the Assumption of Rigid Motion

Registration is the process of transforming two or more data-sets (3D point clouds, images, *etc.*) to the same coordinate system, such that overlapping components of these data sets are aligned together. Under the assumption of rigid motion, the coordinate systems of the data sets are related by a rotation and a translation. There are several ways of representing a rigid motion, in this work the rotation is represented by rotation matrix $\mathbf{R} \in \mathrm{SO}(n)$ and a translation vector $\mathbf{t} \in \mathbb{R}^n$. $\mathrm{SO}(n)$ is the special orthogonal group, which include all the orthogonal matrices with determinant value of 1. The action of the group on a set of points $X \subset \mathbb{R}^n$ is defined by

$$\mathbf{R}X = \{\mathbf{R}\mathbf{x} \ : \ \mathbf{x} \in X\}. \tag{3.1}$$

Two sets of points related by a rigid motion are therefore defined by $Y = \mathbf{R}X + \mathbf{t}$ where each corresponding pair of elements $\mathbf{y} \in Y, \mathbf{x} \in X$ is related by $\mathbf{y} = \mathbf{R}\mathbf{x} + \mathbf{t}$. The scope of this work is on registration of three dimensional sets, *i.e.* 3D point clouds.

3.2 Point Cloud Rigid Registration

A point cloud $\mathcal{P}$ is a finite set of points in $\mathbb{R}^3$. In many applications these points are samples from a physical object, $\mathcal{O} \subseteq \mathbb{R}^3$ (we may think of it as a surface or a manifold). Viewing point clouds as sets of samples, the registration problem may be formulated as follows: Let $\mathcal{O} \subseteq \mathbb{R}^3$ be a physical object and $T(\mathbf{x}) = \mathbf{R}\mathbf{x} + \mathbf{t}$ a rigid map ($\mathbf{R} \in \mathrm{SO}(3)$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is translation vector). We consider the transformed object

$$T(\mathcal{O}) := \{T(\mathbf{x}) \ : \ \mathbf{x} \in \mathcal{O}\}. \tag{3.2}$$

Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two point clouds sampled from the object $\mathcal{O}$ and the transformed object $T(\mathcal{O})$, respectively. In the registration problem, the goal is to estimate the transformation parameters $\mathbf{R}$ and $\mathbf{t}$ given only $\mathcal{P}_1$ and $\mathcal{P}_2$. Since point clouds are generated by a sampling procedure, the effects of sampling must be addressed, when solving the registration problem. Ideally, the relation between the two sampled point clouds $\mathcal{P}_1$ and $\mathcal{P}_2$ (sampled from $\mathcal{O}$ and $T(\mathcal{O})$) satisfies the relation $\mathcal{P}_2 = T(\mathcal{P}_1)$. Unfortunately, when point clouds are sampled at two different poses of some object, it is unlikely that the same set of object points is sampled, in both. In fact, if we assume a uniformly distributed sampling pattern on a continuous surface, it may be easily proved that the probability to have such a relation is null. The sampling differences are reflected as a registration error, in a different manner form the AWGN on the coordinates.

When dense point clouds are considered, the sampling noise effect has on the registration performance is lessen. Yet, as mentioned in chapter 1, it is critical to reduce the size of the point clouds to improve computational efficiency and reduce communication costs. As a result, many registration applications make use of under-sampled point clouds. On such point clouds, the sampling noise effect is dominant and significantly deteriorate registration performance. We therefore approach registration under sampling noise scenarios for the case of under-sampled point clouds.

We consider the registration problem in the following scenarios:

- Full intersection (Vanilla model) - where $\mathcal{P}_2 = T(\mathcal{P}_1)$.
- Sampling noise - Two cases are considered: Partial intersection, where $\mathcal{P}_2$ and $T(\mathcal{P}_1)$ may intersect, but are not identical; Zero intersection, where $\mathcal{P}_2$ and $T(\mathcal{P}_1)$ have no samples in common.
- Gaussian noise - $\mathcal{P}_2 = T(\mathcal{P}_1) + \mathcal{N}$, where $\mathcal{P}_2$ is a result of a rigid transformation of $\mathcal{P}_1$ with its coordinates perturbed by AWGN.

Chapter 4

UME

In this paper we adopt the Universal Manifold Embedding (UME [Hagege and Francos, 2010]) framework designed for registering two functions $g, h : \mathbb{R}^n \to \mathbb{R}$, with compact supports related by a geometric transformation (rigid, affine) parameterized by $\mathbf{A}$. Zero and first order moments (integrals) are evaluated in constructing the $(n+1) \times D$ UME matrix (where $D > n+1$) such that the UME matrices of g and h are co-variant to the transformation $\mathbf{A}$, as detailed below.

4.1 Continuous UME

In this section we briefly review the principles of the UME for observations related by an affine transformation. Let O be the space of observations. Let Φ be the group of affine transformations, and let S be a set of known objects. Every observation is the result of applying a geometric deformation $\phi \in \Phi$ to an object $s \in S$. The parameters of the affine transformation completely

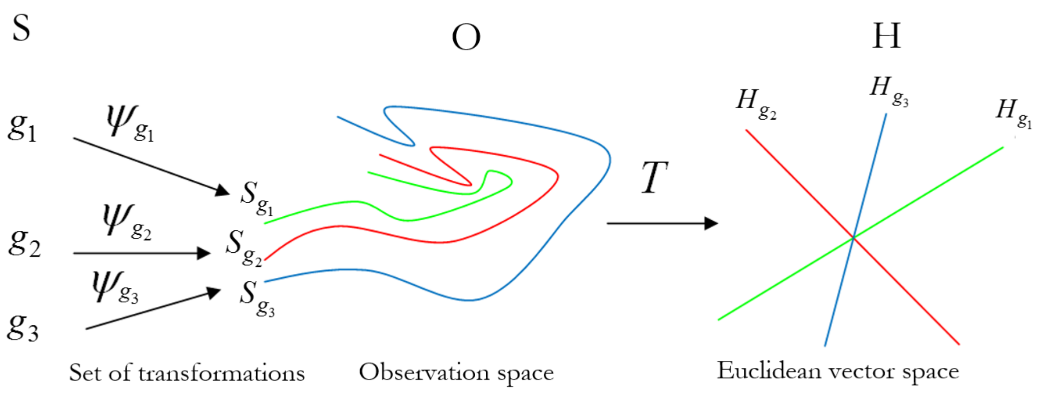

Figure 4.1: The Universal Manifold Embedding framework (from left to right): The physical model that generates the observations - applying the set of possible deformations to some object g produces S_g which is the set of all possible observations on g. S_g is a subset of the space of observations O. The UME - all observations in S_g are non-linearly mapped by $\mathbf{T}$ to a unique linear subspace $H_g = \mathbf{T}(S_g)$.

specify the action of the group of geometric transformations the object may undergo. We denote by $\phi(s) \in O$ the set of all possible observations on an object s. Thus, $\phi(s)$ is the orbit of s under Φ.

The universal manifold embedding is a map $\mathbf{T} : O \to H$ from the space of observations into a low dimensional Euclidean space, H, such that the set $\mathbf{T}(\phi(s))$ is a linear subspace of H for any s (see Figure 4.1). Thus, the UME reduces the dimension of any problem concerning the multiplicity of appearances of objects from the high dimensional space of observations O to the low dimensional linear space H and allows for the usage of classical linear theory in solving the highly non-linear problems of deformable object registration.

Next, the mapping $\mathbf{T}$ is described: Consider the case where the finite support functions $h(x), g(x)$ are observations on the same object related by an affine transformation, i.e.

$$h(\mathbf{x}) = g(\mathbf{A}\mathbf{x} + \mathbf{c}),\ \ \mathbf{A} \in \mathrm{GL}(n);\ \ \mathbf{c}, \mathbf{x} \in \mathbb{R}^n \tag{4.1}$$

Let $\mathbf{y} = \mathbf{A}\mathbf{x}+\mathbf{c}$, $\mathbf{x} = \mathbf{A}^{-1}\mathbf{y}+\mathbf{b}$ where $\mathbf{b} = -\mathbf{A}^{-1}\mathbf{c}$. Let $\mathbf{y}' = [1, y_1, \ldots, y_n]^T$ then $\mathbf{x} = \mathbf{D}\mathbf{y}'$ where $\mathbf{D} = [\mathbf{b}; \mathbf{A}^{-1}]$ is an $n \times (n+1)$ matrix. Let $P \in \mathbb{N}$ and let $w_l,\ l = 1, \ldots, P$ be a set of bounded, Lebesgue measurable functions $w_l : \mathbb{R} \to \mathbb{R}$. Since by definition $h(x) = 0, x \notin \mathit{supp}(h)$, and similarly for g, by a change of variables we obtain the following identities:

$$\int\limits_{\mathbb{R}^n} w_l \circ h(\mathbf{x}) d\mathbf{x} = \left|\mathbf{A}^{-1}\right| \int\limits_{\mathbb{R}^n} w_l \circ g(\mathbf{y}) d\mathbf{y} \tag{4.2}$$

$$\int\limits_{\mathbb{R}^n} \mathbf{x} w_l \circ h(\mathbf{x}) d\mathbf{x} = \left|\mathbf{A}^{-1}\right| \int\limits_{\mathbb{R}^n} (\mathbf{D}\mathbf{y}') w_l \circ g(\mathbf{y}) d\mathbf{y} \tag{4.3}$$

Let f be some observation on a deformable object and let

$$\mathbf{T}(f) = \begin{bmatrix} \int\limits_{\mathbb{R}^n} w_1 \circ f(\mathbf{y}) d\mathbf{y} & \int\limits_{\mathbb{R}^n} y_1 w_1 \circ f(\mathbf{y}) d\mathbf{y} & \cdots & \int\limits_{\mathbb{R}^n} y_n w_1 \circ f(\mathbf{y}) \\ & \vdots & & \\ \int\limits_{\mathbb{R}^n} w_P \circ f(\mathbf{y}) d\mathbf{y} & \int\limits_{\mathbb{R}^n} y_1 w_P \circ f(\mathbf{y}) d\mathbf{y} & \cdots & \int\limits_{\mathbb{R}^n} y_n w_P \circ f(\mathbf{y}) \end{bmatrix} \tag{4.4}$$

Let $\mathbf{D}' = [\mathbf{e}_1; \mathbf{D}^T]$, where $\mathbf{e}_1 = [1\ 0\ \ldots\ 0]^T$, be the matrix representation of an affine transformation in homogeneous coordinates. Rewriting (4.2), (4.3) for $l = 1, \ldots, P$ in a matrix form, we have:

$$\mathbf{T}(g)\mathbf{D}' \left|\mathbf{A}^{-1}\right| = \mathbf{T}(h) \tag{4.5}$$

To find the matrix $\mathbf{D}'$ (and thus recover the parameters of the affine transformation) we notice that (4.5) is in fact an over determined linear equation system. Hence, the least squares solution for $\mathbf{D}'$ is given by

$$|\mathbf{A}^{-1}|\mathbf{D}' = (\mathbf{T}(g)^T\mathbf{T}(g))^{-1}\mathbf{T}(g)^T\mathbf{T}(h) \tag{4.6}$$

and $|\mathbf{A}^{-1}|$ is recovered directly from (4.2) by

$$\left|\mathbf{A}^{-1}\right| = \frac{\int\limits_{\mathbb{R}^n} w \circ h(\mathbf{x})d\mathbf{x}}{\int\limits_{\mathbb{R}^n} w \circ g(\mathbf{y})d\mathbf{y}} \tag{4.7}$$

4.2 Discrete UME

Following the principles of the UME framework, we derive a new discrete closed-form implementation of the UME for the registration of point clouds undergoing rigid transformations. More specifically, let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two point clouds related by a rigid transformation T. Then, an invariant feature (function) $\mathcal{F}$ on $\mathcal{P}_1$ and $\mathcal{P}_2$ is a function that assigns any point $\mathbf{p} \in \mathcal{P}_1$ and the transformed point $T(\mathbf{p}) \in \mathcal{P}_2$ with the same value. A simple example for such an invariant feature is the one that assigns each point with its distance from the point cloud center of mass.

Since for finite support objects, it is straightforward to reduce the problem of computing the rigid transformation $T(\mathbf{p}) = R\mathbf{p} + \mathbf{t}$ to a rotation-only problem, *i.e.* $\mathbf{t} = 0$, we show that the moment integral calculations involved in evaluating the UME operator, may be replaced by computing moments of the invariant functions using summations.

Theorem 4.2.1. *Let* $\mathbf{R}$ *be a rotation matrix and* $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *be two point* *clouds satisfying the relation $\mathcal{P}_2 = \mathbf{R} \cdot \mathcal{P}_1$. Let $\mathcal{F}$ be an $SO(3)$ invariant feature on $\mathcal{P}_1$ and $\mathcal{P}_2$, namely*

$$\mathcal{F}(\mathbf{p}) = \mathcal{F}(\mathbf{Rp}), \quad \forall \mathbf{p} \in \mathcal{P}_1, \tag{4.8}$$

then

$$\mathbf{M}_{\mathcal{P}_2}(\mathcal{F}) = \mathbf{R} \cdot \mathbf{M}_{\mathcal{P}_1}(\mathcal{F}), \quad \textit{where } \mathbf{M}_{\mathcal{P}_i}(\mathcal{F}) = \frac{1}{|\mathcal{P}_i|} \begin{bmatrix} \sum_{\mathbf{p} \in \mathcal{P}_i} p_1 \mathcal{F}(\mathbf{p}) \\ \sum_{\mathbf{p} \in \mathcal{P}_i} p_2 \mathcal{F}(\mathbf{p}) \\ \sum_{\mathbf{p} \in \mathcal{P}_i} p_3 \mathcal{F}(\mathbf{p}) \end{bmatrix}, \ \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}. \tag{4.9}$$

In order to prove Theorem 4.2.1, we first state and prove a discrete version of the UME theorem from which Theorem 4.2.1 follows immediately as a special case. We begin by presenting the continuous UME theorem, [Efraim and Francos, 2019], in the specific case where the translation vector $\mathbf{t} = 0$.

Definition 4.2.1.1. *Let $k : \mathbb{R}^n \to \mathbb{R}$ be a compactly supported measurable function and $w_1, \dots, w_D : \mathbb{R} \to \mathbb{R}$ are measurable functions. The $n \times D$ UME matrix of k with respect to $w_1, \dots, w_D$ is defined by*

$$[\mathbf{UME}_k]_{i,j} := \int_{\mathbb{R}^n} x_i w_j(k(\mathbf{x})) d\mathbf{x}. \tag{4.10}$$

Theorem 4.2.2. ***(Continuous UME)*** *[Efraim and Francos, 2019], Let $f, g : \mathbb{R}^n \to \mathbb{R}$, be two functions with compact supports related by a rotation $\mathbf{R}$, i.e. $g(\mathbf{x}) = f(\mathbf{Rx})$ for all $\mathbf{x} \in \mathbb{R}^n$. Then, for any set of D measurable* *functions $w_1, \ldots, w_D$ such that $w_i(0) = 0$ for all i,*

$$\mathbf{UME}_f = \mathbf{R} \cdot \mathbf{UME}_g \,. \tag{4.11}$$

We next provide the discrete analog of Theorem 4.2.2 where the continuous functions f and g are replaced by the invariant functions estimated from the observed point clouds using the DNN, and integration is replaced by summation on the elements in the point clouds.

Definition 4.2.2.1. *Let $\mathcal{P} \subseteq \mathbb{R}^3$ be a finite point cloud, $\mathcal{F}$ a feature (function) on $\mathcal{P}$ and $w_1, \ldots, w_d : \mathbb{R} \to \mathbb{R}$ measurable functions. The discrete $n \times D$ UME matrix of $\mathcal{P}$ and $\mathcal{F}$ with respect to $w_1 \ldots, w_D$ is defined to be*

$$[\mathbf{UME}_{\mathcal{P}}^{\mathcal{F}}]_{i,j} = \sum_{\mathbf{p} \in \mathcal{P}} p_i w_j(\mathcal{F}(\mathbf{p})). \tag{4.12}$$

Proposition 4.2.2.1. *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two point clouds satisfying $\mathcal{P}_2 = \mathbf{R}\mathcal{P}_1$, and $\mathcal{F}$ is an invariant feature on $\mathcal{P}_1$ and $\mathcal{P}_2$. For any D functions $w_1, \ldots, w_d : \mathbb{R} \to \mathbb{R}$ satisfying $w_i(0) = 0$ for all i*

$$\mathbf{UME}_{\mathcal{P}_2}^{\mathcal{F}} = \mathbf{R} \cdot \mathbf{UME}_{\mathcal{P}_1}^{\mathcal{F}} \,. \tag{4.13}$$

We note that if we take w_1 to be the identity function and denote the first column of $\mathbf{UME}_{\mathcal{P}_1}^{\mathcal{F}}$ and $\mathbf{UME}_{\mathcal{P}_2}^{\mathcal{F}}$ by $[\mathbf{UME}_{\mathcal{P}_1}^{\mathcal{F}}]_1$ and $[\mathbf{UME}_{\mathcal{P}_2}^{\mathcal{F}}]_1$ respectively, we have

$$\frac{1}{|\mathcal{P}_i|}[\mathbf{UME}_{\mathcal{P}_i}^{\mathcal{F}}]_1 = \mathbf{M}_{\mathcal{P}_i}(\mathcal{F}), \quad i = 1, 2. \tag{4.14}$$

Hence, Theorem 4.2.1 is followed by Proposition 4.2.2.1 trivially, as a special case.

Proof of Proposition 4.2.2.1 The main idea in our proof is to approximate the discrete sums defining the discrete UME matrices in (4.13) by continuous integrals and apply the continuous UME theorem. Given $\varepsilon > 0$, $\mathcal{P}_1$, $\mathcal{P}_2$ and the invariant feature $\mathcal{F}$, we construct two compactly supported measurable functions $f_\varepsilon, g_\varepsilon : \mathbb{R} \to \mathbb{R}$ such that $g_\varepsilon(\mathbf{x}) = f_\varepsilon(R^{-1}\mathbf{x})$. We then apply Theorem 4.2.2 to f_ε and g_ε, and taking $\varepsilon \to 0$ we will conclude.

Denote the ball of radius ε centered at a point $\mathbf{p} \in \mathbb{R}^3$ by $B_\varepsilon(\mathbf{p})$. Define the functions f_ε is and g_ε by

$$f_\varepsilon(\mathbf{x}) := \sum_{\mathbf{p}\in\mathcal{P}_1} \mathcal{F}(\mathbf{p})\mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{x}), \quad g_\varepsilon(\mathbf{x}) := \sum_{\mathbf{p}\in\mathcal{P}_2} \mathcal{F}(\mathbf{p})\mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{x}). \tag{4.15}$$

See the illustration of $f_\varepsilon(\mathbf{x})$ in Figure 4.2. We now prove that the desired relation $g_\varepsilon(\mathbf{x}) = f_\varepsilon(\mathbf{R}^{-1}\mathbf{x})$ holds: Directly from the definition of f_ε,

$$f_\varepsilon(\mathbf{R}^{-1}\mathbf{x}) \overset{(4.15)}{=} \sum_{\mathbf{p}\in\mathcal{P}_1} \mathcal{F}(\mathbf{p})\mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{R}^{-1}\mathbf{x}). \tag{4.16}$$

Since a rigid transformation maps any ball to a ball with the same radius, we have that $\mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{R}^{-1}\mathbf{x})$ is non-zero if and only if $\mathbf{R}^{-1}\mathbf{x} \in B_\varepsilon(\mathbf{p})$:

$$\mathbf{R}^{-1}\mathbf{x} \in B_\varepsilon(\mathbf{p}) \iff \mathbf{x} \in \mathbf{R}(B_\varepsilon(\mathbf{p})) \iff \mathbf{x} \in B_\varepsilon(\mathbf{R}\mathbf{p}). \tag{4.17}$$

It is immediately follows that

$$\mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{R}^{-1}\mathbf{x}) = \mathbb{1}_{B_\varepsilon(\mathbf{Rp})}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{R}^3. \tag{4.18}$$

Substituting (4.18) into (4.16), and using the SO(3) invariance of $\mathcal{F}$ we have

$$f_\varepsilon(\mathbf{R}^{-1}\mathbf{x}) = \sum_{\mathbf{p}\in\mathcal{P}_1} \mathcal{F}(\mathbf{p})\mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{R}^{-1}\mathbf{x}) \tag{4.19}$$

$$= \sum_{\mathbf{p}\in\mathcal{P}_1} \mathcal{F}(\mathbf{Rp})\mathbb{1}_{B_\varepsilon(\mathbf{Rp})}(\mathbf{x}) \tag{4.20}$$

$$= \sum_{\mathbf{p}\in\mathcal{P}_2} \mathcal{F}(\mathbf{p})\mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{x}) = g_\varepsilon(\mathbf{x}). \tag{4.21}$$

We have so far proved that $f_\varepsilon(\mathbf{x}) = g_\varepsilon(\mathbf{Rx})$ for all $\mathbf{x}$. By Theorem 4.2.2 we have

$$\mathbf{UME}_{g_\varepsilon} = \mathbf{R}\cdot\mathbf{UME}_{f_\varepsilon}\,. \tag{4.22}$$

For sufficiently small ε, the balls defining f_ε do not intersect and therefore we have,

$$[\mathbf{UME}_{f_\varepsilon}]_{ij} \overset{(4.15)}{=} \int_{\mathbb{R}^3} x_i w_j \left(\sum_{\mathbf{p}\in\mathcal{P}_1} \mathcal{F}(\mathbf{p}) \mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{x}) \right) d\mathbf{x} \tag{4.23}$$

$$= \int_{\biguplus_{\mathbf{v}\in\mathcal{P}_1} B_\varepsilon(\mathbf{v})} x_i w_j \left(\sum_{\mathbf{p}\in\mathcal{P}_1} \mathcal{F}(\mathbf{p}) \mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{x}) \right) d\mathbf{x} \tag{4.24}$$

$$+ \int_{\mathbb{R}^3 \setminus \biguplus_{\mathbf{v}\in\mathcal{P}_1} B_\varepsilon(\mathbf{v})} x_i \underbrace{w_j \left(\sum_{\mathbf{p}\in\mathcal{P}_1} \mathcal{F}(\mathbf{p}) \underbrace{\mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{x})}_{=0} \right)}_{w_j(0)=0} d\mathbf{x} \tag{4.25}$$

$$= \sum_{\mathbf{v}\in\mathcal{P}_1} \int_{B_\varepsilon(\mathbf{v})} x_i w_j \left(\sum_{\mathbf{p}\in\mathcal{P}_1} \mathcal{F}(\mathbf{p}) \mathbb{1}_{B_\varepsilon(\mathbf{p})}(\mathbf{x}) \right) d\mathbf{x} \tag{4.26}$$

$$= \sum_{\mathbf{v}\in\mathcal{P}_1} w_j(\mathcal{F}(\mathbf{v})) \int_{B_\varepsilon(\mathbf{v})} x_i d\mathbf{x}. \tag{4.27}$$

where $\biguplus$ denotes a disjoint union of sets and the last equality stems from the fact that $\mathcal{F}(\mathbf{p})$ is constant on $B_\varepsilon(\mathbf{v})$. By (4.27) and the integral mean value theorem we have that

$$\lim_{\varepsilon\to 0} \frac{1}{\mathrm{Vol}(B_\varepsilon)} [\mathbf{UME}_{f_\varepsilon}]_{ij} = \lim_{\varepsilon\to 0} \frac{1}{\mathrm{Vol}(B_\varepsilon)} \sum_{\mathbf{p}\in\mathcal{P}_1} w_j(\mathcal{F}(\mathbf{p})) \int_{B_\varepsilon(\mathbf{p})} x_i d\mathbf{x}$$
$$= \sum_{\mathbf{p}\in\mathcal{P}_1} w_j(\mathcal{F}(\mathbf{p})) \underbrace{\lim_{\varepsilon\to 0} \frac{1}{\mathrm{Vol}(B_\varepsilon)} \int_{B_\varepsilon(\mathbf{p})} x_i d\mathbf{x}}_{p_i} = \sum_{\mathbf{p}\in\mathcal{P}_1} p_i w_j(\mathcal{F}(\mathbf{p})) \tag{4.28}$$

This shows that

$$\mathbf{UME}^{\mathcal{F}}_{\mathcal{P}_1} = \lim_{\varepsilon\to 0} \frac{1}{\mathrm{Vol}(B_\varepsilon)} \mathbf{UME}_{f_\varepsilon}, \tag{4.29}$$

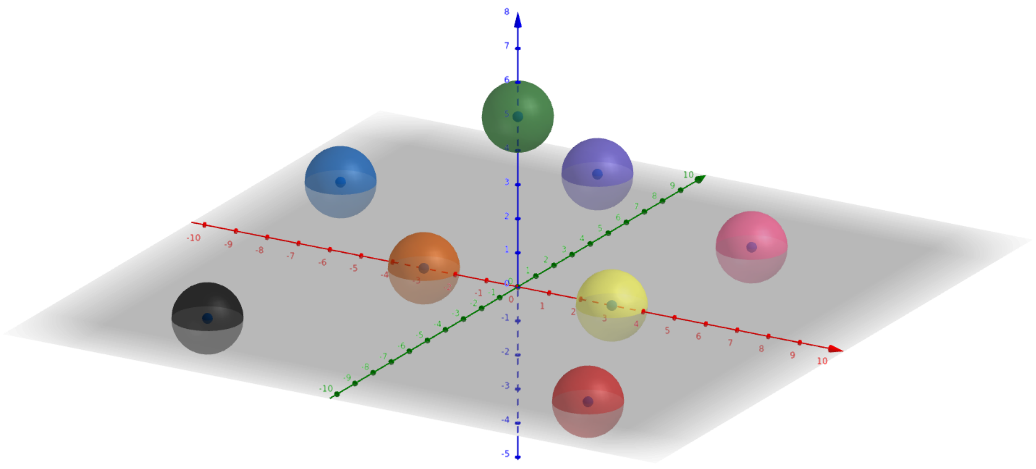

Figure 4.2: $B_{\varepsilon(\mathbf{p})}$ for $\mathbf{p} \in \mathcal{P}$. Balls centers are the point cloud points, and each ball color represents the value of $\mathcal{F}(\mathbf{p})$.

and similarly it is easily proved that

$$\mathbf{UME}_{\mathcal{P}_2}^{\mathcal{F}} = \lim_{\varepsilon \to 0} \frac{1}{\mathrm{Vol}(B_\varepsilon)} \mathbf{UME}_{g_\varepsilon} . \tag{4.30}$$

Finally, applying Theorem 4.2.2 on f_ε and g_ε we obtain:

$$\mathbf{UME}_{\mathcal{P}_2}^{\mathcal{F}} = \lim_{\varepsilon \to 0} \frac{1}{\mathrm{Vol}(B_\varepsilon)} \mathbf{UME}_{g_\varepsilon} = \lim_{\varepsilon \to 0} \frac{1}{\mathrm{Vol}(B_\varepsilon)} \mathbf{R} \cdot \mathbf{UME}_{f_\varepsilon} \tag{4.31}$$

$$= \mathbf{R} \cdot \lim_{\varepsilon \to 0} \frac{1}{\mathrm{Vol}(B_\varepsilon)} \mathbf{UME}_{f_\varepsilon} = R \cdot \mathbf{UME}_{\mathcal{P}_1}^{\mathcal{F}} . \tag{4.32}$$

This completes the proof.

Chapter 5

DeepUME

Following the strategy of integrating the UME registration methodology into a deep neural network, we both adjust the UME method [Hagege and Francos, 2010] adapting it to a DNN framework and design our architecture to optimize the UME performance.

In the UME registration framework the input is composed of two point clouds satisfying the relation

$$\mathcal{P}_2 = \mathbf{R} \cdot \mathcal{P}_1 + \mathbf{t} \tag{5.1}$$

and an invariant feature (that is, a function $\mathcal{F}$ defined on a point cloud $\mathcal{P}$ such that $f(\mathbf{p}) = f(\mathbf{R} \cdot \mathbf{p} + \mathbf{t})$ for any $\mathbf{p} \in \mathcal{P}$). Applying the UME operator on each of the point clouds, two matrices $\mathbf{M}_{\mathcal{P}_1}$ and $\mathbf{M}_{\mathcal{P}_2}$ are obtained, such that

$$\mathbf{M}_{\mathcal{P}_2} = \mathbf{R} \cdot \mathbf{M}_{\mathcal{P}_1} \,. \tag{5.2}$$

The geometric nature of the UME motivates as to use it as a basis for our

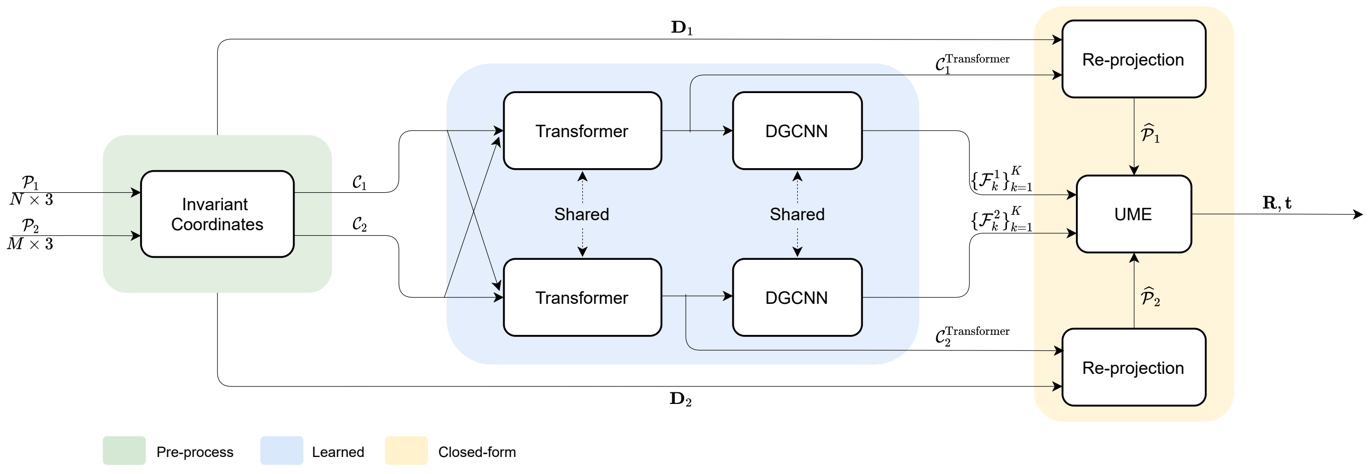
$\mathbf{D}_1$
$\mathcal{P}_1$ $N \times 3$
$\mathcal{P}_2$ $M \times 3$
Invariant Coordinates
$\mathcal{C}_1$
$\mathcal{C}_2$
Transformer
Shared
Transformer
DGCNN
Shared
DGCNN
$\mathcal{C}_1^{\text{Transformer}}$
$\mathcal{C}_2^{\text{Transformer}}$
$\{\mathcal{F}_k^1\}_{k=1}^K$
$\{\mathcal{F}_k^2\}_{k=1}^K$
Re-projection
$\widehat{\mathcal{P}}_1$
UME
$\widehat{\mathcal{P}}_2$
Re-projection
$\mathbf{R}, \mathbf{t}$
$\mathbf{D}_2$
Pre-process
Learned
Closed-form

Figure 5.1: DeepUME network architecture.

framework. While many registration methods find corresponding points in the reference and the transformed point clouds in order to solve the registration problem, in the UME methodology a new set of "corresponding points" is constructed by evaluating low order geometric moments of the invariant feature. This is a significant advantage in noisy scenarios, where point correspondences between the reference and transformed point clouds, may not exist at all. The use of moments allows us to exploit the geometric structure of the objects to be registered, which is invariant under sampling (as long as the sampling is reliable) resulting in an improved immunity to sampling noise.

The goal of the deep neural network in our framework is to construct multiple high-quality invariant features, in order to maximize the performance of the UME in various noisy scenarios. We adapt DCP architecture [Wang and Solomon, 2019a], which has been proved to be very efficient in creating high-dimensional embedding for point clouds. Conceptually, we may divide our framework into three main parts, each responsible of a different aspect of the registration process. The first part is a pre-process designed to overcome DCP limitation in preforming the learning process over the entire rotation range. The second part is an unsupervised deep neural network responsible for learning features that are subsequently employed for parameter estimation. The final part consists of the UME method for parameter extraction. The framework is illustrated in Figure 5.1, and explained in details in the following.

5.1 UME registration

Relying on the original UME method, we proved the closed-form formula presented in equation (4.9):

$$\mathbf{M}_{\mathcal{P}_2}(\mathcal{F}) = \mathbf{R} \cdot \mathbf{M}_{\mathcal{P}_1}(\mathcal{F}), \quad \text{where } \mathbf{M}_{\mathcal{P}_i}(\mathcal{F}) = \frac{1}{|\mathcal{P}_i|} \begin{bmatrix} \sum_{\mathbf{p}\in\mathcal{P}_i} p_1 \mathcal{F}(\mathbf{p}) \\ \sum_{\mathbf{p}\in\mathcal{P}_i} p_2 \mathcal{F}(\mathbf{p}) \\ \sum_{\mathbf{p}\in\mathcal{P}_i} p_3 \mathcal{F}(\mathbf{p}) \end{bmatrix}, \ \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}.$$

We call $\mathbf{M}_{\mathcal{P}_i}$ the moment vector of the transformation invariant function $\mathcal{F}$ defined on $\mathcal{P}_i$.

Given two sets of points in $\mathbb{R}^3$, $\{\mathbf{v}_i\}_{i=1}^k$ and $\{\mathbf{u}_i\}_{i=1}^k$ $k \geq 3$, satisfying the relation $\mathbf{v}_i = \mathbf{R} \cdot \mathbf{u}_i$ for all i, we may find $\mathbf{R}$ by a standard procedure proposed by [Horn et al., 1988a; Myronenko and Song, 2009] (see section 2.1.3). Hence, we conclude from (4.9) that in the absence of noise, finding a set of invariant functions $\mathcal{F}_1, \ldots, \mathcal{F}_k$, such that $k \geq 3$, yields a closed-form solution to the registration problem. We remark that in order to use the invariant functions $\mathcal{F}_1, \ldots, \mathcal{F}_k$ by the discrete UME method efficiently, they need be linearly independent. More precisely, for Horn's transformation extraction to work, the dimension of the space created by the moments vector, span $\{M_P(\mathcal{F}_j)\}_{j=1}^{k}$, must be at least 3. For registration in noisy scenarios, we would like $\mathcal{F}_1, \ldots, \mathcal{F}_k$ to be diverse, and therefore allow us to exploit the geometric structure of the shapes represented by our sampled point clouds.

However in the presence of noise (sampling or additive) (4.9) no longer holds as $\mathcal{P}_2$ and $\mathbf{R} \cdot \mathcal{P}_1$ are not identical anymore and in fact, with a high probability they do not share any point in common. Therefore, the estimated rotation matrix is noisy.

This is the point where a deep neural network comes into play. The registration error obviously depends on the difference between $\mathbf{M}_{\mathcal{P}_2}$ and $\mathbf{R} \cdot \mathbf{M}_{\mathcal{P}_1}$, and on the function $\mathcal{F}$ being rigid transformation invariant despite the noise. To that extent, we employ a deep neural network in order to *learn* how to construct good transformation-invariant functions. These invariant functions are designed to exploit the geometry of the point cloud so that their invariance to the transformation is minimally affected by the noise, resulting in a smaller registration error.

5.2 Features learning performed by DGCNN

Towards obtaining noise resilient SO(3)-invariant functions for effectively evaluating the UME moments, we aim at learning features capturing the geometric structure of the point cloud. This structure is determined by the point cloud coordinates (global information) and the neighborhood of each point (local information). For that reason, we adopt the same architec-

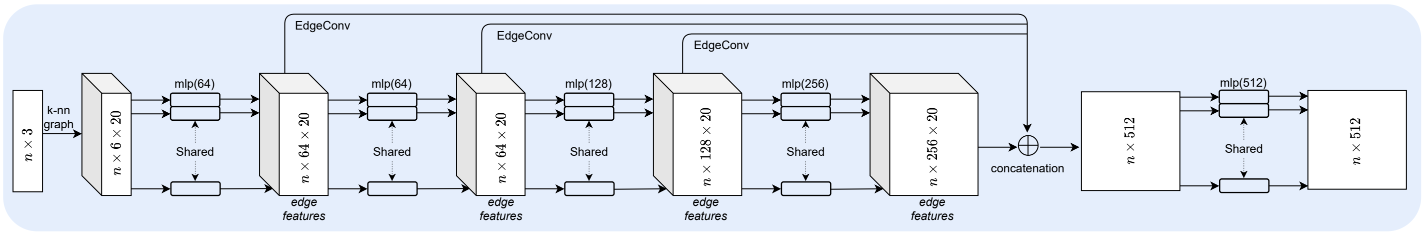

Figure 5.2: DGCNN network architecture.

ture used by the DCP embedding network - DGCNN [Wang et al., 2019b]. DGCNN is designed to preform a per-point embedding such that information on neighboring points is well incorporated.

A key to DGCNN local information extraction into features is its input structure - instead of a raw point cloud, the input is the point clouds self-looped k-NN graph $\mathcal{G}$. As such, each point in the point cloud is characterized by the coordinates of points in its neighborhood in addition to its own coordinates. This approach results in a new representation for each point using a $6 \times k$ matrix. Each column of that matrix represents one of the k nearest neighbors and consists of the coordinates of the observed point stacked on top of the coordinated of the neighbor point.

5.2.1 DGCNN architecture

The version of DGCNN architecture used in this work is shown in Figure 5.2. DGCNN approach is inspired by PointNet [Qi et al., 2017a] and convolution operations. Instead of working on individual points like PointNet, it exploits local geometric structures by constructing a local neighborhood graph and applying convolution-like operations on the edges connecting neighboring pairs of points, in the spirit of graph neural networks.

Edge Convolution

Let $\mathcal{P}$ be an F-dimensional point cloud with N points, denoted by

$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \subseteq \mathbb{R}^F \tag{5.3}$$

In the simplest setting of $F = 3$, each point contains 3D coordinates

$$\mathbf{p}_i = (p_x, p_y, p_z); \tag{5.4}$$

it is also possible to include additional coordinates representing color, surface normal, and so on. In a deep neural network architecture, each subsequent layer operates on the output of the previous layer, so more generally the dimension F represents the feature dimensionality of a given layer. A directed graph $\mathcal{G}$ representing local point cloud structure,

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \quad \mathcal{V} = \{1, \ldots, N\}, \quad \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \tag{5.5}$$

where $\mathcal{V}$ and $\mathcal{E}$ are the vertices and edges, respectively. $\mathcal{G}$ is computed in its simplest case, as the k-nearest neighbor (k-NN) graph of $\mathcal{P}$ in $\mathbb{R}^F$. The graph includes self-loop, meaning each node also points to itself. *Edge features* are defined as

$$\mathbf{e}_{ij} = h_\theta(\mathbf{p}_i, \mathbf{p}_j), \quad h_\theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'} \tag{5.6}$$

where h_θ is a nonlinear function with a set of learnable parameters θ. EdgeConv operation is then defined by applying a channelwise symmetric aggregation operation $\square$ (*e.g.*, $\sum$ or max) on the edge features associated with all the edges emanating from each vertex. The output of EdgeConv at the i-th vertex is thus given by

$$\mathbf{p}_i' = \underset{j:(i,j)\in\mathcal{E}}{\square} \mathbf{e}_{ij} = \underset{j:(i,j)\in\mathcal{E}}{\square} h_\theta(\mathbf{p}_i, \mathbf{p}_j). \tag{5.7}$$

(5.7) is analogy to convolution along images, if we regard $\mathbf{p}_i$ as the central pixel and $\{\mathbf{p}_j : (i,j) \in \mathcal{E}\}$ as a patch around it. Overall, given an F-dimensional point cloud with n points, EdgeConv produces an F'-dimensional point cloud with the same number of points.

Formally, DGCNN is a neural network comprised of a total of L layers. Let h_θ^l denote the nonlinear function parameterized by a multi-layer perceptron in the l-th layer, and let $\mathbf{p}_i^l$, denote the embedding of the point cloud i-th point. The network forward pass is given by

$$\mathbf{p}_i^l = f\left(\left\{h_\theta^l\left(\mathbf{p}_i^{l-1}, \mathbf{p}_j^{l-1}\right)\ \forall j \in \mathcal{N}_i\right\}\right) \tag{5.8}$$

where $\mathcal{N}_i$ is the i-th point neighbourhood in $\mathcal{G}$ and f represents channel-wise max aggregation. The network final layer result with per point features or representations, noted

$$\mathcal{F}_{\mathcal{P}} = \{\mathbf{p}_1^L, \mathbf{p}_2^L, \ldots, \mathbf{p}_i^L, \ldots, \mathbf{p}_N^L\}. \tag{5.9}$$

DGCNN embeds one point cloud at a time, sharing its weights between the two clouds, i.e. the same implementation with the same parameters transform all input streams. Ideally, this embedding network should result in identical feature (function value) for two corresponding points in the reference point cloud and in its transformed version.

5.3 SO(3)-invariant coordinate system

Since DGCNN weights are shared, when the coordinates of corresponding points between two point clouds are significantly different, the learning process fails. Clearly, this situation occurs when transformations of large magnitude, *e.g.* large rotations, are considered (see section 6). Therefore DGCNN in its existing design cannot be employed towards constructing invariant features when rotations by large angles are considered.

We overcome this inherent difficulty by transferring the input point clouds to an alternative coordinates system, which is SO(3)-invariant. By doing so, the network inputs are two "almost similar" point clouds, where "almost" is due to sampling noise presence, in these under-sampled input point clouds. Whenever the point clouds to be registered are sufficiently dense (or noise-free), the difference of this new representation between the input point clouds is negligible (and does not exist in the noise free case). However, in our setting, where the point clouds to be registered are sparse and differently sampled, this is not the case anymore. The loss of information caused by low sampling rate makes the resulting representations of the clouds significantly different. Despite that, even when sparse point clouds are considered, the similarity of the resulting point clouds in our new coordinates system is sufficient in order to enable a learning process of multiple high-quality invariant features to occur.

Formally, Given a point cloud $\mathcal{P}$, the cloud center of mass (denoted by

$\mathbf{m}_\mathcal{P}$) is subtracted from each point coordinates, to obtain a centered representation $\mathcal{P}'$. The axes of the new coordinate system are the principle vectors of the point cloud covariance matrix given by

$$\mathbf{H}_{\mathcal{P}'} = \sum_{\mathbf{p}\in\mathcal{P}'} \mathbf{p}\mathbf{p}^T \tag{5.10}$$

For a point $\mathbf{p} \in \mathcal{P}'$, the new coordinates of $\mathbf{p}$ are defined to be $\mathbf{c_p} = \mathbf{D}^T_{\mathcal{P}'} \cdot \mathbf{p}$ where $\mathbf{D}_{\mathcal{P}'}$ is a matrix whose columns are principle vectors. Formally, $\mathbf{D}_{\mathcal{P}'}$ is an orthogonal matrix for which $\mathbf{H}_{\mathcal{P}'} = \mathbf{D}_{\mathcal{P}'}\mathbf{\Lambda}\mathbf{D}^T_{\mathcal{P}'}$ for a diagonal matrix $\mathbf{\Lambda}$. The resulting point cloud new coordinates are denoted by $\mathcal{C}$.

It is easy to verify that the new axes (columns of the PCA matrix) are co-variant under a rigid transformation:

$$\mathbf{H}_{\mathbf{R}\mathcal{P}'} = \sum_{\mathbf{p}\in\mathbf{R}\mathcal{P}'} \mathbf{R}\mathbf{p}\mathbf{p}^T\mathbf{R}^T = \mathbf{R}\mathbf{H}_{\mathcal{P}'}\mathbf{R}^T = \mathbf{R}\mathbf{D}_{\mathcal{P}'}\mathbf{\Lambda}(\mathbf{R}\mathbf{D}_{\mathcal{P}'})^T. \tag{5.11}$$

That is,

$$\mathbf{D}^{\mathbf{R}\mathcal{P}'} = \mathbf{R}\mathbf{D}_{\mathcal{P}'}. \tag{5.12}$$

Using (5.12), we easily prove that the projection coefficients on the new axes are rotation invariant. Given $\mathcal{P}_1$ and $\mathcal{P}_2$ related by a rigid motion, we have

$$\begin{aligned} \mathcal{C}_1 &= \left\{ \left(\mathbf{D}_{\mathcal{P}_1'}\right)^T \mathbf{p} : \mathbf{p} \in \mathcal{P}_1 \right\} = \left\{ \left(\mathbf{R}^T\mathbf{D}_{\mathcal{P}_2'}\right)^T \mathbf{p} : \mathbf{p} \in \mathcal{P}_1 \right\} \\ &= \left\{ \left(\mathbf{D}_{\mathcal{P}_2'}\right)^T \mathbf{R}\mathbf{p} : \mathbf{p} \in \mathcal{P}_1 \right\} = \left\{ \left(\mathbf{D}_{\mathcal{P}_2'}\right)^T \mathbf{p} : \mathbf{p} \in \mathcal{P}_2 \right\} = \mathcal{C}_2. \end{aligned} \tag{5.13}$$

5.3.1 Axes sign ambiguity

For a point cloud $\mathcal{P}$, the principal vectors defining $\mathbf{D}_{\mathcal{P}'}$, are defined up to a sign. Hence, the equality in (5.12) is true up to multiplication of the the columns by ± 1. That is to say, only one from the 8 possibilities for the principal vectors matrix satisfies the desired equality (5.12). We eliminate this sign ambiguity by considering all 8 possible new axes systems $\left\{ D^i_{\mathcal{P}'_2} \right\}_{i=1}^{8}$ given by different sign multiplication constellations. We choose the one axes system that satisfies (5.13) by

$$i = \underset{j=1,\ldots,8}{\arg\min}\, d_{\mathrm{C}}(\mathcal{C}_1, \mathcal{C}_2^j) \tag{5.14}$$

where d_{C} stands for Chamfer distance.

Furthermore, since the change of coordinate system is invertible, the original point cloud can be reconstructed.

In the noisy case, the relation $\mathcal{C}_1 = \mathcal{C}_2$ does not hold anymore. Due to the noise, axes are no longer co-variant and thus the projections are no longer invariant. However, the difference between $\mathcal{C}_1$ and $\mathcal{C}_2$ is sufficiently small so that a successful learning process of invariant features may be applied using DGCNN.

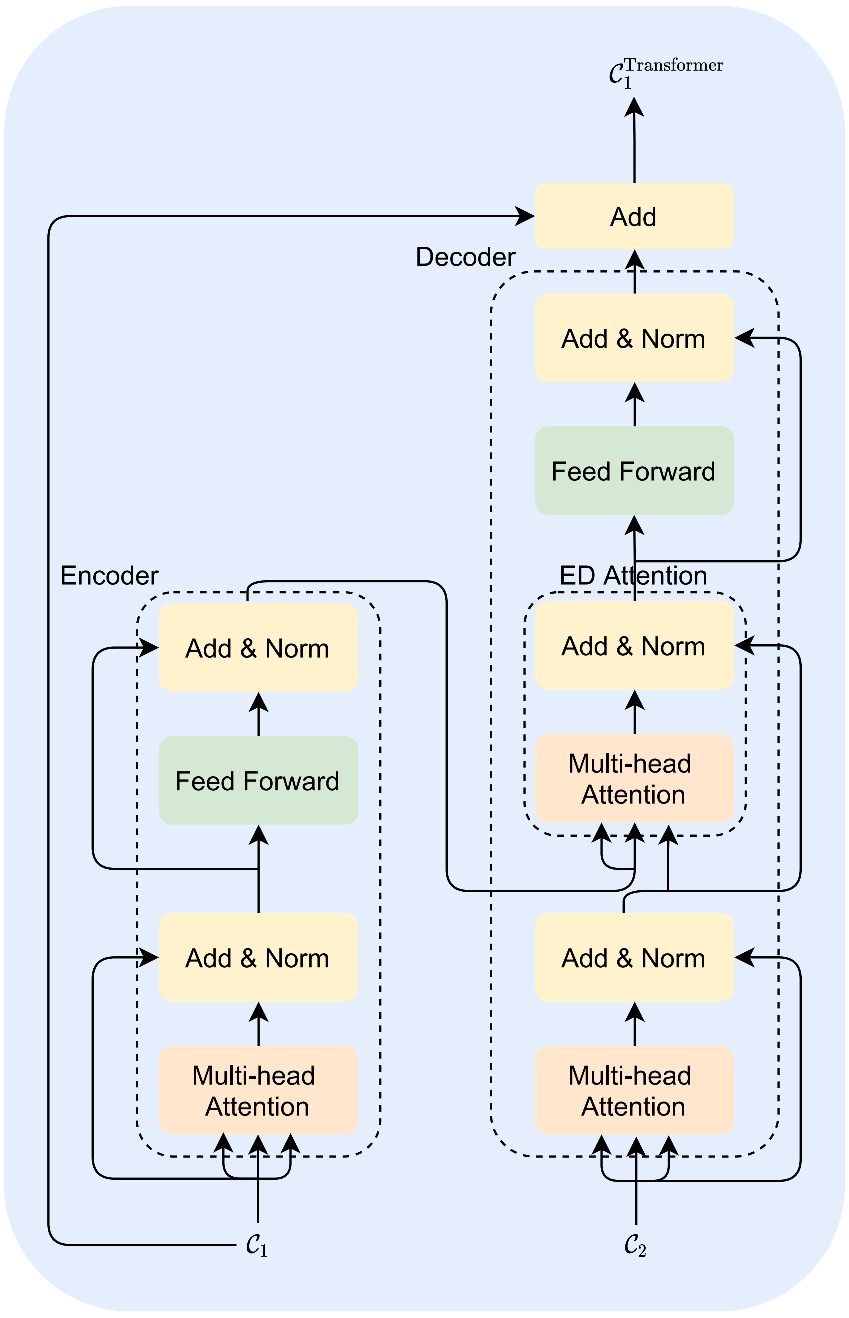

Figure 5.3: Transformer network architecture. ED stands for Encoder-Decoder.

5.4 A joint resampling preformed by a Transformer

In order to minimize the sampling noise impact on the registration results, we employ a strategy inspired by DCP. In [Wang and Solomon, 2019a] it is concluded that more reliable features for point clouds registration can be learned once their embeddings are processed simultaneously, by a Transformer.

This observation is motivated by the observation that the most useful features for rigid alignment are learned jointly from local and global information. These features for matching, can be further improve by making them *task-specific.* That is, changing the features by employing on the particularities of $\mathcal{P}_1$ and $\mathcal{P}_2$ together rather than embedding $\mathcal{P}_1$ and $\mathcal{P}_2$ independently. Inspired by the recent success of BERT [Devlin et al., 2018], non-local neural networks [Wang et al., 2018], and relational networks [Santoro et al., 2017] using attention-based models, a module to learn co-contextual information by capturing self-attention and conditional attention can be added. The matching problem encountered in rigid alignment is analogous to the the NLP sequence-to-sequence problem, other than its need for positional embeddings to describe where words are in a sentence. As the Transformer [Vaswani et al., 2017] was originally developed to solve the latter, DCP employ it in its architecture to perform a joint-resampling in the feature space for a further improvement in features matching.

We employ DCP Transformer for learning a joint-resampling strategy of the projected samples in $\mathcal{C}_1$ such that the resampling depends on the sampling of $\mathcal{C}_2$, and vice versa.

5.4.1 Attention mechanisms

Attention mechanisms, and particularly transformer-based deep networks [Vaswani et al., 2017], are nowadays the leading model in natural language processing, and its dominance is only growing. The rationale behind attention draws inspiration from attention cues in biology. A simplistic model of how human attention is divided treats the sensing of the environment, which is carried out non-voluntarily, as sensory inputs (referred to henceforth as *values*) and their corresponding *keys*, *i.e.*, the non-voluntarily cues which have led to the acquisition of the inputs. Our ability to focus our attention on particular parts of the environment is related to the current task, and is represented by the *query* cues. Our ability to focus our attention on particular portions of the sensory values is thus determined by both the queries and the keys. Attention architectures rely on three main building blocks:

1. Attention Pooling

   An attention pooling is used to model interactions between queries and keys based on an attention scoring function, and is given by a weighted sum of the values based on defined attention weights.

2. Multi-Head Attention

   Attention pooling are used as an intermediate mechanism in a larger model referred to as multi-head attention. Here, instead of performing a single attention pooling, the queries, keys, and values are transformed with a set of independently learned linear projections. Then these projected queries, keys, and values are fed into attention pooling in parallel. In the end, the attention pooling outputs are concatenated and transformed with another learned linear projection to produce the final output.

3. Self-Attention

   The treated attention mechanisms - queries, keys, and values are not obliged to be provided as inputs. These mechanisms can be converted into layers of neural networks by setting all these inputs to be the same input vector, referred to as *self-attention layer*.

Arguably, the leading application of self-attention mechanisms to design deep architectures is the transformer model. Though originally proposed for sequence to sequence learning on text data, Transformers have been pervasive in a wide range of modern deep learning applications, such as in areas of language, vision, speech, and reinforcement learning.

Transformer architecture

The overall architecture of the Transformer as used in this work, is illustrated in Figure 5.3. Each dashed encoder-decoder block in Figure 5.3 can be repeated multiple times. In our description (and implementation) we consider a single encoder-decoder block.

Encoder The encoder block is comprised of two sublayers. The first is a multi-head self-attention pooling block and the second is a 1-D convolutional neural network (typically comprised of two layers), implementing a feed-forward network. Specifically, in the encoder self-attention, queries, keys, and values are all from the the outputs of the previous encoder layer. Inspired by ResNet architectures [Khan et al., 2020], adding skip connection is a widelyused technique to improve the performance and the convergence of deep neural networks, which is believed to relieve the difficulty in optimization due to non-linearity by propagating a linear component through the neural network layers. Therefore, a residual connection is employed around both sublayers, which is followed by a normalization layer.

Decoder The transformer decoder is also a stack of multiple identical layers with residual connections and layer normalizations. Besides the two sublayers described in the encoder, the decoder inserts a third sublayer, known as the encoder-decoder attention, between these two. In the encoder-decoder attention, queries are from the outputs of the previous decoder layer (in our context, they form the target sampling locations of the input), and the keys and values are from the Transformer encoder outputs. In the decoder self-attention, queries, keys, and values are all from the the outputs of the previous decoder layer.

5.4.2 A joint-resampling

Loosely speaking, the action of the transformer is meant to improve performances in learned methods for various tasks, by jointly creating weighted sums that change the original input. Formally, take $\mathcal{C}_1$ and $\mathcal{C}_2$ to be the SO(3)-invariant coordinates generated by the module in 5.3; these representations are computed independently of one another. Our attention model learns a function

$$\phi : \mathbb{R}^{N\times 3} \times \mathbb{R}^{M\times 3} \to \mathbb{R}^{N\times 3} \tag{5.15}$$

that provides new coordinates for the point clouds

$$\mathcal{C}_1^{\text{Transformer}} = \mathcal{C}_1 + \phi(\mathcal{C}_1, \mathcal{C}_2), \quad \mathcal{C}_2^{\text{Transformer}} = \mathcal{C}_2 + \phi(\mathcal{C}_2, \mathcal{C}_1). \tag{5.16}$$

Notice we treat ϕ as a residual term, providing an additive change to $\mathcal{C}_1$ and $\mathcal{C}_2$ depending on the order of its inputs. The idea here is that the map $\mathcal{C}_1 \mapsto \mathcal{C}_1^{\text{Transformer}}$ modifies the features associated to the points in $\mathcal{C}_1$ in a fashion that is knowledgeable about the structure of $\mathcal{C}_2$; the map $\mathcal{C}_2 \mapsto \mathcal{C}_2^{\text{Transformer}}$ serves a symmetric role. The asymmetric function ϕ is given by a Transformer.

In principle, the additional terms, $\phi(\mathcal{C}_1, \mathcal{C}_2)$ and $\phi(\mathcal{C}_2, \mathcal{C}_1)$, added to $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively, are deformations meant to infuse information from one point cloud to the other. In our learning process this deformation terms are optimized to improve registration. A-priory, any deformation can be learnt by the Transformer, as long as it improves the registration performances. In the beginning of the learning process, the Transformer's action indeed badly distorts the point clouds shape. Curiously, as the learning process converges, the additional terms seem to preserve the shapes structure rather then deform it. This process of changing the point clouds without distorting the shape may be intuitively referred as a resampling process. One way to see the similarity between the shapes of Transformer's input and output is by considering the Chamfer distance. We average the Chamfer distance between $\mathcal{C}_i$ and $\mathcal{C}_i^{\text{Transformer}}$ over the ModelNet40 dataset, and get a distance of about 0.004 (where all point clouds are scaled to the unit sphere). This shows that a different point cloud is obtained after the Transformer's action, however the small Chamfer distance between the input and output point clouds (relatively to the point clouds size) shows that this difference is due to a change of sampling points rather a change in shape.

An alternative way to characterize a resampling of the point cloud is by considering the one sided Chamfer distance between the new set of points and the sampled object. The one sided Chamfer distance of a point cloud $\mathcal{P}_1$ from another point clouds $\mathcal{P}_2$ is given by

$$d_{\text{C}}(\mathcal{P}_1, \mathcal{P}_2) = \frac{1}{|\mathcal{P}_1|} \sum_{\mathbf{p} \in \mathcal{P}_1} \min_{\mathbf{p}' \in \mathcal{P}_2} \|\mathbf{p} - \mathbf{p}'\|_2 \tag{5.17}$$

and measures how much any point in $\mathcal{P}_1$ is far from its closest neighbor in $\mathcal{P}_2$, in average. Whenever our new set of points indeed represent a new set of samples from a physical object, its one sided Chamfer distance from a densely sampled version of this point cloud is small. On FAUST dataset, which is densely sampled (80K points per object), such a test is feasible. We average the one sided Chamfer distance between $\mathcal{C}_i$ (the Transformer's input) and $\mathcal{P}_i$, and compare it to the distance between $\mathcal{C}_i^{\text{Transformer}}$ (the Transformer's output) and $\mathcal{P}_i$; where $\mathcal{P}_i$ is the corresponding densely sampled version of the original object, consists of 80K points. As expected, in both cases this distance is small (relatively to the clouds scale): For the original point cloud $\mathcal{C}_i$ the average distance is 0.00081 and for its the transformed version $\mathcal{C}_i^{\text{Transformer}}$, it is 0.00144 (where here again all point clouds are scaled to the unit sphere). We speculate that the distances difference is due to the sampling method used for creating FAUST dataset. Unlike ModelNet40, where point clouds are artificially created by software, FAUST was captured by a real scanner.

Therefore, the original point cloud, which contain samples taken in a similar way to the compared densely sampled version, show better resemblance to the dense version compared to the transformed version, which is now sampled in a different way, due to the Transformers action.

The dual resampling process executed by the Transformer has two objectives in the UME framework. The first, as mentioned, is to provide DGCNN a good 'starting point' for constructing better invariant features. The second is related to the actual evaluation of the UME moments, where we use the re-sampled point clouds produced by the Transformer and re-project them on the corresponding principle axes to obtain

$$\widehat{\mathcal{P}}_1 = \mathbf{D}_1 \cdot \mathcal{C}_1^{\text{Transformer}} + \mathbf{m}_{\mathcal{P}_1}, \quad \widehat{\mathcal{P}}_2 = \mathbf{D}_2 \cdot \mathcal{C}_2^{\text{Transformer}} + \mathbf{m}_{\mathcal{P}_2}. \tag{5.18}$$

The point clouds, $\widehat{\mathcal{P}}_1$ and $\widehat{\mathcal{P}}_2$, are re-sampled versions of the original points clouds, related by the same rigid transformation that relates $\mathcal{P}_1$ and $\mathcal{P}_2$. We therefore apply the UME registration to the re-sampled point clouds $\widehat{\mathcal{P}}_1$ and $\widehat{\mathcal{P}}_2$, using the DGCNN generated functions designed to be SO(3) invariant in the presence of observation sampling noise. As demonstrated by the experimental results, applying UME registration on the re-projected re-sampled point clouds provides improved performance.

5.5 Loss

In order to overcome the possible ambiguity problem of symmetric objects, discussed in Section 6, we adopt the Chamfer distance [Barrow et al., 1977] as our loss function. That is, if $\hat{T}$ is the estimated transformation,

$$\mathcal{L}(\hat{T}) = d_C(\hat{T}(\mathcal{P}_1), \mathcal{P}_2). \tag{5.19}$$

Using this loss, the ambiguous examples such as those in ModelNet40 [Wu et al., 2015] do not damage the learning process, as even if an ambiguity exists and the registration is successful, the Chamfer distance will be small. The Chamfer loss function has another advantage as no labels are required for the learning process, what makes it unsupervised.

5.6 Implementation Details

The architecture of DeepUME is shown in Figure 2 in the paper. Similarly to DCP [Wang and Solomon, 2019a], we use 5 *EdgeConv* (denoted as DGCNN [Wang et al., 2019b]) layers. The number of filters in each layer are $[64, 64, 128, 256, 512]$. The number of heads in multi-head attention is 1 and the embedding dimension is 64. We use LayerNorm [Ba et al., 2016] without Dropout [Srivastava et al., 2014]. Adam [Kingma, 2014] is used to optimize the network parameters, with initial learning rate 0.001. We divide the learning rate by 10 at epochs $75, 150$ and 200, training for a total of 250 epochs. As the modules of the pre-processing and the UME are of closed-form and non iterative or brute force, their impact on the computational time is negligible. Training on DeepUME takes about 11 hours to converge with Pyotrch [Paszke et al., 2019] and an NVIDIA Quadro RTX6000 GPU. All point clouds renderings were produced by Mitsuba [Nimier-David et al., 2019].

Chapter 6

Experiments

In this section selected experiments showing key concepts or performance gains of methods discussed in previous chapters are presented.

We conduct experiments on three datasets: ModelNet40, FAUST and Stanford 3D Scanning Repository where the last two are used only for testing. We train our network using ModelNet40, which consists of 12,311 CAD models, in 40 categories where (80%) are used for training and the rest for testing.

Following previous works experimental settings, we uniformly sample 1,024 points from each model's outer surface and further center and rescale the model into the unit sphere. In our training and testing procedure, for each point cloud, we randomly (uniformly) choose a rotation drawn uniformly from the full range of Euler angles and a translation vector in $[-0.5, 0.5]$ in each axes. We apply the rigid transformation obtained from the resulting parameters on $\mathcal{P}_1$, followed by a random shuffling of the points order, and get $\mathcal{P}_2$. In noisy scenarios, a suitable noise is applied to $\mathcal{P}_2$. We train our framework in the scenario of Bernoulli noise (see bellow), in the specific case where $p_1 = p_2 = 0.5$, and test all scenarios using the trained configuration.

Each experiment is evaluated using four metrics: The RMSE metric, both for the rotation and the translation is defined by

$$\mathrm{RMSE}(\mathbf{R}) = \sqrt{\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^{\mathrm{gt}}\|_2^2},\ \boldsymbol{\theta}^{\mathrm{gt}} = [\theta_x^{\mathrm{gt}}\ \theta_y^{\mathrm{gt}}\ \theta_z^{\mathrm{gt}}]^{\mathrm{T}} \tag{6.1}$$

$$\mathrm{RMSE}(\mathbf{t}) = \sqrt{\|\hat{\mathbf{t}} - \mathbf{t}^{\mathrm{gt}}\|_2^2} \tag{6.2}$$

(where gt stands for ground-truth) as well as the Chamfer distance and the Hausdorff distance [Liu et al., 2018; Urbach et al., 2020], proposed next as alternative metrics where ambiguity issues are resolved. We compare our performances with the basic implementation of the UME method (coded by ourselves), ICP (implemented in Intel Open3D [Zhou et al., 2018]), and four learned methods; PointNetLK and DCP (benchmarks point cloud registration networks) as well as the recently proposed DeepGMR, [Yuan et al., 2020] and RGM, [Fu et al., 2021]. We retrain the baselines, adapting the code released by the authors, and test all compered methods in exactly the same setting.

6.1 The Effect of Sampling Rate on Registration under Sampling Noise Scenarios

We now demonstrate that the proposed method overcome the problem of a severe under-sampling. Figure 6.1 shows the registration performance of different methods under different points densities. Among the compared performances, we add here those of PCA, in which a set of corresponding points is obtained from the clouds principal vectors (see section 5.3).

As mentioned in chapters 1 and 3, many point cloud registration applications process under-sampled point clouds. Figure 6.1 shows that on dense point clouds, the compared registration methods achieve comparable results; But as the sampling rate decreases, the sampling noise effect is dominant and severely damage their results, while the ones of our model remain unchanged.

We would like to remark that for the closed form registration methods considered, where computational limitations are indulgent, registration is testable with up to 80,000 sample points per cloud. In that scenario, PCA and UME registration achieve RMSE($\mathbf{R}$) errors of 5.147 and 2.573 respectively. This is particularly interesting since it shows that PCA and UME, which are both critical parts in our framework, require about 160 times more samples from our method (80k vs 500) in order to achieve similar performances. It is further implied that the main effect of the proposed method is equivalent to the interpolation of sub-sampled point clouds where the quality criterion of the interpolation is its efficiency for registration purposes.

6.2 The Ambiguity Problem in ModelNet40

An ambiguity problem arises in point cloud registration whenever symmetric objects are considered, as more than a single rigid transformation (depending on the degree of symmetry of the object) can correctly align the two point clouds. In the noise free scenario such an ambiguity is trivially handled since a fixed point constellation undergoes a rigid transformation which in

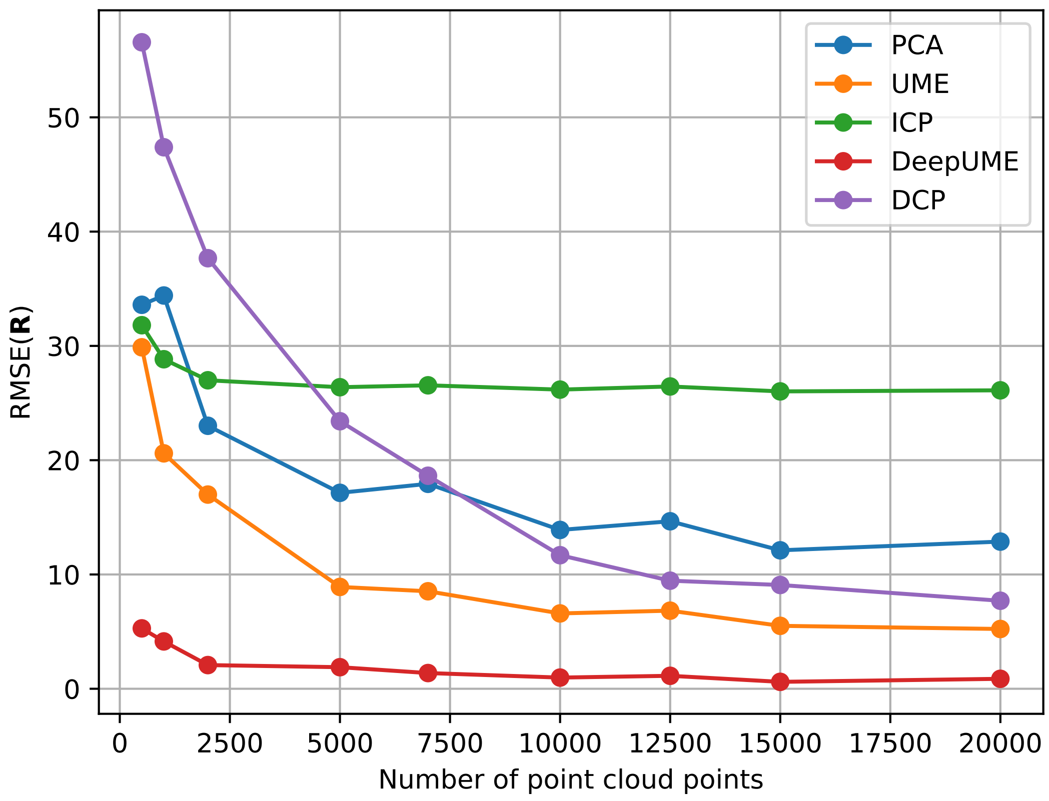

Figure 6.1: Registration performance under zero-intersection noise model and different points densities on the FAUST dataset. On dense point clouds, the compared registration methods achieve comparable results. As the sampling rate decreases, the sampling noise effect is dominant and severely damage their results, while the ones of our model remain unchanged.

the case of nonuniform sampling guaranties the uniqueness of the solution. However, when observations are noisy, the ambiguity is harder to resolve as the constellation structure breaks. In that scenario, $\mathcal{P}_2$ is a noisy version of $T(\mathcal{P}_1)$, and possibly no rigid transformation can perfectly align the point clouds. In that case, if the point clouds to be aligned represent a symmetric shape, there are multiple transformations that approximately align the two clouds together.

For symmetric shapes, which are very common in ModelNet40 dataset, the rotation angles RMSE metric may assign large errors to successful registrations. To resolve this ambiguity we suggest to replace this metric by the well known Chamfer and Hausdorff distances defined by

$$d_{\mathrm{C}}(\mathcal{P}_1, \mathcal{P}_2) = \frac{1}{|\mathcal{P}_1|} \sum_{\mathbf{p} \in \mathcal{P}_1} \min_{\mathbf{p}' \in \mathcal{P}_2} \|\mathbf{p} - \mathbf{p}'\|_2 + \frac{1}{|\mathcal{P}_2|} \sum_{\mathbf{p}' \in \mathcal{P}_2} \min_{\mathbf{p} \in \mathcal{P}_1} \|\mathbf{p}' - \mathbf{p}\|_2 \tag{6.3}$$

$$d_{\mathrm{H}}(\mathcal{P}_1, \mathcal{P}_2) = \max_{\mathbf{p} \in \mathcal{P}_1} \min_{\mathbf{p}' \in \mathcal{P}_2} \|\mathbf{p} - \mathbf{p}'\|_2 + \max_{\mathbf{p}' \in \mathcal{P}_2} \min_{\mathbf{p} \in \mathcal{P}_1} \|\mathbf{p}' - \mathbf{p}\|_2. \tag{6.4}$$

In Figure 6.2, we demonstrate the ambiguity problem. The box shaped bookshelf (as many other examples in ModelNet40) is symmetric under rotations of 180° about the z axes. Therefore, as shown the in Figure 6.2, in the sampling noise scenario under zero-intersection model (see below) two possible registration solutions (obtained from one to another by applying a 180° rotation about the z axes) are possible. Both solutions do align the point clouds, yet one achieves zero RMSE($\mathbf{R}$) error, while the other yields a much higher one. Nevertheless, both Chamfer and Hausdorff distances achieve small errors which implies they are better suited for measuring registration

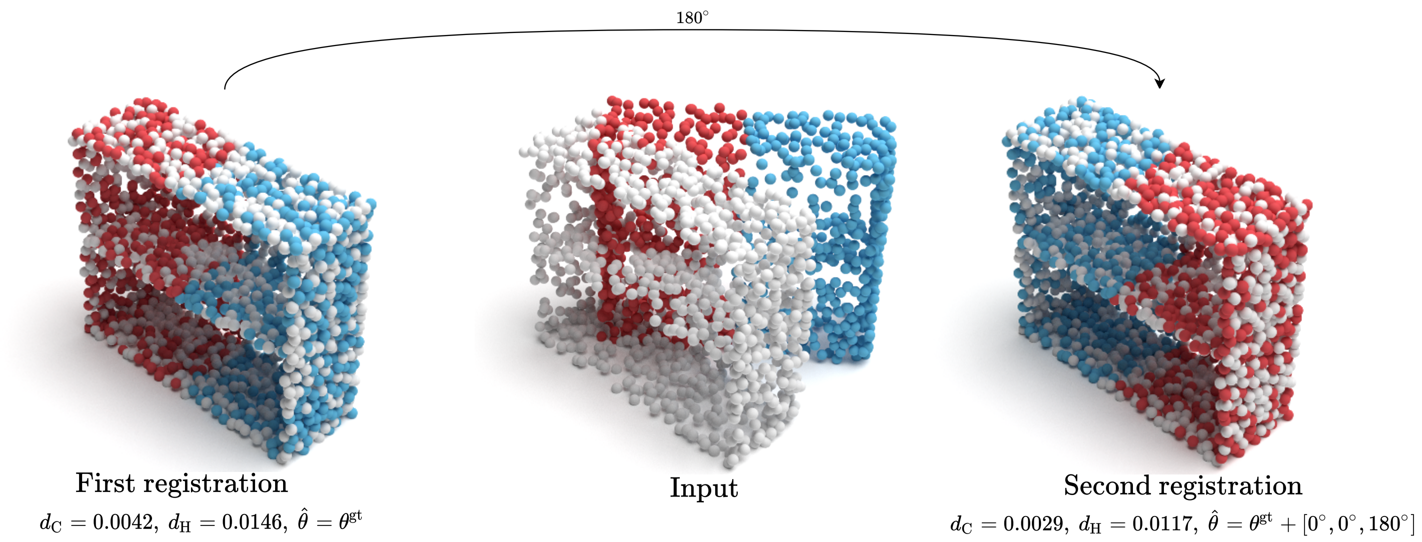
180°
First registration
$d_{\mathrm{C}} = 0.0042,\ d_{\mathrm{H}} = 0.0146,\ \hat{\theta} = \theta^{\mathrm{gt}}$
Input
Second registration
$d_{\mathrm{C}} = 0.0029,\ d_{\mathrm{H}} = 0.0117,\ \hat{\theta} = \theta^{\mathrm{gt}} + [0^\circ, 0^\circ, 180^\circ]$

Figure 6.2: ModelNet40 registration ambiguity arising in sampling noise scenarios. Due to the sampling noise there is no one true registration solution for the input, but any that aligns the clouds together. Therefore, the two presented solutions should result in a low error metric, which is reflected in the Chamfer and Hausdorff distances and not in RMSE($\mathbf{R}$).

error on symmetric shapes.

Using ModelNet40, we test performance, in four scenarios: noise free model, sampling noise (Bernoulli noise and zero-intersection noise) and additive white Gaussian coordinate noise. We note that in the presence of sampling noise we indeed observe large errors in RMSE(R) due to the symmetry of objects although the symmetric shapes are well aligned. Moreover, in the unseen datasets, where real world data is used, which is naturally asymmetric, the rotation RMSE is indeed small and indicates successful registration. Therefore we consider the Chamfer and Hausdorff distances to be more reliable metrics for registration in the presence of symmetries. Figure 6.7 shows registration results of several baseline methods and our proposed method on a representative example from the unseen dataset FAUST. All experimental results are summarized in Tables 6.1 and 6.2.

6.3 Seen Dataset

6.3.1 Noise free model

We examine the case where no noise is applied to the measurements. In that case, we see that the estimation error is practically null. DeepGMR is shown to be the second best learned method, while all other tested methods yield large errors, meaning that the registration fails when large range of rotation angles is considered.

6.3.2 Bernoulli noise

The Bernoulli noise case is related to the scenario of sampling noise, in which the point clouds contain different number of points. In that case, we choose randomly (uniformly) two numbers p_1 and p_2 in $[0.2, 1]$. Then, each point in $\mathcal{P}_i$ is removed with probability $1 - p_i$, independently from the rest of the points. We perform registration on the resulting point clouds $\mathcal{P}_1^B$ and $\mathcal{P}_2^B$. We note that the number of points in $\mathcal{P}_i^B$ averages to $p_i \cdot 2048$, and is likely to be different in the two clouds. The number of corresponding points between the resulting clouds averages to $p_1 p_2 \cdot 2048$ (as the probability for the two point clouds to share a specific point is $p_1 p_2$). We note that we were not able to evaluate RGM performance in that scenario.

The error surfaces depicted in Figure 6.3 describe the rotation RMSE with respect to the probabilities q_1 and q_2 in all three datasets tested. As

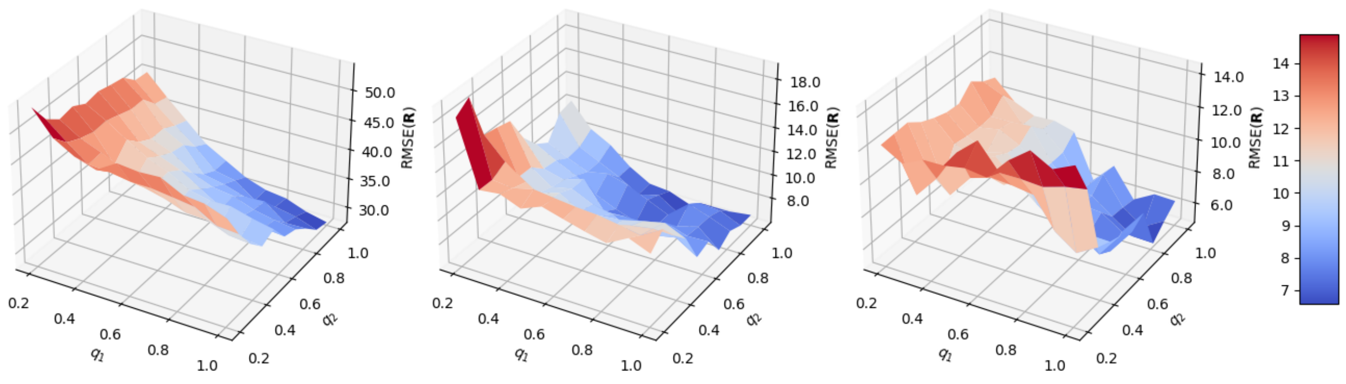

Figure 6.3: DeepUME performance (measured in RMSE($\mathbf{R}$)) under Bernoulli noise model with respect to the probabilities q_1 and q_2 in all three datasets tested. The surfaces visualization implies that the sparseness of the sparsest point clouds is the dominant cause of error, rather then the difference between the clouds densities.

expected, the error increases as q_1 and q_2 decrease and the point clouds are made sparser. Note that in Figure 6.3, in all the datasets considered, the rotation RMSE becomes large even when only one of the clouds is sparse (when the probability for keeping a point is small on one cloud and large on the other cloud). That is, the rotation RMSE does not increase much when we take the second cloud to be sparse as well. This might suggest that the sparseness of the sparsest point cloud is the dominant cause of error, rather then the difference between the clouds densities.

For a visualization of the effect of Bernoulli noise on a point cloud in extreme values of the probabilities q_1 and q_2 we refer the reader to Figure 6.4.

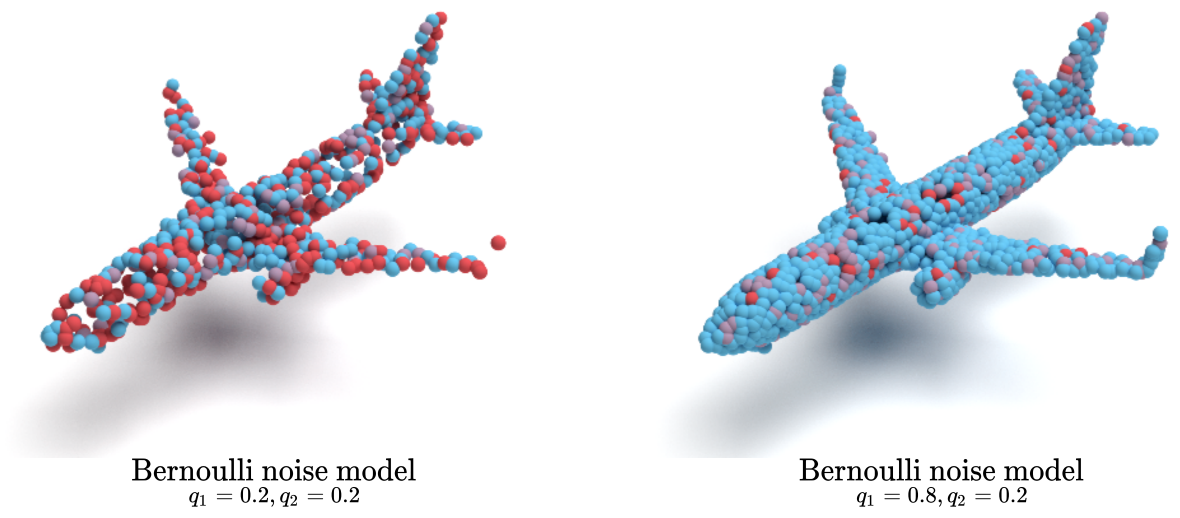

Figure 6.4: The effect of Bernoulli noise on a point cloud in extreme values of the probabilities q_1 and q_2. Red points denote $\mathcal{P}_1$, blue points denote $\mathcal{P}_2$ and purple points belong to both.

6.3.3 Zero-intersection noise

Zero intersection noise is the extreme (and most realistic) case of sampling noise. In that case, we randomly choose 1024 points from $\mathcal{P}_1$ to be removed. Next, we remove the 1024 points from $\mathcal{P}_2$ that correspond to the points in $\mathcal{P}_1$. Thus, by construction, no point in one cloud is the result of applying a rigid transformation to a point in the other cloud.

6.3.4 Additive Gaussian Coordinate Noise

In these set of experiments each coordinate of every point in $\mathcal{P}_2$ is perturbed by an additive white Gaussian random variable drawn from $\mathcal{N}(0, \sigma)$ where σ is chosen randomly in $[0, 0.04]$. All noise components are independent and no value clipping is performed. The results in Table 6.1 indicate that since the UME is an integral (summation) operator the registration error of both the

UME and the DeepUME is lower than the error of the other tested methods.

One of the most intriguing observations considers the effect of different types of noise on registration, and in particular, Additive White Gaussian Noise (AWGN) versus sampling noise. In Figure, 6.5 we evaluate the effect of the noise variance on the rotation RMSE. An interesting observation is that registration on ModelNet40 dataset [Wu et al., 2015] is significantly more affected by AWGN, than other types of data. Recalling the ambiguity problem of ModelNet40 this is quite expected: Noise makes the inherent ambiguity problem harder to resolve. In the presence of noise, many of the ambiguous examples are falsely registered and therefore the average error increases dramatically.

We find that for sufficiently large noise variance, the rotation RMSE in the AWGN case and the rotation RMSE for both sampling noise scenarios, are comparable. In order to illustrate the very different impact of the different types of noise on registration, we refer the reader to Figure 6.6. In order to have the same registration error as in the zero-intersection model on the Stanford dataset, an AWGN with variance of approximately 0.09 is required. As shown in Figure 6.6, such a noise causes a sever distortion, which practically makes the original shape unrecognizable. On the other hand, the original shape of the bunny in the sampling noise scenario is well preserved.

6.4 Unseen Dataset

The generalization ability of a model to unseen data is an important aspect for any learning based framework. In order to demonstrate that our frame-

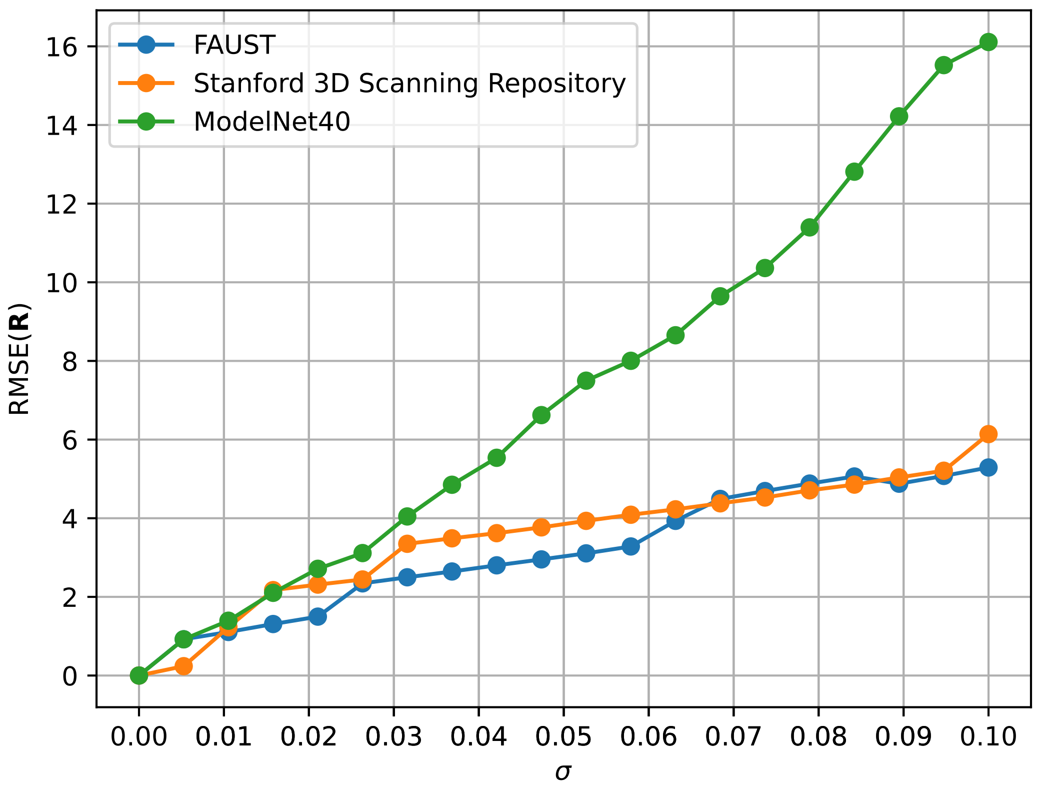

Figure 6.5: The effect of the WAGN variance on DeepUME performance (measured in RMSE($\mathbf{R}$)) in all three datasets tested. In the presence of noise, more symmetric examples in ModelNet40 dataset are incorrectly registered and therefore the average error increases dramatically.

	Noise free				Bernoulli noise				Gaussian noise			
Model	$d_{\mathbf{C}}$	$d_{\mathbf{H}}$	**RMSE**(R)	**RMSE**(t)	$d_{\mathbf{C}}$	$d_{\mathbf{H}}$	**RMSE**(R)	**RMSE**(t)	$d_{\mathbf{C}}$	$d_{\mathbf{H}}$	**RMSE**(R)	**RMSE**(t)
UME [Efraim and Francos, 2019]	<1e-04	<7e-04	0.193	<1e-05	0.0581	0.394	74.164	0.015	0.019	0.151	27.684	0.002
ICP [Besl and McKay, 1992a]	0.275	1.446	83.039	0.276	0.297	1.480	86.381	0.286	0.266	1.436	83.073	0.277
PointNetLK [Aoki et al., 2019]	0.027	0.142	80.360	1.0105	0.029	0.157	83.280	1.073	0.026	0.153	81.843	1.037
DCP [Wang and Solomon, 2019a]	0.059	0.470	92.285	0.014	0.067	0.483	92.818	0.020	0.055	0.456	90.715	0.014
DeepGMR [Yuan et al., 2020]	<7e-06	<9e-05	0.193	<5e-05	0.033	0.224	72.447	0.018	0.011	0.085	42.515	0.004
RGM [Fu et al., 2021]	0.255	1.333	99.937	0.388	N/A	N/A	N/A	N/A	0.254	1.331	100.117	0.389
DeepUME (ours)	**<1e-07**	**<1e-07**	**<3e-04**	**<1e-07**	**0.010**	**0.083**	**40.357**	**0.015**	**0.002**	**0.012**	**2.425**	**0.001**

Table 6.1: ModelNet40 experimental results. Our method achieves substantial performance gains compared to the competing techniques for all metrics in all the examined scenarios. $d_{\mathbf{C}}$ and $d_{\mathbf{H}}$ stands for Chamfer and Hausdorff distances respectively. Best results are **bold** and second best are underlined.

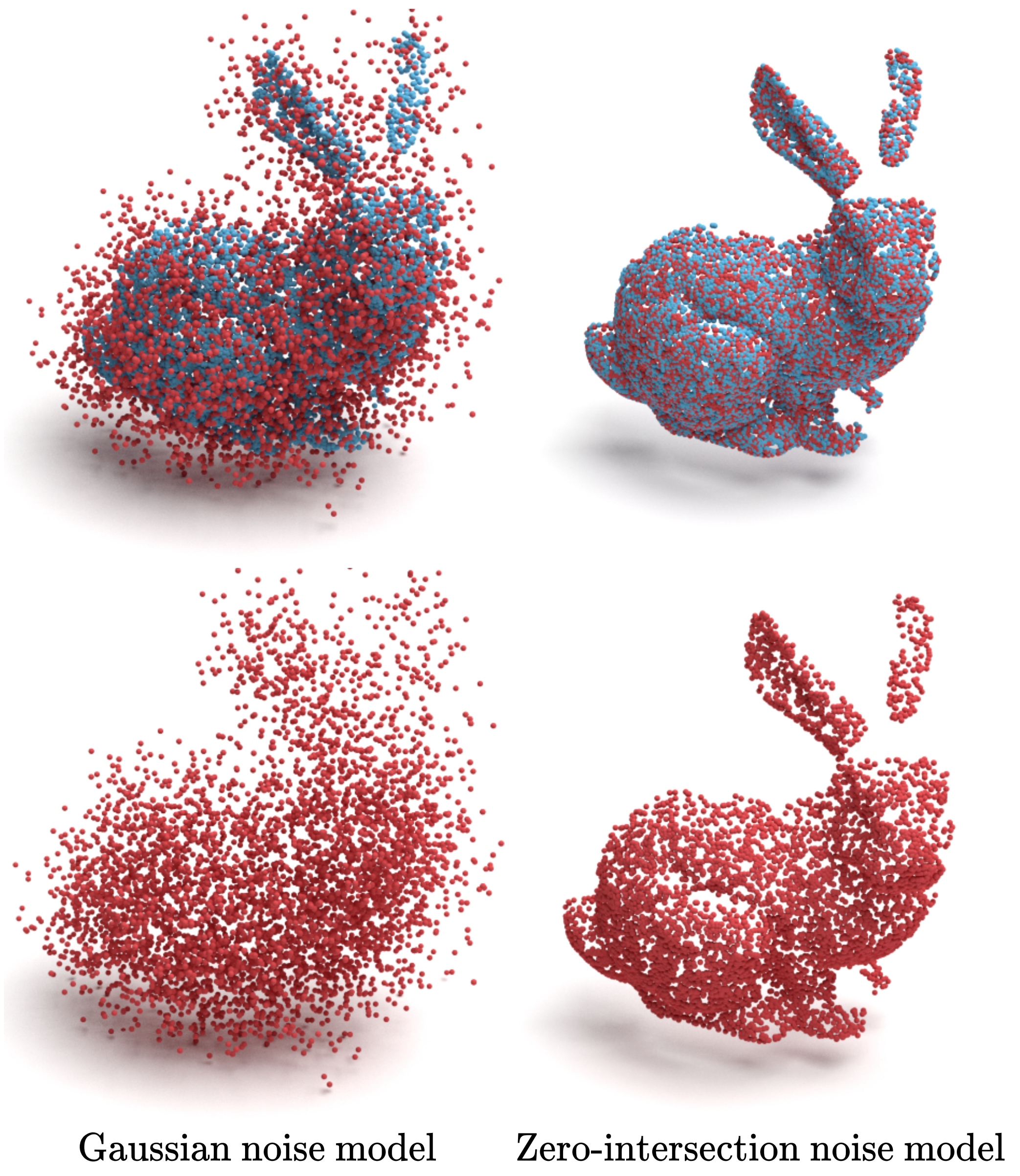

Figure 6.6: In the top row $\mathcal{P}_1$ and $\mathcal{P}_2$ are presented together, and in the bottom row only $\mathcal{P}_2$ is shown. In order to cause the same registration error as in the zero-intersection noise model on the Stanford dataset, an AWGN with variance of approximately 0.09 is required. While the former preserves well the original shape, the latter sever distortion makes it barely recognizable.

	ModelNet40 [Wu et al., 2015]				FAUST [Bogo et al., 2014]				Stanford 3D Scanning Repository [Stanford Scanning Repository, 1994]			
Model	$d_{\mathbf{C}}$	$d_{\mathbf{H}}$	**RMSE**(R)	**RMSE**(t)	$d_{\mathbf{C}}$	$d_{\mathbf{H}}$	**RMSE**(R)	**RMSE**(t)	$d_{\mathbf{C}}$	$d_{\mathbf{H}}$	**RMSE**(R)	**RMSE**(t)
UME [Efraim and Francos, 2019]	0.051	0.373	80.331	0.010	0.007	0.085	35.983	0.044	0.033	0.267	48.716	0.010
ICP [Besl and McKay, 1992a]	0.276	1.448	82.948	0.277	0.376	1.643	84.544	0.279	0.288	1.337	87.292	0.277
PointNetLK [Aoki et al., 2019]	0.028	0.147	80.858	1.023	0.018	0.170	90.512	1.120	0.040	0.289	84.520	1.147
DCP [Wang and Solomon, 2019a]	0.059	0.475	93.221	0.014	0.046	0.516	94.315	0.137	0.072	0.522	99.328	0.011
DeepGMR [Yuan et al., 2020]	0.026	0.117	**67.282**	0.010	0.003	0.027	27.941	0.020	0.005	0.119	39.402	0.012
RGM [Fu et al., 2021]	0.254	1.335	100.970	0.388	0.385	1.677	114.496	0.418	0.278	1.257	104.872	0.368
DeepUME (ours)	**0.011**	**0.094**	70.818	**0.009**	**0.002**	**0.024**	**8.630**	**0.019**	**0.002**	**0.110**	**5.625**	**0.010**

Table 6.2: Zero-intersection noise results on seen (ModelNet40) and unseen datasets (FAUST and Stanford 3D Scanning Repository). Our method outperforms the competing techniques in all scenarios for all metrics, only except RMSE($\mathbf{R}$) for ModelNet40.

work indeed generalizes well, we test it on two unseen datasets. The first is the FAUST dataset that contains human scans of 10 different subjects in 30 different poses each with about 80,000 points per shape, and the other is the Stanford 3D Scanning Repository. We generate the objects to be registered using a similar methodology to that employed for the ModelNet40 dataset. Our framework achieves accurate registration results in all scenarios checked, and shows superior performance over the compared methods.

6.5 Additional Registration Results

6.5.1 Small rotation angles range

In our experiments, all DGCNN-based networks, except ours, provide poor registration. This is expected, as without a proper pre-processing procedure such networks fail to create features that are invariant under large rotations. The experimental results demonstrate that on the average, when large rotations are allowed, the registration results of the proposed method outperform

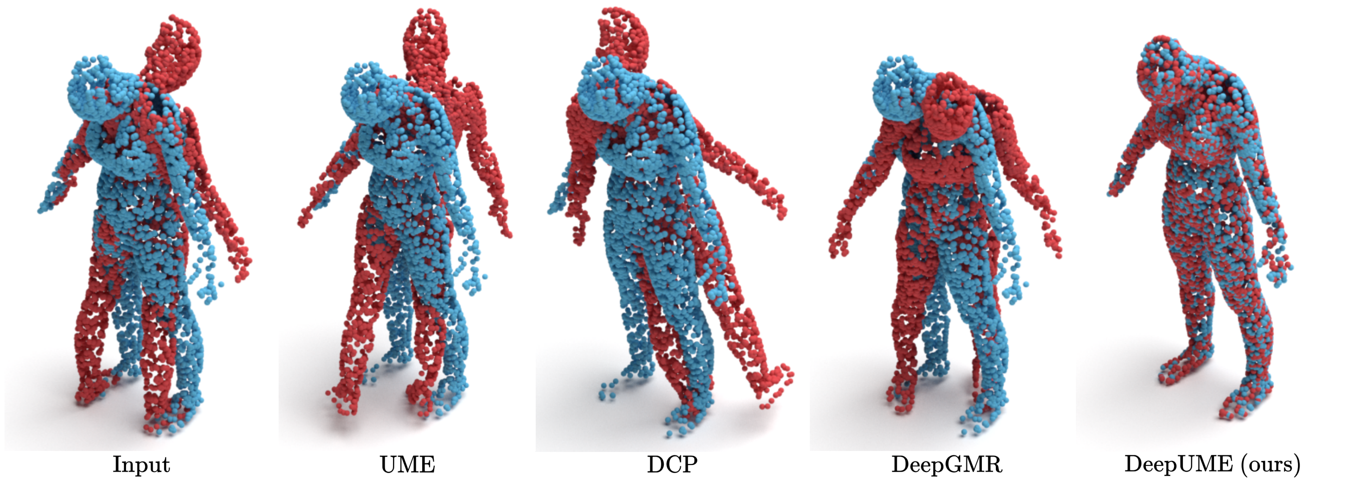

Figure 6.7: Registration results on an unseen dataset, where the observations are subject to a large relative rotation and sampling noise (zero-intersection model). While UME [Efraim and Francos, 2019] and DCP [Wang and Solomon, 2019a] fail to align the objects, and DeepGMR [Yuan et al., 2020] results with a substantial registration error, the proposed method successfully aligns the shapes.

the alternatives. However, for a more comprehensive overview, we also compare our performance separately for the case of registration under rotation by small angles (up to $60°$ about each axes).

For a fair comparison, we use the pre-trained models released by the authors and test all the proposed methods with rotation in the range $[0, 60]$ degrees about each of the axes, for the noise free and zero-intersection noise models, in a similar manner to the above described experiments. We note that in the small rotations experiment, none of the tested methods encounters the ambiguity problem of ModelNet40.The symmetry in the examples of ModelNet40 creates ambiguity where rotations by angles larger then $90°$ are considered. Since our method is designed for arbitrary rotations, we do suffer from the ambiguity problem in ModelNet40. Hence, for generating a reliable comparison, we perform that experiment on the Stanford 3D

	Noise free				Zero-intersection noise			
Model	$d_{\mathbf{C}}$	$d_{\mathbf{H}}$	**RMSE**(R)	**RMSE**(t)	$d_{\mathbf{C}}$	$d_{\mathbf{H}}$	**RMSE**(R)	**RMSE**(t)
UME [Efraim and Francos, 2019]	<9e-05	<4e-04	0.333	<5e-06	0.030	0.269	42.559	0.010
ICP [Besl and McKay, 1992a]	0.280	1.274	72.265	0.277	0.267	1.256	71.532	0.276
PointNetLK [Aoki et al., 2019]	<7e-04	0.005	0.729	0.005	0.012	0.113	11.504	0.095
DCP [Wang and Solomon, 2019a]	<4e-04	0.003	5.739	0.002	0.065	0.479	71.307	0.011
DeepGMR [Yuan et al., 2020]	<4e-06	<2e-05	0.110	<6e-05	0.294	0.270	33.128	0.015
RGM [Fu et al., 2021]	0.268	1.203	0.025	<2e-04	0.267	1.187	**4.690**	0.032
DeepUME (ours)	**<1e-07**	**<1e-07**	**<8e-05**	**<1e-07**	**0.002**	**0.118**	5.159	**0.010**

Table 6.3: Free and zero-intersection noise models results on the unseen dataset Stanford 3D Scanning Repository in small rotation angles regime of $[0, 60]$ degrees about each axes. Our method outperforms the competing techniques in the examined scenarios for all metrics, only except the RMSE($\mathbf{R}$) in the zero-intersection noise model. These results suggest that our framework improves the state-of-the-art methods not only on the average over the entire range, but also in the small angles scenario, where most of the methods were designed to be optimal.

Scanning Repository [Stanford Scanning Repository, 1994], which does not contain ambiguous examples.

From the results summarized in Table 6.3, we conclude that the proposed method outperforms all compared methods in all metrics examined (except for one case, where RGM method [Fu et al., 2021] achieves slightly smaller rotation RMSE). This shows that the proposed framework outperforms state-of-the-art methods not only on the average on the entire transformation range, but also for the small angles scenario.

6.6 Ablation Study

We conduct several ablation studies, removing components of the proposed DeepUME and replacing each part with an alternative, to better evaluate our design. The studies were tested for the Gaussian noise and zero-intersection noise models on the unseen FAUST [Bogo et al., 2014] dataset, in a similar manner to the experiments described above. The results of this section are summarized in Table 6.4.

6.6.1 With or without invariant coordinates?

We first try to evaluate whether the new transformation-invariant coordinates generated for the point cloud at the pre-processing phase, provide value over the original representation of the point cloud. We therefore remove the pre-processig module (presented in Figure 5.1) and compare the resulting performance to that obtained using the full model. Table 6.4 demonstrates that DeepUME performs consistently better with the inclusion of the pre-processig module.

6.6.2 Coordinates joint-resampling or features joint-resampling?

In our proposed framework, the Transformer layer executes a joint-resampling procedure in the coordinated space of $\mathbb{R}^3$. This is unlike other networks, where the joint-resampling process is executed in feature space. Applying the Transformer in the coordinate space has a notable computational advantage as in this case, the embedding size of each point is significantly lower. In our our network the embedding size of each point on the coordinate space is 3 while in the feature space it is 512. Hence, performing a coordinate sampling allows for a dramatic decrease in computational complexity, both in the train and evaluation processes (twice faster). We compare our framework

	Zero-intersection noise				Gaussian noise			
Model	$d_{\mathbf{C}}$	$d_{\mathbf{H}}$	**RMSE**(R)	**RMSE**(t)	$d_{\mathbf{C}}$	$d_{\mathbf{H}}$	**RMSE**(R)	**RMSE**(t)
No pre-processing	0.094	0.913	83.506	0.140	0.087	0.896	82.260	0.140
Features joint-sampling	0.007	0.082	14.913	0.027	0.001	0.012	1.533	0.002
MLP instead of UME	<u>0.003</u>	<u>0.046</u>	<u>12.964</u>	<u>0.023</u>	<u>0.001</u>	<u>0.011</u>	<u>1.488</u>	<u>0.003</u>
DeepUME (full model)	**0.002**	**0.024**	**8.630**	**0.019**	**0.001**	**0.011**	**1.069**	**0.002**

Table 6.4: Ablation study results on the unseen dataset FAUST [Bogo et al., 2014]. DeepUME full model achieves equal or better registration results in all tested scenarios and in all metrics.

with the two strategies - resampling in coordinate space and resampling in features space. The results presented in Table 6.4 show that the decrease in computational complexity does not cause a decrease in registration accuracy.

6.6.3 UME or MLP?

While MLP (Multi Layer Perceptron) provides, in principle, a universal approximation, the UME integration into a DNN framework is designed to provide an accurate computation of a rigid motion under noisy sampling of point clouds. A natural question to ask is whether the UME parameter extraction may be replaced by a general learned module, such that registration with comparable accuracy is achieved. As expected, Table 6.4 shows that the model performs better with the UME layer than a general MLP.

Chapter 7

Discussion

7.1 Integrating closed-form methods into a DNN framework

The deep learning technique could serve as a feature extraction tool to replace the original point coordinate. The conventional optimization could provide a theoretical guarantee for the convergence. Firstly, advanced loss calculation strategies are developed to apply an optimization strategy to calculate an estimated transformation from the learned feature. Secondly, calculate the loss between the estimated transformation and ground truth. Many existing methods [Wang and Solomon, 2019a], [Huang et al., 2020] demonstrate that combining both advantages could achieve both high accuracy and efficiency. For instance, deep closest point (DCP [Wang and Solomon, 2019a]) uses deep features to estimate correspondences and use SVD to calculate the transformation. FMR [Huang et al., 2020] applies deep learning to extract global feature and uses Lukas-Kanade (LK) algorithm to minimize the feature difference. [Fey et al., 2020] uses deep learning to calculate the soft correspondences and use message passing network to refine the correspondences. DeepGMR [Yuan et al., 2020] uses deep learning to calculate the correspondences between Gaussian models and points and optimize the transformation based on GMM optimization.

These existing approaches provide initial trials on conventional optimization and deep neural networks to solve registration problems. Combining conventional optimization theory and recent deep neural networks is a promising way to provide high accuracy and efficiency and theoretically guarantee current deep learning based registration methods. The research direction is to design advanced loss calculation strategies to optimize the neural network by combining the existing optimization strategies [Huang et al., 2021].

7.2 Conclusions

We derived a novel solution to the highly practical problem of aligning differently sampled and noisy point clouds. The solution integrates the closed form UME registration into a DNN framework. Since the UME is an operator defined on functions of the coordinates, in order to enable registration, these functions need to be invariant to the transformation. We generate such invariant features using an unsupervised deep neural network architecture designed to jointly resample the two point clouds to minimize the effect of the sampling noise. The derived new discrete version of the UME operator and its integration into the DNN framework enable us to achieve accurate registration in various noisy scenarios. In addition, projecting the point clouds on transformation invariant coordinate system, removes a major obstacle in the learning process of DGCNN-based networks when registration under large rotations is considered.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Aiger, D., Mitra, N. J., and Cohen-Or, D. (2008a). *4-points congruent sets for robust pairwise surface registration.*

Aiger, D., Mitra, N. J., and Cohen-Or, D. (2008b). 4-points congruent sets for robust surface registration. *ACM Transactions on Graphics*, 27(3):#85, 1–10.

Aoki, Y., Goforth, H., Srivatsan, R. A., and Lucey, S. (2019). Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7163–7172.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., and Wolf, H. C. (1977).

Parametric correspondence and chamfer matching: Two new techniques for image matching. Technical report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER.

Besl, P. J. and McKay, N. D. (1992a). Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics.

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

Bogo, F., Romero, J., Loper, M., and Black, M. J. (2014). Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3794–3801.

Bookstein, F. L. (1989). Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585.

Bresson, G., Alsayed, Z., Yu, L., and Glaser, S. (2017). Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220.

Bülow, H. and Birk, A. (2018). Scale-free registrations in 3d: 7 degrees of freedom with fourier mellin soft transforms. *International Journal of Computer Vision*, 126(7):731–750.

Campbell, D. and Petersson, L. (2016). Gogma: Globally-optimal gaussian mixture alignment. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5685–5694.

Campbell, D., Petersson, L., Kneip, L., Li, H., and Gould, S. (2019). The alignment of the spheres: Globally-optimal spherical mixture alignment for camera pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11796–11806.

Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155.

Chetverikov, D., Stepanov, D., and Krsek, P. (2005). Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. *Image and vision computing*, 23(3):299–309.

Chetverikov, D., Svirko, D., Stepanov, D., and Krsek, P. (2002). Object recognition supported by user interaction for service robots.

Choy, C., Dong, W., and Koltun, V. (2020). Deep global registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2514–2523.

Collis, R. (1970). Lidar. *Applied optics*, 9(8):1782–1788.

Darom, T. and Keller, Y. (2012). Scale-invariant features for 3-d mesh models. *IEEE Transactions on Image Processing*, 21(5):2758–2769.

Darom, T. and Keller, Y. (2012). Scale-invariant features for 3-d mesh models. *IEEE Transactions on Image Processing*, 21(5):2758–2769.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Deng, H., Birdal, T., and Ilic, S. (2018a). Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 602–618.

Deng, H., Birdal, T., and Ilic, S. (2018b). Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 195–205.

Deng, H., Birdal, T., and Ilic, S. (2019). 3d local features for direct pairwise registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3244–3253.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhawale, A., Shankar, K. S., and Michael, N. (2018). Fast monte-carlo localization on aerial vehicles using approximate continuous belief representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5851–5859.

Drost, B., Ulrich, M., Navab, N., and Ilic, S. (2010). Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005. Ieee.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110.

Eckart, B., Kim, K., and Kautz, J. (2018). Hgmr: Hierarchical gaussian mixtures for adaptive 3d registration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 705–721.

Efraim, A. and Francos, J. M. (2019). The universal manifold embedding for estimating rigid transformations of point clouds. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5157–5161. IEEE.

Elbaz, G., Avraham, T., and Fischer, A. (2017). 3d point cloud registration for localization using a deep neural network auto-encoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640.

Evangelidis, G. D. and Horaud, R. (2017). Joint alignment of multiple point sets with batch and incremental expectation-maximization. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1397–1410.

Fan, H., Su, H., and Guibas, L. J. (2017). 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.

Faugeras, O. and Hebert, M. (1986). *The Representation, Recognition, and Locating of 3-D Objects.*

Ferencz, I. H. and Shimshoni, I. (2017a). Registration of 3d point clouds using mean shift clustering on rotations and translations. In *2017 International Conference on 3D Vision (3DV)*, pages 374–382. IEEE.

Ferencz, I. H. and Shimshoni, I. (2017b). Registration of 3d point clouds using mean shift clustering on rotations and translations. *2017 Int. Conf. 3D Vis. (3DV)*, pages 374–382.

Fey, M., Lenssen, J. E., Morris, C., Masci, J., and Kriege, N. M. (2020). Deep graph matching consensus. *arXiv preprint arXiv:2001.09621.*

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Fitzgibbon, A. W. (2003). Robust registration of 2d and 3d point sets. *Image and vision computing*, 21(13-14):1145–1153.

Fu, K., Liu, S., Luo, X., and Wang, M. (2021). Robust point cloud registration framework based on deep graph matching. *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR).*

Gao, W. and Tedrake, R. (2019). Filterreg: Robust and efficient probabilistic point-set registration using gaussian filter and twist parameterization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11095–11104.

Gojcic, Z., Zhou, C., Wegner, J. D., and Wieser, A. (2019). The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of* *the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5545–5554.

Granger, S. and Pennec, X. (2002). Multi-scale em-icp: A fast and robust approach for surface registration. In *European Conference on Computer Vision*, pages 418–432. Springer.

Groß, J., Ošep, A., and Leibe, B. (2019). Alignnet-3d: Fast point cloud registration of partially observed objects. In *2019 International Conference on 3D Vision (3DV)*, pages 623–632. IEEE.

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., and Kwok, N. M. (2016). A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89.

Hagege, R. and Francos, J. M. (2010). Parametric estimation of affine transformations: An exact linear solution. *Journal of Mathematical Imaging and Vision*, 37(1):1–16.

Hagege, R. R. and Francos, J. M. (2016). Universal manifold embedding for geometrically deformed functions. *IEEE Transactions on Information Theory*, 62(6):3676–3684.

Hill, D. L., Batchelor, P. G., Holden, M., and Hawkes, D. J. (2001). Medical image registration. *Physics in medicine & biology*, 46(3):R1.

Horn, B. K., Hilden, H. M., and Negahdaripour, S. (1988a). Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135.

Horn, B. K. P. (1984). Extended gaussian images. *Proceedings of the IEEE*, 72(12):1671–1686.

Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642.

Horn, B. K. P., Hilden, H., and Negahdaripour, S. (1988b). Closed-form solution of absolute orientation using orthonormal matrices. *JOURNAL OF THE OPTICAL SOCIETY AMERICA*, 5(7):1127–1135.

Huang, X., Mei, G., and Zhang, J. (2020). Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11366–11374.

Huang, X., Mei, G., Zhang, J., and Abbas, R. (2021). A comprehensive survey on point cloud registration. *arXiv preprint arXiv:2103.02690*.

Isola, P., Xiao, J., Torralba, A., and Oliva, A. (2011). What makes an image memorable? In *CVPR 2011*, pages 145–152. IEEE.

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

Johnson, A. E. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449.

Khan, A., Sohail, A., Zahoora, U., and Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516.

Kingma, D. P. (2014). Ba j.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 531.

Kuo, C.-C. J. and Chen, Y. (2018). On data-driven saak transform. *Journal of Visual Communication and Image Representation*, 50:237–246.

Kuo, C.-C. J., Zhang, M., Li, S., Duan, J., and Chen, Y. (2019). Interpretable convolutional neural networks via feedforward design. *Journal of Visual Communication and Image Representation*, 60:346–359.

Lang, I., Manor, A., and Avidan, S. (2020). Samplenet: Differentiable point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7578–7588.

Lawin, F. J., Danelljan, M., Khan, F. S., Forssén, P.-E., and Felsberg, M. (2018). Density adaptive point set registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3829–3837.

Li, J., Zhang, C., Xu, Z., Zhou, H., and Zhang, C. (2020). Iterative distance-aware similarity matrix convolution with mutual-supervised point elimination for efficient point cloud registration. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pages 378–394. Springer.

Liu, Y., Kong, D., Zhao, D., Gong, X., and Han, G. (2018). A point cloud registration algorithm based on feature extraction and matching. *Mathematical Problems in Engineering*, 2018.

Low, K.-L. (2004). Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4(10):1–3.

Lu, W., Wan, G., Zhou, Y., Fu, X., Yuan, P., and Song, S. (2019). Deepvcp: An end-to-end deep neural network for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–21.

Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision. In *Proc. of the Intl. Joint Conference on Artificial Intelligence*. Vancouver, British Columbia.

Maes, C., Fabry, T., Keustermans, J., Smeets, D., Suetens, P., and Vandermeulen, D. (2010a). Feature detection on 3d face surfaces for pose normalisation and recognition. In *2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–6. IEEE.

Maes, C., Fabry, T., Keustermans, J., Smeets, D., Suetens, P., and Vandermeulen, D. (2010b). Feature detection on 3d face surfaces for pose normalisation and recognition. In *2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–6. IEEE.

Magnusson, M., Lilienthal, A., and Duckett, T. (2007). Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics*, 24(10):803–827.

Mellado, N., Aiger, D., and Mitra, N. J. (2014a). *Super 4pcs fast global pointcloud registration via smart indexing*. Wiley Online Library.

Mellado, N., Aiger, D., and Mitra, N. J. (2014b). Super 4pcs fast global pointcloud registration via smart indexing. *Computer Graphics Forum*, 33(5):205–215.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., et al. (2008). Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597.

Myronenko, A. and Song, X. (2009). On the closed-form solution of the rotation matrix arising in computer vision problems. *arXiv preprint arXiv:0904.1613*.

Nimier-David, M., Vicini, D., Zeltner, T., and Jakob, W. (2019). Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)*, 38(6):1–17.

Pais, G. D., Ramalingam, S., Govindu, V. M., Nascimento, J. C., Chellappa, R., and Miraldo, P. (2020). 3dregnet: A deep neural network for 3d point registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7193–7203.

Papadopoulo, T. and Lourakis, M. I. (2000). Estimating the jacobian of the singular value decomposition: Theory and applications. In *European Conference on Computer Vision*, pages 554–570. Springer.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). *Automatic differentiation in pytorch*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pomerleau, F., Colas, F., and Siegwart, R. (2015). A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104.

Pottmann, H., Leopoldseder, S., and Hofer, M. (2004a). Registration without icp. *Computer Vision and Image Understanding*, 95(1):54–71.

Pottmann, H., Leopoldseder, S., and Hofer, M. (2004b). Registration without icp. *Computer Vision and Image Understanding*, 95(1):54 – 71.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). 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.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*.

Rusinkiewicz, S. (2019). A symmetric objective function for icp. *ACM Transactions on Graphics (TOG)*, 38(4):1–7.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE.

Rusu, R. B., Blodow, N., Marton, Z. C., and Beetz, M. (2008). Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*.

Segal, A., Haehnel, D., and Thrun, S. (2009). *Generalized-icp*. Seattle, WA.

Seo, H. J. and Milanfar, P. (2010). Action recognition from one example. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):867–882.

Sipiran, I. and Bustos, B. (2011a). Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. *The Visual Computer*, 27(11):963–976.

Sipiran, I. and Bustos, B. (2011b). Harris 3d: A robust extension of the harris operator for interest point detection on 3d meshes. *The Visual Computer*, 27:963–976.

Smisek, J., Jancosek, M., and Pajdla, T. (2013). 3d with kinect. In *Consumer depth cameras for computer vision*, pages 3–25. Springer.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Stanford Scanning Repository (1994). The Stanford 3D Scanning Repository.

Stoyanov, T., Magnusson, M., Andreasson, H., and Lilienthal, A. J. (2012). Fast and accurate scan registration through minimization of the distance between compact 3d ndt representations. *The International Journal of Robotics Research*, 31(12):1377–1393.

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., and Kautz, J. (2018). Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2530–2539.

Tabib, W., O'Meadhra, C., and Michael, N. (2018). On-manifold gmm registration. *IEEE Robotics and Automation Letters*, 3(4):3805–3812.

Tam, G. K., Cheng, Z.-Q., Lai, Y.-K., Langbein, F. C., Liu, Y., Marshall, D., Martin, R. R., Sun, X.-F., and Rosin, P. L. (2012). Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. *IEEE transactions on visualization and computer graphics*, 19(7):1199–1217.

Taylor, J., Tankovich, V., Tang, D., Keskin, C., Kim, D., Davidson, P., Kowdle, A., and Izadi, S. (2017). Articulated distance fields for ultra-fast tracking of hands interacting. *ACM Transactions on Graphics (TOG)*, 36(6):1–12.

Urbach, D., Ben-Shabat, Y., and Lindenbaum, M. (2020). Dpdist: Comparing point clouds using deep point cloud distance. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J., editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XI*, volume 12356 of *Lecture Notes in Computer Science*, pages 545–560. Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, L., Chen, J., Li, X., and Fang, Y. (2019a). Non-rigid point set registration networks. *arXiv preprint arXiv:1904.01428*.

Wang, X., Girshick, R., Gupta, A., and He, K. (2018). Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803.

Wang, Y. and Solomon, J. M. (2019a). Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3523–3532.

Wang, Y. and Solomon, J. M. (2019b). Prnet: Self-supervised learning for partial-to-partial registration. *arXiv preprint arXiv:1910.12240*.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019b). Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.

Yang, H., Shi, J., and Carlone, L. (2020a). Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333.

Yang, J., Cao, Z., and Zhang, Q. (2016). A fast and robust local descriptor for 3d point cloud registration. *Information Sciences*, 346-347:163 – 179.

Yang, J., Li, H., Campbell, D., and Jia, Y. (2015). Go-icp: A globally optimal solution to 3d icp point-set registration. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2241–2254.

Yang, J., Xiao, Y., and Cao, Z. (2018). Toward the repeatability and robustness of the local reference frame for 3d shape matching: An evaluation. *IEEE Transactions on Image Processing*, 27(8):3766–3781.

Yang, J., Zhang, Q., Xian, K., Xiao, Y., and Cao, Z. (2017). Rotational contour signatures for both real-valued and binary feature representations of 3d local shape. *Computer Vision and Image Understanding*, 160:133 – 147.

Yang, J., Zhao, C., Xian, K., Zhu, A., and Cao, Z. (2020b). Learning to fuse local geometric features for 3d rigid data matching. *Information Fusion*, 61:24–35.

Yang, Y., Chen, W., Wang, M., Zhong, D., and Du, S. (2020c). Color point cloud registration based on supervoxel correspondence. *IEEE Access*, 8:7362–7372.

Yang, Z., Pan, J. Z., Luo, L., Zhou, X., Grauman, K., and Huang, Q. (2019). Extreme relative pose estimation for rgb-d scans via scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4531–4540.

Yew, Z. J. and Lee, G. H. (2018). 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 607–623.

Yew, Z. J. and Lee, G. H. (2020). Rpm-net: Robust point matching using learned features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11824–11833.

Yuan, W., Eckart, B., Kim, K., Jampani, V., Fox, D., and Kautz, J. (2020). Deepgmr: Learning latent gaussian mixture models for registration. In *European Conference on Computer Vision*, pages 733–750. Springer.

Zabulis, X., Lourakis, M. I., and Koutlemanis, P. (2018). Correspondence-free pose estimation for 3d objects from noisy depth data. *The Visual Computer*, 34(2):193–211.

Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., and Funkhouser, T. (2017). 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1802–1811.

Zhang, Z. (1994a). Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152.

Zhang, Z. (1994b). Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152.

Zhou, J., Wang, M., Mao, W., Gong, M., and Liu, X. (2020). *Siamesepointnet: A siamese point network architecture for learning 3d shape descriptor*. Wiley Online Library.

Zhou, Q.-Y., Park, J., and Koltun, V. (2016). Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer.

Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*.

Zhu, M., Atanasov, N., Pappas, G. J., and Daniilidis, K. (2014). Active deformable part models inference. In *European Conference on Computer Vision*, pages 281–296. Springer.