UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUSTAVO MARQUES NETTO

Robust Point-Cloud Registration based on Dense Point Matching and Probabilistic Modeling

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. Manuel Menezes de Oliveira Neto

Porto Alegre
2022

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Marques Netto, Gustavo

Robust Point-Cloud Registration based on Dense Point Matching and Probabilistic Modeling / Gustavo Marques Netto. – Porto Alegre: PPGC da UFRGS, 2022.

99 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2022. Orientador: Manuel Menezes de Oliveira Neto.

1. Registro de nuvens de pontos. 2. Registro rígido. 3. Registro não-rígido. 4. Correspondência densa de pontos. I. Menezes de Oliveira Neto, Manuel. II. Título.

"If I have seen farther than others,
it is because I stood on the shoulders of giants."

— Sir Isaac Newton

RESUMO

Esta dissertação apresenta duas técnicas para o registro de nuvens de pontos 3D que são robustas a ruído e a nuvens de pontos parciais. Nossas técnicas abordam o registro rígido e o não rígido e exploram as vantagens do uso de aprendizado profundo para a estimativa de correspondência densa entre pontos. Para ambos os tipos de registro, nós propomos uma única rede neural. Nossa rede usa um mecanismo de atenção recentemente proposto e considera explicitamente a falta de correspondências entre os pontos, o que é crítico para a sua performance. Além disso, nós aplicamos avanços recentes em modelagem probabilística para refinar as correspondências criadas por nossa rede durante o registro não rígido. Tal combinação de aprendizado profundo e modelagem probabilística produz sensibilidade a contextos e também gera uma deformação coerente dos pontos, o que torna nossa abordagem resiliente a ruído e a perda de informação. Nós demonstramos a efetividade das nossas técnicas ao compará-las com métodos no estado da arte. Nossas comparações usam bases de dados com ruído e nuvens de pontos parciais ou com amostragem irregular. Os experimentos mostram que em geral, nós obtemos resultados superiores. Por exemplo, nossas abordagens alcançam um erro até 45% menor que outras técnicas no registro não rígido de nuvens de pontos parciais, ou até 49% menor no registro rígido. Nós também discutimos alguns aspectos extras da nossa técnica como a robustez a níveis diferentes de ruído e a números diferentes de amostras nas nuvens de pontos. Por último, nós abordamos a falta de bases de dados que forneçam o registro correto entre as nuvens de pontos. Essas bases são críticas no treinamento supervisionado de modelos de registro não rígido. Para resolver essa escassez, nós propomos uma estratégia de autoaprendizado baseada em deformações randômicas.

Palavras-chave: Registro de nuvens de pontos. registro rígido. registro não-rígido. correspondência densa de pontos.

ABSTRACT

This thesis presents techniques for 3D point-cloud registration that are robust to outliers and missing regions. They tackle non-rigid and rigid registration and exploit the advantages of deep learning for dense point matching. This is done by proposing a single new neural network to solve both registration types. Our network uses a recently proposed attention mechanism and explicitly accounts for missing correspondences, which is key to its performance. Additionally, we use recent advances in probabilistic modeling to further refine the correspondences created by our network during non-rigid registration. Such a combination of deep learning and probabilistic modeling produces context awareness and enforces motion coherence, which makes our approach resilient to outliers and missing information. We demonstrate the effectiveness of our techniques by comparing them to state-of-the-art methods. Our comparisons use datasets containing noise, partial point clouds, and irregular sampling. The experiments show that our techniques obtain superior results in general. For example, our approaches achieve a registration error up to 45% smaller than other techniques in partial point clouds for non-rigid registration, and up to 49% smaller on rigid registration. We also discuss additional aspects of our techniques such as robustness to different levels of noise, and different numbers of samples in the point clouds. Finally, we tackle the lack of datasets with ground truth for supervised training of non-rigid registration models by presenting a self-supervised strategy based on random deformations.

Keywords: Point-cloud registration, rigid registration, non-rigid registration ,dense point matching.

LIST OF ABBREVIATIONS AND ACRONYMS

SLAM	Simultaneous Localization and Mapping
ICP	Iterative Closest Point
SVD	Singular Value Decomposition
RANSAC	Random Sample Consensus
GMM	Gaussian Mixture Model
LIDAR	Light Detection and Ranging
EM	Expectation-Maximization
OT	Optimal Transport
EPE	End-to-end Point Error
MIE	Mean Isotropic Error
MAE	Mean Anisotropic Error
SGD	Stochastic Gradient Descent
P/D	Projection or Direct Generation
MLP	Multi-Layer Perceptron

LIST OF SYMBOLS

$Y, \hat{Y}, y_m$	Source Point Cloud, Aligned Source Point Cloud, Single Source Cloud Point
$X, \hat{X}, x_n$	Target Point Cloud, Permutated Target Cloud, Single Target Cloud Point
SO	Subgroup Orthogonal Matrices with Determinant $+1$
λ	Motion Coherence Intensity
$\mathbf{P}, \bar{\mathbf{P}}$	Probability/Soft-Assignment/Correspondence Matrix, Ground-Truth Hard-Assignment Matrix
$\mathbf{v}$	Non-Rigid Vectors
$\mathbf{S}_{\mathbf{final}}$	Estimated Hard-Assignment Matrix
ν, ν'	Number of Target Points Matched to each Source, Probability a Target Point is Outlier
β	RBF Kernel Free Parameter
τ	Convergence Tolerance
ω	Outlier Probability
γ	Initial scaling of $sigma_2$
κ	Dirichlet Shape
ψ	Digamma Function
$\sigma^2, \hat{\sigma}^2, \sigma_m^2$	GMM Covariance, Final Residual-Covariance, Point-Wise GMM Covariance
τ	Convergence Tolerance
O	Sinkhorn Iterations
L	Number of Attention Layers
$\mathbf{s}$	Scaling Factor
$\mathbf{w}_i$	Point-Wise Weight
$\mathbf{G}$	$M \times M$ Motion-Coherence Matrix

$\mathbf{R}, \mathbf{R}_{GT}, \mathbf{R}^{eul}$	Rotation Matrix, Ground-Truth Rotation Matrix, Rotation as Vector of Euler Angles
$\mathbf{t}, \mathbf{t}_{GT}$	Translation Vector, Ground-Truth Translation Vector
$\|.\|$	Determinant
T	Rigid + Non-Rigid Transformation
$\mathbf{I_D}$	Identity Matrix of Dimension D
$diag, tr$	Diagonal Matrix/Trace
$\mathsf{x}_n, \mathsf{y}_m$	Deep Features
h	Neural Network Linear Layer + Normalization + ReLU
N_m	Number of Neighbors
$\mathsf{f}_m^Y, \mathsf{f}_n^X$	Features Enriched by Attention Module
ψ_k	Point Cloud Rigidly Transformed in RMANet
δ_{x_n}	Point-Wise Attention Module Residual
$\mathsf{q}_n, \mathsf{k}_n, \mathsf{v}_n$	Point-Wise Attention Module Query, Key, Value
$\varnothing$	Element-Wise Division of Matrices
$\mathbf{1}_{N_c}, \mathbf{0}_{M_r}$	Column Vector N 1s, Row Vector of M 0s
$\mathcal{V}, \mathbf{\Lambda}$	Eigenvectors, Eigenvalues

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The recent popularization of 3D scanners and LIDARs has led to a growing interest on the manipulation and processing of 3D point clouds. A fundamental task when processing point clouds is registration, which aims at aligning different sets of samples. It is a basic block for applications such as SLAM (BRESSON et al., 2017), scene/object reconstruction (NASEER; KHAN; PORIKLI, 2018; BERGER et al., 2017), automated driving (WANG; WU; NIU, 2019), motion estimation (LIU et al., 2020), among others.

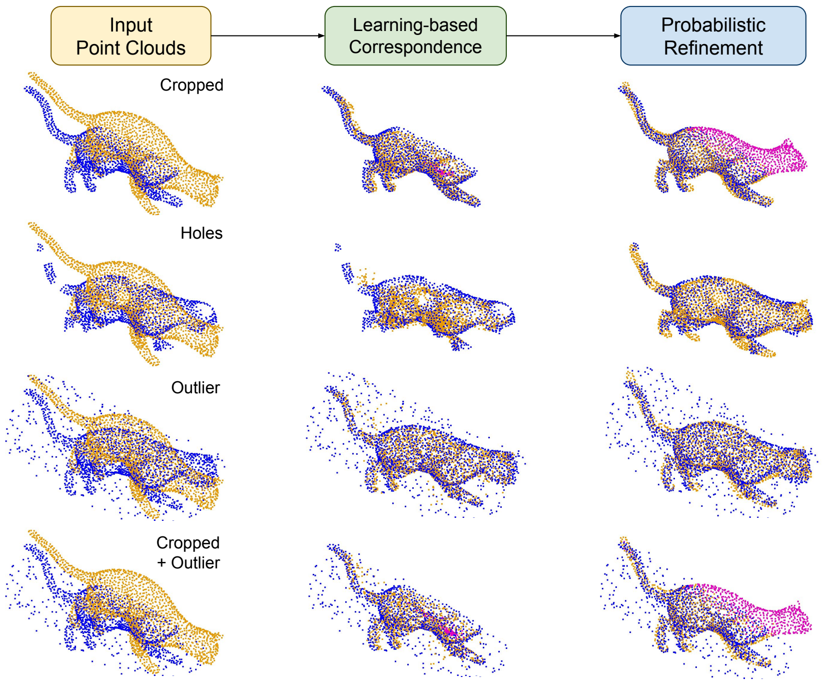
In *rigid registration*, two point clouds to be aligned typically only differ by a rotation and/or a translation, although scaling is also often treated under this class. ICP (BESL; MCKAY, 1992) and its variants are perhaps the most widely used solutions for this problem. In *non-rigid registration* portions from a source point cloud can be deformed independently from others, such as in articulated or organic figures. Furthermore, the registration can also be isometric, thus preserving the distances between the points in the cloud. Here, we assume the more general problem of non-isometric transformation between the clouds. In this case, one seeks for an as-close-as-possible alignment between the two point clouds. The registration problem becomes considerably harder in the presence of noise, and when portions of the point clouds to be aligned are missing. Approaching these challenges is key for having a robust registration, able to deal with real-world scenarios.

Recently, deep learning techniques targeted at rigid registration have obtained impressive results (FENG et al., 2021). However, little work has been done on non-rigid registration. State-of-art works (FENG et al., 2021; EISENBERGER et al., 2021) still lack resilience to critical cases, e.g., when dealing with noisy point clouds and/or when there are only partial matches between them. This is mostly due to the difficulty of establishing correspondences among the available parts. The same observation applies to state-of-the-art non-learning-based methods, such as BCPD (HIROSE, 2020b).

We present robust techniques for performing non-rigid as well as rigid registration of point clouds. Our techniques combine the advantages of deep learning models for obtaining dense point correspondences with recent advances in probabilistic modeling (Fig. D.1). They achieve state-of-the-art results for challenging scenarios involving noisy and missing regions, leading to better and faster registration. We also present a self-supervised training approach for non-rigid registration with on-par results to supervised training.

Figure D.1 illustrates the use of our technique, trained using self-supervision, for

Figure 1.1: Non-rigid registrations produced by our method for challenging scenarios. We present the results after each stage of our pipeline. The source clouds are in orange and the target ones are in blue. Points without correspondence in the target are shown in magenta.



Source: The Authors

non-rigid registration of point clouds under multiple difficult situations. It shows the results produced by the steps of our method for scenarios including: (i) missing a large portion of one of the point clouds (*Cropped* - first row); (ii) several holes across one point cloud (*Holes* - second row); (iii) noisy point cloud (*Outliers* - third row); (iv) a combination of missing large portions and noise (*Cropped* + *Outlier* - fourth row). Note how our method consistently achieves good results across all these situations, properly handling points without correspondences (shown in pink).

1.1 Thesis Statement

Non-rigid registration of a pair of 3D point clouds can be significantly improved by breaking the problem in two steps. The first one is a forward pass through a neural network that produces a correspondence matrix between points from the different clouds. The second is a probabilistic optimization, which from the initial correspondences produces the final registration. Additionally, applying the same neural network can yield state-of-art results on rigid registration of pairs of 3D point clouds. In this case, the network can improve robustness to noise and partial clouds when employed iteratively with cyclic consistency.

1.2 Thesis Demonstration

In order to demonstrate our statement, we treat the two registration types separately. In the non-rigid case, we use a modified SuperGlue neural network (SARLIN et al., 2020) to generate the correspondence matrix. This is followed by an adapted version of the BCPD algorithm (HIROSE, 2020b), which produces the final registration. When dealing with partial or noisy point clouds this strategy produces better alignments. Additionally, it allows classifying points from the source point clouds as having correspondences or not in the target. In the rigid case, we adopt the framework proposed by RGM (FU et al., 2021) which instead of using the correspondence matrix directly, applies a refinement with Hungarian algorithm over the correspondences. The final registration uses the SVD method to estimate the rigid-transformation matrix.

1.3 Contributions

The **contributions** of this dissertation include:

- Learning-based techniques for non-rigid (Section 3.1) and rigid (Section 5.1) registration of point clouds that are robust to large missing portions, as well as to noise. Both techniques outperform previous approaches in these challenging scenarios;
- An adaptation of the SuperGlue network to produce correspondences between pairs of 3D point clouds (Chapter 4);
- A self-supervised training strategy for robust non-rigid registration of point clouds (Section 6.1.3).

1.4 Structure of this Thesis

The remaining of this dissertation is split into six chapters and a set of appendices. Chapter 2 introduces the notion of rigid and non-rigid registration and reviews works related to our approach. For the ones we directly compare with, we present a detailed explanation. Chapter 3 presents how we adapted an existing technique to create our solution for non-rigid registration. In the same chapter, we describe how our solution interacts with the proposed neural network. In Chapter 4, we explain our network, and how it relates to and differs from other learning-based models. Chapter 5 shows how we solve rigid registration by having an existing framework to use our proposed neural network. Chapter 6 reports quantitative and qualitative registration results of the proposed technique compared to state-of-art approaches with noisy and partial point clouds. We further experiment with different configurations of our algorithms and types of point clouds. Finally, Chapter 7 discusses the existing limitations of our technique and future directions to explore. There are three appendices, the first offers additional qualitative results comparing the alignment of the proposed work to others. The last two investigate improvements over the non-rigid registration algorithm and its input parameters.

2 BACKGROUND ON POINT CLOUD REGISTRATION AND RELATED WORKS

This chapter has three sections. The first briefly introduces the problems tackled in this thesis. The remaining two discuss the techniques mostly related to our work. The related works are classified as either rigid or non-rigid, and as traditional or learning-based approaches. We also address in more details works that we directly compare against ours.

2.1 Rigid and Non-Rigid Registration

Rigid registration of point clouds aims at aligning a point cloud $Y = \{y_m\}$, $y_m \in \mathbb{R}^D$, $m \in 1,..,M$ by typically a rotation and a translation towards a target cloud $X = \{x_n\}$, $x_n \in \mathbb{R}^D, n \in 1,..,N$. y_m and x_n are the spatial coordinates of the points in the corresponding clouds, M and N are the number of such points, and D is the dimension of the space in which they are located. Non-rigid registration of point clouds aims at deforming a point cloud Y by a transformation T towards a target cloud X, where Y and X follow the presented definition. The deformation is pointwise and the resulting displacement can be given $\mathbf{v} = T(y) - y$. Regarding the point-cloud dimentionality, in this work we approach the 3D case so $D = 3$.

2.2 Related Works on Rigid Registration

Traditional Approaches: ICP (BESL; MCKAY, 1992) is one of the most popular approaches for rigid registration, and several improvements have been proposed to solve it, including optimizations for estimating correspondences (Tr-ICP (FITZGIBBON, 2003)), and modeling using maximum likelihood estimation (EM-ICP (GRANGER; PENNEC, 2002)). Probabilistic approaches like CPD (MYRONENKO; SONG, 2010), FilterReg (GAO; TEDRAKE, 2019), GMM-Tree (ECKART; KIM; KAUTZ, 2018), and Branch-and-Bound-based methods, such as Go-ICP(YANG et al., 2015), have shown improved results. In another direction, FGR (ZHOU; PARK; KOLTUN, 2016) optimizes the transformation given by the correspondences over a non-convex objective function. Feature matching (TOMBARI; SALTI; STEFANO, 2010) followed by RANSAC is largely used to better prune the correspondences. In general, these approaches rely on iterative optimization, which becomes expensive when dealing with challenging scenarios, such as registration under partial correspondences. For an in-depth discussion of these techniques, see (POMERLEAU; COLAS; SIEGWART, 2015).

Learning-Based Models: Recently, neural networks have been used to address the limitations of traditional approaches. Interestingly, new works tend to build on concepts from traditional ones to achieve state-of-art results. DCP (WANG; SOLOMON, 2019) takes the correspondence and transformation estimation from ICP with dense matching of deep-learned features followed by a differential SVD layer. RPM-Net (YEW; LEE, 2020) builds on it by having an iterative process but with a differential Sinkhorn layer solving an optimal transport (OT) problem, which explicitly accounts for partial clouds. Deep-GMR (YUAN et al., 2020) presents a neural network to solve an expectation-maximization (EM) problem on a small number of Gaussian Mixture Models (GMM) which are estimated by the network. Works using deep feature descriptors typically followed by RANSAC include Predator (HUANG et al., 2021), which uses an attention mechanism on features from KPConv convolutions (THOMAS et al., 2019). Other works on this direction include D3Feat (BAI et al., 2020) and FCGF (CHOY; PARK; KOLTUN, 2019). Finally, RGM (FU et al., 2021) employs a graph matching network to estimate the correspondences and binary-assignment loss function, instead of directly outputting a rigid transformation.

2.2.1 Rigid Registration based on Correspondence and SVD

In this section we detail the SVD formulation to solve the rigid alignment originally proposed by Arun *et al.* (ARUN; HUANG; BLOSTEIN, 1987). This is not only used by our rigid registration technique, but also used by ICP implementations and adapted by BCPD, RGM, and others.

Given a set of corresponding points $\mathcal{Y} = \{y_j\} \in Y$ and $\mathcal{X} = \{x_j\} \in X$, with J being the total number of correspondences, the objective is to optimally align them in a least square sense

$$(\mathbf{R}, \mathbf{t}) = \underset{\mathbf{R}\in SO(D), \mathbf{t}\in\mathbb{R}^D}{\operatorname{argmin}} \sum_{j=1}^{J} \mathbf{w_j} \|(\mathbf{R}y_j + \mathbf{t}) - x_j\|^2, \tag{2.1}$$

where $\mathbf{w}_j \geq 0$ are weights for each pair. $SO(D)$ is the subgroup of orthogonal matrices with determinant $+1$ which also represents all rotation matrices.

2.2.1.1 Translation

Assuming $\mathbf{R}$ is fixed and having $F(t) = \sum_{j=1}^{J} \mathbf{w}_j \|(\mathbf{R}y_j + \mathbf{t}) - x_j\|^2$, we can find the optimal translation by taking the derivative of F w.r.t. $\mathbf{t}$ and finding its roots,

$$0 = \frac{\partial F}{\partial \mathbf{t}} = 2\sum_{j=1}^{J} \mathbf{w}_j(\mathbf{R}y_j + \mathbf{t} - x_j) = 2\mathbf{t}\sum_{j=1}^{J} w_j + 2\mathbf{R}\sum_{j=1}^{J} \mathbf{w}_j y_j - 2\sum_{j=1}^{J} \mathbf{w}_j x_j. \quad (2.2)$$

Considering $\bar{y} = \frac{\sum_{j=1}^{J} \mathbf{w}_j y_j}{\sum_{j=1}^{J} \mathbf{w}_j}$ and $\bar{x} = \frac{\sum_{j=1}^{J} \mathbf{w}_j x_j}{\sum_{j=1}^{J} \mathbf{w}_j}$, the optimal translation is $\mathbf{t} = \bar{x} - R\bar{y}$. By plugging this result back into Eq. (2.1) we obtain a formulation independent of translation: $(\mathbf{R}, \mathbf{t}) = \underset{\mathbf{R} \in SO(D), \mathbf{t} \in \mathbb{R}}{\operatorname{argmin}} \sum_{j=1}^{J} w_j \|\mathbf{R}(y_j - \bar{y}) - (x_j - \bar{x})\|^2$. This allows us to break the problem into a translation followed by a rotation. In the remaining of this section we will refer to y_j, x_j as if they have already been translated by $\mathbf{t}$.

2.2.1.2 Rotation

The next step is to minimize $\|\mathbf{R}y_j - x_j\|^2$, and this expression can be rewritten as $y_j^T y_j - 2y_j^T \mathbf{R} x_j + x_j^T x_j$ since the rotations in the first term cancel each other. Substituting it back to Eq. (2.1)

$$\mathbf{R} = \underset{\mathbf{R} \in SO(D)}{\operatorname{argmin}} \sum_{j=1}^{J} \mathbf{w}_j (y_j^T y_j - 2y_j^T \mathbf{R} x_j + x_j^T x_j) = \underset{\mathbf{R} \in SO(D)}{\operatorname{argmin}} \sum_{j=1}^{J} -2(\mathbf{w}_j y_j^T \mathbf{R} x_j) \quad (2.3)$$

$$= \underset{\mathbf{R} \in SO(D)}{\operatorname{argmax}} \sum_{j=1}^{J} \mathbf{w}_j y_j^T \mathbf{R} x_j, \quad (2.4)$$

where the first equality is valid because the first and last terms do not depend on $\mathbf{R}$, so minimizing the expression will lead them to zero. Next we can have the resulting sum written in a matrix form as $tr(\mathbf{W}\mathcal{Y}^T \mathbf{R} \mathcal{X})$, where $\mathcal{Y}, \mathcal{X} \in \mathbb{R}^{D \times J}$, $\mathbf{R} \in \mathbb{R}^{D \times D}$, and $\mathbf{W} = diag(\mathbf{w}_j)$ is a $\mathbb{R}^{j \times j}$ matrix whose diagonal are the weights. Finally, tr is the trace of a matrix.

The objective becomes finding a rotation maximizing $tr(\mathbf{W}\mathcal{Y}^T \mathbf{R} \mathcal{X})$. Using the trace property $tr(\mathbf{AB}) = tr(\mathbf{BA})$ we have $tr(\mathbf{R}\mathcal{X}\mathbf{W}\mathcal{Y}^T)$, which by having $\mathbf{S} = \mathcal{X}\mathbf{W}\mathcal{Y}^T$ and decomposing it by SVD we get the cost expression as

$$\mathbf{R} = \underset{\mathbf{R} \in SO(D)}{\operatorname{argmax}}\, tr(\mathbf{R}\mathbf{\Phi}\mathbf{S}'\mathbf{\Psi}^T) = \underset{\mathbf{R} \in SO(D)}{\operatorname{argmin}}\, tr(\mathbf{S}'\mathbf{\Psi}^T \mathbf{R}\mathbf{\Phi}). \quad (2.5)$$

Note that since $\mathbf{R}$ is by definition an orthogonal matrix as are $\mathbf{\Phi}$, $\mathbf{\Psi}^T$, by SVD properties, we have $\mathbf{M} = \mathbf{\Psi}^T\mathbf{R}\mathbf{\Phi}$ being an orthogonal matrix as well. This means that the columns μ_d in $\mathbf{M}$ are orthonormal vectors with $\mu_d^T\mu_d = 1$ and all entries $\mu_{d,d}$ of $\mathbf{M}$ are ≤ 1. By also considering $\mathbf{S}'$ a diagonal matrix of non-negative numbers whose elements are $\{\sigma_d\} \geq 0$, we have that

$$tr(\mathbf{S}'\mathbf{M}) = \sum_{d=1}^{D} \sigma_d\mu_{d,d} \leq \sum_{d=1}^{D} \sigma_d. \tag{2.6}$$

This means that the expression is maximized if $\mu_{dd} = 1$. As $\mathbf{M}$ is also an orthogonal matrix, in order to maximize the resulting value $\mathbf{M}$ has to be the identity matrix, $\mathbf{I_D}$. The resulting optimal rotation matrix can then be computed by

$$\mathbf{I_D} = \mathbf{M} = \mathbf{\Psi}^T\mathbf{R}\mathbf{\Phi} \Rightarrow \mathbf{\Psi} = \mathbf{R}\mathbf{\Phi} \Rightarrow \mathbf{R} = \mathbf{\Psi}\mathbf{\Phi}^T. \tag{2.7}$$

2.2.1.3 Orientation Rectification

The process described previously finds an optimal orthogonal matrix that could contain reflections in addition to rotations. A reflection matrix has the same properties as a rotation, however, its determinant is -1. Therefore, if $det(\mathbf{\Psi}\mathbf{\Phi}^T) = +1$ the result is a rotation, otherwise it contains reflections. Assuming $det(\mathbf{\Psi}\mathbf{\Phi}^T) = -1$, this means we could not find a rotation maximizing $tr(\mathbf{S}'\mathbf{M})$. In this case, it is necessary to build an approximation of $\mathbf{M}$ such that $\mathbf{R}$ is a rotation. This can be obtained by flipping the signal of the smallest singular value of $\mathbf{S}'$. For more details, we direct the reader to the original work (ARUN; HUANG; BLOSTEIN, 1987) and the notes by Sorkine and Rabinovich (SORKINE-HORNUNG; RABINOVICH, 2017). The resulting computation can be written in a single expression

$$\mathbf{R} = \mathbf{\Psi} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \cdots & \\ & & & |\mathbf{\Psi}\mathbf{\Phi}^T| \end{bmatrix} \mathbf{\Phi}^T \Rightarrow \mathbf{R} = \mathbf{\Psi} diag(1, ..., 1, |\mathbf{\Psi}\mathbf{\Phi}^T|)\mathbf{\Phi}^T. \tag{2.8}$$

2.2.2 DCP

DCP (WANG; SOLOMON, 2019) stands for *Deep Closest Point* and it tackles the rigid-registration of 3D point clouds. It revisits parts of the ICP algorithm through the perspective of deep learning by having a model that can be split into three parts, each responsible for:

- Map the input clouds (source and target) to permutation/rigid invariant features;
- Enrich the features with an attention based module;
- Predict the rigid transformation with a differentiable SVD layer.

ICP approaches the registration problem by alternating between estimating correspondences among points from source and target clouds ($Y = \{y_m\}$, $y_m \in \mathbb{R}^D, m \in 1, .., M$, $X = \{x_n\}$, $x_n \in \mathbb{R}^D, n \in 1, .., N$), and applying a rigid transformation (that is typically estimated with SVD, Section 2.2.1). DCP employs a learned model to recover the final registration in a single forward pass to the network, such that no iterative refinement is necessary.

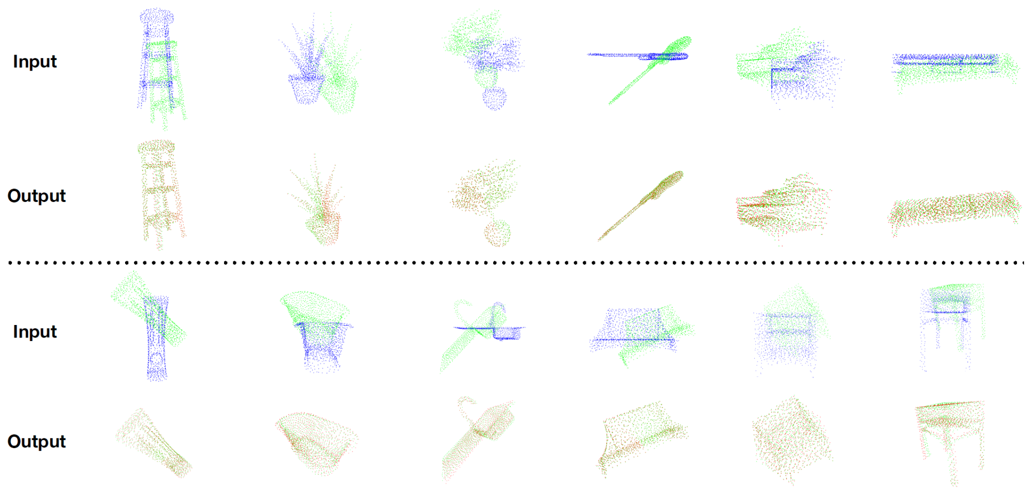
DCP selects between two possible approaches to select the network to estimate the deep features, *Dynamic Graph CNN* (DGCNN) (WANG et al., 2019) and PointNet (QI et al., 2017), where first consistently reports the best results. DGCNN builds a graph given the KNN of each point in the cloud, applies non-linearity to the values of each edge, and vertex-wise aggregate the values (more details at Section 4.1). These features only account for local information given a nearest neighborhood. Furthermore, the features from source and target point clouds are independent of each other since they are applied separately to the network. To remedy these two points, DCP applies the attention module proposed by Point Transformer (VASWANI et al., 2017). This attention module comes from the natural language processing (NLP) domain where it was designed for word matching. It enriches a feature of a given point or word by taking similar features from not only the current point cloud, for example the source, but from the target one as well.

Differently from ICP, the correspondences between points are substituted for matches across the resulting attention features, named $\mathsf{f}_m^Y, \mathsf{f}_n^X$ for source and target clouds respectively. In addition, DCP relies on soft-assignment correspondences given by a probability matrix. This means the assignment/correspondence certainty works as a probability, thus the sum of matching probabilities cannot be greater than 1. This matrix is estimated by

$$\tilde{\mathbf{P}}_{m,n} = \text{softmax}(\mathsf{f}_m^Y(\mathsf{f}_n^X)^T), \tag{2.9}$$

where the softmax guarantees the soft-assignment requirements. Finally, by using this matrix to project the target points (called back-projection), we create a permutated target

Figure 2.1: Examples of registration using DCP on pairs of clouds from the ModelNet40 dataset (WU et al., 2015). The *input* row has source clouds in blue and target ones in green. *Output* rows have the registered clouds in orange.



Source: Wang and Solomon (2019)

that matches the source according to the learned-feature similarities

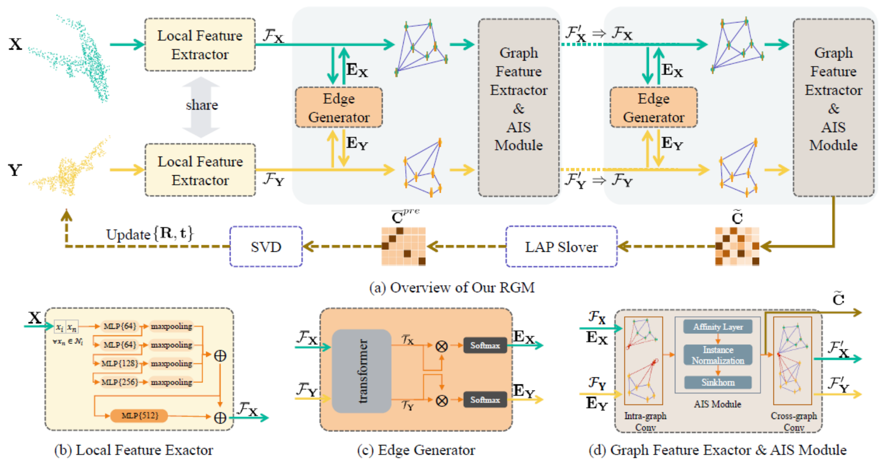
$$\hat{X} = X^T \tilde{\mathbf{P}}_{m,n}. \tag{2.10}$$

This results in a one-to-one set of correspondences which is fed into a fully-differentiable SVD module. Finally, DCP employs supervised learning which directly compares the final estimated rotation and translation to the ground truths,

$$Loss = \|\mathbf{R}_{GT}^T \mathbf{R} - 1\| + \|\mathbf{t}_{GT} - \mathbf{t}\|. \tag{2.11}$$

Figure 2.1 show results using DCP on the ModelNet40 dataset, which consists of point clouds from CAD models of objects. The results have examples on mostly complete point clouds, but we highlight that DCP report results with partial point clouds. This indicates that the network can in some degree adapt to this challenge. On the other hand, it lacks any component directly dealing with missing correspondences between points (neither the loss function nor the network has parts considering this). Consequently, techniques such as RGM and Predator have better performance than DCP because they address registration of partial clouds.

Figure 2.2: Diagram of the neural network and framework for rigid registration of 3D point clouds proposed by RGM.



Source: Fu *et al.* (2021)

2.2.3 RGM

RGM (FU et al., 2021) stands for *Robust Point Cloud Registration Framework Based on Deep Graph Matching* and it rigidly registers 3D point clouds. RGM does not propose an end-to-end neural network to produce the registration matrix as DCP does. Instead, it concentrates on generating the correspondence matrix, $\tilde{\mathrm{C}}$, that is fed into SVD to estimate the rigid transformation. Furthermore, it uses cycle consistency and multiple iterations to better refine the assignment and the registration matrix. The entire process is presented in Figure 2.2.

The main idea of RGM is to solve the matching of points adapting the architecture originally proposed for deep graph matching named LCE (WANG; YAN; YANG, 2019). To estimate the vertices of each graph, RGM uses a DGCNN (WANG et al., 2019) network that for each point produces a feature invariant to translation and rotation as it is done by DCP. For the edges, E_Y and E_X, the features produced by the Point Transformer network (VASWANI et al., 2017), f_m^Y and f_m^X, are independently used as

$$\mathsf{E}_Y = \mathrm{softmax}(\mathsf{f}_m^Y(\mathsf{f}_m^Y)^T). \tag{2.12}$$

The next steps consist of two independent but similar calls to what in Fig. 2.2 is named *Graph Feature Extractor & AIS Module*. The first call computes a self-correlation feature, $\mathsf{F}_{x_i}^{self}$ and $\mathsf{F}_{y_i}^{self}$, meaning deep features from the source cloud do not consider target-cloud information. These features are used to produce an affinity matrix, $\mathbf{A}_{ij}$, which is the first version of the correspondence matrix

$$\mathbf{A}_{ij} = \mathsf{F}_{x_i}^{self} \mathbf{W} \mathsf{F}_{y_i}^{self}. \tag{2.13}$$

Finally, the affinity matrix is made into a valid probability matrix by using the Sinkhorn operator (SINKHORN; KNOPP, 1967; MENA et al., 2018). The second call as displayed in Fig. 2.2 applies the new found permutation matrix to back-project the target cloud features F_X. Which in turn produces a cross-correlation feature that is used to produce the final soft correspondence matrix $\tilde{\mathbf{C}}$.

The network is trained by only comparing the correspondence matrix to a ground truth, instead of the registration results. This can be seen as a classification problem as well, given that a source point can have a label/target point associated. As a result, RGM uses the cross-entropy loss function that is common in classification tasks,

$$Loss = -\sum_{i}^{N}\sum_{j}^{M}(\mathbf{C}_{i,j}^{GT}\log\tilde{\mathbf{C}}_{i,j} + (1-\mathbf{C}_{i,j}^{GT})\log(1-\tilde{\mathbf{C}}_{i,j})). \tag{2.14}$$

RGM remedies issues from DCP, as its loss function and the Sinkhorn operator directly target partial correspondences between points. However, the two passes on the *Graph Feature Extractor Module* have independent features and result in a network twice the size of our proposed model and DCP. This results in a runtime time two times higher but with worse performance when compared to our proposed network (see Section 6.1.7).

2.2.4 Predator

Predator (HUANG et al., 2021) focuses on solving the rigid-registration problem on point clouds with low overlap. It tackles this problem by proposing a neural network that produces four outputs. Two of them are sets of point-wise features and are used in RANSAC to compute the rigid registration matrix. The third is a point-wise overlapping score that indicates the probability of a point to belong to an overlapping region. The last one is a point-wise *matchability* score telling the probability of a point to be correctly matched given the target cloud. The network can be broken into three main modules:

- *Encoding module*, that generates the sets of superpoints and corresponding deep features from the input point clouds;
- *Overlapping attention module*, that produces features dependent of both point clouds,

in addition to scores indicating how much each point and its correspondence belong to the overlapping region.

- *Decoding module*, that from the features and scores produced before generates pointwise descriptiors and overlap/*matchability* scores.

The first module uses a series of residual blocks with KPConv convolutions (THOMAS et al., 2019). This use the original points to create superpoints and the associated features. The resulting features are taken to the attention module which first enriches the features by a self-attention layer using the aggregation proposed by DGCNN. This layer can be summarized by:

$$\mathsf{x}_n^{k+1} = \max_{j:(n,j)} h_\theta^{k+1}(\{\mathsf{x}_n^k, \mathsf{x}_j^k - \mathsf{x}_n^k\}) \; \forall j \in \mathsf{N}_n, \mathsf{x}_n^{self} = h_\theta^{self}(\{\mathsf{x}_n^0, \mathsf{x}_n^1, \mathsf{x}_n^2\}), \tag{2.15}$$

where h_θ^{k+1} is a linear layer, and in the case of $\mathsf{x}_n^0, \mathsf{x}_n^1, \mathsf{x}_n^2$, the linear tranformation is followed by an instance normalization, and LeakyReLU. N_n are the k-nearest neighbor of each point x_n whose deep features are x_n^k. The next step is to account for the other point-cloud features, this constitutes the cross-attention module. Predator adapts the Transformer network (VASWANI et al., 2017), which results in the features $\mathsf{f}_m^Y, \mathsf{f}_n^X$. For details about this module, we direct the user to the original paper and for Section 4.2 where a similar approach is taken.

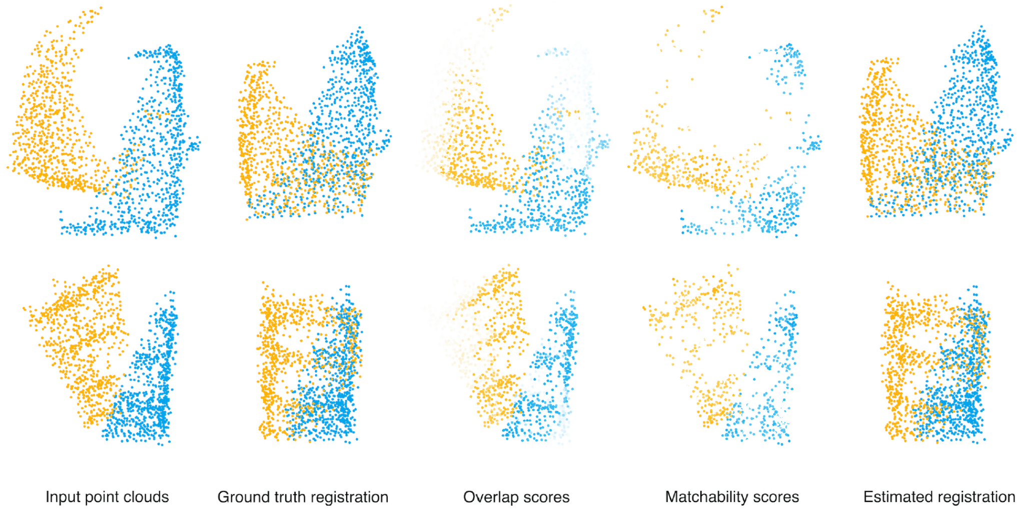
Predator is trained end-to-end with a loss function composed of three terms, one per network output type,

$$Loss = \lambda_c Loss_c + \lambda_o Loss_o + \lambda_m Loss_m. \tag{2.16}$$

$Loss_c$ is the circle loss function (SUN et al., 2020) and it takes the clouds aligned by the ground truth. Given each point position, it enforces points that are close in the Cartesian coordinates to similarly near in the deep-feature space. $Loss_o$ uses a cross-entropy loss function since this score classifies whether each source and target point is in the overlapping region. $Loss_m$ also uses a cross-entropy loss function and as the *matchability* score it tells if a source point has a correspondence in the aligned target cloud. Finally, $\lambda_c, \lambda_o, \lambda_m$ are the weights of each loss term.

Figure 2.3 presents registrations using the Predator network on the ModelNet40 dataset. We notice the alignment in the last column is precise even though the estimation of the scores has problems. This is a drawback of the multiple-term loss function instead of the single one used by RGM. In Predator, a well-defined and meaningful set of features

Figure 2.3: Examples of registration using Predator on pairs of clouds from the ModelNet40 dataset (WU et al., 2015). All columns have source clouds in blue and target ones in orange. The overlap and *matchability* scores are demonstrated as shades of each color. This means that points with a higher score have more saturated colors.



Source: Huang *et al.* (2021)

estimated by the network will not necessarily translate into good overlap scores, for example. Furthermore, tunning the different weights in the loss function or the parameters of the circle loss term, $Loss_c$, includes extra complexity to the training process compared to RGM and our loss function (Section 4.4).

2.3 Related Works on Non-Rigid Registration

Traditional Approaches: Non-rigid ICP (AMBERG; ROMDHANI; VETTER, 2007) expands the original algorithm to this domain, and recent extensions include (YAO et al., 2020). CPD (MYRONENKO; SONG, 2010) models the problem as a GMM where the target points are generated by Gaussian distributions induced by source points. The alignment is estimated by solving a maximum likelihood problem through EM. GLTP (GE; FAN; DING, 2014; GE; FAN, 2015) expands on CPD by including an extra term to deal with highly-articulated models. Cao et al. (CAO et al., 2014) also build on CPD by classifying the aligned points as reliable or not. To register the remaining points, it optimizes over a cost function regularized to ensure coherent motion of adjacent points. GMM-Reg (JIAN; VEMURI, 2010) presents a framework for rigid and non-rigid registration to support large amounts of noise. It represents both clouds as GMM's and minimizes the statistical discrepancy between them. Ma et al. presented a series of papers on the

topic. Ma et al. (MA; ZHAO; YUILLE, 2015) use a GMM similar to CPD, but on top of the global cloud alignment, they employ descriptor matching as initial correspondence probabilities. MR-RPM (MA et al., 2017; MA et al., 2018) builds on the set of correspondences by learning a transformation under manifold regularization using EM. Recently, BCPD (HIROSE, 2020b; HIROSE, 2020a) extended CPD using variational Bayesian inference and Nyström acceleration, instead of EM. It presents a superior performance and runtime compared to previous methods, while simplifying the CPD parameter selection.
Learning-Based Models: Techniques focusing on non-rigid 3D deformations have targeted mostly scene flow, and they typically consider point clouds from LIDARs. In this domain, clouds can be projected into 2.5D images (2D images with a depth channel), and deformations are usually semi-rigid objects moving in a scene, *e.g.*, cars and bicycles. This deformation type is considerably simpler than the more general ones handled by non-rigid registration methods, which focus on single objects.

The seminal work of FlowNet3D (LIU; QI; GUIBAS, 2019) on scene flow uses a cost volume to learn the flow in an end-to-end fashion. This was followed by extensions covering non-supervised learning (MITTAL; OKORN; HELD, 2020), Tishchenko *et al.* (TISHCHENKO et al., 2020), pyramid refinement with an improved cost volume (PointPWC-Net) (WU et al., 2020), and recurrent networks (FlowStep3D) (KITTENPLON; ELDAR; RAVIV, 2021). FLOT (PUY; BOULCH; MARLET, 2020) estimates the scene flow modeling it as an OT problem using deep features. Ouyang and Raviv (OUYANG; RAVIV, 2021) expanded PointPWC to explicitly account for occlusions in the clouds by considering them in the cost volume and in the loss function. Although these works present impressive results on LIDAR point clouds, the cost-volume layer has issues learning large deformations as it uses nearest-neighbors similarities.

Focusing on non-rigid registration, CPD-Net (WANG et al., 2019b) predicts the complete deformation using PointNet (QI et al., 2017) features and an unsupervised Chamfer distance loss function. PR-Net (WANG et al., 2019a) expands on TPS (BOOKSTEIN, 1989) by learning the shape correlation based on features of the voxelized point clouds. Recently, RMANet (FENG et al., 2021) presented an unsupervised learning approach based on a recurrent model which learns weights and rigid transformations at each iteration. It offers superior results to CPD-Net and PR-Net, and can be directly applied to rigid registration. These works do not tackle directly partial point clouds, or propose ways how to do it.

A related problem is shape correspondence (SAHILLIOĞLU, 2020), whose goal is finding correspondences between pairs of mesh vertices while keeping the existing edges. State-of-the-art learning-based works include CorrNet3D and NeuroMorph which are based on point-clouds correspondence and can be extended to the registration case. CorrNet3D (ZENG et al., 2021) presents an unsupervised approach based on estimating dense correspondences obtained using a correlation matrix estimated from DGCNN (WANG et al., 2019) features. The unsupervised loss function uses mirrored transformations by having the deformed source as similar as possible to the target cloud and vice-versa. NeuroMorph (EISENBERGER et al., 2021) also tackles mesh interpolation. It presents a similar unsupervised approach but builds on DGCNN by including a global max-pooling when estimating the features. Its unsupervised learning function considers registration, geodesic distance, and as-rigid-as-possible deformation. Similar to previously mentioned works, both approaches do not account for partial correspondences, when training or applying their models.

2.3.1 Bayesian Coherent Point Drift

To solve the non-rigid registration problem BCPD (HIROSE, 2020b) models it in a probabilistic fashion, so Y is assumed to define a probability distribution that generates X. This leads to solving the registration problem being equivalent to finding the distribution parameters that maximize the probability of the target points being sampled from said distribution. Under this modeling, four assumptions are made:

1 A target point x_n is either a point of the deformed Y or an outlier generated with a probability of ω;

2 If x_n is an outlier, it was generated by a distribution given by $p_{out}(x_n)$;

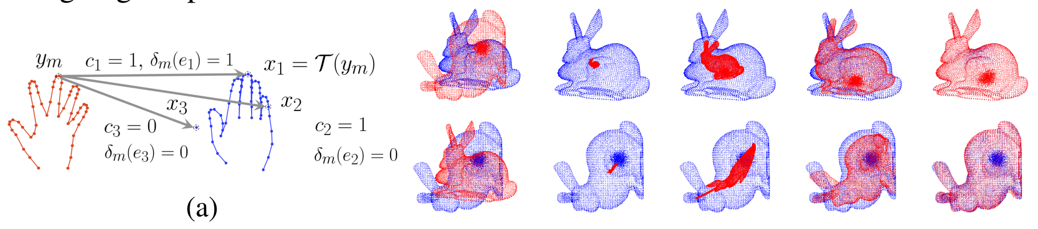
3 If x_n belongs to the deformed Y, an index $m \in 1, ..., M$ indicating that x_n corresponds to y_m is sampled with probability α_m;

4 In addition, x_n is generated from a multivariate normal distribution whose mean vector is $T(y_m)$ and the covariance matrix is $\sigma^2 \mathbf{I}_D$;

5 X is generated by repeating the previous steps for each x_n.

The listed assumptions imply multiple probability distributions, forming a full joint one. These are controlled by variables whose values are expected to be inferred given the observed x_n and are deemed latent variables, represented by θ. The full joint distribution takes the form $p(X, Y, \theta)$. This means θ controls the probability distribution that models the generation of each x_n by all y. Below we present how BCPD defines the joint

Figure 2.4: (a) Source point cloud in red and target in blue. The variable $\mathbf{c}_n \in \{0, 1\}$ indicates whether x_n is an outlier. Variable $\mathbf{e}_n \in \{1, ..., M\}$ speficies the index of a point y_m that correspond to x_n. Target points x_1, x_2, and x_3 represent the point that correspond to y_m, a non-outlier that does not correspond to any point, and an outlier. (b) Example combining rigid and non-rigid registration. Each rows displays the two possible directions when aligning the point-clouds.



(a)

(b)

Source: Hirose *et al.* (2020)

probability equation using conditional probabilities over θ.

BCPD models the transformation not only by a non-rigid displacement vector, but a rigid transformation too. This results in a deformed source given by $\hat{Y} = T(y_m + \mathbf{v}_m) = \mathbf{sR}(y_m + \mathbf{v}_m) + \mathbf{t}$, where $\mathbf{s}$, $\mathbf{R}$, $\mathbf{t}$ are respectively a scale factor, rotation matrix and a translation vector. Following item 4, the target is generated by a Gaussian Mixture Model (GMM), so if x_n corresponds to y_m the probability distribution of x_n is

$$\phi(x_n; T(y_m + \mathbf{v}_m), \sigma^2\mathbf{I}_D) = |2\pi\sigma^2\mathbf{I}_D|^{-\frac{1}{2}} exp\{-\frac{1}{2\sigma^2}\|x_n - T(y_m + \mathbf{v}_m)\|\}. \tag{2.17}$$

The final GMM expands on the normal distribution (item 4) by considering how x_n can be an outlier (item 1 and 2), and how it can correspond to a source point y_m (item 3):

$$p(x_n, \mathbf{e}_n, \mathbf{c}_n | y, \mathbf{v}, \alpha, \mathbf{s}, \mathbf{R}, \mathbf{t}, \sigma^2) = \{\omega p_{out}(x_n)\}^{1-\mathbf{c}_n}\{(1-\omega)\Pi_{m=1}^{M}(\alpha_m\phi_{mn})^{\delta_m(\mathbf{e}_n)}\}^{\mathbf{c}_n}. \tag{2.18}$$

We highlight two points. First, $\mathbf{c}_n$ indicates a Bernoulli distribution, so if $\mathbf{c}_n = 1$, x_n is not an outlier; and if $\mathbf{c}_n = 0$, it is. Second, $\mathbf{e}_n$ implies a categorical distribution, which generalizes Bernoulli to multiple categories. This is represented by $\delta_m(\mathbf{e}_n)$, an indicator function which is only equal to 1 when $\mathbf{e}_n = m$, meaning the match is correct between y_m and x_n. Figure 2.4a illustrates the role of these variables.

The remaining latent variables are ($\mathbf{v}_m$, α_m, $\mathbf{s}$, $\mathbf{R}$, $\mathbf{t}$, σ^2) and from them only $\mathbf{v}_m$ and α_m are assumed to have prior distributions. This means that further constraints modeling each respective parameter are only assumed to the non-rigid displacement field and to the probability of correspondence between source to target points. v_m is modeled by a Gaussian distribution which enforces motion coherence, meaning near points should move similarly. The prior distribution of $\mathbf{v}_m$ is given by:

$$p(\mathbf{v}|y) = \phi(\mathbf{v}; 0, \lambda^{-1}\mathbf{G}), \tag{2.19}$$

where the variance $\lambda\mathbf{G}$ is composed of $\mathbf{G}$, an $M{\times}M$ positive-definite matrix generated by an RBF kernel, and β, a free parameter. $p(\alpha)$ is given by a Dirichlet distribution $p(\alpha) = Dir(\alpha|\kappa\mathbf{1}_M)$, where κ>0 and is an input parameter. The choice of such distribution comes from BCPD using Bayesian inference to solve the estimation of θ. Given α is the probability of a modeled categorical distribution, choosing Dirichlet simplifies the updating equations of the adopted framework and it is the default selection in such cases (BISHOP, 2006). The final joint distribution takes the following form

$$p(X, Y, \theta) \propto p(v|y)p(\alpha)\Pi_{n=1}^{N} p(x_n, \mathbf{e}_n, \mathbf{c}_n|y, \mathbf{v}, \alpha, \mathbf{s}, \mathbf{R}, \mathbf{t}, \sigma^2). \tag{2.20}$$

Given θ = ($\mathbf{v}$, α, $\mathbf{c}$, $\mathbf{e}$, $\mathbf{s}$, $\mathbf{R}$, $\mathbf{t}$, σ^2) and the observed X and Y, the objective is to find θ maximizing $p(\theta|X, Y)$. The problem can also be interpreted as estimating the expectation of θ over $p(\theta|X, Y)$. An analytic form does not exist to solve this problem because of the joint nature of Eq. (2.20). Furthermore, evaluating the expectation over all possible combinations of θ is costly and cannot be used. To solve this issue, BCPD applies Variational Bayesian Inference (VBI) which simplifies the problem by introducing a new distribution $q(\theta)$. This distribution has the property of having its expectation easily computed. The problem than becomes finding $q(\theta)$ that more closely approximates $p(\theta|X, Y)$ instead of θ. This process is done iteratively by alternately updating factorized formulations of q, *i.e.*, $q_i(\theta_i)$, where i is a component of θ. During the updates, the remaining q_j are kept fixed. For more details on how VBI is used to obtain the BCPD algorithm, we direct the reader to the original paper, its supplemental material, and Chapter 10 of Bishop (BISHOP, 2006).

The resulting Algorithm 1 consists of a loop that at each iteration estimates the matching probability matrix, $\mathbf{P} = \{\mathbf{p}_{mn}\}$, $\mathbf{p}_{mn} \in \mathbb{R}^{M\times N}$, between all deformed source and target points. This matrix can also be treated as a correspondence matrix because it informs the probability of each possible correspondence between the point clouds. In the next step, P informs the updates to the values of variance and rigid/non-rigid transformations. In Algorithm 1, $\mathbf{1}_{Nc}$ is a column vector consisting of N 1s, and $\mathbf{1}_{Mr}$ is a row vector consisting of M 1s. $Diag$ is the diagonal of a given matrix. σ_m^2 is the mth diagonal element of Σ (defined in the algorithm). ψ is the digamma function (the logarithmic derivative of the gamma function) and V is the volume of the bounding box of all points in X. An example of registration produced by the BCPD algorithm is shown at Fig. 2.4b.

```
Inputs  : $X = \{x_n\}, Y = \{y_m\}$, $\lambda$ (motion coherence intensity parameter),
          $\beta$ (RBF kernel free parameter), $\gamma$ (initial scaling of $\sigma^2$),
          $\tau$ (convergence tolerance), $\kappa$ (Dirichlet shape parameter)
Outputs: $\hat{Y} = \mathbf{s}\mathbf{R}(y + \mathbf{v}) + \mathbf{t}$
$\alpha_m = \frac{1}{M}$, $\mathbf{G} = \exp \frac{1}{2\beta^2}\|(y - y')\|^2$, $p_{out}(x_n) = 1/V$, $\hat{Y} = Y$
$\sigma^2 = \frac{\gamma}{MND} \sum_{n=1}^{N} \sum_{m=1}^{M} \|x_n - y_m\|^2$, $\hat{\sigma}^2 = 0$
while $\|\sigma^2 - \hat{\sigma}^2\| > \tau$ do
    $\phi_{mn} = \phi(x_n; \hat{y}_m, \sigma^2 I_D) exp\{-\frac{s^2}{2\sigma^2} Tr(\sigma_m^2 \mathbf{I}_D)\}$
    $\mathbf{p}_{mn} = \frac{(1-\omega)\alpha_m \phi_{mn}}{\omega p_{out}(x_n) + (1-\omega)\sum_{m'=1} \alpha_{m'} \phi_{m'n}}$
    $\nu = \mathbf{P}\mathbf{1}_{N_c}$, $\nu' = \mathbf{P}^T \mathbf{1}_{M_r}$, $\hat{N} = \nu \mathbf{1}_{M_r}$
    $\hat{X} = diag(\nu)^{-1} \mathbf{P} X$
    $\Sigma^{-1} = \lambda \mathbf{G}^{-1} + \frac{s^2}{\sigma^2} d(\nu)$, $\mathbf{v} = \frac{s^2}{\sigma^2} \Sigma diag(\nu)(T^{-1}(\hat{x}) - y)$
    $\hat{u} = y + \mathbf{v}$, $\alpha_m = exp\{\psi(\kappa + \nu_m) - \psi(\kappa M + \hat{N})\}$
    $\bar{x} = \frac{1}{\hat{N}} \sum_{m=1}^{M} \nu_m \hat{x}_m$, $\bar{u} = \frac{1}{\hat{N}} \sum_{m=1}^{M} \nu_m \hat{u}_m$, $\bar{\sigma}^2 = \frac{1}{\hat{N}} \sum_{m=1}^{M} \nu_m \sigma_m^2$
    $\mathbf{S}_{xu} = \frac{1}{\hat{N}} \sum_{m=1}^{M} \nu_m (\hat{x}_m - \bar{x})(\hat{u}_m - \bar{u})^T$
    $\mathbf{S}_{uu} = \frac{1}{\hat{N}} \sum_{m=1}^{M} \nu_m (\hat{u}_m - \bar{u})(\hat{u}_m - \bar{u})^T + \bar{\sigma}^2 \mathbf{I}_D$
    $\mathbf{\Phi} \mathbf{S}'_{xu} \Psi^T = \mathrm{SVD}(\mathbf{S}_{xu})$, $\mathbf{R} = \mathbf{\Phi} diag(1, ..., 1, |\mathbf{\Phi}\mathbf{\Psi}^T|)\mathbf{\Psi}^T$
    $\mathbf{s} = tr(\mathbf{R}\mathbf{S}_{xu})/Tr(\mathbf{S}_{uu})$, $\mathbf{t} = \bar{x} - \mathbf{s}\mathbf{R}\bar{u}$, $\hat{y} = \hat{T}(y + \hat{v})$
    $\hat{\sigma}^2 = \sigma^2$, $\sigma^2 = \frac{1}{D\hat{N}} \left(x^T diag(\nu')x - 2x^T \mathbf{P}^T \hat{y} + \hat{y}^T diag(\nu)\hat{y}\right) + \mathbf{s}^2 \bar{\sigma}^2$
end
```

Algorithm 1: BCPD Algorithm

2.3.2 RMANet

RMANet stands for *Recurrent Multi-view Alignment Network for Unsupervised Surface Registration* and it solves the problems of rigid and non-rigid registration of 3D point clouds with a deep neural network. To avoid dealing with ground truths, which are hard to obtain for non-rigid registration, the authors proposed an unsupervised approach. In addition, to enforce motion coherence among neighbor points, RMANet does not use a pointwise deformation vectors, the more common choice. Instead, it builds the deformed cloud by iteratively applying rigid transformations with skinning weights. This means, that given source and target clouds, $Y = \{y_m\}$, $y_m \in \mathbb{R}^D, m \in 1, .., M$, $X = \{x_n\}$,

$x_n \in \mathbb{R}^D, n \in 1, .., N$, the network estimates

$$\hat{Y}^k = \sum_{r=1}^{K} \mathbf{w}_r^k \psi_r, \tag{2.21}$$

where $\mathbf{w}_k \in \mathbb{R}^M$ is a set of skinning weights for each rigid transformation, ψ_r, and $\hat{Y}^k$ is the registered cloud.

The number of variables to be estimated is considerably large, $M{\times}K$ plus $6{\times}K$, for the weights and registration respectively, and this motivates a recurrent approach. Differently from estimating all at once, at each step, a pair $\mathbf{w}_r^k, \psi_r$ is computed given the previous $\mathbf{w}_{r-1}^k, \psi_{r-1}$. The resulting updated registered cloud, $\hat{Y}$, is given by:

$$\mathbf{w}_r^k = (1 - \mathbf{w}_k^k)\mathbf{w}_r^{k-1}, \tag{2.22}$$

$$\psi_k = \text{RMANet}(\hat{Y}^{k-1}), \tag{2.23}$$

$$\hat{Y}^k = (1 - \mathbf{w}_k^k)\hat{Y}^{k-1} + \mathbf{w}_k^k \psi_k. \tag{2.24}$$

The neural network to estimate the weights and registration matrix consists of an adaptation of the Gated Recurrent Units (GRU) (CHO et al., 2014), in a similar fashion to works targeting optical flow (TEED; DENG, 2020). For details about network architecture, we direct the reader to the implementation, more even than the original paper.

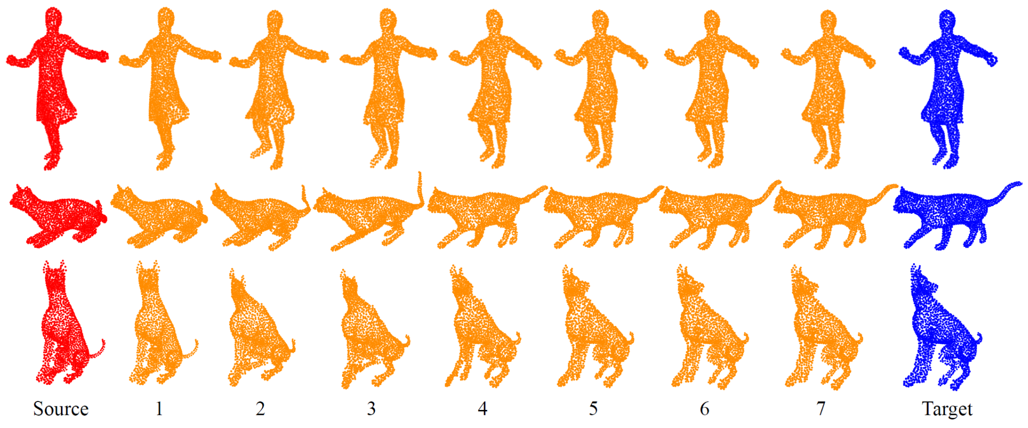
Regarding the unsupervised training, RMANet avoids classic metrics such as Chamfer and Earth Mover's distance. Its main idea is to compare 2D depth maps of both target and deformed point clouds. This means that each point cloud is projected into multiple 2D planes, each with a different point of view. The resulting deformed source and target are then compared. The loss function consists of

$$Loss^k = Loss_{depth} + \lambda_{mask} Loss_{mask} + \lambda_{arap} Loss_{arap} + \lambda_{tran} Loss_{tran} + \lambda_{sparse} Loss_{sparse}, \tag{2.25}$$

where $\lambda_{mask}, \lambda_{arap}, \lambda_{tran}, \lambda_{sparse}$ weight each loss and are input parameters. $Loss_{depth}$ is the mentioned depth map comparison, $Loss_{mask}$ compares the binary maps given the occupancy of each depth map pixel. $Loss_{arap}$ is a regularization term enforcing neighbor points to move similarly, whereas $Loss_{tran}$ limits the translations norm. Finally, $Loss_{sparse}$ limits the L1 norm of the skinning weights.

Figure 2.5 show examples of registration using the RMANEt network. In addition to the final registered cloud, the figure shows the intermediary alignments at each step k.

Figure 2.5: Examples of registration using RMANet on pairs of clouds from their proposed dataset. Figures 1 to 7 show the partial registration at each step k.



Source: Feng *et al.* (2021)

Differently from mentioned rigid registration networks, there is no focus on partial clouds or addressing the presence of any noise in the point clouds. This can be observed in the loss function, where examples of outliers, for example, would hinder any learning. Moreover, fine tunning the weights λ_{mask} and the number of iterations k adds extra complexity during training. We acknowledge these factors contributed to our failures when training the model on different datasets to the one provided by the authors.

2.3.3 FLOT

FLOT is a deep learning technique originally intended for scene-flow scenarios. These scenarios typically consist of a depth image or a point cloud generated by a LiDAR, and the objective is to estimate the motion from source cloud, $Y = \{y_m\}$, $y_m \in \mathbb{R}^D, m \in 1, .., M$, to target cloud, $X = \{x_n\}$, $x_n \in \mathbb{R}^D, n \in 1, .., N$. FLOT tackles the scene-flow problem by modeling it as

$$Y + \mathbf{V} = \mathbf{P}X, \mathbf{P} \in \{0, 1\}, \tag{2.26}$$

where $\mathbf{P}$ is a permutation matrix, and $\mathbf{V}$ is the flow. In this case, to estimate the flow, FLOT focus on the permutation matrix. Computing this matrix is modeled as a optimal transport problem. This means that given a set of source and target options, and a cost matrix, FLOT wants to move mass (points) from one to another in the cheapest way possible. The matrix solving this problem is called *transport plan* $\mathbf{P}' \in \mathbb{R}^{M \times N}$, and in this case each point in the source would have a mass of M^{-1} and in the target N^{-1}. The overall modeling can be written as

$$\mathbf{P}' \in \operatorname*{argmin}_{\mathbf{U}\in\mathbb{R}^{M\times N}} \sum_{i,j}^{M,N} \tilde{\mathbf{P}}_{ij}\mathbf{U}_{i,j} \text{ subject to } \mathbf{U}\mathbf{1}_{N_c} = \mathbf{1}_{M_r}N^{-1} \text{ and } \mathbf{U}^T\mathbf{1}_{M_r} = \mathbf{1}_{N_c}M^{-1}, \quad (2.27)$$

where $\tilde{\mathbf{P}}_{i,j} \geq 0$ is the cost matrix from moving mass (a point in the scene flow context) from a source to a target point. $\mathbf{U}_{i,j}$ are the estimated assignments to be optimized. The remaining parts enforce that no mass is lost during the transport.

In an ideal world, all points in the source cloud would have a corresponding one in the target. However, this is not the case, as there can exist partial clouds, for example. This leads FLOT for a different formulation of the problem with a regularization (CHIZAT et al., 2018) controlled by a parameter λ'. This is adjusted by the user and the larger the value of λ', the more strict to mass preservation the optimization becomes. This optimal transport problem is solved with the iterative Sinkhorn algorithm (CUTURI, 2013).

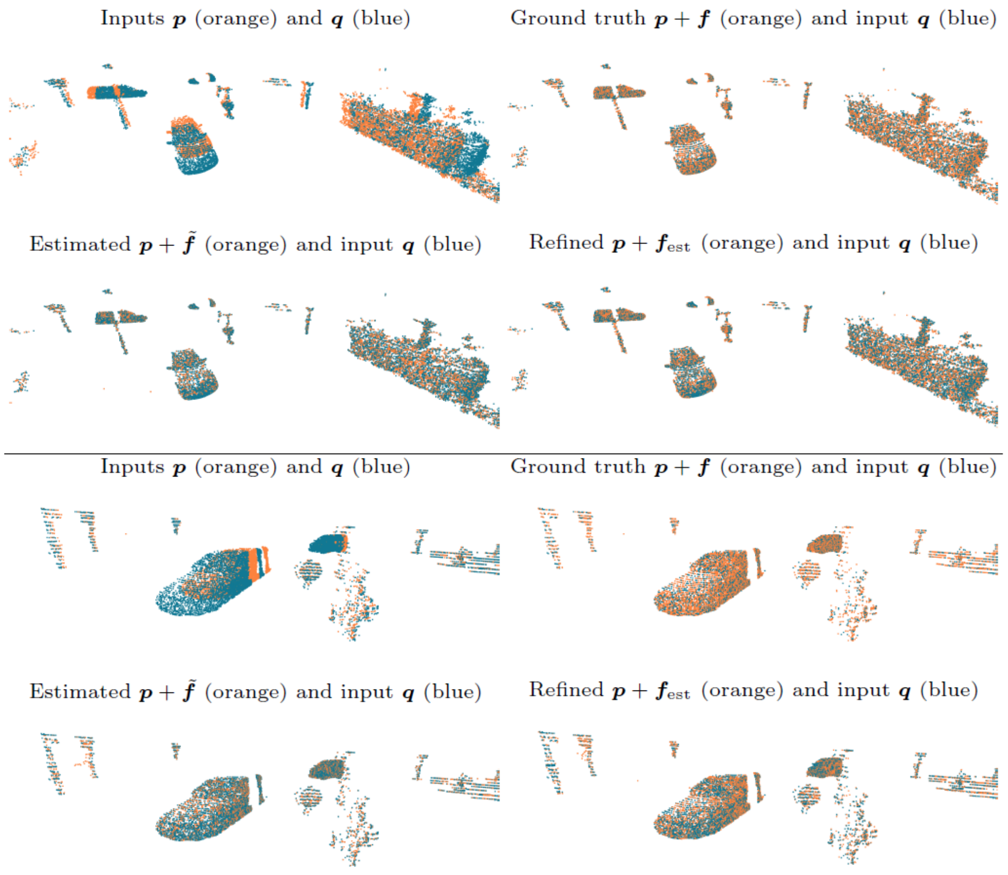
Given this modeling, FLOT focus on training a neural network that is composed of a deep feature estimation module, followed by a cost matrix computation that is fed into an unrolled Sinkhorn module, and a residual refinement of the estimated flow. The first module uses a variation of the PointNet++ network to produce the deep features, $\mathsf{y}_m, \mathsf{x}_n$. The cost matrix is then defined as

$$\tilde{\mathbf{P}}'_{i,j} = \left(1 - \frac{\mathsf{y}_m^T\mathsf{x}_n}{||\mathsf{y}_m||||\mathsf{x}_n||}\right). \quad (2.28)$$

After the Sinkhorn algorithm is run for a specified number of iterations, the resulting matrix, $\mathbf{P}'$, is used to estimate the flow by $\tilde{f} = \tilde{\mathbf{P}}X - Y$. Finally, $\tilde{f}$ is refined by $f_{est} = \tilde{f} + h'(\tilde{f})$, where the residual module, h', uses the same architecture that computes the deep features. We refer the reader to the FLOT author implementation for details about the network architecture. Finally, FLOT is trained in a supervised fashion using L1 distance to the ground truth as loss function.

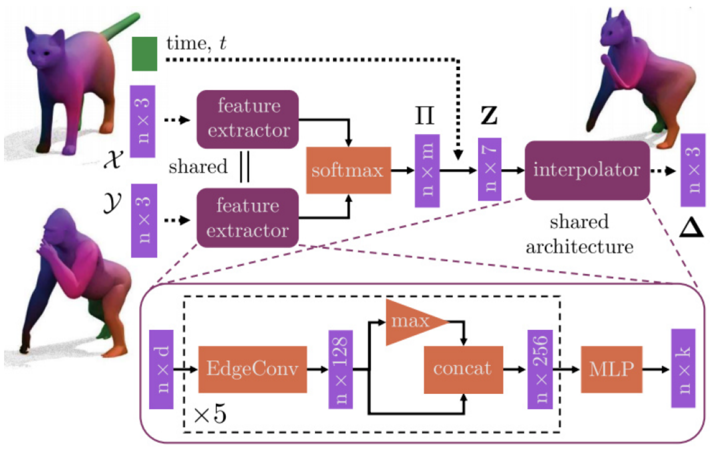
Figure 2.6 presents results on the scene flow dataset from KITTI (GEIGER et al., 2013), where FLOT offered state-of-art results. This domain consists mostly of complete point clouds where the source and target clouds are sampled in a short time interval. As a result, most cases in the dataset consist of different objects with almost independent rigid motion. Regardless of this specific application, FLOT does consider the possibility of no correspondences between points in its formulation of the optimal transport problem by using residual refinement network. However, we notice that the flow refinement module

Figure 2.6: Figure extracted from FLOT (PUY; BOULCH; MARLET, 2020). It presents two examples of registration using FLOT on point clouds from the KITTI dataset (GEIGER et al., 2013). Notice they use a different notation compared to the adopted in this thesis, and we follow it solely in this respective section. Source cloud is denoted by p and target by q.


Inputs $\boldsymbol{p}$ (orange) and $\boldsymbol{q}$ (blue)
Ground truth $\boldsymbol{p}+\boldsymbol{f}$ (orange) and input $\boldsymbol{q}$ (blue)
Estimated $\boldsymbol{p}+\tilde{\boldsymbol{f}}$ (orange) and input $\boldsymbol{q}$ (blue)
Refined $\boldsymbol{p}+\boldsymbol{f}_{\text{est}}$ (orange) and input $\boldsymbol{q}$ (blue)
Inputs $\boldsymbol{p}$ (orange) and $\boldsymbol{q}$ (blue)
Ground truth $\boldsymbol{p}+\boldsymbol{f}$ (orange) and input $\boldsymbol{q}$ (blue)
Estimated $\boldsymbol{p}+\tilde{\boldsymbol{f}}$ (orange) and input $\boldsymbol{q}$ (blue)
Refined $\boldsymbol{p}+\boldsymbol{f}_{\text{est}}$ (orange) and input $\boldsymbol{q}$ (blue)

Source: Puy, Boulch, and Marlet (2020)

Figure 2.7: Diagram of the neural network for point-to-point correspondence and mesh interpolation on 3D meshes proposed by NeuroMorph. In addition, examples of source and target meshes, and the result interpolation.



Source: Eisenberger *et al.* (2021)

might not be enough to generalize to more challenging noise and deformation scenarios. We attribute this to its architecture, the same used on feature estimation. Thus, the refinement does not have additional elements to deal with these challenges.

2.3.4 NeuroMorph

NeuroMorph tackles two different but related problems: point-to-point correspondences in a mesh and smooth interpolation between meshes. It proposes a single neural network that is trained end-to-end on both tasks at once and in an unsupervised fashion. The first problem aims at estimating a correspondence matrix $\mathbf{P}$, which relates source vertices Y to target ones X. The second problem has the objective of learning how to produce intermediate deformed vertices Y_k given a value $k \in [0, 1]$ such that Y_k is always smooth. For example, if $k = 0.5$, one would expect $Y_{0.5}$ to be half deformed into X.

The model proposed to solve the mentioned scenarios is presented in Fig. 2.7. The feature extractor module is composed of two parts. The first, *EdgeConv*, was inspired by DGCNN and focuses on the mesh edge connections. The second part does a maxpolling over all features and aims at including global contextual information into the features. The two modules can be respectively described by

$$\mathsf{x}_m^{k+1} = \max_{j:(m,j)} h_\theta^k(\{\mathsf{x}_m^k, \mathsf{x}_m^k - \mathsf{x}_j^k\}) \; \forall j \in \mathcal{E}_m, \mathsf{x}_m^{k+2} = (\{\mathsf{x}_m^{k+1}, \max \mathsf{x}_n^{k+1}\}), \tag{2.29}$$

where h_θ^k are a small residual networks, $\{,\}$ is the concatenation operator, and $\mathcal{E}$ are the edges of each vertice x_n whose feature is x_m.

Neuromorph computes the correspondence matrix by taking the cosine similarity of the resulting deep features $\mathsf{y}_m, \mathsf{x}_n$ (similar to Eq. (2.28)). This is normalized by a softmax operator and results in a soft-assignment matrix. However, to feed the interpolation module the neural network concatenates the correspondence with the discrete time interval resulting in $Z = (Y, \mathbf{\Pi}X - Y, t)$.

Finally, for the unsupervised training, the number of time intervals to be interpolated starts with a single one, $k = 1$, and it increases logarithmically. The loss function is defined as

$$Loss = \lambda_{reg} Loss_{reg} + \lambda_{arap} Loss_{arap} + \lambda_{geo} Loss_{geo}, \tag{2.30}$$

where $Loss_{reg}$ is the unsupervised registration loss and it is given by $||\mathbf{\Pi}X - Y_1||^2$, this is the distance between the target vertices back projected by the soft-assignment matrix and the final interpolation of the source vertices. $Loss_{arap}$ is used between interpolation intervals Y_k, Y_{k+1}, and it penalizes motions that are not rigid transformations (SORKINE; ALEXA, 2007). The last term tries to preserve the pairwise geodesic matrices, D_Y, D_X, given the correspondence matrix Π by limiting $||\mathbf{\Pi} D_X \mathbf{\Pi}^T - D_X||^2$.

NeuroMorph offers impressive results on both tasks it approaches, vertice matching and mesh interpolation. In addition, the usage of unsupervised training shows that for well-behaved meshes, pointwise matching is not necessary. Similar to RMANet and FLOT, we notice a lack of components focused in guaranteeing robustness. Furthermore, the original paper does not have results targeting meshes/clouds with any noise. We experienced this limitation when generalizing to the more challenging cases when we could not train our implementation (there is no publicly available implementation) on scenarios with outliers or partial point clouds.

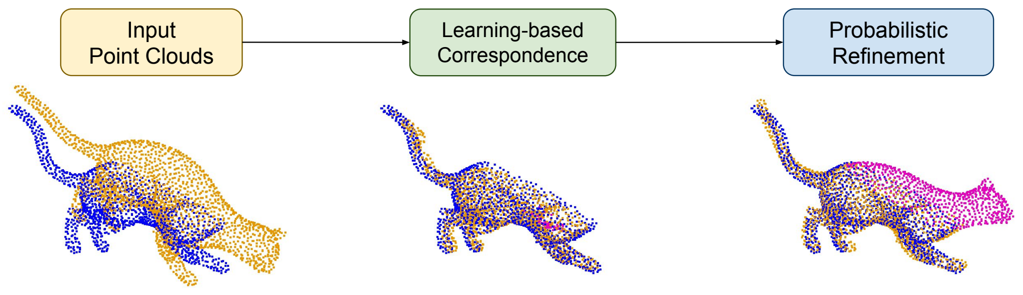
2.4 Summary

This chapter introduced the main concepts related to rigid and non-rigid registration in 3D point clouds. The first section gave a short and general description of them. This showed how open the scope can be for possible solutions, which allows different modeling strategies and consequently, largely distinct solutions. These where presented in the subsequent sections, which gave an overall view of the related and latest works. In each of these sections, we presented a in-depth view of traditional approaches (Section 2.2.1,Section 2.3.1) and learning-based ones (the remaining subsections). For these, we highlight how recent works on rigid registration are designed to deal with partial point clouds (Sections 2.2.2 to 2.2.4). On the other hand, learning based non-rigid ones have focused on dealing with larger deformations (Sections 2.3.2 to 2.3.4) but not with partial clouds. Furthermore, while traditional approaches can be robust to noise (Section 2.3.1), learning-based ones also lack robustness, thus being restricted to well-behaved scenarios.

3 ROBUST DENSE NON-RIGID REGISTRATION

This chapter presents our technique for registration of 3D point cloud. We refer to it as *robust dense registration* (RDR) and it has two variations: one for rigid and one for non-rigid registration. Since rigid registration can be framed as a special case of the non-rigid one, we discuss non-rigid registration in this chapter, and cover rigid registration in Chapter 5. Here, we show how we adapted the original BCPD (HIROSE, 2020b) algorithm to work together with a neural network. We then show the RDR algorithm for non-rigid registration and how it can work with cyclic consistency, or in an optimization loop.

Figure 3.1: Examples of non-rigid registration after each step of RDR on partial point cloud. The source cloud is in orange and the target one is in blue. Points without correspondence in the target are shown in magenta.



Source: The Authors

3.1 Non-rigid Registration

Our non-rigid registration technique consists of two parts: *learning-based correspondences* followed by *probabilistic refinement* (Fig. 3.1). To obtain correspondences among the samples of the source and target point clouds, we modified the SuperGlue network (SARLIN et al., 2020) (originally designed to establish matches among keypoints in pairs of 2D images). The changes consist of adding a feature encoding to allow dense match of 3D point clouds, and a new Optimal Transport module that improved the registration performance by 43% (Section 6.1.7). For the probabilistic refinement, we adapted the BCPD algorithm to use the soft-correspondence assignments produced by the modified SuperGlue network, thus replacing BCPD's point matching process by a more robust alternative. Next, we briefly review the original BCPD algorithm, providing the context required to understand our RDR technique for non-rigid registration, which is summa-

Table 3.1: Main symbols used at Algorithm 2

Name	Description
$diag$/Tr	Diagonal matrix/Trace
λ	Motion coherence intensity parameter
β	RBF kernel free parameter
γ/τ	Initial scaling of σ^2/Convergence tolerance
$\mathbf{G}$	$M \times M$ matrix defining motion coherence over v
ν	Number of target points matched to each source
ν'	Probabilities that each target point is an outlier
$\hat{X}$	$M \times 3$ target cloud permutated by P
σ^2	Covariance of each GMM
$\hat{\sigma}^2$	Final residual-covariance matrix for a given P

Source: The Authors

rized by Algorithm 1.

BCPD models the registration problem using a GMM. It iteratively optimizes $\sigma^2, \mathrm{s}, \mathbf{R}, \mathbf{t}, \mathbf{v}$, which are, respectively, the single variance of all 3D Gaussian distributions in the GMM (one Gaussian per source point), the scale, the rotation, the translation, and the non-rigid deformation vectors required for the matching. Notice that differently from the Algorithm 1, but following the BCPD implementation we do not use the term $\bar{\sigma}^2$ which accounts for the variance of the estimated $\mathbf{v}$. The final registration is given by $\hat{Y} = \mathbf{s}\mathbf{R}(y_m + \mathbf{v}_m) + t$, where $Y = \{y_m\}$,$y_m \in \mathbb{R}^3, m \in 1, .., M$ is the source point cloud to be aligned to a target cloud $X = \{x_n\}$, $x_n \in \mathbb{R}^3, n \in 1, .., N$. y_m and x_n are the coordinates of the points in the corresponding clouds, and M and N are the number of such points. x_n may not have a corresponding $y_m \in Y$, and can also be an outlier with some given probability.

A Gaussian distribution prior enforces motion coherence over the non-rigid deformation vectors, meaning that near points should move similarly. For this, BCPD assumes the prior to have variance $\lambda\mathbf{G}$, where $\mathbf{G}$ is an $M \times M$ positive-definite matrix generated by a RBF kernel and λ is a free parameter. The algorithm iteratively estimates a matching probability matrix $\mathbf{P}$ between all deformed source and target points. $\mathbf{P}$ is used to update the values of variance, rigid and non-rigid transformations. While BCPD produces good registration results, it does not handle large deformations.

Our technique uses the soft assignments produced by our adapted SuperGlue network as an approximation to BCPD's matrix $\mathbf{P}$. BCPD assumes that the probabilities in $\mathbf{P}$ came from a GMM. Although this is not the case for the soft assignments, we argue that it is feasible to use them as BCPD's $\mathbf{P}$ matrix encodes correspondence and outlier information as matching probabilities. The critical aspect for our technique is then to iteratively update $\sigma^2, s, \mathbf{R}, \mathbf{t}$, and $\mathbf{v}$ from $\mathbf{P}$ as well as update $\mathbf{P}$ from these variables. To achieve this, we split the BCPD original loop into two nested ones. The outer loop updates $\mathbf{P}$ based on the latest variable values. The inner one optimizes the five variables given the estimate of $\mathbf{P}$. This setup reduced the registration error for partial clouds on our Custom Dataset (Section 6.1.3) by 22% on average, when compared to (the original) BCPD. In our experiments, we use a single outer loop iteration, and we show this is enough to achieve state-of-the-art results. Section 6.1.8 presents further analysis on these loop options and results with cycle consistency on $\mathbf{P}$. Algorithm 1 summarizes our robust non-rigid registration technique, where $\mathbf{1}_{Nc}$ is a column vector consisting of N 1s, and $\mathbf{1}_{Mr}$ is a row vector consisting of M 1s. For estimating $\mathbf{P}$ in line 4, we refer to a generic *Neural Network Model*, as this framework can be used with other models. In practice, we found that our adapted SuperGlue network produces good results and was used in all experiments and examples reported in the paper.

3.2 Summary

This chapter introduced our algorithm for non-rigid registration. Our technique builds on BCPD by substituting their original dense correspondence estimation between source and target clouds with a matrix generated by a deep-learning model. Even though this violates BCPD's assumption of the probabilities coming from a GMM, we argue that this is still a valid option. Another difference in the algorithm is the inner and outer loops, which also motivates splitting the optimization of σ^2. The result is an algorithm that can work with one or multiple iterations of the outer loop.

```
Inputs  : $X = \{x_n\}$, $Y = \{y_m\}$, $\lambda$, $\beta$, $\gamma$, $\tau$, $outer\_max$, $inner\_max$
Outputs: $\hat{Y} = T(y + v) = \mathbf{s}\mathbf{R}(y + \mathbf{v}) + \mathbf{t}$
$\mathbf{G} = \exp -\frac{1}{2\beta^2}\|(y - y')\|^2$, $\hat{Y} = Y$
$\sigma^2 = \frac{\gamma}{MN3}\sum_{n=1}^{N}\sum_{m=1}^{M}\|x_n - y_m\|^2$, $\hat{\sigma}^2 = \tilde{\sigma}^2 = 0$, $\tilde{\tau} = \hat{\tau} = \sigma^2$
// loop for multiple correspondence estimation with our model
while $i < outer\_max$ and $\hat{\tau} > \tau$ do
    $\mathbf{P} =$ Neural Network Model$(\hat{Y}, X)$ ; // call trained neural network
    $\nu = \mathbf{P}\mathbf{1}_{N_c}$, $\nu' = \mathbf{P}^T\mathbf{1}_{M_r}$, $\hat{N} = \nu\mathbf{1}_{M_r}$
    $\hat{X} = diag(\nu)^{-1}\mathbf{P}X$ ; // permutate the target cloud based on the correspondences
    // nested loop to avoid variances instability
    while $j < inner\_max$ and $\tilde{\tau} > \tau$ do
        $\mathbf{G}' = \frac{\lambda\sigma^2}{\mathbf{s}^2}diag(\nu^{-1}) + \mathbf{G}$
        $\mathbf{r} = \mathbf{G}'^{-1}(T^{-1}(\hat{x}) - y)$, $\mathbf{v} = G\mathbf{r}$, $\hat{u} = Y + \mathbf{v}$ ; // infer motion coherence for low confidence correspondences
        $\bar{x} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m\hat{x}_m$, $\bar{u} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m\hat{u}_m$
        $\mathbf{S}_{xu} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m(\hat{x}_m - \bar{x})(\hat{u}_m - \bar{u})^T$
        $\mathbf{S}_{uu} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m(\hat{u}_m - \bar{u})(\hat{u}_m - \bar{u})^T$
        $\mathbf{\Phi}\mathbf{S}'_{xu}\mathbf{\Psi}^T$ = SVD$(S_{xu})$
        // rigid transformation and scale estimation
        $\mathbf{R} = \mathbf{\Phi}diag(1, ..., 1, |\mathbf{\Phi}\mathbf{\Psi}|)\mathbf{\Psi}^T$
        $\mathbf{s} = tr(\mathbf{R}\mathbf{S}_{xu})/Tr(\mathbf{S}_{uu})$
        $\mathbf{t} = \bar{x} - \mathbf{s}\mathbf{R}\bar{u}$, $\hat{y} = \hat{T}(y + \mathbf{v})$
        $\tilde{\sigma}^2 = \frac{1}{3\hat{N}}\left(x^T diag(\nu')x - 2x^T\mathbf{P}^T\hat{y} + \hat{y}^T diag(\nu)\hat{y}\right)$ ; // optimize over variance, the weighted average distance to target
        $\tilde{\tau} = |\sigma^2 - \tilde{\sigma}^2|$, $\sigma^2 = \tilde{\sigma}^2$, j++
    end
    $\hat{\tau} = |\sigma^2 - \hat{\sigma}^2|$,$\hat{\sigma}^2 = \sigma^2$, i++
end
```

Algorithm 2: RDR Non-Rigid Registration of 3D Point Clouds Algorithm

4 LEARNING-BASED CORRESPONDENCE NETWORK

This chapter describes how we adapted the SuperGlue network (SARLIN et al., 2020) to learn the probability matrix P, which indicates the dense soft correspondences between point clouds Y and X. The original SuperGlue network estimates correspondences of sparse keypoints in pair of 2D images, and has three main modules. The first, *feature embedding*, learns keypoint features while considering their RGB and positional information. This is followed by an *attention module* to propagate knowledge within keypoints in the same image (*self*), and across different images (*cross*). Finally, in the *Sinkhorn operator with dustbins module*, the matrix representing the similarity between descriptors is obtained as the product $\mathsf{f}^{Y}(\mathsf{f}^{X})^{T}$ of the extracted features. To account for visibility, this score matrix is refined with a Sinkhorn operator (MENA et al., 2018). Next, we discuss the changes we made to the feature embedding and optimal transport modules to adapt the original SuperGlue network for defining correspondences between point clouds. Additionally, we describe in details the attention module and the loss function we used for training. The adapted network is shown at Fig. 5.1.

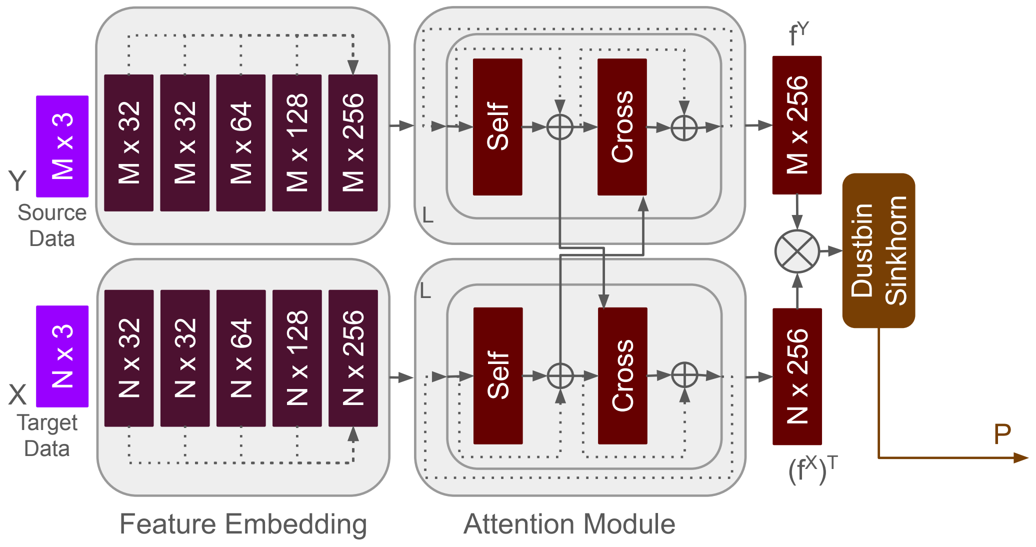
4.1 Feature Embedding

Differently from SuperGlue, we densely estimate the initial point-cloud features using the DCP approach (WANG; SOLOMON, 2019). This is based on the DGCNN (WANG et al., 2019) network and aims to account for local geometry information. The embeddings for the different layers, l, are given by:

$$\begin{aligned} \mathsf{x}_m^0 &= max_{j:(m,j)} h_\theta^0(\{x_m, x_j\}) \; \forall j \; \in \; \mathsf{N}_m, \\ \mathsf{x}_m^l &= max_{j:(m,j)} h_\theta^l(h_\theta^{l-1}), \\ \mathsf{x}_m^{final} &= h_\theta^l(\{\mathsf{x}_m^{l-1}, ..., \mathsf{x}_m^0\}), \end{aligned} \tag{4.1}$$

where N_m are the k nearest neighbors of x_m, and $\{,\}$ is the concatenation operator. h_θ^l is a non-linear function consisting of a multi-layer perceptron (MLP: 2D Convolution with kernel size of 1 + Batch Normalization + ReLU) which is followed by maxpolling across the neighbors. We use five layers (Fig. 5.1); the first computes the embedding from the point cloud coordinates (x_m^0). The next three layers iteratively refine the embedding (x_m^l). The last one takes the concatenation of all previous four layers and does not max-pool

Figure 4.1: Adapted SuperGlue network for learning-based correspondences. Non-rigid registration uses a single iteration of the brown modules, producing a soft-assignment matrix P as output.



Source: The Authors

(x_m^{final}).

4.2 Attention Module

We directly use the attention mechanism of SuperGlue to include more contextual information in the deep features. Differently from NeuroMorph's global max-pooling, we consider not only information about the current point cloud, but across clouds as well. As a result, the deep feature x_n receives contributions from X (*self*) and Y (*cross*). In the context of point-cloud registration, DCP was the first to use attention by adapting the work of Vaswani *et al*. (VASWANI et al., 2017). In our case, attention enhances the dense estimated features by alternatively applying multiple independent layers of *self* and *cross* modules in a residual manner

$$x_n^{l+1} = x_n^l + \delta_{x_n}^l, \; \delta_{x_n}^l = \begin{cases} attention(x_n^l, x_n^l), & \text{if } self, \\ attention(x_n^l, y_m^l), & \text{if } cross, \end{cases} \tag{4.2}$$

where x_n^l is the current feature values and $attention$ is the actual module.

The $attention$ function is a neural network that aggregates features from one of its inputs, x_n^l or y_m^l, based on a learned similarity between this same input and the queried one, x^l_n in this case. The same is done to y^l_m and we apply alternately *self* and *cross* L times, resulting in $\mathsf{f}^Y_m, \mathsf{f}^X_n$ (Fig. 5.1). The module consists of a leaner version of the Transformer network (VASWANI et al., 2017). Its residual increments, $\delta^l_{x_n}$, come from an MLP (3×: Linear Layer + Batch Normalization + ReLU) whose last layer only does the convolution. The main objective of this perceptron is to join the current feature with the message, m_n. This message is the value v_m weighted by how similar are queries q_n, and keys k_m. The term q_n is the result of a linear layer applied to the first *attention* input. k_m and v_m use the same layer type but with the second input. Thus the attention module learns how to take into account information across all points from source and target clouds, as they interchangeably provide the update message. The overall updates in a *cross* setup are given by

$$\delta^l_{x_n} = \mathrm{MLP}^l(\{\mathsf{x}_n, \mathsf{m}_n\}),\ \mathsf{m}_n = h^l_{\mathsf{m}_n}\left(\frac{softmax(\mathsf{q}_n \mathsf{k}^T_m)\mathsf{v}_m}{\sqrt{256}}\right),\ \text{where} \tag{4.3}$$

$$\mathsf{q}_n = h^l_q \mathsf{x}_n,\ \mathsf{k}_m = h^l_k \mathsf{y}_m,\ \mathsf{v}_m = h^l_v \mathsf{y}_m. \tag{4.4}$$

In these equations, $\sqrt{256}$ is the square root of the number of dimensions in the final feature, and it was proposed by the original Transformer network(VASWANI et al., 2017) for more stability. In addition, h^l are linear layers and they together with MLP^l have independent weights for each iteration $l \in L$.

4.3 Sinkhorn Operator with Dustbins

Given the refined descriptors, a direct way to establish correspondences is through scores given by the dot products $\tilde{\mathbf{P}}_{m,n} = \langle \mathsf{f}^Y_m, \mathsf{f}^X_n \rangle$, where $\mathsf{f}^Y_m, \mathsf{f}^X_n$ are 1×256 feature vectors corresponding to the rows of the tensors f^Y and f^X. This is also how DCP, CorrNet3D, and NeuroMorph establish their matching scores. Although this produces good results for one-to-one correspondences, it is unable to directly account for points without correspondences. In addition, the dot product by itself does not produce a valid probability matrix, which in our case should be a relaxed doubly stochastic matrix. This means the sum of the probabilities in the rows and columns should be less or equal to one. To tackle this situation, we follow RPM-Net (YEW; LEE, 2020) on using the Sinkhorn operator proposed by Mena *et al.* (MENA et al., 2018), which builds on the Sinkhorn and Knopp theorem (SINKHORN; KNOPP, 1967). The possible lack of correspondences is handled by an extra possibility of assignment in both sets of descriptors, called dustbins (Fig. 5.1).

The resulting Sinkhorn operator can be defined as

$$\mathbf{P}^{0}_{M+1,N+1} = \{\{exp(\tilde{\mathbf{P}}), \mathbf{0}_{M_r}\}, \mathbf{0}_{N_c+1}\}, \tag{4.5}$$

$$\mathbf{P}^{o}_{M+1,N+1} = \{\mathbf{P}^{o-1}_{M,N+1} \varnothing (\mathbf{P}^{o-1}_{M,N+1}\mathbf{1}_{N_c+1}\mathbf{1}^{T}_{N_c+1}), \mathbf{P}^{o-1}_{m,:}\}, \tag{4.6}$$

$$\mathbf{P}^{o+1}_{M+1,N+1} = \{\mathbf{P}^{o}_{M+1,N} \varnothing (\mathbf{1}_{M_r+1}\mathbf{1}^{T}_{M_r+1}\mathbf{P}^{o}_{M+1,N}), \mathbf{P}^{o}_{:,n}\}, \tag{4.7}$$

$$\mathbf{P} = \mathbf{P}^{o+1}_{M,N}. \tag{4.8}$$

This operator is iterative and it converges to the desired probability matrix in the limit, $o \to \infty$. Initially we append $\tilde{\mathbf{P}}$ with an extra column and row where all elements are 0, $\mathbf{0}_{M_r}, \mathbf{0}_{N_c+1}$. These are called dustbins and are considered another correspondence possibility. Differently from SuperGlue, we do not append to all dustbins a single learnable parameter or define an expected number of matches for them. $\varnothing$ denotes the element-wise division, and $\{,\}$ is the concatenation parameter. We notice that the updates , $\mathbf{P}^{o}_{M+1,N+1}$ and $\mathbf{P}^{o+1}_{M+1,N+1}$, consist on alternately normalizing rows and columns to sum to 1. However, when updating the rows, we leave the dustbin for columns out (the last row, $\mathbf{P}^{o-1}_{m,:}$) and vice-versa for columns. This guarantees they are not bound to the constraint of summing to 1 since they could be matched by any number of points.

4.4 Loss Function

We introduce a new term to the cross-entropy loss used by RGM to compare the estimated matrix $\mathbf{P}$ and the ground truth $\bar{\mathbf{P}}$ (a hard-assignment matrix). This is inspired by loss functions proposed by Ouyang and Raviv (OUYANG; RAVIV, 2021) and Predator (HUANG et al., 2021), and it aims at classifying if a source point $y_m \in Y$ has a correspondence to any target point $x_n \in X$. The new loss term compares the sum of the matching probabilities associated with each source point, $\nu = \mathbf{P}\mathbf{1}_{N_c}$, to the ground truth $\bar{\nu} = \bar{\mathbf{P}}\mathbf{1}_{N_c}$. Even though this new term is indirectly covered by the original loss function, we justify it because we want to minimize the occurrence of false matches. False matches tend to create artifacts as the whole neighborhood around each incorrectly matched points is deformed as well. The complete loss function is given by:

$$\begin{aligned} Loss = &-\sum_{n=1}^{N}\sum_{m=1}^{M}(\bar{\mathbf{P}}_{m,n}\log\mathbf{P}_{mn} + (1-\bar{\mathbf{P}}_{m,n})\log(1-\mathbf{P}_{m,n})) \\ &-\sum_{m=1}^{M}(\bar{\nu}_m\log\nu_m + (1-\bar{\nu}_m)\log(1-\nu_m)). \end{aligned} \tag{4.9}$$

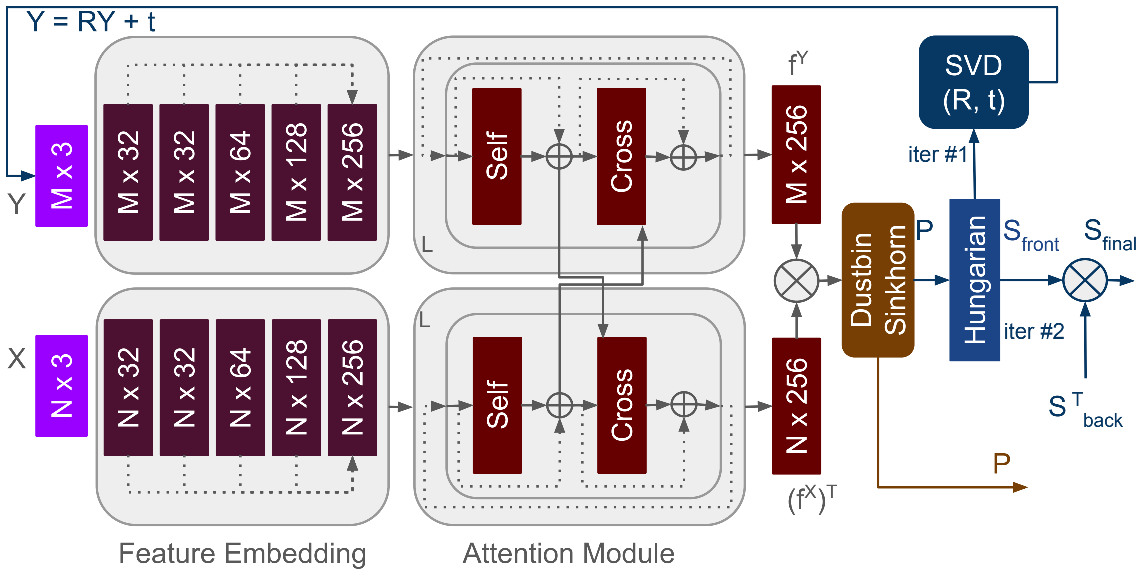
4.5 Summary

This chapter presented our neural network used by the two registration algorithms Chapters 3 and 5. It modifies SuperGlue (SARLIN et al., 2020) in three main aspects. The first is DGCNN as the feature encoding. This is followed by replacing the layer used to produce the final correspondence matrix. Instead of modeling this part as an optimal transport problem and solving it with Sinkhorn algorithm (CUTURI, 2013), we employ the Sinkhorn operator with dustbins. Finally, we include an extra term to the cross-entropy loss function that is focused on reducing false positive correspondences. The resulting network still uses the Superglue sequence of modules: *feature estimation* + *attention module* + *refinement with dustbins*. However, our proposed changes are vital for good performance in 3D point clouds.

5 ROBUST DENSE RIGID REGISTRATION

This chapter presents a variation of our *robust dense registration* (RDR) technique for rigid point cloud registration. The framework used is similar to the proposed by RGM (FU et al., 2021) but with our neural network instead of their proposed one (Figure 5.1). It applies cyclic consistency to estimate a consolidated set of correspondences that produce the final registration. Moreover, in each direction of the cycle, we have multiple iterations over the network to refine the partial matches. In all our examples and experiments we adopt a default of two iterations.

Figure 5.1: Adapted SuperGlue network for learning-based correspondences. For rigid registration, the network in Fig. 4.1 has the extra blue modules, and performs two iterations, producing a hard assignment matrix $\mathbf{S}_{final}$ as output.



Source: The Authors

5.1 Rigid Registration

Our rigid registration model aims to minimize the following error function

$$E = \sum_{n=1}^{N} \sum_{m=1}^{M} \mathbf{c}_n \|\mathbf{R} y_m + \mathbf{t} - x_n\|^2, \tag{5.1}$$

where $\mathbf{c}_n \in \{0, 1\}$. $\mathbf{c}_n = 1$ if the adapted SuperGlue network is confident that x_n has some correspondence in the source point cloud; $\mathbf{c}_n = 0$, otherwise. Since this problem is simpler compared to non-rigid registration, our model only uses learning-based correspondence, dropping the probabilistic refinement step. In this case, the adapted SuperGlue network is slightly modified with respect to the version used for non-rigid registration, and iterates twice (as opposed to a single iteration used for the non-rigid case). This is illustrated in Fig. 5.1 by the blue modules (Hungarian and SVD) and lines at the right portion of the network. During the first iteration, the Hungarian algorithm converts the soft-assignment matrix $\mathbf{P}$ into a hard assignment matrix S, which undergoes an SVD decomposition to estimate the parameters $\mathbf{R}$ and $\mathbf{t}$ of a rigid transformation that maps Y into X (this same SVD based algorithm is used internally in Algorithm 2, lines 10-16). $\mathbf{R}$ and $\mathbf{t}$ are used to obtain a transformed version of the source point cloud Y for the second iteration. The output of the second iteration is a hard assignment matrix $\mathrm{S}_{final} = \mathrm{S}_{front}\,\mathrm{S}^T_{back}$, where S_{front} is the output of the Hungarian module at the second iteration. S^T_{back} is the transpose of a similar matrix produced by a mirrored version of the network shown in Fig. 5.1, obtained swapping the roles of point clouds Y and X. The final $\mathbf{R}$ and $\mathbf{t}$ are estimated applying SVD to S_{final}.

5.2 Summary

This chapter presented a variation of our RDR algorithm for rigid registration. We apply cyclic consistency and multiple calls to our neural network together with the SVD method to estimate rigid registration from a set of correspondences. Differently from the soft-assignment matrix used in the non-rigid case, here, we estimate the transformation from a hard-assignment correspondence matrix.

6 RESULTS

This chapter presents quantitative and qualitative results on non-rigid and rigid registration. Regarding non-rigid, we introduce the compared approaches, their parameters, and the datasets used in the comparisons (Sections 6.1.1 and 6.1.2). The next sections explain our proposed self-supervised training (Section 6.1.3) and the adopted metrics (Section 6.1.4). In terms of actual results and comparisons, we start by showing and discussing the registrations with full-body point clouds. These clouds have larger deformations than the finer-grained details of the subsequent section, which concerns datasets focused on faces (Sections 6.1.5 and 6.1.6). Finally, we present a set of auxiliary experiments. The first is an sensitivity and ablation study that justifies our choice of architecture and parameters for the proposed neural network (Section 6.1.7). The following sections discuss the use of multiple outer-loop iterations and cyclic consistency, the selection of BCPD parameters, training with mixed types of noises, and generalization to a different number of points during training and evaluation (Sections 6.1.8 to 6.1.11).

On rigid registration, we initially present the approaches compared to ours, the dataset used in the experiments, and the different noise scenarios (Section 6.2). Next, we introduce the adopted metrics and discuss the qualitative and quantitative results (Sections 6.2.1 and 6.2.2). Finally, we focus on an sensitivity study regarding the number of iterations of the rigid algorithm (Section 6.2.3).

We implemented our models using PyTorch and Python and used them to register a large number of point clouds under challenging configurations. All reported experiments were performed on a 3.2 GHz PC with 32 GB of memory and an Nvidia GTX 1070 GPU with 8 GB of memory.

6.1 Non-Rigid Registration

The experiments included four types of point clouds: (i) *Clean*, containing no noise; (ii) *Cropped*, where contiguous regions covering 30% of the points were removed; (iii) *Outliers*, where 20% of target points consist of random uniformly distributed points inside the target bounding box; and (iv) *Holes*, where 25% of the points in the target point cloud were removed creating random holes around seeds. For experiments involving *Cropped* point clouds, we argue that it is important to classify regions in the source cloud with no correspondences to target.

Figure D.1 illustrates the results produced by our method for challenging scenarios. It consistently produced good registration for all these cases, which include significant amount noise, and points without correspondences (shown in pink). We compare our approach to five others designed, or adapted by us, to perform non-rigid registration of point clouds: RMANet (FENG et al., 2021), a variant of our method based on RGM's $\mathbf{P}$ matrix (FU et al., 2021), a variant of our method based on NeuroMorph's $\mathbf{P}$ matrix (EISENBERGER et al., 2021), FLOT (PUY; BOULCH; MARLET, 2020), and BCPD (HIROSE, 2020b). RMANet is a recurrent network which presents state-of-the-art results. RGM explicitly learns a matrix $\mathbf{P}$ and applies it to rigid registration. We adapted Algorithm 2 to use RGM's $\mathbf{P}$ matrix and refer to this variant of our method as Ours-RGM. NeuroMorph was proposed for mesh correspondences and interpolation. Since it has no publicly available implementation, we implemented it and adapted its correspondence module to use with point clouds, producing a $\mathbf{P}$ matrix. Similar to RGM, we also adapted Algorithm 2 to use NeuroMorph's $\mathbf{P}$ matrix and refer to this variant of our method as Ours-Neuro. We include these comparisons to show that the $\mathbf{P}$ matrix generated by our adapted SuperGlue network (Fig. 5.1) leads to superior results compared to the ones obtained using RGM's and NeuroMorph's $\mathbf{P}$ matrices. FLOT is a scene-flow technique that can be applied to non-rigid deformation. BCPD is a state-of-the-art method for non-rigid registration based on iterative optimization.

The $\mathbf{P}$ matrices generated by our method, RGM, and NeuroMorph can be used to obtain the deformed cloud using a projection operation defined as $\hat{Y} = \mathbf{P}X$. RMANet and FLOT directly generate the deformed point clouds. In Table 6.1, we indicate the results obtained by each model on *Clean* point clouds, either with the projection operation (P) or with the direct generation (D), in the column *P/D*.

6.1.1 Parameters and Training Details

We used the following parameters when performing the comparisons. Techniques and parameters not mentioned had their default values used:

- Our Neural Network Model: neighborhood size $=$ 20, 20 Sinkhorn iterations ($O =$ 20);
- Ours, Ours-RGM, Ours-Neuro Probabilistic Refinement: $\lambda = 2$, $\gamma = 3$, $\beta = 2$, $K = 100$, $inner\_max$=50, $outer\_max$=1, and $\tau = 0.001$;
- NeuroMorph: Adam optimizer and learning rate of 0.00025;

- BCPD (HIROSE, 2020a): $\lambda = 20$, $\gamma = 3$, $\beta = 1$, $\omega = 0.1$, $K = 300$, $max.iter. = 300$, and $\tau = 0.0001$.

The exception is the *Clean* setup, where $\beta = 0.5$ for Ours, Ours-RGM, and Ours-Neuro. K controls the Nystrom sampling used to accelerate the probabilistic module (HIROSE, 2020b). For our probabilistic refinement and BCPD parameters, we selected values based only on minimizing the end-to-end point error in the *Cropped* scenario for the RMANet dataset. The exception is the choice of β, because the value of 2 is a compromise between lowering the error and inferring the registration of missing parts (Section 6.1.9). We highlight we could achieve even better results with a more advanced strategy to select the parameter values of our probabilistic refinement. Furthermore, we could benefit from using an iterative sweep strategy to select our neural network hyperparameters.

We trained all compared works for 25, 75, and 40 epochs, or until they stopped learning in the RMANet, Custom, and CoMA datasets. For the ModelNet40 dataset, our approach was trained for 165 epochs. Every training and evaluation used a batch size of 8. The training required 14 hours for the RMANet dataset, 16 hours for Custom, and 10 hours for CoMA. For the ModelNet40 dataset, training took 16 hours.

6.1.2 Dataset Details

We used three datasets for training and testing. The first one is the dataset proposed by RMANet. From it we selected 20,000 pairs of clouds for training and 2,000 pairs for evaluation. We compiled the second dataset (referred to as *Custom*) and it contains point clouds sampled from three other datasets: ModelNet40 (WU et al., 2015), TOSCA (BRONSTEIN; BRONSTEIN; KIMMEL, 2008) and HumanMotion (VLASIC et al., 2008). The first group has pairs generated from ModelNet40 point clouds deformed by the same self-training strategy described in Section 5.1.1 of the main paper. The parameter ρ was chosen randomly between $[5, 30]$. To generate the second group, we used the technique proposed by RMANet (FENG et al., 2021) over clouds from TOSCA. The last group of point clouds is from HumanMotion. They are distributed in the following way:

- Training: 4,000 pairs from ModelNet40 (first 32 categories), 2,400 pairs from TOSCA (cat and dog), and 2,400 pairs from HumanMotion (sequences *squat2*, *samba*, *handstand*, and *jumping*);
- Validation: 375 pairs from the ModelNet40, 250 from TOSCA, 250 from Human-

Motion, all using the same training set categories;

- Evaluation: 1100 pairs from ModelNet40 (last 8 categories), 475 from TOSCA (horse), and 500 from HumanMotion (march sequence).

The third dataset was compiled from the face meshes proposed by CoMA (RANJAN et al., 2018) joined with the already mentioned *Custom* dataset. CoMA generates face meshes using learned autoencoders. In our case, we sampled vertices from their training meshes. The meshes consist of 12 subjects doing 12 different poses each: moving lips, opening mouths, showing teeth, etc. Ten poses from 10 subjects were sampled to the training dataset, whereas the remaining two poses and subjects were used for evaluation. We highlight that the original meshes contain the entire head (Fig. 6.2a). However, we selected for training and validation only points relative to faces (Fig. 6.4a). On the other hand, for validation, we consider both options (referred to as *Head* and *Face* respectively. More details in Section 6.1.6). The final distribution of clouds was:

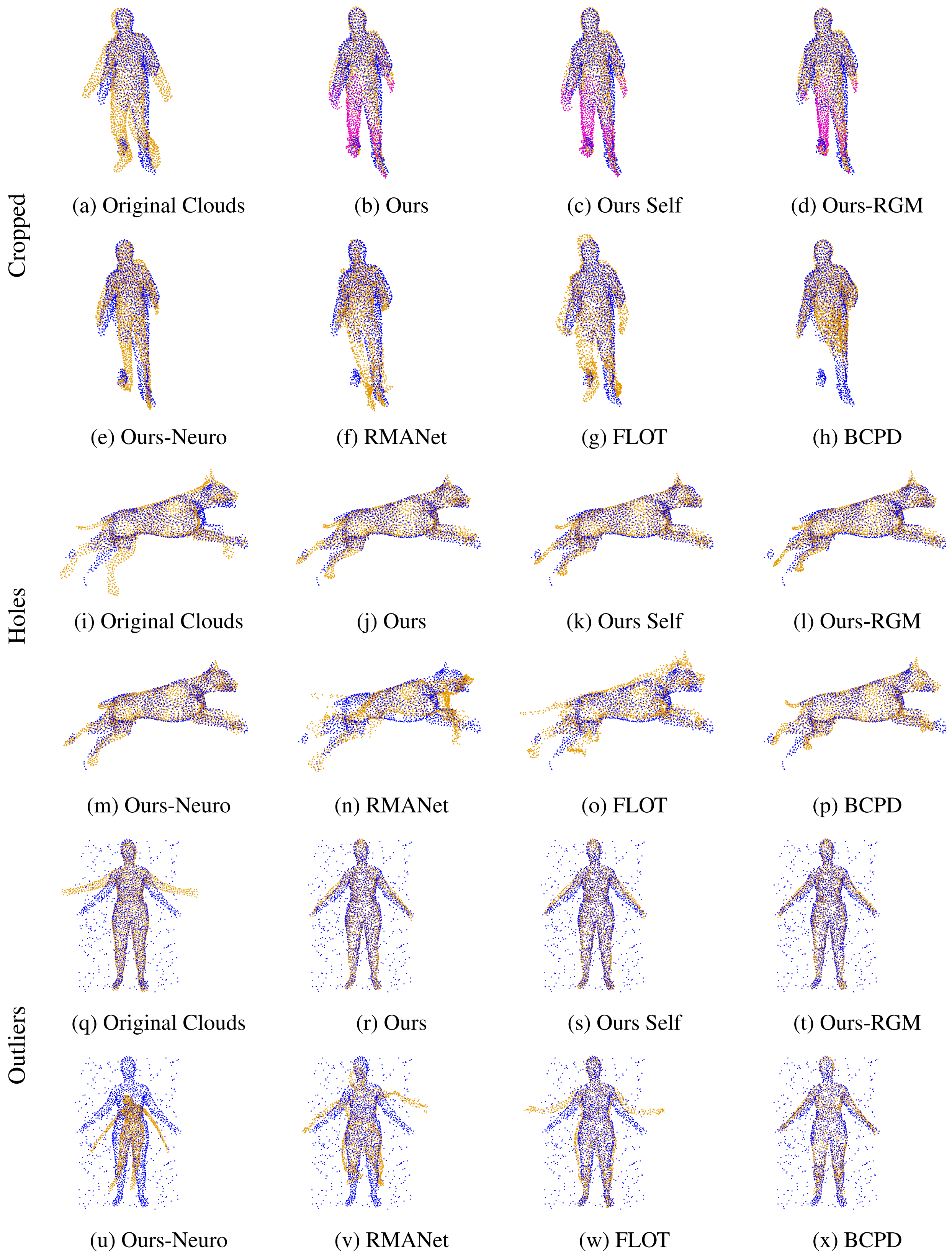
- Training: 8,000 pairs of clouds from the *Custom* dataset joined with 2,500 pairs of clouds selected across 10 subjects from CoMA, whereas for each subject we sampled from 10 of 12 possible poses.
- Validation: 300 pairs from the same training CoMA meshes joined with 800 clouds fro *Custom*.
- Evaluation: 2,000 random pairs from the previously unconsidered meshes.

6.1.3 Self-Supervised Learning and Training

An existing problem when training networks on non-rigid deformation is creating datasets. Rigid-transformation approaches can randomly sample rotations, translations, and crops for example. Learning-based non-rigid techniques, on the other hand, currently rely on non-supervised learning, using loss functions based on Chamfer and Earth-Moving distance for point clouds, or geodesic distance for meshes. However, these are unable to explicitly account for points with no correspondences. To address this issue, we trained our model adding noise to point clouds, and mainly using the artificial deformation method proposed by Hirose (HIROSE, 2020a) to generate dense correspondences, given by

$$X = X + \mathcal{V}\mathbf{\Lambda}\mathbf{J} | \mathbf{J} \in \mathcal{N}(0,1); \mathbf{\Lambda}, \mathcal{V} = eig(\rho^{-1}\mathbf{G}), \tag{6.1}$$

Figure 6.1: Results on *Cropped*, *Holes*, and *Outliers* types of point clouds on top of the RMANet dataset. Source cloud is in orange and target is in blue. Source points classified as having no matches are shown in magenta.



Source: The Authors

where $\mathbf{\Lambda}$ and $\mathcal{V}$ are respectively the eigenvalues and eigenvectors of the covariance matrix $\mathbf{G}$ (defined in Algorithm 2). ρ is a parameter controlling the deformation level, and $\mathbf{J}$ are 3D points sampled from the normal distribution. We refer to this dataset as *self-supervised.*

For RMANet and Custom datasets, we trained all learning based methods in a supervised fashion. The exception is RMANet on Custom dataset. There, we used its pre-trained model, as any further fine-tuning or training with the Custom dataset worsened its performance. We also trained our model (adapted SuperGlue network) with the self-learning strategy (*Ours-Self*). Ours, Ours-RGM, and FLOT were trained using *Cropped* (more challenging) point clouds. NeuroMorph could not learn in this setup, so it was trained with *Clean* point clouds. We trained our model using SGD and learning rate of 0.001 for 25 epochs (RMANet dataset) and for 60 epochs (Custom dataset).

For the CoMA dataset, we followed the same strategy of training with *Cropped* point clouds. The difference lies in fine-tuning the models trained on Custom with the self-supervised learning strategy applied to the CoMA point clouds. In this case, instead of *Ours-Self*, we have *Ours-GT* which refers to our model using the ground-truth pairs. Finally, the restrictions on NeuroMorph and RMANet are still valid. We trained our model using SGD and a learning rate of 0.0005 for 40 epochs.

6.1.4 Metrics

The main metric used to assess the registration quality of the evaluated algorithms is the end-to-end point error (EPE), which computes the mean Euclidean distance between the deformed points and their correspondences. For experiments involving *Cropped* point clouds, we argue that it is important to classify regions in the source cloud with no correspondences to target. Thus, we report Precision and Recall for the classification of existence of correspondence. EPE is reported for points with matches.

Precision and Recall of points classified as not having correspondences in Sec. 5.1 are given by

$$\begin{gathered}\text{Precision} = \frac{(\tilde{\nu} \circ \bar{\nu})\mathbf{1}_{M_r}}{\tilde{\nu}\mathbf{1}_{M_r}},\ \text{Recall} = \frac{(\tilde{\nu} \circ \bar{\nu})\mathbf{1}_{M_r}}{\bar{\nu}\mathbf{1}_{M_r}} \\ \tilde{\nu} = \neg\mathbf{S}\mathbf{1}_{N_c},\ \bar{\nu} = \neg\bar{\mathbf{P}}\mathbf{1}_{N_c},\end{gathered} \tag{6.2}$$

where $\circ$ is the Hadamard product, $\mathbf{1}_{Nc}$ is a column vector consisting of N 1s, and $\mathbf{1}_{Mr}$ is a row vector consisting of M 1s. $\mathbf{S}$ is the hard assignment matrix obtained from $\mathbf{P}$ after applying the Hungarian algorithm, similar to what is done for rigid registration (Sec. 3.2 of the main paper). $\bar{\mathbf{P}}$ is the ground truth assignment matrix, that is also used when training the model. The part of the numerator shared by both metrics, $(\tilde{\nu} \circ \bar{\nu})\mathbf{1}_{M_r}$, is the number of source points correctly predicted. The denominators are the number of predicted points without any matches, used in the computation of Precision, and its ground truth value, used for computing the Recall.

Table 6.1: Performance comparison using EPE as well as Precision and Recall metrics for several methods on different datasets. Columns refer to the RMANet dataset with the point cloud types described in Section 6.1. *BCPD* indicates the errors after BCPD post-processing. *Cropped* variations also report Precision and Recall when classifying if source points have no correspondences. (✗) means the technique did not handle the given dataset, (-) means the comparison is not applicable. The best results are shown in **bold**.

Method	Clean		Outliers	Holes	Cropped		
	P/D	BCPD				Prec.	Recall
Ours	0.020	**0.020**	**0.023**	**0.028**	**0.018**	**0.98**	**0.98**
Ours-Self	0.026	**0.020**	0.026	**0.029**	0.020	0.93	**0.98**
Ours-RGM	0.027	**0.021**	0.027	0.033	0.021	0.92	0.94
Ours-Neuro	**0.008**	0.023	0.160	0.042	0.140	0.46	0.22
RMANet	0.012	-	0.101	0.149	✗	✗	✗
FLOT	0.043	-	0.053	0.042	0.045	-	-
BCPD	-	0.042	0.045	0.057	✗	✗	✗

Source: The Authors

6.1.5 Evaluation on RMANet and Custom datasets

Figure 6.1 compares several techniques considering point clouds of type *Cropped* (rows 1 and 2), *Holes* (rows 3 and 4) and *Outliers* (rows 5 and 6). For each type, the source and target (Original) point clouds are shown on the top left. These are followed by the results produced by: our method (*Ours*), our method trained using self learning (*Ours-Self*), Ours-RGM, Ours-Neuro, RMANet, FLOT, and BCPD.

For the *Cropped* example, our method and its variants are able to map source points without correspondences in the target cloud to plausible locations. FLOT produces a reasonable result, although one observes misalignment in the head, arms and legs. RMANet does a good job for the upper body, but the legs are off. BCPD was unable to map points without correspondences.

Table 6.2: Performance comparison using EPE as well as Precision and Recall metrics for several methods on different datasets. Columns refer to the Custom dataset with the point cloud types described in Section 6.1. *BCPD* indicates the errors after BCPD post-processing. *Cropped* variations also report Precision and Recall when classifying if source points have no correspondences. (✗) means the technique did not handle the given dataset, (-) means the comparison is not applicable. The best results are shown in **bold**.

Method	Clean		Outliers	Holes	Cropped		
	P/D	BCPD				Prec.	Recall
Ours	0.060	**0.036**	0.060	0.078	0.040	**0.90**	0.92
Ours-Self	0.052	**0.035**	0.069	0.058	**0.037**	**0.90**	**0.94**
Ours-RGM	0.084	0.047	0.092	0.076	0.073	0.84	0.85
Ours-Neuro	**0.019**	0.038	0.147	0.064	0.206	0.52	0.32
RMANet	0.131	-	0.201	0.222	✗	✗	✗
FLOT	0.111	-	0.141	0.102	0.110	-	-
BCPD	-	0.159	0.130	0.180	✗	✗	✗

Source: The Authors

For the *Holes* example, RMANet, followed by FLOT, are the ones for which misalignment is most noticeable. The *Outlier* test case also illustrates the effectiveness of our method and its variants Ours-Self and Ours-RGM. Ours-Neuro did not produce a satisfactory result, indicating that the $\mathbf{P}$ matrix generated by NeuroMorph does define reliable correspondences in the presence of noise. Both RMANet and FLOT produced unsatisfactory results, while BCPD achieved good registration. These results highlight the fact that our method is consistently robust across a range of challenging scenarios. It also shows that the $\mathbf{P}$ matrix generated by our model leads to better results than RGM's (see Fig. 6.1l) and NeuroMorph's (see Fig. 6.1u). Moreover, it shows that although BCPD nicely complements our model, it cannot, just by itself, handle difficult cases involving large missing regions (Fig. 6.1h). We report addition qualitative results in Appendix A.

Tables 6.1 and 6.2 summarizes the results comparing the different approaches with *Clean*, *Cropped*, *Holes*, and *Outlier* configurations in the two datasets. Our method (our learning model with the BCPD refinement) yields the best general results when considering all point cloud types. The use of self-learning leads to comparable results with respect to supervised learning, especially in the Custom dataset. For Clean point clouds, our implementation of NeuroMorph obtained the best scores. However, NeuroMorph cannot handle missing parts, and could not be trained with such datasets.

The second closest to ours is Ours-RGM followed by BCPD. BCPD obtained scores close to ours in the RMANet dataset, but could not handle *Cropped* clouds. We also highlight how the BCPD-based probabilitic refinement step improves the results of

Table 6.3: End-to-End Point Error metric computed for multiple scenarios (*Cropped*/*Outliers*/*Holes*) and different ratios, obtained using our model on the RMANet dataset.

	Method	Cropped		Outliers	Holes
		BCPD	Prec./Recall	BCPD	BCPD
Crop 90% Outlier 20% Holes 25%	Ours	0.021	0.90/0.94	0.023	0.028
	Ours Self	0.022	0.84/0.95	0.027	0.029
	Ours RGM	0.029	0.84/0.84	0.027	0.033
Crop 70% Outlier 40% Holes 40%	Ours	0.018	0.98/0.98	0.028	0.031
	Ours Self	0.020	0.93/0.98	0.036	0.037
	Ours RGM	0.021	0.92/0.94	0.043	0.041
Crop 50% Outlier 60% Holes 50%	Ours	0.038	0.94/0.84	0.039	0.044
	Ours Self	0.059	0.91/0.70	0.050	0.058
	Ours RGM	0.182	0.69/0.92	0.080	0.055

Source: The Authors

ours and RGM's network, as can be observed in the *Clean* column under the *Custom* dataset (Table 6.2). Finally, on *Outliers* and *Holes* in the same table, we notice that even with the proposed network, the precision is significantly affected. We attribute this to the more challenging scenarios compared to Table 6.1. Improving precision is explored at Section 6.1.10 by training with different noise types.

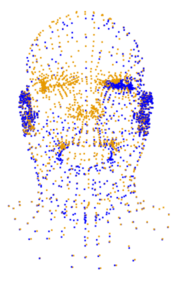
Table 6.3 report results of our model with supervised and self-supervised training, as well as with supervised RGM for various ratios of cropped/outliers/holes. The largest difference in performance as expected happen when moving to the more challenging scenarios (last partition of the table). In general, our approach metrics holds better compared to supervised RGM, which deteriorates considerably, even in the self-supervised case.

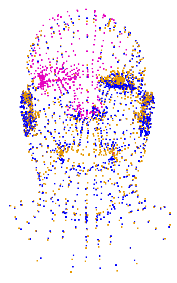
6.1.6 Evaluation on CoMA and TOSCA datasets

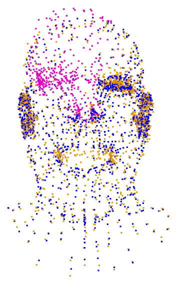
Figures 6.2 and 6.3 compare results among different techniques similarly to Fig. 6.1. The main difference is that all works fine-tuned their models (originally trained on the Custom dataset) on the CoMA dataset with self-supervised training. The exception is Ours-GT, which uses supervised training. The clouds used from the CoMA dataset were sampled from the entire original mesh which included not only the face but the entire head. For the *Cropped* examples, Ours, Ours-GT, and Ours-RGM correctly classified the missing parts. While Ours-GT had issues keeping the head shape of the cropped region, Ours-RGM misclassified some points around the mouth. Even though the three best

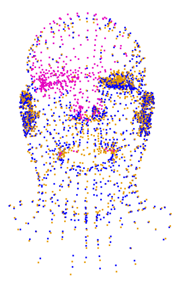
Figure 6.2: Results on *Cropped*, *Holes*, and *Outlier* types of point clouds on top of the CoMA *Head* dataset. Source cloud is in orange and target is in blue. Source points classified as having no matches are shown in magenta.

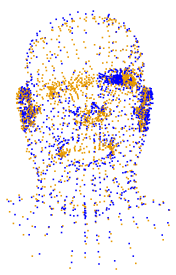
Cropped

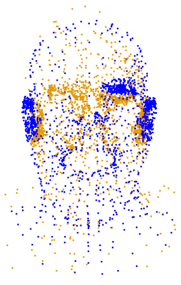

(a) Original Clouds

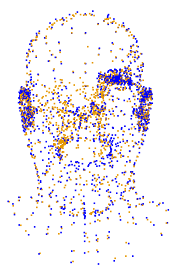

(b) Ours

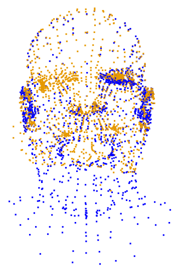

(c) Ours GT


(d) Ours-RGM

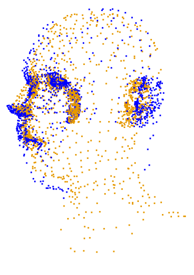

(e) Ours-Neuro

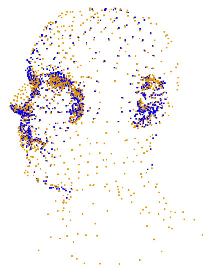

(f) RMANet

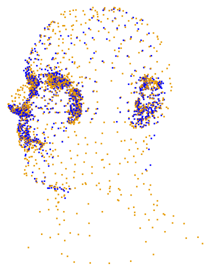

(g) FLOT

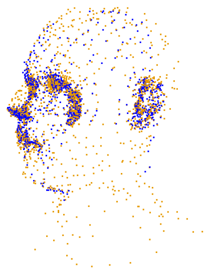

(h) BCPD

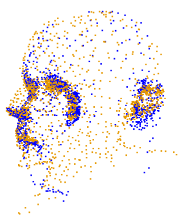
Holes

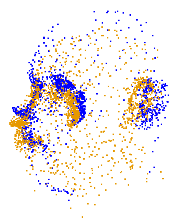

(i) Original Clouds

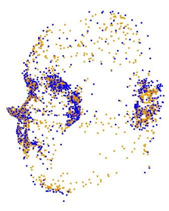

(j) Ours

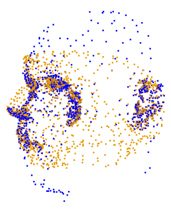

(k) Ours GT


(l) Ours-RGM


(m) Ours-Neuro


(n) RMANet

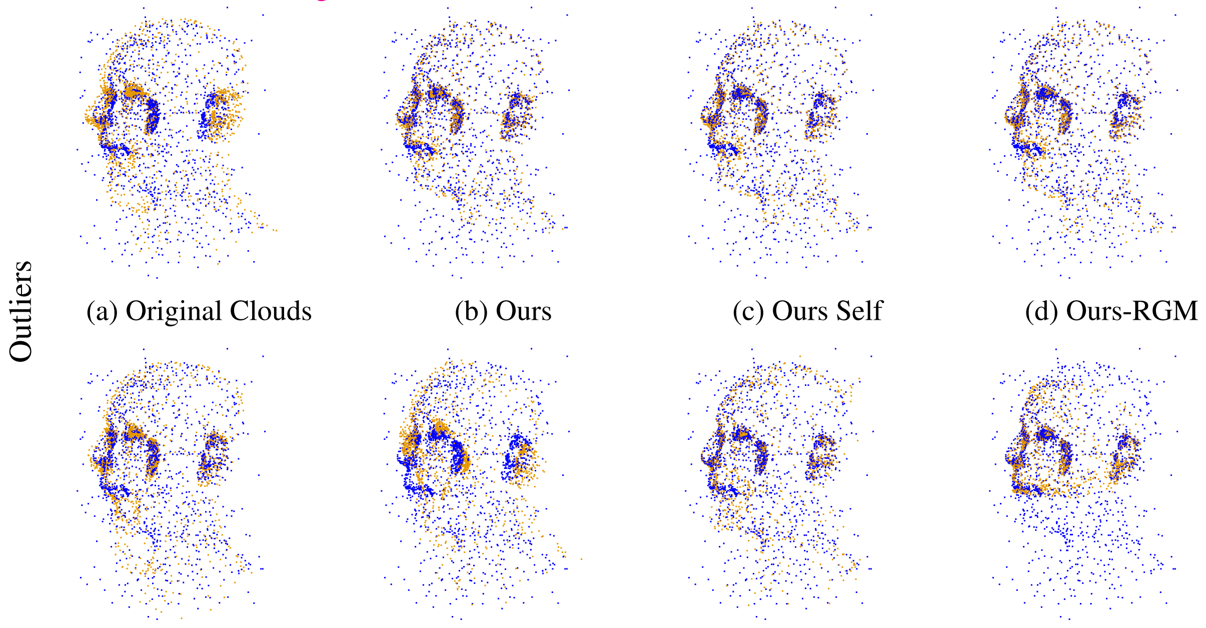

(o) FLOT


(p) BCPD

Source: The Authors

works did not fail dramatically as the others, none of them were able to perfectly deform the mouth. For the *Holes* case, the same three approaches had the best results. FLOT was also able to register, but it deformed the source point cloud to produce similar holes to the found in the target. The *Outliers* tests show the importance of the probabilistic module together with dustbins (Section 4.3), as the three approaches with the best results used the module (Ours, Ours-GT, and Ours-RGM). FLOT again was able to deal with most of the outliers, however, the registration still had artifacts. In general RMANet, Ours-Neuro and BCPD failed in all these examples.

Figure 6.3: Results on the *Outlier* type of point clouds on top of the CoMA *Head* dataset. Source cloud is in orange and target is in blue. Source points classified as having no matches are shown in magenta.



(a) Original Clouds (b) Ours (c) Ours Self (d) Ours-RGM

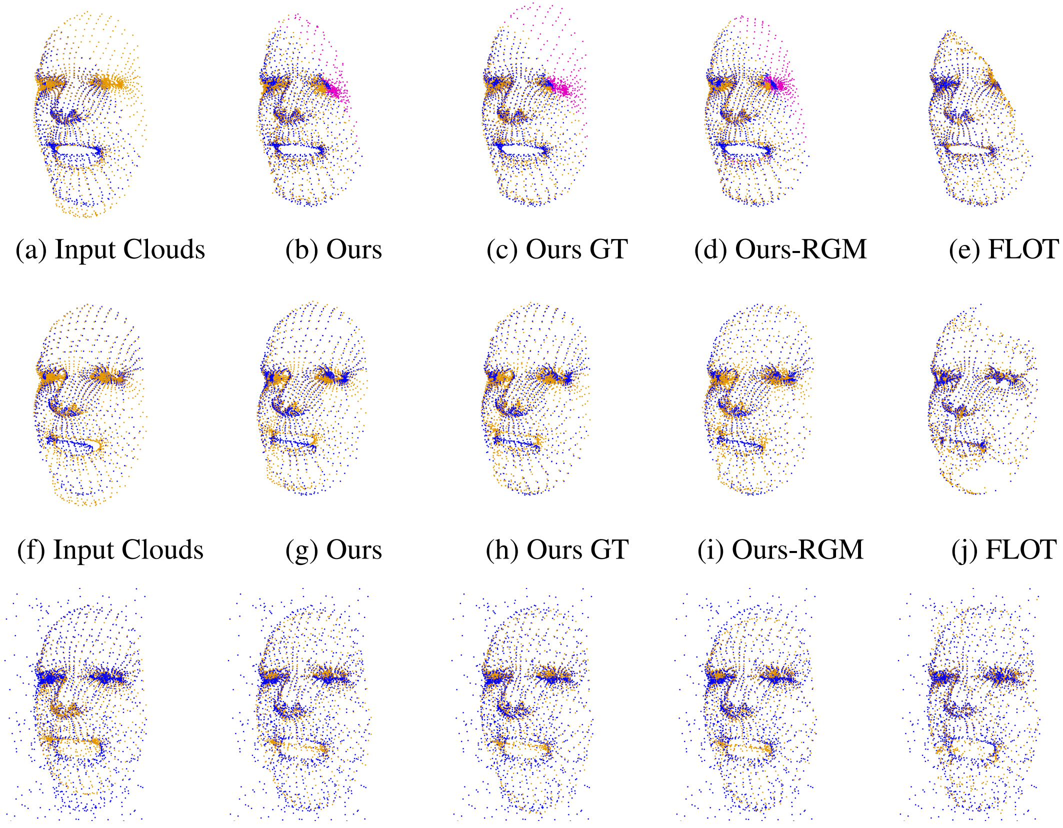
(e) Ours-Neuro (f) RMANet (g) FLOT (h) BCPD

Source: The Authors

Table 6.4 summarizes the results comparing the different approaches with the scenarios similar to Tables 6.1 and 6.2. The two variations of Our method, trained with self-supervised and supervised learning, created the best results in general. For *Clean* point clouds, NeuroMorph had the best results. However, as shown by the other results, it could not handle missing regions or outliers. *Outliers* had the largest difference when using ground-truth instead of self-supervised training. In this case, the improvement with the supervision was noticeable and Ours-RGM had similar results to Ours.

Figure 6.4 present qualitative results comparing different approaches with *Cropped*, Holes, and *Outlier*. Differently from Fig. 6.2, we sample points from only the face region and we did not include BCPD, Ours-Neuro, and RMANet because they had the worst

Figure 6.4: Results on *Cropped*, *Holes*, and Outlier types of point clouds on top of the CoMA *Face* dataset. Source cloud is in orange and target is in blue. Source points classified as having no matches are shown in magenta.



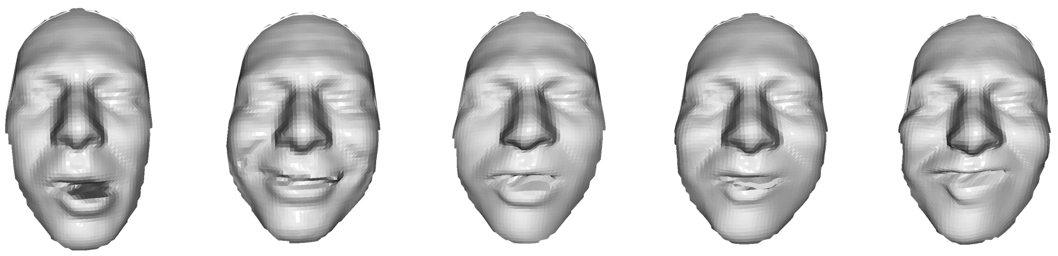
(a) Input Clouds (b) Ours (c) Ours GT (d) Ours-RGM (e) FLOT

(f) Input Clouds (g) Ours (h) Ours GT (i) Ours-RGM (j) FLOT

(k) Input Clouds (l) Ours (m) Ours GT (n) Ours-RGM (o) FLOT

Source: The Authors

Figure 6.5: Registering facial expressions using source (a) and target (b) point clouds. Registration results produced by our model trained on: the Custom dataset (c), fine tuning the weights on faces (d), and fine-tuning on faces plus some clouds from Custom (e). The latter keeps the original performance while better dealing with faces.



(a) Source (b) Target (c) Pre-trained (d) Only-Faces (e) Mixed

Source: The Authors

Table 6.4: Performance comparison using EPE as well as Precision and Recall metrics for several methods on different datasets. Columns refer to the CoMA *Head* dataset with the point cloud types described in Section 6.1. *BCPD* indicates the errors after BCPD post-processing. *Cropped* variations also report Precision and Recall when classifying if source points have no correspondences. (✗) means the technique did not handle the given dataset, (-) means the comparison is not applicable. The best results are shown in **bold**.

Method	Clean		Outliers	Holes	Cropped		
	P/D	BCPD				Prec.	Recall
Ours	0.029	**0.013**	0.019	**0.029**	**0.014**	**0.93**	**0.97**
Ours-GT	0.028	**0.014**	**0.016**	**0.028**	**0.014**	**0.94**	**0.97**
Ours-RGM	0.028	0.018	0.018	0.038	0.017	0.91	0.94
Ours-Neuro	**0.003**	0.020	0.035	0.085	0.049	0.50	0.59
RMANet	0.040	-	0.115	0.160	✗	✗	✗
FLOT	0.009	-	0.031	0.137	0.032	-	-
BCPD	-	0.043	0.038	0.061	✗	✗	✗

Source: The Authors

results in the previous comparisons. For the *Cropped* scenario, Ours, Ours-GT and Ours-RGM were able to correctly classify the missing region, but only Ours-GT preserved the shape of it. In addition, as it happened to the *Head* cases, points around the mouth were wrongly classified by RGM. In terms of registration for the points with correspondences, FLOT presented the best results by correctly deforming the lower lip.

For *Holes*, Ours and Ours-RGM could deal with the small missing parts and deformed the jaw. However, they failed to close the mouth. This is a typical issue in scenarios with a change in the topology of the original mesh and it is an open challenge for our work. FLOT (Fig. 6.4j) was able to close the mouth, but, it could not keep the lip shapes and created holes in the source cloud to exactly match the target.

Finally, in terms of *Outliers*, FLOT had issues because it deformed the points toward the outliers. RGM, on the other hand, translated the face down in order to fit the bottom part which took the forehead away from the target. The best results were from Ours and Ours-GT, although they failed to keep the upper lip position.

As in Table 6.4, we compared different approaches in Table 6.5 but only focusing on the face region. Differently from considering the entire head, Ours-RGM with self-supervised training yielded the best results alongside Ours-GT. Since the original training dataset for all works only considered the face region, Ours with self supervised started to overfit sooner than RGM.

Overfitting does not result from our proposed self-supervision but from the interaction between the network and the CoMA dataset. This conclusion follows from the following observations: first, RGM got results on par with Ours-GT while using the self-supervising training. In addition, we obtained results on par with Ours-GT by training our neural network from scratch with self-supervision. For example, our registrations had an EPE of 0.018 on *Outliers*, and of 0.015 on *Cropped* with Precision/Recall reaching 0.84/0.90. These lower errors show that the overfitting comes from the network, the dataset, and the training strategy. Finally, this problem alongside the issues regarding varying topology (Fig. 6.4h) and fine-grained details (Fig. 6.4b) have the most space for improvements.

Table 6.5: Performance comparison using EPE as well as Precision and Recall metrics for several methods on different datasets. Columns refer to the CoMA *Face* dataset with the point cloud types described in Section 6.1. *BCPD* indicates the errors after BCPD post-processing. *Cropped* variations also report Precision and Recall when classifying if source points have no correspondences. (✗) means the technique did not handle the given dataset, (-) means the comparison is not applicable. The best results are shown in **bold**.

Method	Clean		Outliers	Holes	Cropped		
	P/D	BCPD				Prec.	Recall
Ours	0.030	**0.015**	0.021	0.022	0.017	0.78	0.85
Ours-GT	0.023	**0.015**	**0.018**	**0.018**	0.015	**0.90**	**0.94**
Ours-RGM	0.032	**0.015**	**0.018**	**0.018**	0.015	0.80	0.92
Ours-Neuro	**0.005**	0.020	0.026	0.020	0.036	0.40	0.52
FLOT	0.009	-	**0.019**	0.088	**0.012**	-	-

Source: The Authors

Fig. 6.5 shows registration results on a real dataset of different facial expressions from TOSCA (BRONSTEIN; BRONSTEIN; KIMMEL, 2007). Starting from only 7 pairs of scanned faces, we generated 1,000 face pairs using self-supervised learning. In Fig. 6.5c we used the model pre-trained with the Custom dataset, which can handle the registration but misses details mostly in the mouth and around eyes. We attribute this to the lack of instances of faces (or similar) in the dataset. To improve the results we fine-tuned our model using the *Face* dataset (Fig. 6.5d). We also fine-tuned it using the *Face* dataset plus some clouds from the *Custom* dataset (Fig. 6.5e). Although they produce similar results, the latter maintains its performance on the Custom dataset. This shows how the proposed model can be fine-tuned without becoming too specialized on the new dataset.

Table 6.6: Sensitivity and ablation study using setups trained with the self-learning and evaluated with *Clean* and *Cropped* clouds from the RMANet dataset. The evaluation of the Loss used the Custom dataset.

Method	Clean	Cropped		Time(s)
	BCPD	BCPD	Prec./Recall	1K/8K
Ours L=4,O=20	0.020	0.020	0.93/0.98	0.053/0.395
Ours L=4,O=5	0.020	0.020	0.88/0.96	0.045/0.344
Ours L=4,O=35	0.020	0.020	0.94/0.98	0.060/0.448
Ours L=0,O=20	0.020	0.063	0.78/0.52	0.027/0.180
Ours L=2,O=20	0.020	0.020	0.92/0.97	0.036/0.331
Ours L=6,O=20	0.019	0.019	0.94/0.98	0.067/0.478
SuperGlue OT	0.020	0.035	0.90/0.79	0.044/0.305
Linear (SUN et al., 2021)	0.020	0.024	0.78/0.83	0.043/0.323
Point Transformer (ZHAO et al., 2021) [a]	0.031	0.046	0.84/0.79	0.075/0.483
Our Loss [a]	0.035	0.037	0.90/0.94	-/- [b]
RGM Loss [a]	0.040	0.059	0.84/0.90	-/- [b]

[a] Custom Dataset.
[b] Same architecture and time of Ours L=4,O=20.

Source: The Authors

6.1.7 Sensitivity and Ablation Study

We report results with different configurations of the proposed model in Table 6.6. We show the contributions of each module plus the effect of using different approaches in these modules. The first group of results show how the number of Sinkhorn iterations (O) and attention layers (L) affect error, matching classification, and runtime. Increasing their values directly affect runtime, but beyond the used values not as much the registration quality. We also stress how the attention layers are vital for *Cropped*, as all variations of L and O achieved similar precision on *Clean*. Replacing the SuperGlue attention module with a linear one reduces performance in our application.

Although the SuperGlue Sinkhorn algorithm (SuperGlue OT) and the linear attention module (SUN et al., 2021) produce impressive results in 2D images, they did not perform so well for our application. We also change the feature embedding layers from DGCNN to Point Transformer (ZHAO et al., 2021), although it performs well in the *Clean* scenario, it has issues with partial clouds and it has a worse evaluation runtime. The last group of experiments shows that our loss function outperforms RGM cross-entropy loss.

Table 6.7: Comparison of different non-rigid registration techniques trained on *Cropped* clouds from the Custom dataset. The evaluations use the same parameters as the main paper's experiments. Last column reports times for clouds with 1,000 and 8,000 points.

Method	Clean	Cropped		Time(s)
	BCPD	BCPD	Prec./Recall	1K/8K
Ours	0.036	0.040	0.90/0.92	0.054/0.336
Ours *Cycle*	0.034	0.031	0.87/0.89	0.092/0.621
Ours *Loop*	0.024	0.031	0.88/0.91	0.092/0.834
Ours Self	0.035	0.037	0.90/0.94	0.054/0.336
Ours Self *Cycle*	0.035	0.033	0.89/0.92	0.092/0.621
Ours Self *Loop*	0.024	0.029	0.91/0.94	0.092/0.834
Ours-RGM	0.047	0.073	0.84/0.85	0.111/0.637
Ours-RGM *Cycle*	0.038	0.038	0.82/0.83	0.208/1.210
Ours-RGM *Loop*	0.025	0.054	0.83/0.86	0.263/2.316

Source: The Authors

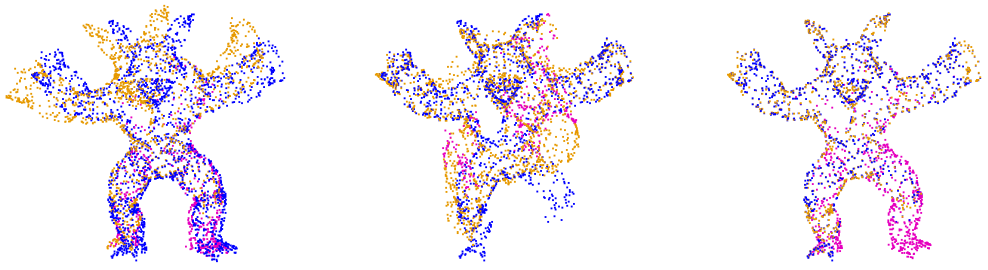
6.1.8 Non-Rigid Registration with Multiple Iterations and Cyclic-Consistency

Table 6.7 reports results comparing our approach with and without self-learning, and Ours-RGM using cyclic consistency and loop over neural network estimations. The models were trained and evaluated on the Custom Dataset. *Loop* refers to using $outter\_max$ $= 10$ as mentioned at Algorithm 1, resulting in estimating $\mathbf{P}$ at each outer iteration. Cyclic consistency refers to having in the final soft assignment matrix only correspondences valid on matching source to target and vice-versa. This is achieved by the following expression $\mathbf{P}_{final} = \mathbf{P}_{front}\mathbf{P}_{back}^{T}$. We note in the original paper this was already used by rigid registration at the end of the second iteration. In this non-rigid case, we use a single iteration to produce $\mathbf{P}_{front}$ and $\mathbf{P}_{back}^{T}$. We refer to this variant by *Cycle* in our experiments.

Both *Loop* and *Cycle* improve EPE in all cases with our approach consistently outperforming Ours-RGM. *Loop* is most effective with *Clean* point clouds, when the correspondence estimation is easier given there are no partial clouds, or points without correspondences. Therefore, the algorithm can converge to a correct and more precise result without accumulating errors during the process. *Cycle*, on the other hand, benefits dealing with partial inputs. This is the result of cyclic consistency improving wrong estimated correspondences created by the partial clouds.

A downside of both *Loop* and *Cycle* is a worse classification of points without correspondences in *Cropped* point clouds. We attribute this to the fact that for *Cropped* point clouds the algorithm may accumulate matching errors across multiple calls to the model.

Figure 6.6: Examples of improvement given by cyclic consistency (*Cycle*) for non-rigid registration of the Stanford Armadillo point clouds. The source cloud is shown in orange while target in blue. Points classified as having no matches are shown in magenta.



(a) Original (b) Ours Self (c) Ours Self *Cycle*

Source: The Authors

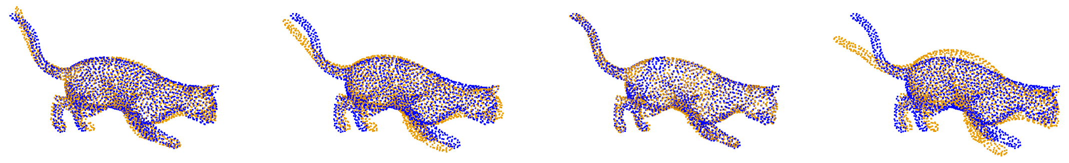
Even with *Cycle*, which improves EPE, Precision and Recall worsen. The improvements are due to the use of *Cycle* reducing the number of false-positive matches (points that should not have correspondences, but received a match nonetheless). These false positives tend to create artifacts as the regions around these points will be deformed incorrectly. However, *Cycle* still accumulates false negatives (points that should have a correspondence, but none were found). For example, if a point p in $\mathbf{P}_{front}$ was assigned no correspondence, the resulting $\mathbf{P}_{final}$ for this point will have lower matching probabilities even if $\mathbf{P}_{back}^{T}$ correctly finds a match to p.

Figure 6.6 presents an example of improvements given by *Cycle*. In this case, Ours-Self cannot correctly register the cropped armadillo with one pass, but by using an additional pass (*Cycle*), our method offers a better registration at a faster time compared to Ours-RGM using a single pass (Table 6.7).

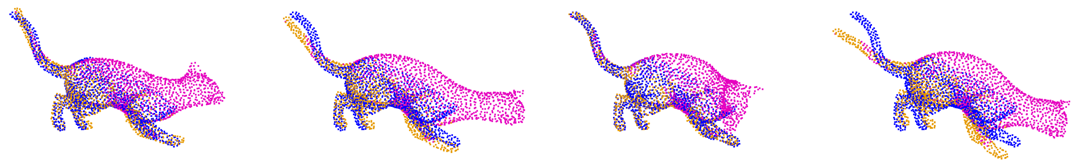
6.1.9 Effects of β and λ on Non-Rigid Registration

Parameters β and λ are from BCPD and control, respectively, the variation of deformation among neighbor points, and the overall intensity of the deformation. Figure 6.7 shows the effects on registration of varying these parameter values, whereas Figure 1 of the main paper has the results obtained with the default values ($\beta = 2$ and $\lambda = 2$). Larger values of β or λ lead to less deformation. However, lowering the value of β or λ makes the registration more sensitive to outliers and partial clouds.

Figure 6.7: Non-rigid registration of *Clean* (top) and *Cropped* (bottom) clouds with our self-learning model for several values of the β and λ parameters. The input point clouds are shown in Figure A.1 (a) and Figure A.2 (a), respectively. Results with the default parameter values ($\beta = 2$ and $\lambda = 2$) are shown in Figure 1 of the main paper. The source cloud is shown in orange while target in blue. Points classified as having no matches are shown in magenta.



(a) λ=0.5, β=2 (b) λ=200, β=2 (c) λ=2,$\beta = 0.5$ (d) λ=2,β=20



(e) λ=0.5, β=2 (f) λ=200, β=2 (g) λ=2,$\beta = 0.5$ (h) λ=2,β=20

Source: The Authors

6.1.10 Training on Mixed Noises

For this experiment, we reimplemented our neural network to support training with batches made of point clouds of different number of points. Thus, the performance of the model trained and evaluated on 1,024 points differs from the previous experiments. This experiment was motivated by the quantitative results with the Custom dataset in Table 6.2 and we report the same metrics in Table 6.8. In them, there is a significant drop of performance on *Outliers* and *Holes*. To solve this problem, during training, for each pair of clouds, we randomly selected among *Outliers*, *Holes*, and *Cropped* (we refer to this strategy by *Mixed*). The first three rows refer to training and evaluating on the Custom dataset, whereas the last rows were trained (where indicated) and evaluated with the CoMA dataset, similar to Table 6.4. In all cases, we used self-supervised learning with the first training step taking 60 epochs and the remaining taking 40 epochs.

Training from scratch on the Custom dataset with *Mixed* resulted in improved results for *Outlier* and Holes. However, it compromised the *Cropped* clouds. This is solved by fine-tuning the model trained on *Cropped* using *Mixed*, achieving then the best of both. For the CoMA dataset case, the improvements were not as noticeable for outliers and holes by having mixed noises during training, although, the results still improved. It is important to notice that having only *Mixed* on CoMA degrades the results. This shows

Table 6.8: Performance comparison using EPE as well as Precision and Recall metrics for several methods on different datasets. Columns refer to the Custom dataset with the point cloud types described in Section 6.1. *BCPD* indicates the errors after BCPD post-processing. *Cropped* variations also report Precision and Recall when classifying if source points have no correspondences. (✗) means the technique did not handle the given dataset, (-) means the comparison is not applicable. The best results are shown in **bold**.

Method	Clean		Outliers	Holes	Cropped		
	P/D	BCPD				Prec.	Recall
Cropped	0.064	0.029	0.061	0.053	0.034	0.90	**0.95**
Mixed	0.042	**0.025**	**0.031**	0.043	0.065	0.76	0.77
Cropped + Mixed	0.066	**0.025**	**0.032**	**0.040**	**0.031**	**0.93**	**0.95**
Cropped + Cropped (CoMA)	0.028	**0.011**	0.019	0.025	**0.13**	**0.93**	**0.97**
Cropped + Mixed +Mixed (CoMA)	0.027	0.015	0.017	0.048	0.023	0.82	0.75
Cropped + Cropped (CoMA) +Mixed (CoMA)	0.027	0.013	**0.015**	**0.023**	**0.013**	**0.94**	**0.97**

Source: The Authors

the same sequence of *Cropped* + *Mixed* is the best approach for better generalization.

6.1.11 Generalization to Different Number of Points

Table 6.9: Performance comparison using EPE and Precision and Recall metrics for our proposed technique. The network was trained with self-supervision on Custom and fine-tuned on COMA *Head* dataset, and evaluated on *Cropped* clouds with different number of points. *EPE 3D* indicates the error for points with correspondence, and Precision and Recall refers to classifying if source points have no correspondences. The best results for each number of points are shown in **bold**.

Method	Trained on 512		Trained on 1024		Trained on 2048	
	EPE 3D	Prec./Recall	EPE 3D	Prec./Recall	EPE 3D	Prec./Recall
256 pts	**0.015**	**0.92/0.96**	0.055	0.81/0.69	0.086	0.55/0.65
512 pts	**0.013**	**0.92/0.97**	0.023	**0.91**/0.92	0.035	0.80/0.83
1,024 pts	**0.014**	**0.92/0.96**	**0.015**	**0.92/0.96**	0.019	0.86/0.88
2,048 pts	**0.015**	**0.89/0.95**	**0.015**	**0.90/0.96**	0.017	0.88/0.93
4,096 pts	**0.017**	0.79/0.81	**0.018**	0.80/0.90	0.019	**0.84/0.92**

Source: The Authors

Tables 6.9 and 6.10 report results comparing our proposed model trained and evaluated on the same setup of Table 6.4, with the CoMA dataset and self-supervised training. In these cases, we trained and evaluated on a variable number of points and the tables show

Table 6.10: Performance comparison using EPE and Precision and Recall metrics for our proposed technique. The network was trained with self-supervision on Custom and fine-tuned on COMA *Head* dataset, and evaluated on *Outlier* and *Holes* clouds with different number of points. The best results for each number of points are shown in **bold**.

Method	Trained on 512		Trained on 1024		Trained on 2048	
	Outlier	Holes	Outlier	Holes	Outlier	Holes
256 pts	**0.021**	**0.032**	0.046	0.046	0.033	0.042
512 pts	**0.019**	**0.028**	0.022	**0.027**	0.026	0.029
1,024 pts	0.020	0.028	**0.017**	**0.026**	0.019	0.031
2,048 pts	0.024	**0.028**	**0.020**	**0.028**	**0.019**	0.032
4,096 pts	0.034	0.038	0.025	0.038	**0.021**	**0.034**

Source: The Authors

how this variation affects the performance when registering point clouds.

For 256 and 512 points, the model trained with 512 points dominates over the others, showing that applying models to less points indeed negatively affects the results. It also indicates that training with more points leads the network to become more dependent on the extra context that larger point clouds give. Following this, it would be expected that for upsampling, the model trained with 512 points would have results on par to 2,048 points, since it learned to rely on less. However, we notice that training with less points result in a model less sensitive to small details on clouds. This can be seen by the lower EPE on models trained with more points, on *Outliers* and *Holes* starting at 2,048 and continuing at 4,096.

However, for *Cropped* the model with better EPE on larger clouds was trained on 512 points, even though for 4,096 the classification of points with no matches has better results on the model trained with 2,048. This might be counterintuitive as the model trained on 512 has worse precision and recall metrics on points with correspondences (0.77/0.76 against 0.82/0.80). However, it is worth mentioning that since we enforce motion coherence, an entire neighborhood can be wrongly deformed by false positives with high confidence. We attribute the best precision of the first model to this.

6.2 Rigid Registration

For the rigid registration experiments, we used the ModelNet40 (WU et al., 2015) dataset which consists of 12,311 point clouds from 40 categories of CAD models. From each cloud, we used 1,024 points to train the models on half of the categories while evaluating them on the other half. This follows the testing strategy used by the compared works RPMNet (YEW; LEE, 2020), RGM (FU et al., 2021), and Predator (HUANG et al., 2021). All clouds were randomly rotated by up to 45°around all axes and also translated by some random amount in the interval [-0.5, 0.5] along all three axes during training and evaluation. The set of point clouds with these random transformations is called *Clean*. The experiments also include a set of *Jitter* point clouds, obtained by adding random Gaussian noise sampled from $\mathcal{N}$(0,0.01) and clipped to [-0.05, 0.05] to *Clean* point clouds. The last set of point clouds, *Crop-Noise*, builds on *Crop-Noise* by keeping only a percentage of the source and target points. All works were trained with the most challenging scenario *Crop-Noise* (keeping 70% of the samples). This evaluates how they generalize to *Clean* and *Jitter* as well. We compare our results against three state-of-the-art methods: Predator, RGM, and RPM-Net. We do not further compare to other methods as these ones have shown to outperform the others. We use SGD to train our network with a 0.001 learning rate.

6.2.1 Metrics

To evaluate rigid registration we adopt the mean isotropic error (MIE) (YEW; LEE, 2020) and the mean anisotropic error (MAE) (WANG; SOLOMON, 2019). These are given respectively by

$$\text{MIE}(\mathbf{R}) = \angle(\mathbf{R}_{GT}^{-1}\mathbf{R}),\ \text{MIE}(\mathbf{t}) = ||\mathbf{t}_{GT} - \mathbf{t}||, \tag{6.3}$$

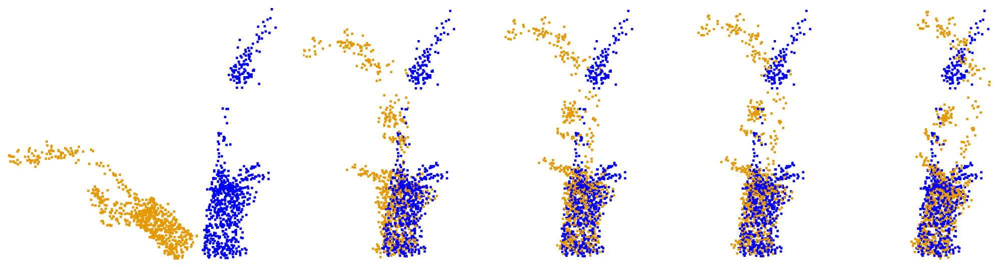
$$\text{MAE}(\mathbf{R}) = \frac{1}{3}\sum |\mathbf{R}_{GT}^{eul} - \mathbf{R}^{eul}|,\ \text{MAE}(\mathbf{t}) = \frac{1}{3}\sum |\mathbf{t}_{GT} - \mathbf{t}|, \tag{6.4}$$

where the MIE$(\mathbf{R})$ and MAE$(\mathbf{R})$ are in degrees, and $\angle(\mathbf{R}) = arccos\left(\frac{tr(\mathbf{R})-1}{2}\right)$. R is the estimated rotation matrix, t is the estimated translation vector, and $\mathbf{R}^{eul}$ is R expressed as a triple of Euler angles. $\mathbf{R}_{GT}, \mathbf{t}_{GT}, \mathbf{R}_{GT}^{eul}$ are their respective ground truths values.

6.2.2 Evaluation on ModelNet40 dataset

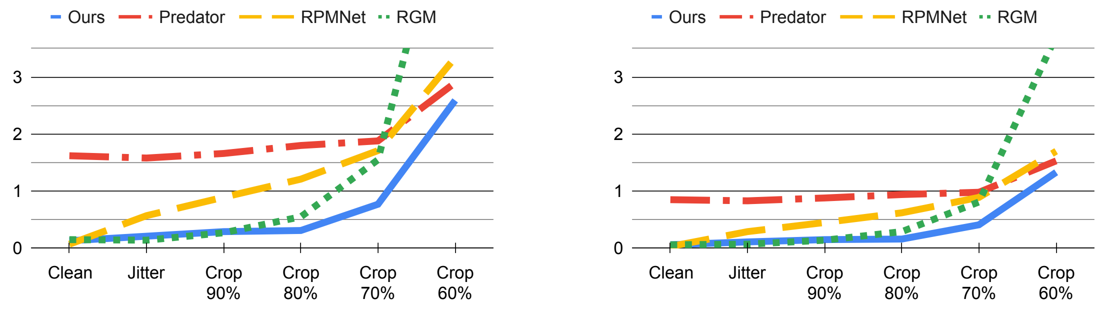
Figure 6.8 compares the results produced by our method, Predator, RPMNet, and RGM for the rigid alignment of incomplete point clouds with partial overlap. For this ex-

Figure 6.8: Rigid registration produced by various models.



Source: The Authors

Figure 6.9: Rotation errors in degrees, MIE and MAE, comparing different approaches for rigid registration on ModelNet40 dataset on *Clean*, *Jitter*, and different percentages of samples kept for *Crop-Noise*.



(a) Mean Isometric Rotation Error (MIE) (b) Mean Anisotropic Rotation Error (MAE)

Source: The Authors

ample, our method produces better registration of the vase and plant, while other methods cause the partial clouds to cross each other. Fig. 6.9 compares the rotation errors in all test cases for Predator, RPMNet, and RGM. It shows that our method has smaller errors for all tested cropping levels.

Table 6.11 compares results with *Clean* clouds where our model and RPMNet have the best results for translations. Except for Predator, all works present a low rotation error, close to 0.1°. On *Jitter* (Table 6.12), RGM has the lead, closely followed by ours. Again, Predator presents larger errors than the others. Finally, for *Crop-Noise* point clouds with Gaussian noise (Table 6.13), our model is the only one to achieve less than one degree of error at MIE(R) and it has the best overall results. These experiments highlights our model's robustness, as it consistently obtains good results across the different types of point clouds.

Table 6.11: Rigid-body registration results on clouds without noise (*Clean*). Isotropic and anisotropic mean errors for rotation ($\mathbf{R}$) and translation ($\mathbf{t}$). The best results are shown in **bold**.

Method	MIE ($\mathbf{R}$)	MIE ($\mathbf{t}$)	MAE ($\mathbf{R}$)	MAE ($\mathbf{t}$)
Ours	0.130	**0.0007**	0.060	**0.0003**
Predator	1.620	0.0100	0.850	0.0070
RGM	0.150	0.0008	0.060	0.0004
RPMNet	**0.072**	**0.0007**	**0.036**	**0.0003**

Source: The Authors

Table 6.12: Rigid-body registration results on clouds with Gaussian noise. Isotropic and anisotropic mean errors for rotation ($\mathbf{R}$) and translation ($\mathbf{t}$). The best results are shown in **bold**.

Method	MIE ($\mathbf{R}$)	MIE ($\mathbf{t}$)	MAE ($\mathbf{R}$)	MAE ($\mathbf{t}$)
Ours	0.210	0.0020	0.110	0.0010
Predator	1.580	0.0150	0.830	0.0070
RGM	**0.140**	**0.0013**	**0.077**	**0.0006**
RPMNet	0.574	0.0050	0.291	0.0020

Source: The Authors

Table 6.13: Rigid-body registration results on clouds with Gaussian noise and random crop preserving 70% of the points. Isotropic and anisotropic mean errors for rotation ($\mathbf{R}$) and translation ($\mathbf{t}$). The best results are shown in **bold**.

Method	MIE ($\mathbf{R}$)	MIE ($\mathbf{t}$)	MAE ($\mathbf{R}$)	MAE ($\mathbf{t}$)
Ours	**0.770**	**0.0060**	**0.410**	**0.0030**
Predator	1.880	0.0190	0.980	0.0090
RGM	1.550	0.0150	0.810	0.0070
RPMNet	1.712	0.0180	0.890	0.0080

Source: The Authors

6.2.3 Sensitivity Study

Table 6.14 reports results on the mentioned scenarios with different number of iterations and the previously described cyclic consistency (Section 5.1). Instead of the two iterations used by RGM, we tested more and fewer repetitions. The Clean scenarios benefit the most from more iterations, as all metrics consistently improve. This is due to the absence of noises that could impact the convergence, so the errors keep reducing. Jitter has no benefit with more iterations which shows that the network cannot produce better correspondences, independently of how much aligned the clouds are. Finally, Crop-Noise reaches the same plateau in the third iteration, meaning that the model cannot deal better with the noise and partial correspondences.

Table 6.14: Rigid-body registration results on clouds on *Clean*, *Jitter* and *Crop-Noise*. Isotropic and anisotropic mean errors for rotation ($\mathbf{R}$) and translation ($\mathbf{t}$). The best results are shown in **bold**.

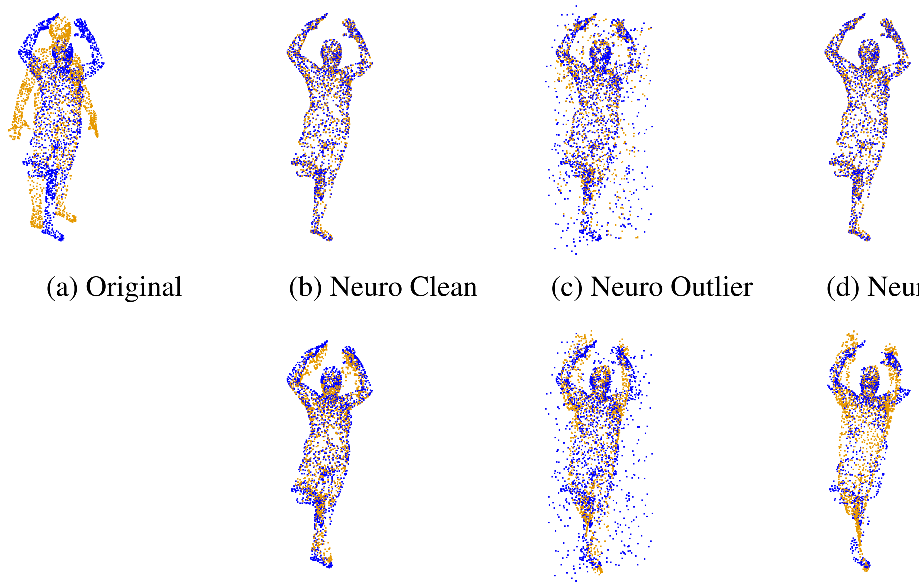
Method	MIE ($\mathbf{R}$)	MIE ($\mathbf{t}$)	MAE ($\mathbf{R}$)	MAE ($\mathbf{t}$)
Ours *Clean*	0.130	0.0007	0.060	0.0003
Ours Iter 3	0.036	0.0002	0.014	0.0001
Ours Iter 4	**0.012**	**0.0000**	**0.001**	**0.0000**
Ours Iter 1	0.966	0.0064	0.468	0.0031
Ours *Jitter*	**0.210**	**0.0020**	**0.110**	**0.0010**
Ours Iter 3	**0.204**	**0.0019**	**0.112**	**0.0010**
Ours Iter 4	**0.206**	**0.0020**	**0.113**	**0.0010**
Ours Iter 1	0.684	0.0053	0.346	0.0026
Ours *Crop-Noise*	0.770	0.0060	0.410	0.0030
Ours Iter 3	**0.583**	**0.0048**	**0.308**	**0.0024**
Ours Iter 4	**0.591**	**0.0051**	**0.313**	**0.0025**
Ours Iter 1	2.640	0.0220	1.364	0.0108

Source: The Authors

6.3 Limitations

Our method for non-rigid registration does not handle very large deformations. Figure 6.10 illustrates this limitation with an example from the FAUST dataset proposed by NeuroMorph (EISENBERGER et al., 2021). This dataset originally focuses on correspondences of meshes, all of them representing people in different poses and with bigger deformations than the ones found in the Custom and RMANet datasets. This shows that

Figure 6.10: Results in the FAUST dataset comparing NeuroMorph to our approach. Even though our algorithm improves performance on noise cases, when there are large deformations and complete point clouds, the inner-product-based NeuroMorph has better results. If we consider *Outlier* and *Holes* though, NeuroMorph as well as ours cannot properly register the clouds.



(a) Original (b) Neuro Clean (c) Neuro Outlier (d) Neuro Holes

(e) Ours Clean (f) Ours Outlier (g) Ours Holes

Source: The Authors

the Sinkhorn module with dustbins and probability model may not handle well correspondences between largely deformed clouds. Our model may not immediately generalize to different point clouds. Currently, this problem is addressed by fine-tuning the model in a self-learning manner with examples of point clouds from the non-handled classes. The other main limitation concerns varied topology between source and target and matching fine details. Examples of these are the problems dealing with opening and closing mouths at Fig. 6.4 or the expressions at Fig. 6.5.

6.4 Summary

This chapter reported experiments validating the overall advantages of our proposed registration methods over the compared approaches. In datasets with larger movements and details such as limbs, RMANet and Custom, we achieved on average superior performance. Even though we trained RDR only with *Cropped* point clouds, it could directly tackle other scenarios (as in the RAMNet dataset case). In situations where it could not generalize so well, as with Outlier and *Holes* in Section 6.1.10, fine-tuning to other noise types improved the registrations.

The experiments also showed that different RDR configurations work better for some scenarios. For example, cyclic consistency tends to improve results on partial point clouds, whereas more complete clouds benefit from multiple loop iterations. Moreover, there are advantages to training for a specific number of points. Thus, if it is possible to predict the size of clouds during evaluation time, one should train on this number. Finally, in the rigid case, a total of 3 iterations yields the best results across all noise types.

The experiments also indicated RDR still fails in fine-detailed scenarios, even though it was better than the other compared approaches on average. In the datasets focusing on faces, we notice problems dealing with finer details. Our approach tends to deal poorly with changes in the topology of the original triangular mesh, such as mouth opening or closing. Another example from the CoMA *Face* dataset was in the classification of points without correspondences, where it overfitted (poor results on *Cropped* Precision/Recall). The last open point for improvements is in the probabilistic post-processing, where parameters, λ, and β, directly affect the final registration (Section 6.1.9).

7 CONCLUSION

We presented a learning-based model for both non-rigid and rigid registration of 3D point clouds while being robust to noise and missing portions. We accomplish this by adapting and improving a neural network designed to match sparse features in 2D images to densely match features in 3D point clouds. This is a considerably harder problem performed on unstructured data, as opposed to on a regular image grid. Our main improvements in the network are a new feature encoding module, and a different approach to handle missing correspondences and to create a valid probability matrix as the network output.

In addition, our technique combines the advantages of deep learning and probabilistic modeling for non-rigid registration. To the best of our knowledge, this is the first time such a combination is explored in the context of non-rigid registration of 3D point clouds. To solve rigid registration, we used our neural network in a framework that uses cyclic consistency and multiple iterations. This achieved state-of-art results on both non-rigid and rigid registration with superior performance on noisy and partial point clouds.

Our experiments showed that our models outperformed previous techniques in terms of robustness in three challenging datasets for non-rigid registration, and one for rigid. In all cases, we tested with partial point clouds and different kinds of noise. Furthermore, we addressed the lack of datasets with ground truth for non-rigid registration by proposing a self-supervised training strategy, which can be combined with random generation of noisy point clouds. This can create fully synthetic datasets that produce robust learning-based models.

We further discussed various aspects of our non-rigid registration method. The first was showing how gracefully it degrades with increasing levels of noise. This was followed by discussing how integrating the calls to our neural network in an optimization loop could improve registration on complete point clouds, and how cyclic consistency could help on partial point clouds. Thirdly, we showed that it is possible to improve our initially reported registration results by training our model simultaneously with point clouds containing missing regions, outliers, and holes. Finally, our experiments show that having similar number of samples in the point clouds used for training and evaluation reduces the registration error.

The results of this work covering rigid and non-rigid registration were reported in an article entitled *Robust Point-Cloud Registration based on Dense Point Matching* *and Probabilistic Modeling* published in the *The Visual Computer* journal (volume 38, pages3217–3230, 2022).

7.1 Future Work

The main directions for future exploration are to target the limitations of the proposed approach. We narrow the constraints into two: problems handling large deformations, and dealing with finer details. A first direction to explore is a better integration of our deep-learning model with the BCPD algorithm. Currently, they are separate and independent steps, so having BCPD able to inform the neural network with more than the input clouds when doing multiple iterations of the network could improve performance. We also tested stagewise training approach for a tighter integration but it did not offer improvements in the CoMA dataset. In this strategy, we fine tuned our model while considering the probabilistic refinement operations. So, instead of comparing the correspodence matrix (Section 4.4), we used the L1 distance between the registered point cloud (produced by our refinement) and the ground truth.

Another direction is to explore delegating to the network the understanding of motion-coherence. This is done in networks like FlowNet3D (LIU; QI; GUIBAS, 2019) and PointPWC (WU et al., 2020) by using cost-volumes and further investigation could lead to a similar behavior for non-rigid registration. Finally, further developments of the BCPD algorithm could improve results, examples of efforts are presented at Appendices B and C. In these chapters, even though we could not achieve improvements, we regard our trials as initial steps to a better BCPD.

REFERENCES

AMBERG, B.; ROMDHANI, S.; VETTER, T. Optimal step nonrigid icp algorithms for surface registration. In: IEEE. **CVPR**. [S.l.], 2007. p. 1–8.

ARUN, K. S.; HUANG, T. S.; BLOSTEIN, S. D. Least-squares fitting of two 3-d point sets. **IEEE Transactions on pattern analysis and machine intelligence**, IEEE, n. 5, p. 698–700, 1987.

BAI, X. et al. D3feat: Joint learning of dense detection and description of 3d local features. In: **CVPR**. [S.l.: s.n.], 2020. p. 6359–6367.

BERGER, M. et al. A survey of surface reconstruction from point clouds. In: WILEY ONLINE LIBRARY. **Comput. Graph. Forum**. [S.l.], 2017. v. 36, n. 1, p. 301–329.

BESL, P. J.; MCKAY, N. D. Method for registration of 3-d shapes. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Sensor fusion IV: control paradigms and data structures**. [S.l.], 1992. v. 1611, p. 586–606.

BISHOP, C. M. Pattern recognition. **Machine learning**, v. 128, n. 9, 2006.

BOOKSTEIN, F. L. Principal warps: Thin-plate splines and the decomposition of deformations. **IEEE TPAMI**, IEEE, v. 11, n. 6, p. 567–585, 1989.

BRESSON, G. et al. Simultaneous localization and mapping: A survey of current trends in autonomous driving. **IEEE Transactions on Intelligent Vehicles**, IEEE, v. 2, n. 3, p. 194–220, 2017.

BRONSTEIN, A. M.; BRONSTEIN, M. M.; KIMMEL, R. Calculus of nonrigid surfaces for geometry and texture manipulation. **IEEE TVCG**, IEEE, v. 13, n. 5, p. 902–913, 2007.

BRONSTEIN, A. M.; BRONSTEIN, M. M.; KIMMEL, R. **Numerical geometry of non-rigid shapes**. [S.l.]: Springer Science & Business Media, 2008.

CAO, V.-T. et al. Non-rigid registration for deformable objects. In: IEEE. **2014 International Conference on Computer Graphics Theory and Applications (GRAPP)**. [S.l.], 2014. p. 1–10.

CHIZAT, L. et al. Scaling algorithms for unbalanced optimal transport problems. **Mathematics of Computation**, v. 87, n. 314, p. 2563–2609, 2018.

CHO, K. et al. On the properties of neural machine translation: Encoder-decoder approaches. **arXiv preprint arXiv:1409.1259**, 2014.

CHOY, C.; PARK, J.; KOLTUN, V. Fully convolutional geometric features. In: **ICCV**. [S.l.: s.n.], 2019. p. 8958–8966.

CUTURI, M. Sinkhorn distances: Lightspeed computation of optimal transport. **NeurIPS**, v. 26, p. 2292–2300, 2013.

ECKART, B.; KIM, K.; KAUTZ, J. Fast and accurate point cloud registration using trees of gaussian mixtures. **arXiv preprint arXiv:1807.02587**, 2018.

EISENBERGER, M. et al. Neuromorph: Unsupervised shape interpolation and correspondence in one go. In: **CVPR**. [S.l.: s.n.], 2021. p. 7473–7483.

FENG, W. et al. Recurrent multi-view alignment network for unsupervised surface registration. In: **CVPR**. [S.l.: s.n.], 2021. p. 10297–10307.

FITZGIBBON, A. W. Robust registration of 2d and 3d point sets. **Image and vision computing**, Elsevier, v. 21, n. 13-14, p. 1145–1153, 2003.

FU, K. et al. Robust point cloud registration framework based on deep graph matching. In: **CVPR**. [S.l.: s.n.], 2021. p. 8893–8902.

GAO, W.; TEDRAKE, R. Filterreg: Robust and efficient probabilistic point-set registration using gaussian filter and twist parameterization. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 11095–11104.

GE, S.; FAN, G. Non-rigid articulated point set registration with local structure preservation. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2015. p. 126–133.

GE, S.; FAN, G.; DING, M. Non-rigid point set registration with global-local topology preservation. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2014. p. 245–251.

GEIGER, A. et al. Vision meets robotics: The KITTI dataset. **The International Journal of Robotics Research**, Sage Publications Sage UK: London, England, v. 32, n. 11, p. 1231–1237, 2013.

GRANGER, S.; PENNEC, X. Multi-scale em-icp: A fast and robust approach for surface registration. In: SPRINGER. **European Conference on Computer Vision**. [S.l.], 2002. p. 418–432.

HIROSE, O. Acceleration of non-rigid point set registration with downsampling and gaussian process regression. **IEEE TPAMI**, IEEE, v. 43, n. 8, p. 2858–2865, 2020.

HIROSE, O. A bayesian formulation of coherent point drift. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, 2020.

HUANG, S. et al. Predator: Registration of 3d point clouds with low overlap. In: **CVPR**. [S.l.: s.n.], 2021. p. 4267–4276.

JIAN, B.; VEMURI, B. C. Robust point set registration using gaussian mixture models. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 33, n. 8, p. 1633–1645, 2010.

KITTENPLON, Y.; ELDAR, Y. C.; RAVIV, D. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In: **CVPR**. [S.l.: s.n.], 2021. p. 4114–4123.

LIU, J. et al. A survey on deep learning methods for scene flow estimation. **Pattern Recognition**, Elsevier, v. 106, p. 107378, 2020.

LIU, X.; QI, C. R.; GUIBAS, L. J. Flownet3d: Learning scene flow in 3d point clouds. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 529–537.

MA, J. et al. Nonrigid point set registration with robust transformation learning under manifold regularization. **IEEE transactions on neural networks and learning systems**, IEEE, 2018.

MA, J. et al. Non-rigid point set registration with robust transformation estimation under manifold regularization. In: **Thirty-First AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2017.

MA, J.; ZHAO, J.; YUILLE, A. L. Non-rigid point set registration by preserving global and local structures. **Transactions on image Processing**, IEEE, v. 25, n. 1, p. 53–64, 2015.

MENA, G. et al. Learning latent permutations with gumbel-sinkhorn networks. In: OPENREVIEW. **ICLR 2018 Conference Track**. [S.l.], 2018. v. 2018.

MITTAL, H.; OKORN, B.; HELD, D. Just go with the flow: Self-supervised scene flow estimation. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2020. p. 11177–11185.

MYRONENKO, A.; SONG, X. Point set registration: Coherent point drift. **transactions on pattern analysis and machine intelligence**, IEEE, v. 32, n. 12, p. 2262–2275, 2010.

NASEER, M.; KHAN, S.; PORIKLI, F. Indoor scene understanding in 2.5/3d for autonomous agents: A survey. **IEEE access**, IEEE, v. 7, p. 1859–1887, 2018.

OUYANG, B.; RAVIV, D. Occlusion guided scene flow estimation on 3d point clouds. In: **CVPR**. [S.l.: s.n.], 2021. p. 2805–2814.

POMERLEAU, F.; COLAS, F.; SIEGWART, R. A review of point cloud registration algorithms for mobile robotics. 2015.

PUY, G.; BOULCH, A.; MARLET, R. Flot: Scene flow on point clouds guided by optimal transport. In: SPRINGER. **ECCV**. [S.l.], 2020. p. 527–544.

QI, C. R. et al. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: **CVPR**. [S.l.: s.n.], 2017. p. 652–660.

RANJAN, A. et al. Generating 3D faces using convolutional mesh autoencoders. In: **ECCV**. [S.l.: s.n.], 2018. p. 725–741.

SAHILLIOĞLU, Y. Recent advances in shape correspondence. **The Visual Computer**, Springer, v. 36, n. 8, p. 1705–1721, 2020.

SARLIN, P.-E. et al. Superglue: Learning feature matching with graph neural networks. In: **CVPR**. [S.l.: s.n.], 2020. p. 4938–4947.

SINKHORN, R.; KNOPP, P. Concerning nonnegative matrices and doubly stochastic matrices. **Pacific Journal of Mathematics**, Mathematical Sciences Publishers, v. 21, n. 2, p. 343–348, 1967.

SORKINE-HORNUNG, O.; RABINOVICH, M. Least-squares rigid motion using svd. **Computing**, v. 1, n. 1, p. 1–5, 2017.

SORKINE, O.; ALEXA, M. As-rigid-as-possible surface modeling. In: **Symposium on Geometry processing**. [S.l.: s.n.], 2007. v. 4, p. 109–116.

SUN, J. et al. Loftr: Detector-free local feature matching with transformers. In: **CVPR**. [S.l.: s.n.], 2021. p. 8922–8931.

SUN, Y. et al. Circle loss: A unified perspective of pair similarity optimization. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2020. p. 6398–6407.

TEED, Z.; DENG, J. Raft: Recurrent all-pairs field transforms for optical flow. In: SPRINGER. **European conference on computer vision**. [S.l.], 2020. p. 402–419.

THOMAS, H. et al. Kpconv: Flexible and deformable convolution for point clouds. In: **ICCV**. [S.l.: s.n.], 2019. p. 6411–6420.

TISHCHENKO, I. et al. Self-supervised learning of non-rigid residual flow and ego-motion. In: IEEE. **Int. Conf. on 3D Vision (3DV)**. [S.l.], 2020. p. 150–159.

TOMBARI, F.; SALTI, S.; STEFANO, L. D. Unique signatures of histograms for local surface description. In: SPRINGER. **European conference on computer vision**. [S.l.], 2010. p. 356–369.

VASWANI, A. et al. Attention is all you need. In: **NeurIPS**. [S.l.: s.n.], 2017. p. 5998–6008.

VLASIC, D. et al. Articulated mesh animation from multi-view silhouettes. In: **ACM SIGGRAPH 2008 papers**. [S.l.: s.n.], 2008. p. 1–9.

WANG, L. et al. Non-rigid point set registration networks. **arXiv preprint arXiv:1904.01428**, 2019.

WANG, L. et al. Coherent point drift networks: Unsupervised learning of non-rigid point set registration. **arXiv preprint arXiv:1906.03039**, 2019.

WANG, R.; YAN, J.; YANG, X. Learning combinatorial embedding networks for deep graph matching. In: **ICCV**. [S.l.: s.n.], 2019. p. 3056–3065.

WANG, Y.; SOLOMON, J. M. Deep closest point: Learning representations for point cloud registration. In: **ICCV**. [S.l.: s.n.], 2019. p. 3523–3532.

WANG, Y. et al. Dynamic graph cnn for learning on point clouds. **ACM TOG**, ACM New York, NY, USA, v. 38, n. 5, p. 1–12, 2019.

WANG, Z.; WU, Y.; NIU, Q. Multi-sensor fusion in automated driving: A survey. **IEEE Access**, IEEE, v. 8, p. 2847–2868, 2019.

WU, W. et al. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In: SPRINGER. **ECCV**. [S.l.], 2020. p. 88–107.

WU, Z. et al. 3d shapenets: A deep representation for volumetric shapes. In: **CVPR**. [S.l.: s.n.], 2015. p. 1912–1920.

YANG, J. et al. Go-icp: A globally optimal solution to 3d icp point-set registration. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 38, n. 11, p. 2241–2254, 2015.

YAO, Y. et al. Quasi-newton solver for robust non-rigid registration. In: **CVPR**. [S.l.: s.n.], 2020. p. 7600–7609.

YEW, Z. J.; LEE, G. H. Rpm-net: Robust point matching using learned features. In: **CVPR**. [S.l.: s.n.], 2020. p. 11824–11833.

YUAN, W. et al. Deepgmr: Learning latent gaussian mixture models for registration. In: SPRINGER. **ECCV**. [S.l.], 2020. p. 733–750.

ZENG, Y. et al. Corrnet3d: Unsupervised end-to-end learning of dense correspondence for 3d point clouds. In: **CVPR**. [S.l.: s.n.], 2021. p. 6052–6061.

ZHAO, H. et al. Point transformer. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [S.l.: s.n.], 2021. p. 16259–16268.

ZHOU, Q.-Y.; PARK, J.; KOLTUN, V. Fast global registration. In: SPRINGER. **European Conference on Computer Vision**. [S.l.], 2016. p. 766–782.

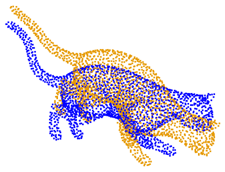
APPENDIX A — FURTHER QUALITATIVE RESULTS

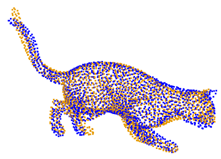
We present additional results at Figures A.1 to A.9 comparing techniques using point clouds from the RMANet and ModelNet40 datasets. Source clouds are in orange and target ones in blue. Source points classified as not having matches are shown in magenta. All cases used the same parameters used in the main paper, with no *Loop* or *Cycle*.

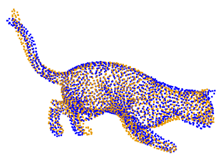
Figures A.1 to A.8 show examples of non-rigid registration. From them, one can note that our method and its self-learning variation produces the best alignments. This is highlighted for the *Cropped* point clouds. Although Ours-RGM can also deal with partial clouds, it is not as precise as our SuperGlue-based model. Figure A.2d is an interesting example where the RGM network has issues indicating points with no matches. This leads to the refinement based on BCPD to not infer the missing part (in magenta) as well as the examples that use our learning model (Figures A.2 (b) and (c)). Furthermore, Figures A.1 and A.5 have results with complete clouds and all works except FLOT produce good results. We notice RMANet on the other hand, can only present good results in this scenario (Figures A.1 (f) and A.5 (f)). This is expected since RMANet was trained with *Clean* clouds (it could not learn with *Cropped*). Figures A.6 to A.8 highlight the benefits of the probabilistic refinement, as techniques that use such refinement produce better results than the ones that do not use it (RMANet and FLOT). The benefits are not only when inferring the registration of missing parts in *Cropped* point clouds, but also when fixing incorrect correspondences in noisy point clouds (*Outliers*) or in point clouds with *Holes*.

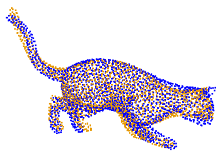
Figure A.9 provides an example of rigid registration and shows a challenging scenario from the ModelNet40 dataset. In this case, the clouds represent a flower and both were cropped. Since the upper part is not as descriptive, only the tip of one leaf indicates the correct alignment. Only our model and Predator consider this detail and produce correct registration. RGM and RPMNet assume the leaf to be the stem which completely changes the final result.

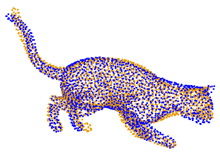
Figure A.1: Comparing various techniques for non-rigid registration on *Clean* point clouds from the RMANet dataset. Our method, its variants and RMANet perform well. FLOT and BCPD miss the tail.

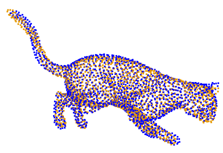

(a) Original Clouds

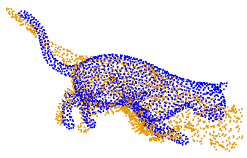

(b) Ours

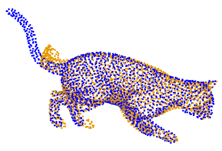

(c) Ours Self


(d) Ours-RGM


(e) Ours-Neuro

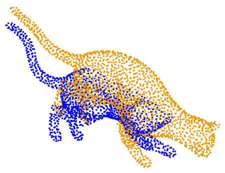

(f) RMANet

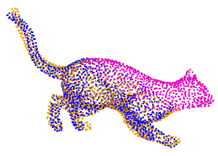

(g) FLOT

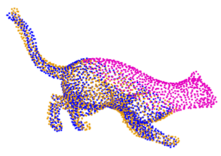

(h) BCPD

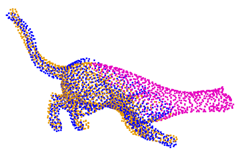
Source: The Authors

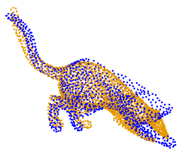
Figure A.2: Comparing various techniques for non-rigid registration on *Cropped* point clouds from the RMANet dataset. Only Ours and Ours-Self obtain good results. Ours-Neuro and BCPD can align the points with correspondences, but are unable to deal with points without matches.

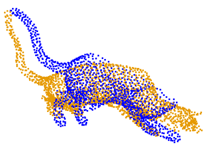

(a) Original Clouds

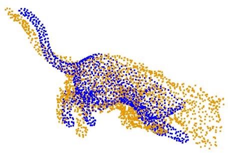

(b) Ours

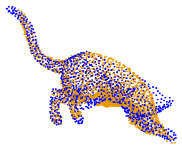

(c) Ours Self


(d) Ours-RGM


(e) Ours-Neuro

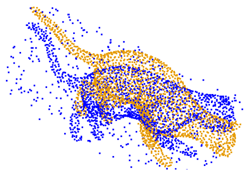

(f) RMANet

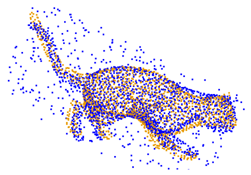

(g) FLOT

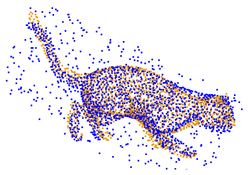

(h) BCPD

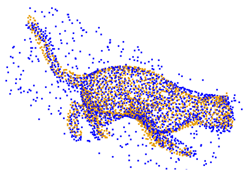
Source: The Authors

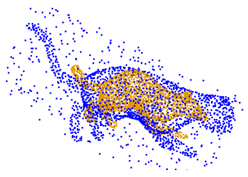
Figure A.3: Comparing various techniques for non-rigid registration on noisy (*Outlier*) point clouds from the RMANet dataset. Ours, Ours-RGM and BCPD achieve good results. Ours-Neuro and RMANet have issues dealing with wrong correspondences. Since they could not train with *Cropped* point clouds and used *Clean* instead, they cannot deal with target points meant to not be matched.

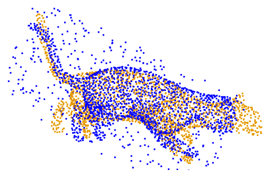

(a) Original Clouds

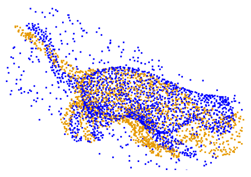

(b) Ours

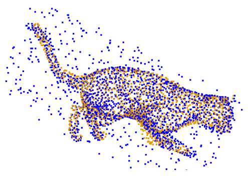

(c) Ours Self


(d) Ours-RGM


(e) Ours-Neuro

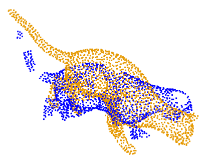

(f) RMANet

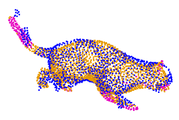

(g) FLOT

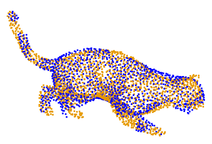

(h) BCPD

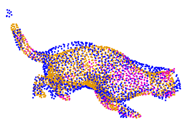
Source: The Authors

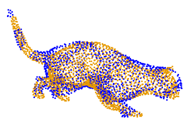
Figure A.4: Comparing various techniques for non-rigid registration on point clouds containing *Holes*. Ours, Ours-RGM and Ours-Neuro present the best results.

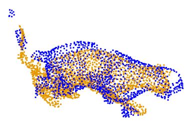

(a) Original Clouds

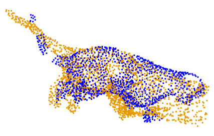

(b) Ours

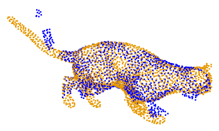

(c) Ours Self


(d) Ours-RGM


(e) Ours-Neuro

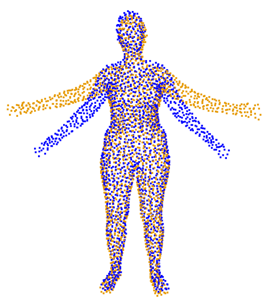

(f) RMANet

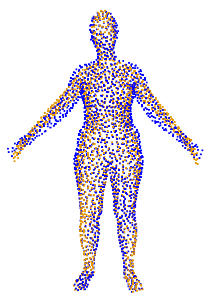

(g) FLOT

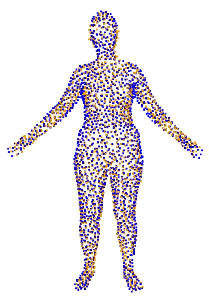

(h) BCPD

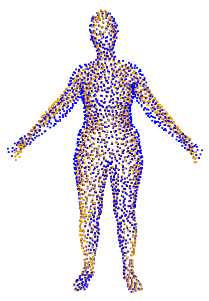
Source: The Authors

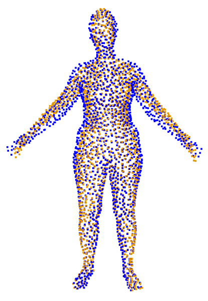
Figure A.5: Another example comparing various techniques for non-rigid registration on *Clean* point clouds from the RMANet dataset. We noticed how most works perform well, except FLOT and BCPD, which misses the feet.

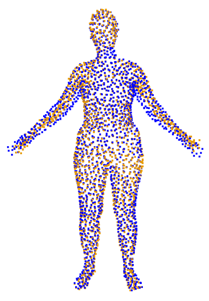

(a) Original Clouds

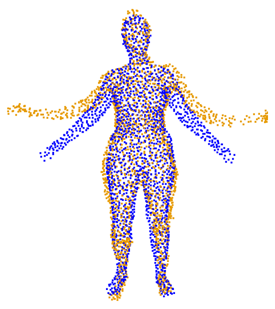

(b) Ours

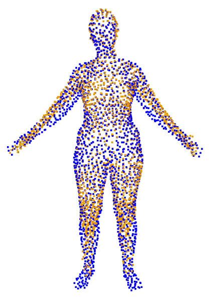

(c) Ours Self


(d) Ours-RGM


(e) Ours-Neuro

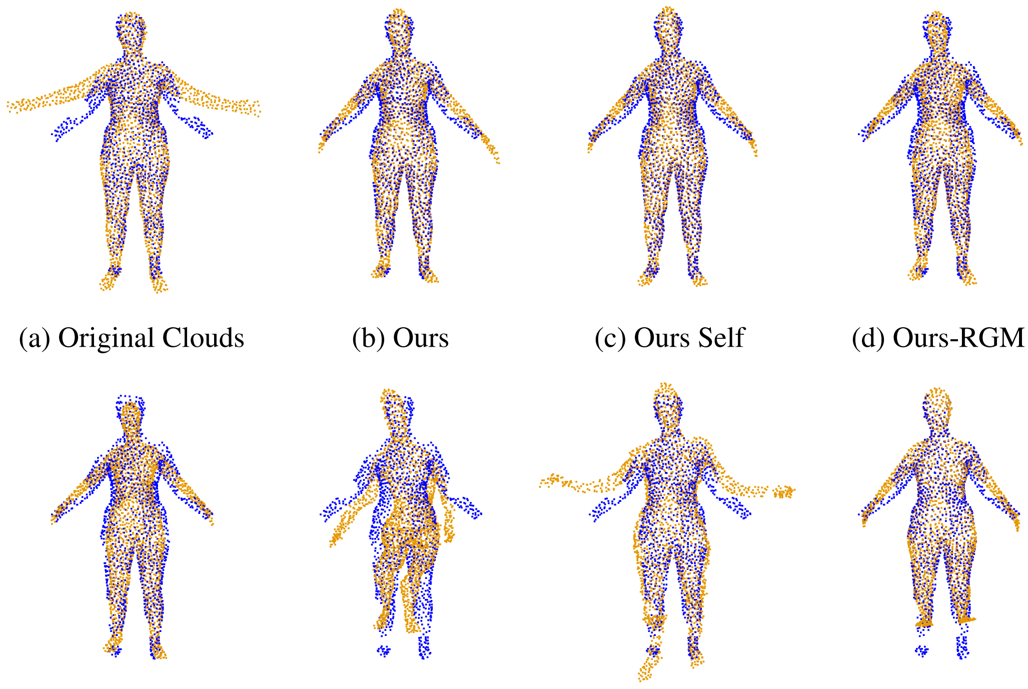

(f) RMANet


(g) FLOT


(h) BCPD

Source: The Authors

Figure A.6: Comparing various techniques for non-rigid registration on point clouds *Holes*. Techniques that use the proposed probabilistic refinement step properly handle this scenario. Even though BCPD correctly aligns the upper body, it misses the feet.

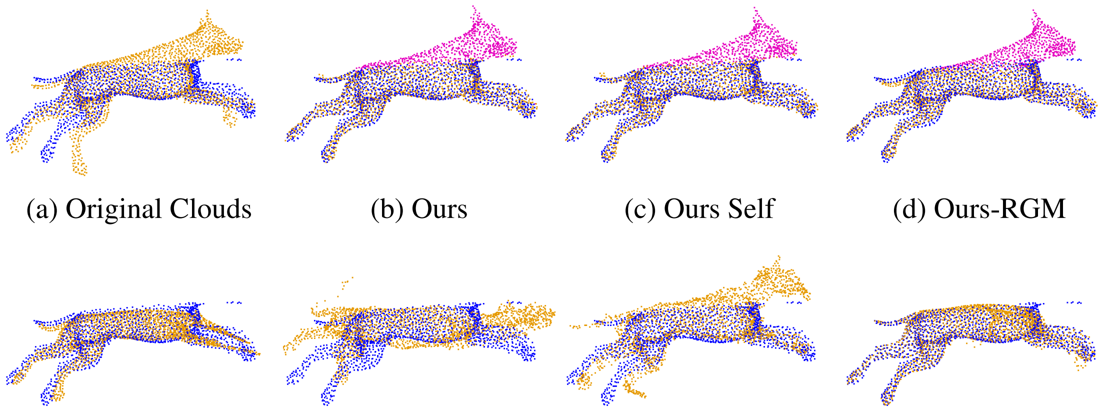


(a) Original Clouds (b) Ours (c) Ours Self (d) Ours-RGM

(e) Ours-Neuro (f) RMANet (g) FLOT (h) BCPD

Source: The Authors

Figure A.7: Comparing various techniques for non-rigid registration on *Cropped* point clouds from the RMANet dataset. Only Ours, Ours-Self, and Ours-RGM achieved good results, with Ours and Ours-Self achieving the best results. Ours-Neuro and BCPD can align the points with correspondences but are unable to deal with points without matches.

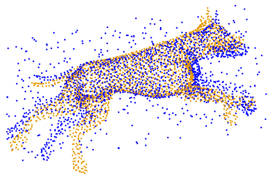


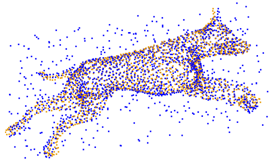
(a) Original Clouds (b) Ours (c) Ours Self (d) Ours-RGM

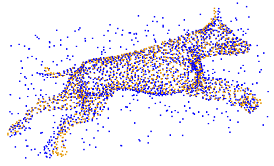
(e) Ours-Neuro (f) RMANet (g) FLOT (h) BCPD

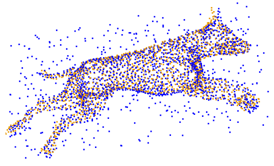
Source: The Authors

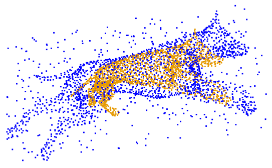
Figure A.8: Comparing various techniques for non-rigid registration on noisy (*Outlier*) point clouds from the RMANet dataset. Ours, Ours-Self, Ours-RGM and BCPD achieve good results. Ours-Neuro and RMANet have issues dealing with wrong correspondences.

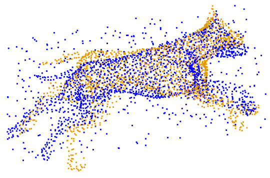

(a) Original Clouds

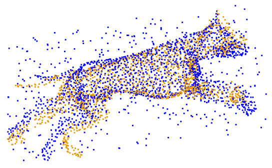

(b) Ours

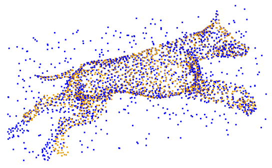

(c) Ours Self


(d) Ours-RGM


(e) Ours-Neuro

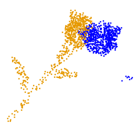

(f) RMANet

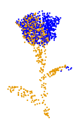

(g) FLOT

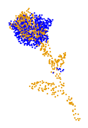

(h) BCPD

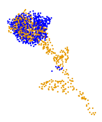
Source: The Authors

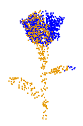
Figure A.9: *Crop-Noise* clouds from the ModelNet40 dataset. Differently from the non-rigid pairs, both clouds were cropped and only the tip of one leaf offers a context to align the upper part. Ours and Predator can deal with this challenge, whereas RPMNet and RGM align the plant stem instead.


(a) Original Clouds


(b) Ours


(c) RGM


(d) RPMNet


(e) Predator

Source: The Authors

APPENDIX B — RESTRICTIONS TO POINTWISE σ^2

This appendix describes some initial effort to improve the BCPD algorith. More specifically, it aims at having a point-wise optimization of the variance, σ^2. We notice the original formulation has a unique value for all GMM components. In this case, we tried to modify the formulation to support possible different value for each component. This means that in addition to considering each source point as the mean value of a Gaussian during optimization, we would also account for individual variances. This did not improve the original results compared to the single-value formulation. Nevertheless, we describe the attempted formulation. We expect this can contribute to different approaches that could yield positive results. All symbols used here follow the notation from Section 2.3.1 and Section 3.1.

The BCPD algorithm can be broken down into three parts (supplemental material of BCPD (HIROSE, 2020b)). The first two focus on optimizing the non-rigid deformation, and only use σ^2. They are not affected in their formulas by this change. The last part, which computes the value of σ^2, requires an update. Our proposed update is a new equation tha replaces *Equation 4* in the BCPD's supplemental material:

$$\begin{aligned}\log q_3 = E(x,y,\theta) = \sum_{n=1}^{N}\sum_{m=1}^{M} \mathbf{c}_n \delta_m(\mathbf{e}_n) ln|2\pi\sigma_m^2 \mathbf{I}_D|^{-\frac{1}{2}} - \\ \frac{1}{2}\sum_{m=1}^{M}\frac{1}{\sigma_m^2}\sum_{n=1}^{N}\mathbf{c}_n\delta_m(\mathbf{e}_n)\|x_n - \mathcal{T}(y_m)\|^2.\end{aligned} \tag{B.1}$$

Following the original formulation, increasing the above function value with respect to σ_m^2 maximizes the lower bound and, consequently, the full joint probability. So, in order to find the maximum value of the above function, we take the derivative w.r.t. σ_m^2 and then set it to zero,

$$\begin{aligned}\frac{\partial \log q_3}{\partial \sigma_m^2} = \sum_{n=1}^{N}\sum_{m=1}^{M}\frac{-1}{2}\mathbf{c}_n\delta_m(\mathbf{e}_n)Tr(\sigma_m^2\mathbf{I}_D)^{-1} + \\ \frac{1}{2}\sum_{m=1}^{M}\frac{1}{(\sigma_m^2)^2}\sum_{n=1}^{N}\mathbf{c}_n\delta_m(\mathbf{e}_n)\|x_n - \mathcal{T}(y_m)\|^2.\end{aligned} \tag{B.2}$$

For the derivative the following property was used: $\frac{\partial |\mathrm{Y}|}{\partial \mathrm{x}} = |\mathrm{Y}|\mathrm{Tr}(\mathrm{Y}^{-1}\frac{\partial \mathrm{Y}}{\partial \mathrm{x}})$, where $|.|$ is the determinant of the given matrix. By setting it to zero we obtain:

$$D\sum_{n=1}^{N}\sum_{m=1}^{M}\mathbf{c}_n\delta_m(\mathbf{e}_n)(\sigma_m^2)^{-1} = \sum_{m=1}^{M}\frac{1}{(\sigma_m^2)^2}\sum_{n=1}^{N}\mathbf{c}_n\delta_m(\mathbf{e}_n)\|x_n - \mathcal{T}(y_m)\|^2. \tag{B.3}$$

In this part, we adopt the same definitions of BCPD, so we have $\sum_{n=1}^{N}\mathbf{c}_n\delta_m(\mathbf{e}_n) = \nu_m$ and $\mathbf{c}_n\delta_m(\mathbf{e}_n) = \mathbf{p}_{mn}$, which results in

$$\sum_{m=1}^{M}\frac{\nu_m}{\sigma_m^2} = \sum_{m=1}^{M}\frac{1}{D(\sigma_m^2)^2}\sum_{n=1}^{N}\mathbf{p}_{mn}\|x_n - \mathcal{T}(y_m)\|^2.$$

Finally, by isolating σ^2 we obtain an expression for estimating σ_m^2 at each step of the algorithm

$$\sigma_m^2 = \frac{1}{D\nu_m}\sum_{n=1}^{N}\mathbf{p}_{mn}\|x_n - \mathcal{T}(y_m)\|^2.$$

Our implementation with individual values of σ^2 did not present satisfactory results. We consider it a topic for future investigation.

APPENDIX C — RESTRICTIONS TO OPTIMIZE λ

From Algorithm 2 and Table 3.1, we highlight a series of input parameters that are meant to be tuned (λ, β, γ). In this appendix, we present an effort to optimize λ. This parameter was chosen because it is the simplest to obtain the update equations. Optimizing β, for example, would require equations based on how the motion coherence term $\mathbf{G}$ is built. This would be more complex than λ, which is directly used when estimating the deformation vectors, $\mathbf{v}$. Nevertheless, we could not optimize λ.

By inspecting Algorithm 2, we notice a lack of metrics showing the quality of the registration. This is understandable since one-to-one correspondences might not exist between points, and this is also a task to be optimized by the algorithm. Therefore, it becomes impossible to have a clear loss function comparing points from source and target point clouds. The closest approximation to this is σ^2, which can be interpreted as the average distance from source to target clouds weighted by the confidence we have on each match. This means that a large distance from a source point y_m to a target one x_n, will be weighted by a small value of $\mathbf{P}_{mn}$. Moreover, BCPD already uses σ^2 as the stop condition for the probabilistic optimization.

Given the role of σ^2 in the algorithm, we search for a value of λ that minimizes σ^2, thus resulting in a better registration. To compute this, we take the derivative of σ^2 w.r.t. λ, given by the expression at (line 17 in Algorithm 2) and set it to zero.

It is important to mention that we assume σ^2 to be pointwise, as we in Appendix B for σ_m^2. Furthermore, the obtained λ is pointwise as well. This was motivated by the simpler equations we obtain when deriving a matrix producing function w.r.t. to a matrix variable. This becomes more critical when using the chain rule in the derivatives, which in our case used the matrix producing function $\hat{y}$. In the original equations with scalars σ^2 and λ, the chain rule would require deriving a scalar by a matrix, followed by a matrix by a scalar. This is avoided with our choices and we always deal with matrices in the optimization.

The original equation in the line 17 of Algorithm 2 is

$$\tilde{\sigma}^2 = \frac{1}{3\hat{N}} \left(x^T diag(\nu')x - 2x^T \mathbf{P}^T \hat{y} + \hat{y}^T diag(\nu)\hat{y} \right), \tag{C.1}$$

and the obtained equation is given by

$$\sigma_m^2 = \frac{1}{3\nu_m}\sum_{n=1}^{N}\mathbf{p}_{mn}\left(x_n x_n^T - 2x_n\hat{y}_m^T + \hat{y}_m\hat{y}_m^T\right), \qquad \text{derivative w.r.t. } \lambda \text{ and in matrix form} \tag{C.2}$$

$$\frac{\partial\sigma_m^2}{\partial\lambda} = (\mathbf{P}X - diag(\nu)\hat{y})\frac{\partial\hat{y}}{\partial\lambda}, \qquad \text{by the chain rule } \frac{\partial\sigma_m^2}{\partial\hat{y}}\frac{\partial\hat{y}}{\partial\lambda} \tag{C.3}$$

$$0 = \mathbf{P}X - diag(\nu)\hat{y}, \qquad \text{after equating it to 0} \tag{C.4}$$

$$0 = \mathbf{P}X - diag(\nu)(\mathbf{s}(\mathbf{R}(Y+\mathbf{v})^T)^T + \mathbf{t}), \qquad \text{where } \hat{y} = \mathbf{s}(\mathbf{R}(Y+\mathbf{v})^T)^T + \mathbf{t} \tag{C.5}$$

$$PX - diag(\nu)t = diag(\nu)\mathbf{s}(Y+v)\mathbf{R}^T \tag{C.6}$$

$$\mathbf{s}^{-1}(diag(\nu)^{-1}\mathbf{P}X - \mathbf{t})\mathbf{R} = Y + \mathbf{v}, \qquad \text{by algebric manipulation} \tag{C.7}$$

$$Xb = \mathbf{s}^{-1}(diag(\nu)^{-1}\mathbf{P}X - \mathbf{t})\mathbf{R}, \qquad \text{first part named } Xb \tag{C.8}$$

$$Xb = Y + \mathbf{G}(\frac{\lambda\sigma^2}{\mathbf{s}^2}diag(\nu)^{-1} + \mathbf{G})^{-1}(T^{-1}(\hat{x}) - Y), \qquad \text{expanding } v \text{ given Algorithm 2} \tag{C.9}$$

$$\mathbf{G}^{-1}(Xb - Y) = (\frac{\lambda\sigma^2}{\mathbf{s}^2}diag(\nu)^{-1} + \mathbf{G})^{-1}(T^{-1}(\hat{x}) - Y) \tag{C.10}$$

$$(\frac{\lambda\sigma^2}{\mathbf{s}^2}diag(\nu)^{-1} + \mathbf{G})\mathbf{G}^{-1}(Xb - Y) = \mathbf{Re}, \qquad \text{where } \mathbf{Re} = T^{-1}(\hat{x}) - Y \tag{C.11}$$

$$Xb - Y + \frac{\lambda\sigma^2}{\mathbf{s}^2}diag(\nu)^{-1}\mathbf{G}^{-1}(Xb - Y) = \mathbf{Re} \tag{C.12}$$

$$Xb - Y = \mathbf{G}diag(\nu)\frac{\mathbf{s}^2}{\sigma^2}\lambda^{-1}(\mathbf{Re} - Xb + Y) \tag{C.13}$$

$$Xb - Y = \mathbf{G}\lambda^{-1}diag(\nu)\frac{\mathbf{s}^2}{\sigma^2}(\mathbf{Re} - Xb + Y) \qquad . \tag{C.14}$$

Notice the final equation has in both sides $M{\times}3$ matrices. $\mathbf{G}$ and λ^{-1} on the other hand are $M{\times}M$ matrix, where λ^{-1} is a diagonal matrix in this case. Estimating λ given this scenario require two steps. The first step is to invert G, and the second is a pointwise division on both side:

$$\mathbf{G}^{-1}(Xb - Y) = \lambda^{-1}diag(\nu)\frac{\mathbf{s}^2}{\sigma^2}(\mathbf{Re} - Xb + Y), \tag{C.15}$$

$$\lambda^{-1} = \frac{\mathbf{G}^{-1}(Xb - Y)}{diag(\nu)\frac{\mathbf{s}^2}{\sigma^2}(\mathbf{Re} - Xb + Y)}. \tag{C.16}$$

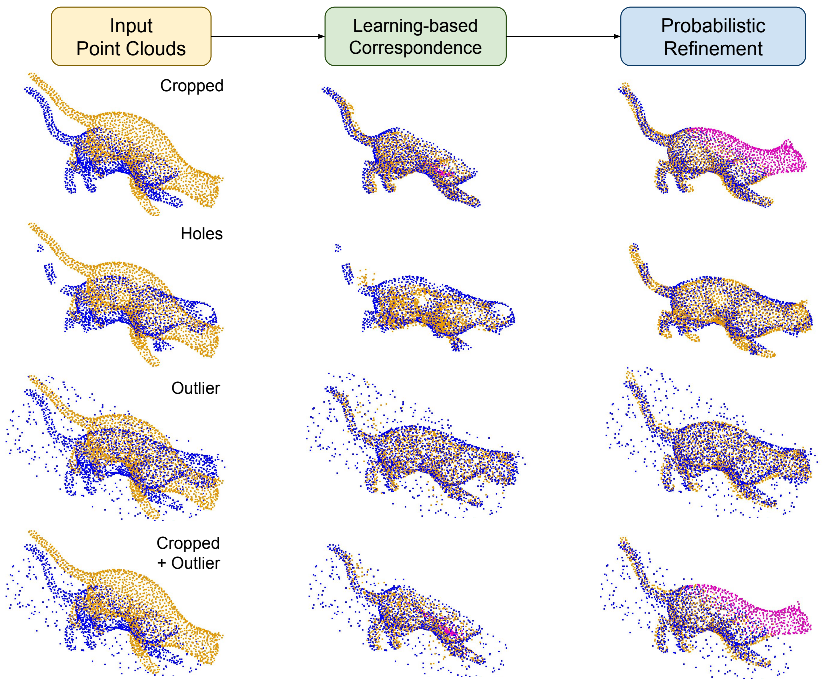
The main issue found is on inverting $\mathbf{G}$. Radial basis function or other tested kernels do not guarantee the existence of an inverse. In most cases, the matrix is ill-conditioned, which creates numerical instability and unexpected results. We leave solving this equation, and mainly not depending on inverting $\mathbf{G}$ as an open topic for future investigations.

APPENDIX D — RESUMO ESTENDIDO

Esta dissertação apresenta duas técnicas para o registro de nuvens de pontos 3D que são robustas a ruído e a nuvens de pontos parciais. Nossas técnicas abordam o registro rígido e o não rígido e exploram as vantagens do uso de aprendizado profundo para a estimativa de correspondência densa entre pontos.

A Figura D.1 mostra resultados produzidos por cada passo da nossa técnica (*Correspondências por Aprendizado de Máquina* e *Refinamento Probabilístico*) para o registro não rígido em diferentes cenários. Esses incluem: (i) uma das nuvens sendo parcial (*Cropped* - primeira linha). (ii) vários buracos em uma das nuvens de pontos (*Holes* - segunda linha); (iii) uma das nuvens de pontos com ruído (*Outliers* - terceira linha); (iv) uma combinação de nuvem parcial e com ruído (*Cropped* + *Outliers* - quarta linha). Perceba como nosso método consistentemente alcança bons resultados em todos os casos, sendo capaz de lidar com pontos sem correspondências de forma apropriada (em magenta).

Figure D.1: Registro não rígido produzido por nosso método em cenários desafiadores. Nós apresentamos os resultados após cada estágio do nosso fluxo. As nuvens iniciais estão em laranja e as de destino estão em azul. Pontos sem correspondências na nuvem de destino são mostradas em magenta.



Fonte: Os Autores

Considerando os resultados mostrados, essa dissertação inclui as seguintes contribuições:

- Técnicas baseadas em aprendizado de máquina para registros não rígidos (Capítulo 3) e rígidos (Capítulo 5) de nuvens de pontos que são robustas a nuvens parciais e a ruído. Ambas as técnicas têm desempenhos superiores a outras abordagens nesses cenários desafiadores;
- Uma adaptação da rede neural SuperGlue para estimar correspondências entre pares de nuvens de pontos 3D (Capítulo 4);
- Uma estratégia de treinamento autosupervisionado para o registro não rígido e robusto de nuvens de pontos (Seção 6.1.3).

Para ambos os tipos de registro, nós propomos uma única rede neural para estimar a correspondência densa entre os pontos. Ela se baseia na rede SuperGlue (SARLIN et al., 2020), mas a modifica em três pontos principais. O primeiro é usando a rede neural DGCNN (WANG et al., 2019) para aprender a estimar descritores iniciais de cada ponto das nuvens. Isso é seguido por uma nova camada para gerar a matriz de correspondências final. Ao invés de modelar essa parte como um problema de transporte ótimo e o resolver com o algoritmo de Sinkhorn (CUTURI, 2013), nós usamos o operador de Sinkhorn (MENA et al., 2018) com alternativas extras de correspondências. Por fim, nós incluímos um termo extra na função de perda com entropia cruzada usada no treinamento da rede neural. Esse termo novo foca em reduzir o número de falsos positivos nas correspondências. A rede neural resultante ainda usa a mesma sequência de módulos propostos por SuperGlue: *estimativa de descritores* + *módulo de atenção* + *refinamento com alternativas extras*. Todavia, nossas mudanças são cruciais para o bom desempenho do modelo em nuvens de pontos 3D.

Além disso, nós aplicamos avanços recentes em modelagem probabilística para refinar as correspondências criadas por nossa rede durante o registro não rígido. Tal combinação de aprendizado profundo e modelagem probabilística produz sensibilidade a contextos e também gera uma deformação coerente dos pontos. Como consequência, nossa abordagem é resiliente a ruído e a perda de informação.

Nosso refinamento é baseado no BCPD (HIROSE, 2020b) e substitui a estimativa densa de correspondências entre as nuvens de pontos iniciais e de destino pela matriz gerada pela nossa rede neural. Mesmo que essa troca viole as suposições feitas por BCPD de que as probabilidades vêm de uma mistura de distribuições Gaussianas, nós argumentamos que essa opção ainda assim é válida tendo em vista os resultados obtidos. Outra diferença no algoritmo de refinamento são os laços interno e externo que são consequências da otimização do registro em dois estágios. O resultado é um algoritmo que pode trabalhar com uma ou múltiplas iterações do laço externo.

Já para o registro rígido, nossa técnica aplica consistência cíclica e múltiplas chamadas para a rede neural proposta. Esses passos são seguidos pela estimativa da matriz de transformação rígida usando o método clássico com SVD. Diferentemente do caso não rígido em que usamos uma matriz fracionária, aqui nós convertemos as correspondências em valores binários e inteiros.

Demonstramos a efetividade das nossas técnicas ao compará-las com métodos no estado da arte. Essas comparações usam bases de dados com ruído e nuvens de pontos parciais ou com amostragem irregular. As nuvens consideradas têm movimentos acentuados e detalhes como membros se movendo em direções diferentes, ou rostos fazendo diversas expressões. Os experimentos mostram que, em geral, nós obtemos resultados superiores às técnicas no estado da arte. Por exemplo, nossas abordagens alcançam um erro até 45% menor que outras técnicas no registro não rígido de nuvens de pontos parciais, ou até 49% menor no registro rígido.

Nós também discutimos alguns aspectos extras da nossa técnica como a robustez a níveis diferentes de ruído e a números diversos de amostras nas nuvens de pontos. Por último, abordamos a falta de bases de dados que forneçam o registro correto entre as nuvens de pontos. Essas bases são críticas no treinamento supervisionado de modelos de registro não rígido. Para resolver essa escassez, nós propomos uma estratégia de autoaprendizado baseada em deformações randômicas.

Finalmente, salientamos que ainda há espaço para melhorias nas nossas técnicas, em especial para o registro não rígido. Um primeiro ponto seria no tratamento de deformações muito grandes ou nuvens com escalas diferentes. Outro aspecto seria em ter uma rede neural produzindo diretamente o registro, ou seja, sem a necessidade de um refinamento posterior. Essa é a tarefa futura mais desafiadora, mas é a que promete os maiores avanços.