Registration of 3D Point Clouds with Low Overlap

Master Thesis

Shengyu Huang

Photogrammetry and Remote Sensing Group
ETH Zürich

Supervisors:
Zan Gojcic, Mikhail Usvyatsov
Prof. Dr. Konrad Schindler

July 1st, 2020

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IGP
Institute of Geodesy and Photogrammetry

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Registration of 3D Point Clouds with Low Overlap

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Huang	Shengyu

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 1st July 2020	Shengyu Huang

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude and appreciation to everyone who support me during this master thesis:

- **Prof. Dr. Konrad Schindler** for offering me the opportunity of working on this project, and providing scientific guidance throughout this four months.
- **Mikhail Usvyatsov** for undertaking the supervision and his continuous passion for this project. Starting from my interdisciplinary project, Mikhail has supervised me for around 1 year and I strongly benefit from our discussions and brainstorming on quite many topics and implementation details. This thesis is impossible without his "puzzle game" idea.
- **Zan Gojcic** for undertaking the supervision and providing continuous assistance. I am grateful for his careful and patient guidance when I start this thesis. Zan is quite experienced in this field and is always willing to answer all my questions anytime during the day. Our fruitful discussions and his valuable inputs contributed markedly to the outcome of this master thesis.

Last but not least, I would like to thank my parents for their unconditional support in the last 25 years, and my girlfriend, for her support and company.

Abstract

Point cloud registration serves as a key component to a wide range of applications include 3D reconstruction and LiDAR odometry and mapping. Existing approaches focus on registration of point clouds with over 30% overlap, lowering this bound is of great benefit to real world applications. For example, under a 3D reconstruction setting, this would allow for reduced acquisition time and fine-grained local geometry reconstruction by leveraging more fragment pairs. In this thesis, we take the first attempt to study registration of low overlapping point clouds. This is challenging because: i). fully convolutional local geometric features are contaminated by irrelevant global context; ii). the inlier ratio of the over-complete correspondence set is upper-bounded by the low overlap ratio p. To mitigate such challenges, we propose a pruning module, which works by first aggregating local features to global patch-wise descriptors, then solving a partial assignment problem with a differentiable optimal transport layer. It directly attends the deep point cloud registration model to the overlap region and explicitly reduces the search space of feature matching. Our pruning module is light and efficient that can be easily plugged into deep point cloud registration models. We demonstrate improved performances from two deep learning models on two benchmarks, the improvement is significant in low overlap regions.

Contents

List of Figures

List of Tables

1

Introduction

Point cloud registration is a key task in point cloud processing, based on the consensus over common parts captured from different sensors or viewpoints, it allows us to reconstruct the whole object/scene from multiple point sets. It has wide applications, including autonomous driving [Zhang and Singh, 2015], 3D reconstruction [Choi et al., 2015], simultaneous localisation and mapping [Mur-Artal et al., 2015], virtual and augmented reality [Newcombe et al., 2011] etc.

Most works [Khoury et al., 2017] only address registration of point clouds with over 30% overlaps, as is shown in Table 1.1, starting from strong local geometric features 3DSN [Gojcic et al., 2019a], the geometric registration performances almost saturate on such benchmark. However, if we take a close look at the registration recall with respect to overlap ratios in Figure 1.1, we find the performances drop quickly in low overlap regions. Registration of point clouds with low overlap is important in two senses: on one hand, it can save acquisition time by reducing total scans required to capture the whole scene; on the other hand, it allows for more fragment pairs to be registered, capturing detailed local geometries and increasing the registration accuracy.

Most recently, there also emerged many end-to-end point cloud registration models [Wang and Solomon, 2019a, Yew and Lee, 2020]. However, they only report performances on highly overlapping object-centric synthetic datasets. Unlike synthetic data, real 3D point cloud scans are influenced by self-occlusion and substantial noise, and could have only a small degree of overlap between scans. The performances of these deep models on real data should be carefully tested.

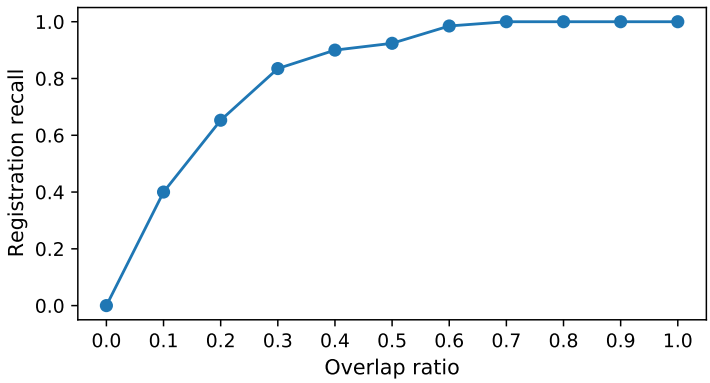

Figure 1.1.: *Registration recall w.r.t. overlap ratios. We use D3Feat [Bai et al., 2020] features and RANSAC [Fischler and Bolles, 1981] for model estimation.*

Table 1.1.: *Registration recall on 3DMatch benchmark [Zeng et al., 2017]. There are in total eight scenes, Kitchen has the most fragment pairs, thus the recall is less biased by randomness.*

	3DMatch	CGF	PPFNet	3DSN	FCGF	MultiReg	D3Feat	TEASER++
Kitchen	0.85	0.72	0.90	0.96	0.95	0.98	0.96	0.98
Home 1	0.78	0.69	0.58	0.88	0.91	0.93	0.93	0.92
Home 2	0.61	0.46	0.57	0.79	0.72	0.73	0.70	0.83
Hotel 1	0.79	0.55	0.75	0.95	0.93	0.97	0.95	0.97
Hotel 2	0.59	0.49	0.68	0.83	0.88	0.90	0.91	0.89
Hotel 3	0.58	0.65	0.88	0.92	0.81	0.89	0.85	0.94
Study	0.63	0.48	0.68	0.84	0.86	0.92	0.85	0.89
MIT Lab	0.51	0.42	0.62	0.76	0.82	0.78	0.69	0.84

1.1. Objective of this Thesis

The aim of this thesis is to develop a deep learning model for registration of 3D point clouds with low overlap. In this thesis, we try to answer the following two questions:

1. Why do current state-of-the-art learning-based methods fail to register point clouds with low overlap?
2. How can we improve learning-based methods for this task?

1.2. Structure of this Thesis

This thesis is organized as follows: Chapter 2 reviews several essential components in a correspondence-based, end-to-end point cloud registration pipeline. Chapter 3 presents the methodological approach employed in this these. In more detail, Section 3.1 introduces fundamental properties of point sets and basic point cloud processing. Feature extraction model is explained in Section 3.2, followed by *hard* and *soft* feature matching methods in Section 3.3. Section 3.4 talks about filtering networks that output associated weights for each correspondence, these weights together with the correspondence set are employed to get final pose estimation, explained in Section 3.5. Our main contributions are elaborated in Section 3.6 and Section 3.7. Chapter 4 presents experimental results and discussions. Eventually, this thesis is summarized in Chapter 5 with conclusions and outlooks.

2

Related Work

This chapter reviews several key components in a complete point cloud registration pipeline. First, we introduce deep learning models on 3D data, which is fundamental to the whole pipeline. Second, we discuss both hand-crafted and learned feature extraction models that generate meaningful feature descriptors from local geometries. Third, we talk about learned outlier filters that classify feature matches into inliers and outliers. Finally, we present end-to-end deep learning models that aimed for point cloud registration.

2.1. Deep learning in 3D

3D data have multiple representations, they can be represented by multiple images with different viewing directions [Su et al., 2015, Kalogerakis et al., 2017, Li et al., 2020], as voxel grids [Maturana and Scherer, 2015, Qi et al., 2016, Choy et al., 2019b], point clouds [Qi et al., 2017a, Qi et al., 2017b, Thomas et al., 2019], meshes [Defferrard et al., 2016, Gkioxari et al., 2019], or implicit functions [Park et al., 2019, Mescheder et al., 2019]. Here, we limit ourselves to point clouds and voxel grids.

2.1.1. Point cloud-based methods

Pointnet [Qi et al., 2017a] was a seminal work for deep learning directly from point clouds. The basic idea is to apply a shared bank of MLPs to the coordinates and attributes of each individual point, as approximations of point kernels to extract point-wise features, then use global max-pooling to abstract to a global representation in a permutation-invariant manner. By design, Pointnet does not capture local structures induced by the metric space the points live in, making it difficult to deal with geometric detail. To overcome this limitation, the follow-up version

Pointnet++ [Qi et al., 2017b] applies Pointnet recursively on a nested partitioning of the input point set, to hierarchically aggregate local information into a compact and fine-grained representation. Pointnets rely on MLPs to process point clouds individually, the receptive field is limited even with a few set abstraction layers, harming the downstream tasks like semantic segmentation. More recently, there have been some attempts to design convolutions directly over point clouds [Thomas et al., 2019, Atzmon et al., 2018, Li et al., 2018]. These methods rely on Euclidean space these points live in to build neighborhood and design spatial kernels, they aggregate local context in a better manner and have shown strong performances on semantic segmentation [Thomas et al., 2019][1] and geometric registration [Bai et al., 2020].

2.1.2. Voxel-based methods

Voxels are a straight-forward 3D generalisation of pixels, but point clouds from 3D sensors are sparse by nature. Instead of applying 3D convolutions on the full volumetric occupancy grid [Maturana and Scherer, 2015], sparse CNNs store the non-empty voxels as sparse tensors and only performs convolutions on such sparse coordinate lists. Graham *et al.* [Graham, 2014] introduce a CNN which takes sparsity into account, but is limited to small resolution (80^3 voxels in their experiments) due to the decrease in sparsity after repeated convolution. To deal with the dilation of non-zero activations, Graham *et al.* [Graham and van der Maaten, 2017] advocate the strategy to store the convolution output only at occupied voxels. Hackel *et al.* [Hackel et al., 2018] explore feature sparsity by selecting only a fixed number of the highest activations. Choy *et al.* [Choy et al., 2019a] introduce MinkowskiEngine, an open-source auto-differentiation library that extends [Graham and van der Maaten, 2017] to 4D spatio-temporal perception. It also proposes hybrid kernels with predefined sparsity patterns to mitigate the exponential increase of parameters. Sparse convolutions have shown strong performances on geometric registration [Choy et al., 2019b, Gojcic et al., 2020], image matching [Rocco et al., 2020], object detection [Gwak et al., 2020], and scene recognition [Huang et al., 2020].

2.2. Feature extraction

2.2.1. Hand-crafted features

Extracting discriminative and robust 3D local descriptors is a fundamental task in the field of 3D computer vision, as this is usually the first step for downstream applications like point cloud registration, 3D reconstruction and LiDAR odometry. 3D point clouds are usually captured by different range cameras or scanners at different viewpoints, in order to match them correctly under arbitrary spatial transformation, rotation invariance is a desired key property of these 3D local descriptors. In general there are two strategies to achieve this. The first is to estimate a unique local reference frame(LRF) and then transform the local patch before feature extraction to obtain a canonical representation, such LRF is typically based on the eigen-

[1] Please refer to two benchmarks: Semantics3d: `http://www.semantic3d.net/view_results.php?chl=1` and Scannet: `http://kaldir.vc.in.tum.de/scannet_benchmark` for more details.

value decomposition of the sample co-variance matrix of the local patch. Methods include SHOT [Tombari et al., 2010b] and USC [Tombari et al., 2010a] rely on this strategy. However, such LRF is non-unique [Guo et al., 2013] by its simplest definition, the second strategy is to rely on the intrinsically rotation-invariant point pair features(PPF) as basis for feature extraction. These methods include FPFH [Rusu et al., 2009] and PPFH [Rusu et al., 2008]. Despite significant progress in the field, hand-crafted 3D local descriptors never reached the performance of their 2D counterparts[2].

2.2.2. Learned features

Inspired by the success of deep learning methods in image matching [Anastasiya Mishchuk, 2017, Dusmanu et al., 2019], there has also been rapid progress in learning local geometric representations from 3D data. CGF [Khoury et al., 2017] and LORAX [Elbaz et al., 2017] utilise deep learning models to reduce dimensions of their handcrafted descriptors. 3DMatch [Zeng et al., 2017] voxelizes the region around each key-point and uses an modified AlexNet to learn local descriptors supervised by a contrastive loss. PPFNet [Deng et al., 2018b] samples different local patches and applies Pointnet to extract local context each, then it relies on max-pooling to fuse global context from all patches and concatenate it to the local context such that each local descriptor is also aware of the global context, local descriptors are finally optimised by a N-tuple loss. PPF-FoldNet [Deng et al., 2018a] achieves feature rotation invariance by using pure point pair features and is learned in a self-supervised way. 3DFeat-Net [Yew and Lee, 2018] jointly learns 3D feature detectors and descriptors for point cloud matching with weak supervision. 3DSmoothNet [Gojcic et al., 2019a] relies on LRF to canonicalise local patches and represents each patch using voxelised smoothed density values, compact local descriptors are finally matched in a siamese deep learning architecture. This patch-based processing is inefficient because intermediate network activations are not reused across adjacent patches. FCGF [Choy et al., 2019b] overcomes this drawback by formulating local feature extraction in a fully convolutional manner, it builds on sparse convolutions and is 290 times faster than 3DSmoothNet [Gojcic et al., 2019a]. USIP [Li and Lee, 2019] is a self-supervised method, it minimises the probabilistic chamfer loss of distances of each point in one point cloud with its nearest neighbor in the other rotated and translated point cloud. D3Feat [Bai et al., 2020] is a combination of D2-Net [Dusmanu et al., 2019] and KPConv [Thomas et al., 2019], it simultaneously outputs salience scores and local descriptors for each point. Li *et al.* [Li et al., 2020] propose an end-to-end learnable local multi-view descriptors for 3D Point Clouds by integrating a differentiable renderer into a neural network.

2.3. Learned outlier filters

In a correspondence set, for each correspondence(point pair) after ground truth transformation, inlier(*good* correspondence) is spatially close to each other and outlier(*bad* correspondence) is distant by at least a margin τ. In spite of powerful learned 3D local descriptors [Choy et al., 2019b], the over-complete correspondence sets obtained by feature matching

[2]Please refer to [Guo et al., 2016] for a comprehensive comparison of these hand-crafted descriptors.

are still seriously contaminated by outliers [Yang et al., 2020]. Heuristics include Lowe's ratio test [Lowe, 2004] and mutual check are effective, deterministic and data-independent preliminary steps to filter outliers. They are usually followed by RANSAC [Fischler and Bolles, 1981] to determine final inlier set and estimate transformation model. In this section, we introduce several deep learning models aimed at filtering outliers.

2.3.1. Outlier filtering networks

Ranftl and Koltun [Ranftl and Koltun, 2018] point out that inlier and outlier distributions over an over-complete correspondence set admit regularity that can be learned. They introduce a deep learning model to directly classify inliers and outliers and thus robustly estimating the fundamental matrices. A concurrent work from Yi *et al.* [Moo Yi et al., 2018] proposes *context normalisation* to incorporate global context in permutation-equivalent networks(Pointnets), we term this *PointCN*. It has the same form as *instance normalisation* [Ulyanov et al., 2016] while plays a different role, aiming to learn the hidden scene geometry and camera motion of image pairs to determine the inlier/outlier matches in multiple view geometry. Pointnets used in PointCN operates on each point individually and thus can't capture local context, OANet [Zhang et al., 2019] overcomes this by mapping correspondences to a fewer nodes, then it applies the weight-sharing perceptrons on the spatial dimension to establish spatial correlations. ACNet [Sun et al., 2019] embeds the attention mechanism into PointCN to exclude the influence of outliers during normalisation.

2.3.2. Differentiable RANSACs

RANdom SAmple Consensus(RANSAC) [Fischler and Bolles, 1981] is a classic and has been widely for model estimation [Hartley and Zisserman, 2003] since it was proposed, there have been many follow-up works including DEGENSAC [Chum et al., 2005], GC-SAC [Barath and Matas, 2018] and MAGSAC [Barath et al., 2019]. It's so strong that even today, with careful hyper-parameter tuning, is difficult to beat [Jin et al., 2020]. In deep learning era, there have also been some attempts to differentiate and directly optimise it for downstream applications. DSAC [Brachmann et al., 2017] mimics the behavior of RANSAC, it differentiates the minimal set proposal and consensus scoring function by substituting the deterministic selection of the highest scoring model hypothesis with a probabilistic selection over the softmax distribution of scores. The scoring network is prone to overfitting, in a follow-up work, Brachmann *et al.* [Brachmann and Rother, 2018] replace it with a soft inlier count. In addition, it uses the Shannon entropy to measure scoring distribution broadness and keep it within a reasonable range to stabilise the training process. More recently, Brachmann *et al.* [Brachmann and Rother, 2019] extend [Brachmann and Rother, 2018] to learn hypothesis search in a principled fashion that directly optimize an arbitrary task loss during training.

2.3.3. Neighborhood consensus networks

Neighborhood consensus network [Rocco et al., 2018] builds on the classic idea of disambiguating feature matches using semi-local constraints [Bian et al., 2017]. It formulates the problem of filtering outliers as identifying spatially consistent matches by analysing neighborhood consensus patterns in 4D space, and such *consensus* could be reflected by the activations from convolutional operations. Convolutions in 4D space is computationally expansive and thus not capable of high-accuracy localised correspondences, Rocco *et al.* [Rocco et al., 2020] identifies the sparse nature of such 4D space and overcomes this drawback using sparse convolutions. Two concurrent works from Choy *et al.* [Choy et al., 2020a, Choy et al., 2020b] interpret this as *recognising geometric patterns in high-dimensional space* and further extend 4D convolutions for image matching to 6D convolutions for 3D point cloud matching.

2.4. Deep point cloud registration

There have been many attempts to do point cloud registration in an end-to-end fashion. PointNetLK [Aoki et al., 2019] is a combination of PointNet [Qi et al., 2017a] and Lucas & Kanade algorithm [Lucas et al., 1981], it unrolls them into a single trainable recurrent deep neural network. Instead of explicitly encoding such alignment process, PCRNet [Sarode et al., 2019] uses Pointnets to directly regress the 3 translation and 4 normalized rotation quaternion. Similarly, CorsNet [Kurobe et al., 2020] regresses the per point offset from one point cloud to another. DeepICP [Lu et al., 2019] takes initial pose estimations from on-board sensors of autonomous-driving cars to establish rough correspondence sets and then uses Pointnets to extract features, final SVD-based refinement is built on the topk matches. DCP [Wang and Solomon, 2019a] uses DGCNN [Wang et al., 2019] to first extract independent features, then it uses a symmetric Transformer [Vaswani et al., 2017] module to learn the co-contextual information between two point clouds, final pose is also solved using singular value decomposition. A follow-up work PRNet [Wang and Solomon, 2019b] extends DCP [Wang and Solomon, 2019a] to partial-to-partial registration by using a *topk* operator to determine points in common before feature matching. RPM-Net [Yew and Lee, 2020] approach the partial-to-partial assignment by introducing a differentiable Sinkhorn layer [Sinkhorn and Knopp, 1967] with extended dustbin rows and columns to deal with non-common points. Gojcic *et al.* [Gojcic et al., 2020] propose the first end-to-end learnable, multiview 3D point cloud algorithm. In its simple form of pairwise registration, it uses FCGF [Choy et al., 2019b] for efficient feature extraction, modified OANet [Zhang et al., 2019] for outliers filtering and weighted SVD for final pose estimation. DGR [Choy et al., 2020a] shares FCGF and weighted-SVD layer with [Gojcic et al., 2020] in common, while it uses a 6D convolutional networks to classify inliers and outliers.

3

Methodology

In this chapter, we first introduce several key components of our model in detail, then we present our complete model aimed for registration of point clouds with low overlap. We mainly follow the pair-wise registration proposed in [Gojcic et al., 2020]. However, due to the nature of low overlapping point clouds, we get much lower inlier ratios and this is problematic for filtering networks. To mitigate this problem, we propose a pruning scheme to sample a few patch pairs that are subsets of input point clouds and are of high probability to cover the overlap region. These patches are further used to determine correspondences, filter outliers and estimate poses. We term this *pruning scheme* as sampling subset $\mathcal{I}$ from input $\mathcal{P}$ could be interpreted as pruning $\mathcal{P}\backslash\mathcal{I}$. Through this operation, we explicitly reduce the search space of feature matching and *potentially*[1] increase the overlap ratio, transforming our originally difficult task into a relatively easier one.

This chapter is organized as follows:

1. We introduce properties of point sets, point cloud sampling methods and 3-dimensional Euclidean space.
2. We present Fully Convolutional Geometric Feature(FCGF) [Choy et al., 2019b] for efficient feature extraction.
3. We talk about Learning to Find Good Correspondence(LTFGC) [Moo Yi et al., 2018] and Order-Aware Network(OANet) [Zhang et al., 2019] for outlier filtering, and their extensions to point cloud registration.
4. We show the Kabsch algorithm [Kabsch, 1976] for 3D pose estimation.
5. We propose a pruning scheme to sample optimal subsets.
6. We present our complete model.

[1] We say *potentially* because current model could fail to sample subsets that cover overlap regions some time.

3.1. Preliminaries

3.1.1. Point sets

Since the introduction of AlexNet [Krizhevsky et al., 2012], deep learning in the form of convolutional neural networks has became the standard approach in image recognition tasks and has almost revolutionized the whole computer vision field in recent 10 years. However, deep learning on 3D data lags far behind. There are several reasons: i). For 3D convolution, memory consumption increases cubically with voxel resolution and current hardware can't afford that yet; ii). Point cloud, as the most common representation of 3D data, is unstructured and imposes great challenge for deep learning algorithms. The former is now partially fixed by sparse convolution [Choy et al., 2019a] but due to its specific batch processing, researchers still encounter many challenges when "reinventing" deep learning models in 3D. Pointnet [Qi et al., 2017a] addresses the latter problem by s shared bank of MLPs and global pooling layer. Here, we introduce several properties of point clouds that should be carefully taken into account when designing algorithms to learn meaningful features:

1. **Unordered**: A point cloud of N points are represented by a point set of cardinality N, such set is unordered and our model should be invariant to the $N!$ permutations of the input set.
2. **Neighborhood**: Each point in a point cloud is not isolated, it forms a subset with its close neighbors measured by Euclidean distance space. Our model could extract meaningful local context from interactions within such subsets.
3. **Sparsity**: Point clouds are mostly sampled from the surface of scenes and objects, they are sparse by nature. A straight-forward discretization of point clouds into 3D grids would lead to massive redundant space and is not efficient. An efficient algorithm should also take such sparsity into consideration.
4. **Invariance under transformations**: Rotating and translating a point cloud of a table doesn't change the fact that the point cloud represents a table. The learned representation should be invariant under certain Euclidean transformation.

3.1.2. Iterative Farthest Point Sampling

Pointnets [Qi et al., 2017a] and dynamic graph neural network [Wang et al., 2019] require fixed number of points as input. Given input point clouds $P = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x_n}\}$ of size n, we can use iterative farthest point sampling (FPS) [Qi et al., 2017b] to choose a subset $P' = \{\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, ..., \mathbf{x}_{i,m}\}$ of arbitrary size m, such that $\mathbf{x}_{i,j}$ is the most distant point (Euclidean distance) from the set $\mathcal{M} = \{\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, ..., \mathbf{x}_{i,j-1}\}$ with regard to the rest $\mathcal{P}\backslash\mathcal{M}$. Compared with random sampling, it has better coverage of the entire point set given the same size. We took the implementation from here[2], the same scene with different levels of sparsity are shown in Figure 3.1.

[2] https://github.com/rusty1s/pytorch_cluster

Figure 3.1.: *Scene representation with different FPS settings. It starts with 1024 points and doubles each time until it reaches the raw resolution.*

3.1.3. Voxelization

Sparse convolution is a voxel-based method and requires the input point clouds to be voxelised into discrete 3D grids. When the voxel resolution is high, it's common that several points with different labels/colors fall into the same voxel. MinkowskiEngine [Choy et al., 2019a] randomly chooses one point and assigns its label to the voxel by default. SparseConv [Graham and van der Maaten, 2017] supports randomly picking one label or averaging the labels.

3.1.4. 3-dimensional Euclidean space

Angle-axis representation A 3D rotation matrix could be represented using angle-axis representation, which is composed of a single rotation angle θ and its axis $\boldsymbol{\omega}$. For a rotation matrix $\mathbf{R}$, the angle-axis $(\theta, \boldsymbol{\omega})$ representation[3] is:

$$\theta = \arccos\left(\frac{\mathrm{Tr}(\mathbf{R}) - 1}{2}\right), \boldsymbol{\omega} = \frac{1}{2\sin\theta}\begin{bmatrix} \mathbf{R}(3,2) - \mathbf{R}(2,3) \\ \mathbf{R}(1,3) - \mathbf{R}(3,1) \\ \mathbf{R}(2,1) - \mathbf{R}(1,2) \end{bmatrix} \tag{3.1}$$

Euler angle and rotation matrix Given the Euler angle representation of a rotation matrix, we can derive the rotation matrix as a sequence of three rotations, one about each principle axis,

[3] https://en.wikipedia.org/wiki/Axis\T1\textendashangle_representation

then it has the form:

$$\begin{aligned}\mathbf{R} &= \mathbf{R}_z(\phi)\mathbf{R}_y(\theta)\mathbf{R}_x(\psi) \\ &= \begin{bmatrix} \cos\theta\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \cos\theta\sin\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi \\ -\sin\theta & \sin\psi\cos\theta & \cos\psi\cos\theta \end{bmatrix} \\ &= \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}\end{aligned} \tag{3.2}$$

To retain the Euler angles from the rotation matrix, starting with $\mathbf{R}_{31}$, we find $\mathbf{R}_{31} = -\sin\theta$, then we get $\theta = -\sin^{-1}(\mathbf{R}_{31})$. There are two distinct values of θ that satisfy the equation. Therefore, we can determine two possible values for Euler angles from rotation matrix[4].

3.2. Feature extraction

Feature extraction provides each point/voxel associated features that describe local and global context, it is the key to feature matching and downstream applications that require correspondences between two images pairs or point clouds. Given a point cloud $\mathbf{P} \in \mathbb{R}^{N\times 3}$, a feature extraction model $\mathcal{F}$ predicts feature $\mathbf{f}_i \in \mathbb{R}^C$ for each point $\mathbf{p}_i \in \mathbb{R}^3$ in $\mathbf{P}$. Ideally, within the same point cloud, the proximity of neighboring points in data space is preserved in feature space; between two point clouds, spatially neighboring points after ground truth transformation are closer to each other than the rest points in feature space. In consideration of both speed and accuracy, we choose Fully Convolutional Geometric Feature(FCGF) [Choy et al., 2019b] as our feature extraction backbone. As its name indicated, it is a fully convolutional neural network that processes the whole point cloud in a single forward pass. In addition, it is a voxel-based method, it uses sparse convolution to build deep networks and is thus memory efficient. In this section, we first introduce sparse convolution, then we talk about metric learning which is fundamental to feature learning, finally we present FCGF [Choy et al., 2019b].

3.2.1. Sparse convolution

Analogous to a dense 3D tensor($H \times W \times C$, H and W represent image height and width, C represents feature channels) used by 2D convolution, a standard input representation for 3D convolution is a dense 4D tensor: three spatial dimensions and one feature dimension. However, most voxels are empty due to the inherent sparsity. Such sparsity[5] increases with the voxel resolution, while the memory also increases cubically, causing huge waste of both storage and computation. Instead, sparse convolution operates on sparse tensors.

[4] https://www.gregslabaugh.net/publications/euler.pdf

[5] Sparsity is defined as the ratio of empty voxels

Formally, we can represent a sparse tensor for input 3D data as coordinates $\mathbf{C}$ and associated features $\mathbf{F}$:

$$\mathbf{C} = \begin{bmatrix} b_1 & x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots & \vdots \\ b_N & x_N & y_N & z_N \end{bmatrix}, \mathbf{F} = \begin{bmatrix} \mathbf{f}_1^T \\ \vdots \\ \mathbf{f}_N^T \end{bmatrix} \tag{3.3}$$

where (x_i, y_i, z_i) is the i-th 3D coordinate and b_i is the batch index which provides an additional dimension for batch processing. $\mathbf{f}_i$ is the feature associated with the i-th coordinate. In the most simple setting, $\mathbf{f}_i$ is set to be 1 to indicate the occupancy of the sparse tensor.

Contrast to normal convolutions, in intermediate layers, a sparse tensor has a feature at location u only if the corresponding coordinate is present in $\mathbf{C}$, this avoids the dilation of non-empty voxels in each layer and keeps the same sparse pattern.

3.2.2. Metric learning

The goal of metric learning is to learn a function $\mathcal{F}_\theta(x) : \mathbb{R}^C \to \mathbb{R}^D$ that keep the neighboring proximity in both input and output space. Let $\mathcal{D}(x, y) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be a metric function measuring distance in the embedding space. Metric learning begins with two constraints: similar features have to be close to each other: $\mathcal{D}(\mathbf{f}_i, \mathbf{f}_j) \to 0 \quad \forall (i, j) \in \mathcal{P}$ and dissimilar features have to be at least a margin away: $\mathcal{D}(\mathbf{f}_i, \mathbf{f}_j) > m \quad \forall (i, j) \in \mathcal{N}$, where $\mathcal{P}$ and $\mathcal{N}$ denote *positive* and *negative* sets that are used to train the network[6]. In practice, we look for the positive pairs that are most distant and negative pairs that are most close, which we term *hard positive mining* and *hard negative mining* respectively. Lin *et al.* [Lin et al., 2015] show that the constraints for positive pairs could lead to over-fitting and propose a margin-based loss for positive pairs:

$$\mathcal{L}(\mathbf{f}_i, \mathbf{f}_j) = I_{ij} \left[\mathcal{D}(\mathbf{f}_i, \mathbf{f}_j) - m_+\right]^2 + \bar{I}_{ij} \left[m_- - \mathcal{D}(\mathbf{f}_i, \mathbf{f}_j)\right]^2 \tag{3.4}$$

where m_+ and m_- are margins for positive and negative pairs, $I_{ij} = 1$ if $(i, j) \in \mathcal{P}$ and 0 if $(i, j) \in \mathcal{N}$, and $\bar{\cdot}$ is the NOT operator. We can also convert the constraint $\mathcal{D}(\mathbf{p}, \mathbf{p}_+) < m_+, \mathcal{D}(\mathbf{p}, \mathbf{p}_-) > m_-$ into a triplet loss: by adding the two constraints, we get $\mathcal{D}(\mathbf{p}, \mathbf{p}_+) + m_- < \mathcal{D}(\mathbf{p}, \mathbf{p}_-) + m_+$, then set $m = m_- - m_+ > 0$, we get $\mathcal{D}(\mathbf{f}, \mathbf{f}_-) - D(\mathbf{f}, \mathbf{f}_+) > m$, finally we reach our triplet loss:

$$\mathcal{L}(\mathbf{f}, \mathbf{f}_+, \mathbf{f}_-) = \left[m + \mathcal{D}(\mathbf{f}, \mathbf{f}_+) - \mathcal{D}(\mathbf{f}, \mathbf{f}_-)\right]^2 \tag{3.5}$$

This loss makes sure that, given an *anchor* point $\mathbf{f}$, the projection of a *positive* point $\mathbf{p}_+$ is closer to the anchor's projection than that of a *negative* point $\mathbf{p}_-$by at least a margin m. Analogous to the hard positive and negative samples for contrastive loss, the mining of hard triplets is also crucial for training. As the dataset gets larger, the possible number of triplets grows cubically, the neural network relatively quickly learns to correctly map most trivial triplets, rendering a large fraction of triplets uninformative. On the other hand, selecting only the hardest triplets would introduce outliers unproportionally and make neural network unable to learn "normal" associations. Hence it is common to only mine moderate negatives and positives. On the other hand, embedding all the training samples and computing all pairwise distances are expensive,

[6]Note that features are all normalised, otherwise by scaling them, the margin could be arbitrarily small or big.

Hermans *et al.* [Hermans et al., 2017] propose the *Batch Hard mining*, which selects the hardest positive and hardest negative samples for each anchor within the batch when forming the triplets to compute loss. Such formation can be considered moderate triplets, since they are the hardest within a small subset of the data, which is exactly what is best for learning with the triplet loss.

3.2.3. Fully Convolutional Geometric Features

Network architecture FCGF [Choy et al., 2019b] feature is a fully convolutional geometric feature built on Minkowski Engine [Choy et al., 2019a] which is an auto-differentiation library that provides support for sparse convolutions and implements all essential deep learning layers. FCGF uses a UNet [Ronneberger et al., 2015] structure with skip connections and residual blocks to extract sparse-convolutional features. The network architecture is shown in Figure 3.2, it consists of a contracting path to gradually capture local and global context, and a symmetric expanding path that enables precise localisation. Skip connections are applied in expanding path which concatenates the low-level features from contracting path to high level features from up-sampling operations. Specifically, the contracting path starts with a convolutional layer with $7 \times 7 \times 7$ convolution kernels to first abstract big local context, then four strided convolutional layers with residual blocks aggregate richer local context and increase the number of feature channels to be 256, we term this *bottleneck*. The expanding path gradually up-samples the high level features from bottleneck with transposed convolution and finally reaches the input resolution. Our final FCGF feature is a compact 32-dimensional feature that is powerful and efficient for feature matching.

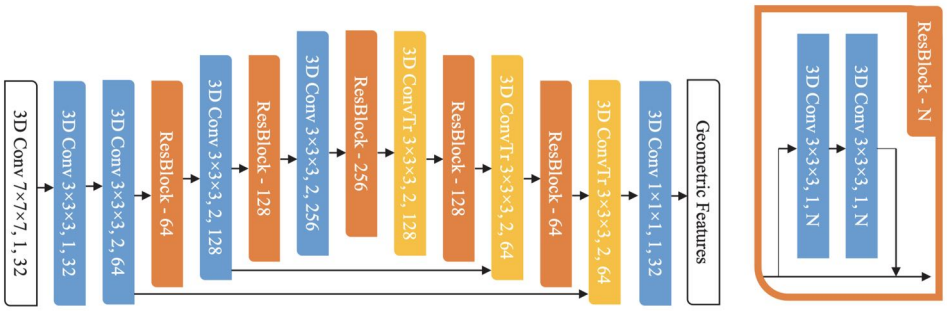

Figure 3.2.: *FCGF network architecture. Each block is characterized by three parameters: kernel size, stride, and channel dimensionality. Figure is taken from [Choy et al., 2019b].*

Loss function Due to inherent properties of a fully convolutional network, metric learning is challenging in two aspects. **First**, batch samples don't follow an independent and identical distribution(i.i.d.) and it is crucial to filter out false negative samples in hard-negative mining. Under a traditional metric learning setting, for example, learning an embedding for object classification task, the feature extraction model consumes the input images/point clouds and outputs a single feature as the global descriptor of the input. These global descriptors are only dependent

on the inputs, as long as the inputs are not highly correlated[7], they follow an i.i.d. However such i.i.d. assumption doesn't hold for a fully convolutional network. The feature extraction model takes the point cloud and outputs associated features for each point, spatially neighboring points produce adjacent features that are also correlated in feature space, thus, hard-negative mining could find features adjacent to anchors, and they are false negatives. **Second**, the number of features used in a fully convolutional network is orders of magnitude larger than in standard metric learning problems and thus it is not feasible to use all pairwise distances within a batch. For instance, FCGF [Choy et al., 2019b] generates 40k features for a pair of scans (this increases proportionally with the batch size and voxel resolution) while a mini-batch in traditional metric learning has only around 1k features.

In consideration of these two challenges, FCGF uses hardest-contrastive and hardest-triplet losses. First, FCGF samples anchor points and a set of candidates for mining per scene. Then, FCGF mines the hardest negatives $\mathbf{f}_i^-, \mathbf{f}_j^-$ for both $\mathbf{f}_i$ and $\mathbf{f}_j$ in a positive pair $(\mathbf{f}_i, \mathbf{f}_j)$ as in Figure 3.3 and remove false negatives that fall within a certain radius from the corresponding anchor. Finally, FCGF uses the pairwise loss for the mined quadruplet $(\mathbf{f}_i, \mathbf{f}_j, \mathbf{f}_i^-, \mathbf{f}_j^-)$ and form the fully-convolutional contrastive loss:

$$\begin{aligned}\mathcal{L}_C = &\sum_{(i,j)\in\mathcal{P}} \left[\mathcal{D}\left(\mathbf{f}_i, \mathbf{f}_j\right) - m_+\right]^2 / |\mathcal{P}| + \lambda_n I_i \left[m_- - \min_{k\in\mathcal{N}} \mathcal{D}\left(\mathbf{f}_i, \mathbf{f}_k\right)\right]^2 / |\mathcal{P}_i| \\ &+\lambda_n I_j \left[m_- - \min_{k\in\mathcal{N}} \mathcal{D}\left(\mathbf{f}_j, \mathbf{f}_k\right)\right]^2 / |\mathcal{P}_j|\end{aligned} \tag{3.6}$$

The normalization term for negative pairs simply averages all valid negative pairs equally. λ_n is a weight for negative losses and FCGF simply uses 0.5 to weight positives and negatives equally. Similarly, FCGF constructs the triplet loss:

$$\begin{aligned}\mathcal{L}_T = \frac{1}{Z} \sum_{(i,j)\in\mathcal{P}} \Bigg(&I\left(i, k_i\right) \left[m + \mathcal{D}\left(\mathbf{f}_i, \mathbf{f}_j\right) - \min_{k\in\mathcal{N}} \mathcal{D}\left(\mathbf{f}_i, \mathbf{f}_k\right)\right] \\ &+I\left(j, k_j\right) \left[m + \mathcal{D}\left(\mathbf{f}_i, \mathbf{f}_j\right) - \min_{k\in\mathcal{N}} \mathcal{D}\left(\mathbf{f}_j, \mathbf{f}_k\right)\right]\Bigg)\end{aligned} \tag{3.7}$$

The above finds the hardest negatives for both $\mathbf{f_i}, \mathbf{f_j} \in \mathcal{P}$. FCGF follows [Hermans et al., 2017] and uses non-squared loss to mitigate features from collapsing into a single point. In practice, false negatives indicator $I((i, k_i)$ and I_i is implemented using efficient hash-based filtering.

Implementation details On 3DMatch dataset [Zeng et al., 2017], FCGF applies different data augmentations for both scans in a pair include random scaling $\in [0.8, 1.2]$ rotating $\in [0°, 360°]$. Such rotation augmentation is a simple and effective way to make FCGF invariant to relative camera pose changes. Using colors or normals as sparse tensor features leads to overfitting as 3DMatch dataset is not diverse enough, despite the data augmentation tricks. In this end, FCGF creates an input sparse tensor with coordinates from a scan and 1-vectors as features. The voxel size is set to be 2.5 centimeters and the geometric feature dimension is 32. FCGF trains the networks for 100 epochs using SGD [Rosenblatt, 1958]. It starts with a learning rate 0.1 and decays with an exponential learning rate schedule with $\gamma = 0.99$.

[7]Batch samples are normally constructed by random sampling from a large and diverse dataset, the batch size is relatively small to the dataset. Therefore, batch samplers are usually not correlated.

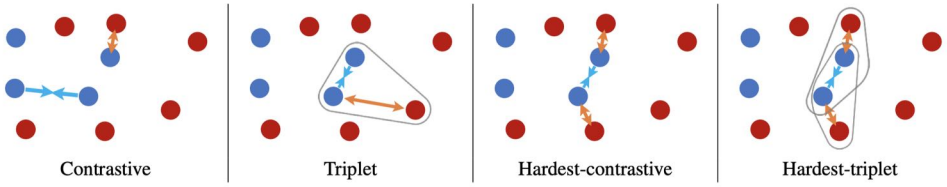

Figure 3.3.: *Sampling and negative-mining strategy for each method. Figure is taken from [Choy et al., 2019b].*

3.3. Feature matching

Feature matching is the step to establish correspondences between two point clouds. For two features $\mathbf{f_i}$ and $\mathbf{f_j}$ from two point clouds, their squared distance: $\mathcal{D}(\mathbf{f_i}, \mathbf{f_j})^2 = \|\mathbf{f_i} - \mathbf{f_j}\|^2 = \|\mathbf{f_i}\|^2 + \|\mathbf{f_j}\|^2 - 2\mathbf{f_i}^T\mathbf{f_j}$. Since FCGF [Choy et al., 2019b] features are normalized such that $\|\mathbf{f_i}\| = \|\mathbf{f_j}\| = 1$, we get $\mathcal{D}(\mathbf{f_i}, \mathbf{f_j}) = 2 - 2\mathbf{f_i}^T\mathbf{f_j}$. We know that $\mathcal{D}(\mathbf{f_i}, \mathbf{f_j}) \to 0$ when two features are similar and $\mathcal{D}(\mathbf{f_i}, \mathbf{f_j}) > m_-$ when two features are dissimilar, thus the inner product $\mathbf{f_i}^T\mathbf{f_j}$ serves as a good measure of the similarity between two features, this score is high when two features are similar and vice versa. In general, we have two ways to determine the final correspondence, they are the *hard* argmax sampler and the *soft* softmax sampler.

3.3.1. Argmax sampler

The behavior of argmax sampler resembles that of nearest neighbor search. In a toy example shown in Figure 3.4, after computing the feature similarity scores between $\mathbf{p_i}$ and candidates $\{\mathbf{q_i}\}, i = 0, 1, 2, 3, 4$, we assign the candidate $\mathbf{q_4}$ to $\mathbf{p_i}$ as its correspondence, as it has the highest similarity score(smallest distance in feature space). Mathematically, this equals to one-hot encoding of the argmax function over the similarity scores vector:

$$m(\mathbf{p_i}, \mathbf{Q}) = \text{onehot}[\text{argmax}(\mathbf{F}_Q\mathbf{f_i}^T)]\mathbf{Q} \tag{3.8}$$

where $m(\mathbf{p}_i, \mathbf{Q})$ is the function to determine the correspondence of $\mathbf{p}_i$ from $\mathbf{Q}$, $\mathbf{F}_Q \in \mathbb{R}^{N\times C}$ represents the features, $\mathbf{Q} \in \mathbb{R}^{N\times 3}$ represents the coordinates.

One drawback of argmax sampler is that it is non-differentiable. Due to the *argmax* operation, the gradients could only pass the argmax-indexed feature during back propagation, this could lead to high variance and cause instability during training and is thus problematic in an end-to-end model. In practice, argmax sampler usually is only used at inference stage to produce sharp correspondences. At training stage, it is replaced by a soft version to improve stability.

3.3.2. Softmax sampler

To alleviate the non-differentiability of argmax sampler, softmax sampler uses a probabilistic approach that generates a "soft map" from one point cloud into the other. In the toy sample shown in Figure 3.4, we assign an associate weight to each candidate $\mathbf{q_i}$ based on its distance to the query $\mathbf{p_i}$ in feature space, these weights are all non-negative and normalized such that

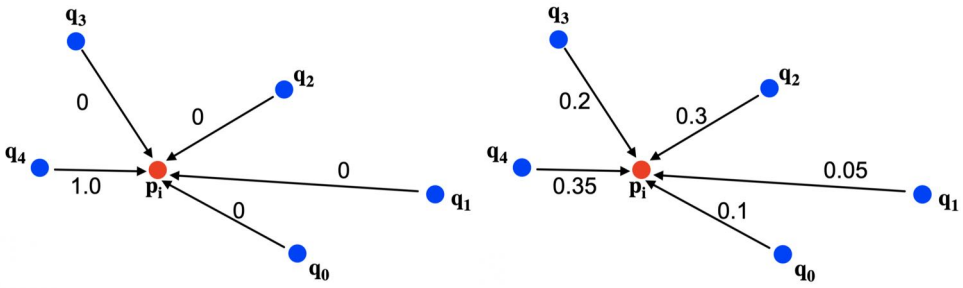

Figure 3.4.: *Toy example of Argmax(left) and Softmax(right) sampler.*

the summation is 1. The final correspondence is the weighted average of all these candidates. Formally, these weights are given by a *softmax* function over the similarity scores:

$$m(\mathbf{p_i}, \mathbf{Q}) = \text{softmax}(\mathbf{F}_Q \mathbf{f_i}^T)\mathbf{Q} \tag{3.9}$$

Softmax function is differentiable everywhere, thus can be used during training to increase stability. However, because softmax sampler uses a "soft map", the weights are spread out that the correspondence is not sharp. In the optimal situation[8], each query has at most one correspondence and its weight should be 1[9]. To control this *hardness* or *smoothness* of the soft map, we can introduce a temperature parameter λ:

$$m(\mathbf{p_i}, \mathbf{Q}) = \text{softmax}(\mathbf{F}_Q \mathbf{f_i}^T/\lambda)\mathbf{Q} \tag{3.10}$$

As is shown in Figure 3.5, in the limit $\lambda \to 0$, the soft map converges to a deterministic nearest neighbor(NN) search [Plötz and Roth, 2018]. In practice, this usually happens when the network is confident with its prediction at the end of the training. In addition, we also find that a proper initialisation of the temperature value λ[10] is important to finally get sharp correspondence.

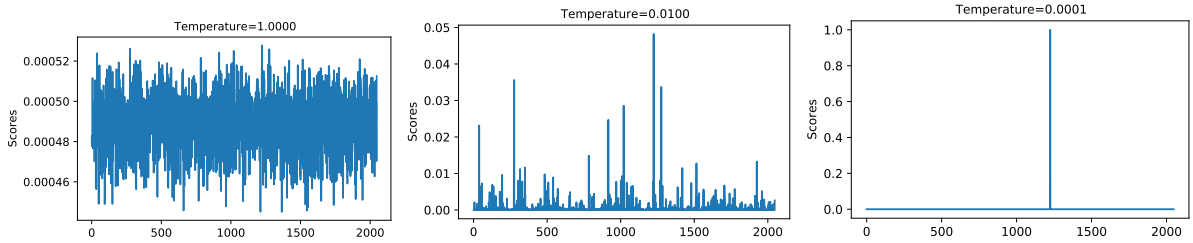

Figure 3.5.: *Influence of temperature value in a soft mapping.*

3.3.3. Gumbel-Softmax sampler

PRNet [Wang and Solomon, 2019b] proposes to use Gumbel-Softmax [Jang et al., 2016] to obtain sharp correspondence at the beginning of training while preserving the differentiability. It

[8] Non-maximum suppression is applied such that there's no ambiguity in feature matching.
[9] Argmax sampler can help but only at inference stage.
[10] Initialise with a small value like 0.01.

works by sampling the softmax weight vector in forward pass and back-propagating through the full weight vector. Formally, the Gumbel–Softmax mapping function is given by:

$$m\left(\mathbf{p}_i, \mathbf{Q}\right) = \text{onehot}\left[\arg\max_j \text{softmax}\left(\frac{\mathbf{F}_Q \mathbf{f_i}^T + g_{ij}}{\lambda}\right)\right] \tag{3.11}$$

where $(g_{i1}, ..., g_{ij}, ..., g_{iN})$ are i.i.d. samples drawing from Gumbel(0,1). The distribution of Gumbel(0,1) is shown in Figure 3.6. The map in Equation 3.11 is not differentiable due to the discontinuity of argmax operator. Instead, PRNet [Wang and Solomon, 2019b] relies on straight-through gradient estimator [Bengio et al., 2013] which yields biased sub-gradient estimates with low variance during back propagation. Specifically, it uses Equation 3.10 to compute loss.

In practice, we find that with normalised FCGF [Choy et al., 2019b] features, the magnitude of similarity scores is much lower than that of Gumbel noise, rendering our features uninformative in the numerator term. Even with un-normalised features, we don't find it useful compared to softmax sampler with temperature. We list it here simply for the sake of completeness of this thesis.

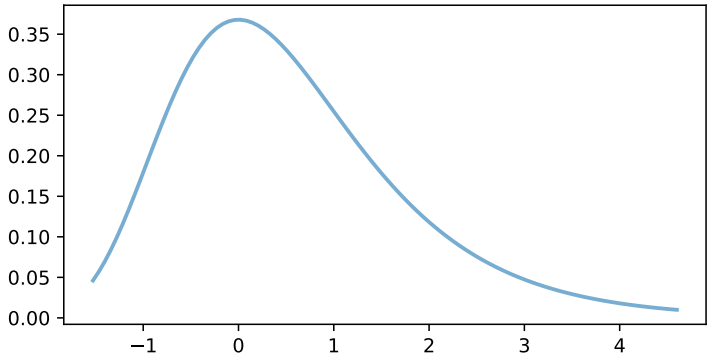

Figure 3.6.: *Probability density function of Gumbel(0,1).*

3.4. Outlier filtering

Outlier filtering filters out outliers from feature matching. For a correspondence set $\mathcal{O}$, a filtering network $\mathcal{F}$ predicts a weight $w \in [0, 1]$ that indicates the likelihood to be an inlier.

There are two reasons for the presence of outliers in an over-complete[11] correspondence sets. The **first** is that by nature, only points in the overlap region could have a good correspondence, rendering many correspondences outliers, especially for point clouds with low overlap. The **second** is that even our features are trained such that the neighboring proximity is kept in feature space, they are still far from perfect and this causes ambiguities during feature matching.

[11] We say "over-complete" because we predict correspondences for all the points in one point cloud. The knowledge of "overlap region" is not accessible to us without ground truth transformation, otherwise we would only need to predict for points in the overlap region.

Consider two fully overlapping point clouds of the same size, in the best case, the score matrix is a permutation matrix that all the elements $\in \{0, 1\}$ and each row and columns sums to 1. However, what we could actually get is a fuzzy matrix that is shown in Figure 3.7, such ambiguities bring us many outliers.

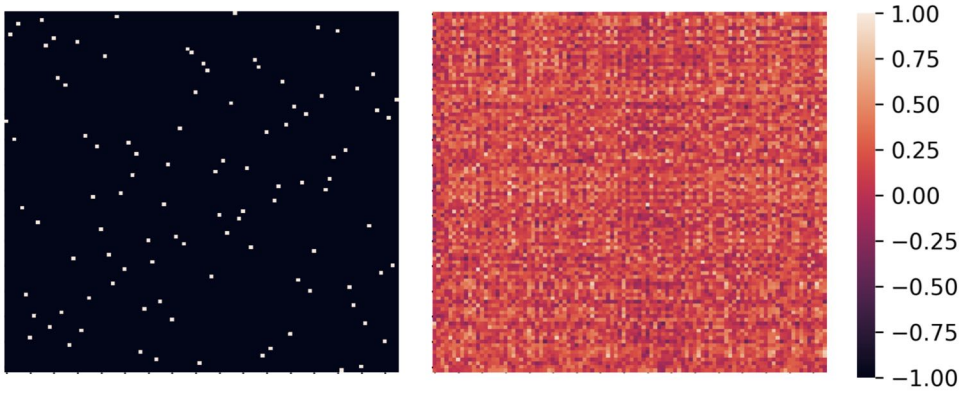

Figure 3.7.: *Visualization of score matrix. Left: desired score matrix. Right: actual score matrix from FCGF [Choy et al., 2019b].*

To filter out outliers, one classic solution is first applying heuristics include ratio test [Lowe, 2004] and mutual check, then using RANSAC [Fischler and Bolles, 1981] for robust estimation. RANSAC works by alternating between hypothesis set proposal and selection based on the sample consensus. The expected iterations k required to get an outlier-free hypothesis set is as follows:

$$k = \frac{log(1-p)}{log(1-w^n)} \tag{3.12}$$

where w is the inlier ratio, n is the minimum number of samples required to estimate a model(cardinality of the hypothesis set), p is be the desired probability that the RANSAC algorithm provides an outlier free hypothesis set. In point cloud registration, to find at least $n = 3$ good correspondences with $p = 99.9\%$, the required iterations $k \approx 860$ when $w = 0.2$, this number is much bigger when the inlier ratio is smaller($k = 55000$ when $w = 0.05$). Therefore, it's time-consuming to get reliable estimation for low overlapping point clouds. In addition, the hypothesis selection is non-differentiable that it can't be directly used to train a deep neural network.

In this section, we present PointCN [Moo Yi et al., 2018, Ranftl and Koltun, 2018] and OANet [Zhang et al., 2019]. These two methods were originally proposed for image-based epipolar geometry tasks and were adopted to point clouds for deformation analysis [Gojcic et al., 2019b] and pair-wise registration [Gojcic et al., 2020]. With *context normalisation*, they turn out to be efficient and can generalise well to unseen scenes.

3.4.1. PointCN

DFE [Ranftl and Koltun, 2018] and LTFGC [Moo Yi et al., 2018] are concurrent works that exploit combining Pointnet [Qi et al., 2017a] and context normalization [Ulyanov et al., 2016] to

classify putative correspondences, which they refer to as *PointCN*. PointCN takes N normalised correspondences between two 2D points $(4 \times N)$ as input, and predicts one weight for each correspondence that encodes the likelihood to be inlier. Weights-sharing Perceptrons(Pointnet) process each correspondence independently and thus are invariant to permutations and flexible to the size of input. Context normalisation integrates first and second moment statistics of intermediate activations within each image pair to encode global context, such as scene geometry and camera motion.

Context normalisation In practice, context normalisation applies the same operation as instance bormalisation [Ulyanov et al., 2016]. Let $o_i^l \in \mathbb{R}^{C^l}$ be the output of layer l for correspondence i, where C^l is the number of neurons in layer l, we normalised o_i^l to be:

$$\mathrm{CN}(o_i^l) = \frac{o_i^l - \mu^l}{\gamma^l} \tag{3.13}$$

where,

$$\mu^l = \frac{1}{N}\sum_i^N o_i^l, \gamma^l = \sqrt{\frac{1}{N}\sum_i^N (o_i^l - \mu^l)^2} \tag{3.14}$$

By normalising each Perceptron's output across correspondences, but separately for each image pair, we efficiently encode global context into the distribution of the feature maps, rendering our model aware of scene geometry and camera motion and thus can distinguish between inliers and outliers.

Network architecture The network architecture of LTFGC [Moo Yi et al., 2018] is shown in Figure 3.8. There are in total 12 Resnet blocks [He et al., 2016], each block contains two sequential sub-blocks that consists of: a Perceptron with 128 neurons, an instance normalisation layer, a batch normalisation [Ioffe and Szegedy, 2015] layer and a ReLU [Hahnloser et al., 2000] non-linear activation function. Finally, a ReLU followed by a tanh function is applied to force the output weights in the range [0,1). As is shown in Figure 3.9, such truncated tanh is preferred over a sigmoid function as it produces more binary weights to completely remove outliers.

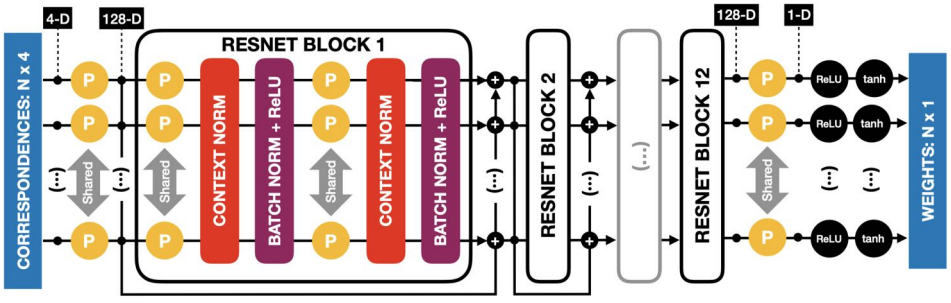

Figure 3.8.: *Network architecture of Learning to Find Good Correspondence. Figure is taken from [Moo Yi et al., 2018].*

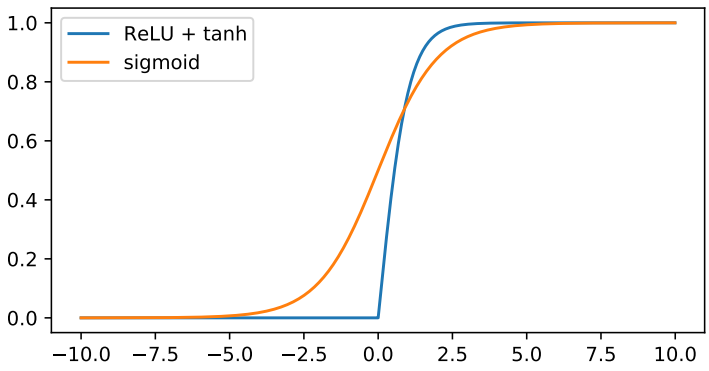

Figure 3.9.: *Comparison between truncated tanh and sigmoid functions.*

Iterative form To refine the estimation, DFE [Ranftl and Koltun, 2018] formulates this task as an iteratively re-weighted least-squares(IRLS) problem. It has in total 5 consecutive estimation modules. As is shown in Figure 3.10, each module consists of two steps: the first step takes correspondences and associated weights from last module to estimate a model, the second step takes correspondences and residuals with respect to the previously estimated model and generates new weights.

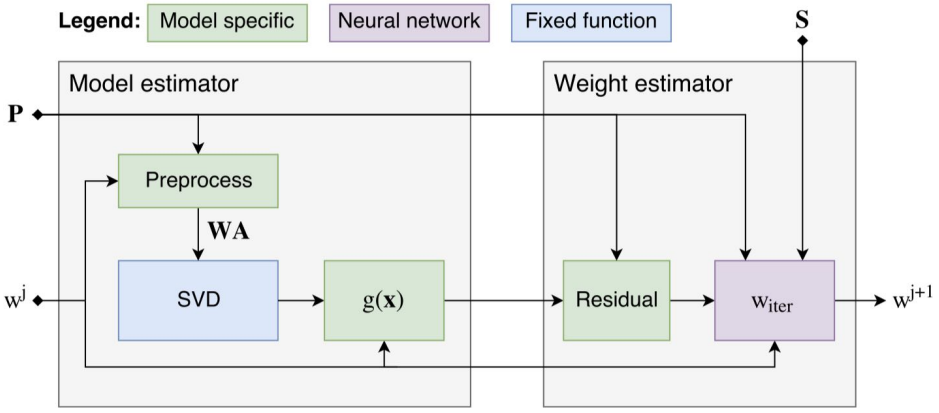

Figure 3.10.: *DFE estimation module. Figure is taken from [Ranftl and Koltun, 2018].*

Loss function We can form two loss terms. **First**, predicting the likelihoods of being inliers can be seen as a binary classification task, our model classifies correspondences as inliers or outliers. Therefore, we can apply the binary cross-entropy loss $\mathcal{L}_c$ to supervise the classification. **Second**, the model estimation module is a weighted least-squares problem and has a closed-form solution that is also differentiable, thus we can form a regression loss $\mathcal{L}_e$ over the final estimated fundamental or essential matrix.

In practice, LTFGC [Moo Yi et al., 2018] trains the filtering network with a hybrid loss function:

$$\mathcal{L}_x(\Phi) = \alpha\mathcal{L}_c\left(\Phi, \mathbf{x}\right) + \beta\mathcal{L}_e\left(\Phi, \mathbf{x}\right) \tag{3.15}$$

where Φ are the network parameters and x is a set of putative correspondences, α and β weight two loss terms.

Classification loss is a binary cross-entropy loss:

$$\mathcal{L}_c\left(\Phi, \mathbf{x}\right) = \frac{1}{N}\sum_{i=1}^{N} \gamma^i \mathcal{H}\left(y^i, \mathcal{S}\left(o^i\right)\right) \tag{3.16}$$

where γ^i is a weight to balance positive and negative examples, o^i is un-normalised logit, $\mathcal{S}$ is the sigmoid function used in conjunction with binary cross entropy $\mathcal{H}$.

Regression loss is a mean squared error loss function:

$$\mathcal{L}_e\left(\Phi, \mathbf{x}\right) = \min\left\{\left\|\mathbf{E}^* \pm \overline{\mathbf{E}}\right\|^2\right\} \tag{3.17}$$

where $\mathbf{E}^*$ is the ground truth essential matrix and $\overline{\mathbf{E}}$ is the estimation.

Classification loss alone proves to be robust while adding the regression loss as a regulariser further improves the performance. Detailed ablation studies are shown in Figure 3.11: *Ours* denotes optimising with two loss terms, *Essential* denotes optimising with regression loss only, *Classification* denotes optimising with classification loss only, *Direct* denotes directly regressing the Essential matrix. *Essential* and *Direct*, which do direct regression without classification, perform worse than *Classification*, as long as the number of keypoints is sufficient. *Ours* outperforms all the others by combining both classification and regression, by a margin of 12-24%. Note that the difference is larger for smaller error thresholds, suggesting that both *Essential* and *Direct* are learning the general trend of the dataset without providing truly accurate poses.

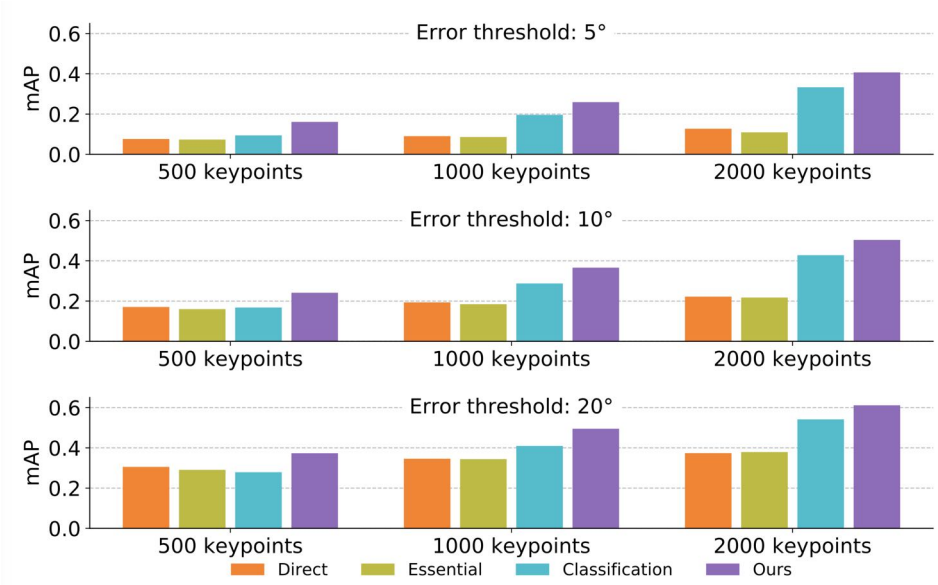

Figure 3.11.: *Comparison between different optimisation strategies. Figure is taken from [Moo Yi et al., 2018].*

3.4.2. Order-Aware Network

Order-Aware Network(OANet) [Zhang et al., 2019] is different from LTFGC [Moo Yi et al., 2018] in two ways. First, it introduces differentiable pooling and unpooling layers to capture local context which is missed in PointCN. Second, it introduces spatial correlation layer which is complementary to PointCN to better capture global context. With these two modifications, it improves outlier filtering results.

Local context One drawback of PointCN is that it applies MLPs to each correspondence individually, hence it can not capture local context, which has been shown important in Pointnet++ [Qi et al., 2017b]. OANet [Zhang et al., 2019] introduces a differentiable pooling layer to map correspondences to clusters. Specifically, to map N correspondences to M nodes, where $N > M$, it directly learns a weight matrix $\mathbf{S}_{pool} \in \mathbb{R}^{N \times M}$ such that:

$$\mathbf{X}_{l+1} = \mathbf{S}_{pool}^{T} \mathbf{X}_{l} \tag{3.18}$$

where $\mathbf{X}_{l+1} \in \mathbb{R}^{M \times D}, \mathbf{X}_{l} \in \mathbb{R}^{N \times D}$. Each row of $\mathbf{S}_{pool}^{T}$ is normalised such that each node in $\mathbf{X}_{l+1}$ can be seen as weighted average of all the samples in $\mathbf{X}_{l}$. Different nodes attend to different local context by learning different weight vectors. Up-sampling M nodes to N correspondences is done by learning another weight matrix $\mathbf{S}_{unpool}$ in the same fashion.

Spatial correlation Another drawback of PointCN is that it encodes the global context by mean and variance of features, this overlooks the underlying complex relationships between correspondences. OANet [Zhang et al., 2019] overcomes this by introducing a differentiable spatial correlation layer.

As is shown in Figure 3.12, N features first go through the differentiable pooling layer to form M nodes which encode different local contexts, then six order-aware filtering networks correlate the nodes to better model the global context. The order-aware filtering block is shown in Figure 3.13, it has two PointCN modules and one spatial correlation layer in between. The motivation for such design is that the pooled nodes are in a canonical order, their relations are potentially useful. Spatial correlation layer applies weight-sharing perceptrons directly on the spatial dimension to encode such relations. Note that PointCN applies weight-sharing perceptrons on the channel dimension, therefore these two operations are orthogonal to each other and are thus complementary. Therefore, the order-aware filtering module combines both of them to better capture the global context.

3.4.3. Extension to point cloud registration

It is straight-forward to extend PointCN [Ranftl and Koltun, 2018, Moo Yi et al., 2018] or OANet [Zhang et al., 2019] to point cloud registration. Apart from the input difference(changed from $4 \times N$ to $6 \times N$), to estimate the final model, the weighted 8-point algorithm is also replaced by weighted kabsch algorithm [Kabsch, 1976], we will detail the latter in Section 3.5. In addition, we follow [Gojcic et al., 2020] to design our filtering network, it consists of an initial step and a refine step, the initial step takes the correspondences set and predicts

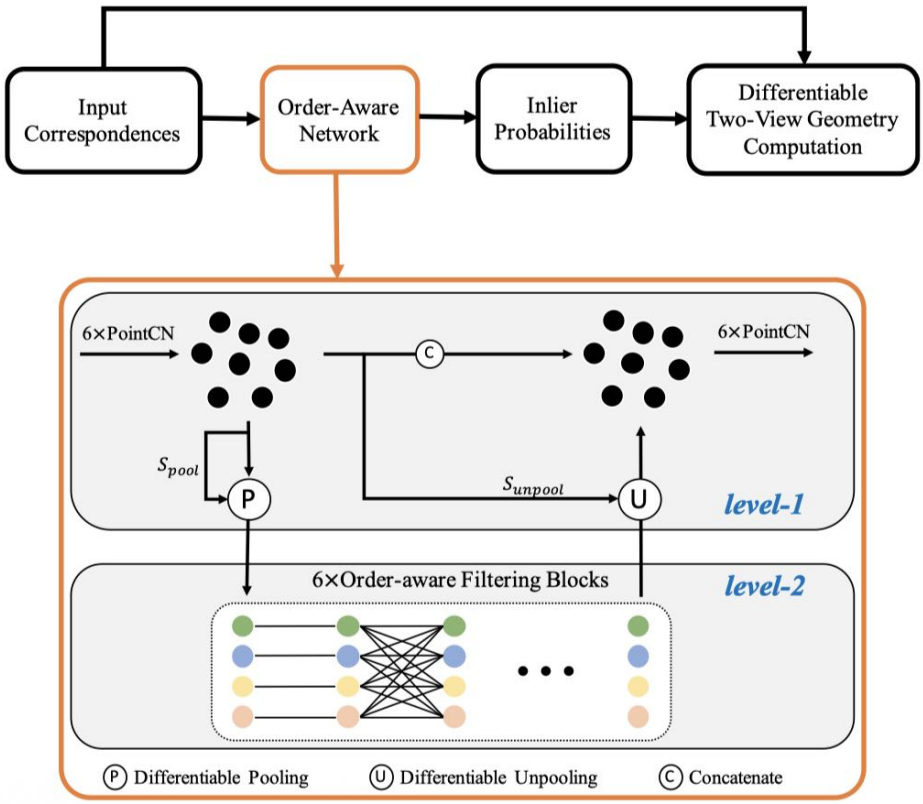

Figure 3.12.: *Network architecture of OANet. Figure is taken from [Zhang et al., 2019].*

weights, the refine step takes the correspondence set together with their weights and residuals with respect to the previously estimated model, then predicts new weights. According to the inlier/outlier classification recall and precision, we see a consistent improvement from the refine step. In consideration of the computational load, we only refine once.

3.5. Transformation estimation

Point cloud registration can be formulated as a weighted least-squares problem. Given a set of correspondences and associated weights, the optimal transformation will minimise their L_2 distances:

$$\hat{\mathbf{R}}, \hat{\mathbf{t}} = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i}^{N_p} w_l \|\mathbf{R}\mathbf{p_i} + \mathbf{t} - m(\mathbf{p}_i, \mathbf{Q})\|^2, \quad \mathbf{p}_i \in \mathbf{P} \tag{3.19}$$

where $m(\mathbf{p_i}, \mathbf{Q})$ determines correspondence of $\mathbf{p}_i$ from $\mathbf{Q}$. For good correspondences, $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ minimise their L_2 distances in Euclidean space, while for bad correspondences, their weights are close to 0 and influences are thus discarded in our minimisation problem. Arun *et al.* [Arun et al., 1987] propose a closed-form solution to Equation 3.19, which we refer to as Kabsch algorithm.

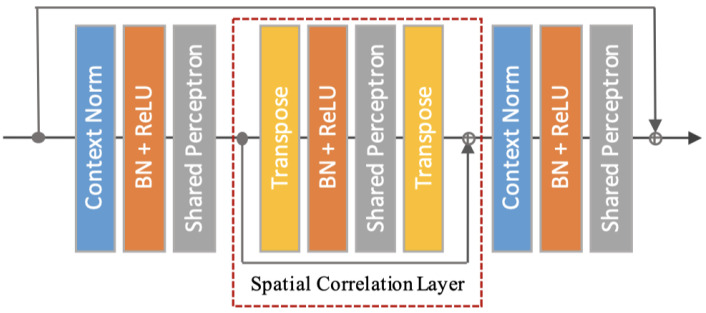

Figure 3.13.: *Order-Aware Filtering block. Figure is taken from [Zhang et al., 2019].*

3.5.1. Kabsch algorithm

Given two point clouds $\mathbf{P}$ and $\mathbf{Q}$, together their feature matching function m and weights $\{w_l\}$, we summarise the closed-form solution to Equation **??**eq: kabsch as follows:

we first determine weighted centroids $\overline{\mathbf{p}}$ and $\overline{\mathbf{q}}$ of $\mathbf{P}$ and $\mathbf{Q}$:

$$\overline{\mathbf{p}} := \frac{\sum_{l=1}^{N_\mathbf{P}} w_l \mathbf{p}_l}{\sum_{l=1}^{N_\mathbf{P}} w_l}, \quad \overline{\mathbf{q}} := \frac{\sum_{l=1}^{N_\mathbf{Q}} w_l \mathbf{q}_l}{\sum_{l=1}^{N_\mathbf{Q}} w_l} \tag{3.20}$$

then, the centered point clouds can be computed as:

$$\widetilde{\mathbf{p}}_l := \mathbf{p}_l - \overline{\mathbf{p}}, \quad \widetilde{\mathbf{q}}_l := \mathbf{q}_l - \overline{\mathbf{q}}, \quad l = 1, \ldots, N \tag{3.21}$$

let $\widetilde{\mathbf{P}} \in \mathbb{R}^{N\times 3}$ and $\widetilde{\mathbf{Q}} \in \mathbb{R}^{N\times 3}$ denote the centered point clouds, then a weighted co-variance matrix $\mathbf{S}$ can be computed as:

$$\mathbf{S} = \widetilde{\mathbf{P}}^T \mathbf{W} \widetilde{\mathbf{Q}} \tag{3.22}$$

where $\mathbf{W} = diag(w_1, ..., w_N)$ is the diagonal weight matrix. Considering the singular value decomposition $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, the solution is given by:

$$\hat{\mathbf{R}} = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det\left(\mathbf{V}\mathbf{U}^T\right) \end{bmatrix} \mathbf{U}^T \tag{3.23}$$

where $\det$ denotes computing the determinant and is used here to avoid creating a reflection matrix. Finally, $\hat{\mathbf{t}}$ is computed as:

$$\hat{\mathbf{t}} = \overline{\mathbf{q}} - \hat{\mathbf{R}}\overline{\mathbf{p}} \tag{3.24}$$

Kabsch algorithm is differentiable [Paszke et al., 2017], thus can be plugged into an end-to-end model for robust estimation.

3.5.2. Geometric loss

In an end-to-end point cloud registration model, given ground-truth and estimated transformations, we have several options to measure the deviation and supervise the training.

Frobenius norm [Gojcic et al., 2020, Wang and Solomon, 2019b, Aoki et al., 2019] use Frobenius norm of relative transformation matrices:

$$\mathcal{L} = \left\| \mathbf{R}^* - \hat{\mathbf{R}} \right\|_F + \left\| \mathbf{t}^* - \hat{\mathbf{t}} \right\|_2 \tag{3.25}$$

where the Frobenis norm of a matrix A is defined as follows:

$$\|\mathbf{A}\|_F := \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} a_{ij}^2} = \|\operatorname{vec}(\mathbf{A})\|_2 = \sqrt{\operatorname{trace}\left(\mathbf{A}^\top \mathbf{A}\right)} \tag{3.26}$$

L_2 norm [Choy et al., 2020a] uses the rotation angle deviation and L_2 norm of the translation vector as geometric loss:

$$\begin{aligned} \mathbf{L}_{\text{rot}}(\hat{\mathbf{R}}) &= \arccos \frac{\operatorname{Tr}\left(\hat{\mathbf{R}}^T \mathbf{R}^*\right) - 1}{2} \\ \mathbf{L}_{\text{trans}}(\hat{\mathbf{t}}) &= \left\| \hat{\mathbf{t}} - \mathbf{t}^* \right\|^2 \end{aligned} \tag{3.27}$$

L_1 norm [Yew and Lee, 2020] uses the L_1 distance between the source point cloud $\mathbf{X}$ transformed using ground-truth and estimated transformations:

$$\mathcal{L} = \frac{1}{J} \sum_{j}^{J} \left| (\mathbf{R}^* \mathbf{x}_j + \mathbf{t}^*) - \left(\hat{\mathbf{R}} \mathbf{x}_j + \hat{\mathbf{t}} \right) \right| \tag{3.28}$$

3.6. Patch-based pruning

As is illustrated in introduction part, with the above-mentioned techniques, state-of-the-art methods fail to get descent results on low overlapping point clouds. The main reason is that low overlapping point clouds inherently have fewer good correspondences, this makes it more challenging for filtering networks to correctly determine inliers from an over-complete correspondence set that is highly contaminated by outliers. To mitigate this problem, we aim to directly increase the overlap ratio, this could be achieved by removing points outside of overlap region and thus decreasing the denominator in Equation 3.29.

$$\text{overlap ratio} = \frac{\text{\# points in the overlap region}}{\text{\# points in the overlap region + \# points outside of overlap region}} \tag{3.29}$$

However, such overlap information is only accessible when given the ground truth transformation. On the other hand, for two subsets $\mathbf{P}_s$ and $\mathbf{Q}_s$ of point clouds $\mathbf{P}$ and $\mathbf{Q}$ that fall into the overlap region, they have similar appearance, and such appearance information is invariant to Euclidean transformation[12]. Therefore, our goal becomes mining the subsets of two point clouds that fall in the overlap region, this could be done by analysing their appearance similarities.

In the following parts, we first present a simple yet effective clustering and set aggregation methods, then we introduce an optimal transport layer used to solve the partial assignment problem.

[12]Rotating and translating the scan of a bunny doesn't change the fact that it looks like a bunny.

3.6.1. Sampling and clustering

The first step is building a few subsets(patches) for each point cloud. The coverage of each patch should not be too big nor too small. If it's too big, in the extreme case, as big as the input point cloud, then we simply go back to the original problem. If it's too small, it becomes difficult to distinguish patches as most of them would be small planar areas.

We build patches in two steps. Consider a point cloud $\mathbf{P}$, **first**, we apply iterative farthest point sampling(FPS) [Qi et al., 2017b] to determine K centroids. Compared with random sampling, FPS has better coverage of the entire point set. This is especially beneficial when we only sample a few centroids. **Second**, we form K patches using k-nearest-neighbor search. The size M of each patch depends on the size of input and the number of patches. In practice, we determine this by running a gird search.

Our proposed FPS and k-nn is fully deterministic and only contain 2 hyper-parameters K and M. This simple strategy turns out to be sufficient for this task. Other advanced methods include geometrically homogeneous partition [Landrieu and Simonovsky, 2018], which partitions the point cloud into geometrically simple yet meaningful shapes, and these shapes don't overlap with each other. However, it requires to compute hand-crafted geometric features for each point beforehand and is thus computationally heavier. In addition, different partitions have different sizes, making it less flexible to be incorporated into highly optimized batch processing under PyTorch framework [Paszke et al., 2017]. We leave this choice to future work.

3.6.2. Set aggregation

Set aggregation computes a compact global descriptor for each patch. We have two options for this: **one** is directly learning a global descriptor from coordinates, for example the bottleneck in Pointnet [Qi et al., 2017a]; **another** is aggregating the point-wise FCGF [Choy et al., 2019b] features. We choose the latter because FCGF [Choy et al., 2019b] features are computed on whole point cloud, it has richer global and local context that is helpful in disambiguate similar patches.

Consider a patch with associated coordinates $\mathbf{C}_s \in \mathbb{R}^{M\times 3}$ and features $\mathbf{F}_s \in \mathbb{R}^{M\times C}$, our goal is aggregating them into a global descriptor $\mathbf{G}_s \in \mathbb{R}^{C'}$. One key property of set aggregation is permutation-invariance, because $\mathbf{C}_s$ and $\mathbf{F}_s$ are order-less. Global max-pooling and average-pooling are typical permutation-invariant operations that are independent of orders of input. However, they treat each element equally and are task-agnostic, are too simple to capture complex global context and adapting to downstream applications. Instead, we learn a data-dependent attentive pooling [Hu et al., 2020] layer to efficiently aggregate set features. Specifically, we first learn a linear layer $\mathbf{W}$ to map feature vectors to associate weights $\mathbf{s}$, then we apply another MLP to the weighted feature vector to generate our final global descriptor:

$$s_i = \psi(\mathbf{f}_i, \mathbf{W}), \quad \mathbf{G}_s = MLP(\sum_i^N s_i \cdot \mathbf{f}_i) \tag{3.30}$$

Actually, max-pooling and average-pooling could be seen as special cases of attentive pooling, where $\mathbf{s}$ equals to $[0, ..., 1, ..., 0]$ and $\mathbf{1}_M/M$ respectively and MLP becomes an identical layer.

Other more complex methods include NetVLAD [Arandjelovic et al., 2016]. The high level idea is first learning several centroids, then each centroid aggregates different local contexts, and finally they are concatenated to a global descriptor. We leave the ablation of NetVLAD [Arandjelovic et al., 2016] in our framework to future work.

3.6.3. Partial assignment

Partial assignment determines *topk* patch pairs to be used for registration, topk $< M$. We term this *partial assignment* because not all patches could find a match, only those intersect with the overlap region are likely to be selected. In practice, we rely on a differentiable Sinkhorn algorithm [Sinkhorn, 1964] to solve this.

Outlier bins In this partial assignment problem, some patches should be ignored as they can't find a match. This is achieved by assigning them to *dustbins* [DeTone et al., 2018]. Specifically, we first build $K \times K$ score matrix $\mathbf{S}$ from global descriptors $\{\mathbf{G}_s\}$, then we extend it to a $(K+1) \times (K+1)$ matrix $\overline{\mathbf{S}}$ that satisfies:

$$\begin{aligned} \overline{\mathbf{S}}_{K+1,i} &= \overline{\mathbf{S}}_{j,K+1} = z, 1 \leq i, j \leq K+1 \\ \overline{\mathbf{S}}_{:K,:K} &= \mathbf{S} \end{aligned} \tag{3.31}$$

where z is a learnable parameter that adapts to the magnitudes of $\mathbf{S}$ as training proceeds.

Optimal transport Given the augmented score matrix $\overline{\mathbf{S}}$, we aim to find a partial assignment matrix $\overline{\mathbf{P}} \in \mathbb{R}^{(K+1)\times(K+1)}$ which defines correspondences between two patch sets. It satisfies in total $2K+2$ constraints:

$$\begin{aligned} \overline{\mathbf{P}}_{(i,:)}\mathbf{1}_{K+1} = \overline{\mathbf{P}}^T_{(:,i)}\mathbf{1}_{K+1} = 1, \forall i, j \leq K \\ \overline{\mathbf{P}}_{(K+1,:)}\mathbf{1}_{K+1} = \overline{\mathbf{P}}^T_{(:,K+1)}\mathbf{1}_{K+1} = K \end{aligned} \tag{3.32}$$

Consider two patch sets $\mathcal{P}_s$ and $\mathcal{Q}_s$ of size K, i-th patch in $\mathcal{P}_s$ is assigned to j-th patch in $\mathcal{Q}_s$ if $\overline{\mathbf{P}}_{i,j}$ is the maximum along both i-th row and j-th column. i-th patch is ignored(assigned to dustbin) if $\overline{\mathbf{P}}_{i,K+1}$ is the maximum along i-th row.

Determining $\overline{\mathbf{P}}$ is a quadratic assignment problem, it's NP-hard and requiring expensive and complex solvers to find an optimal solution. However, from the optimal transport theory [Cuturi, 2013], by adding an entropy regulariser, the assignment matrix can be obtained by solving:

$$\min \langle \overline{\mathbf{S}}, \overline{\mathbf{P}} \rangle - \lambda \mathcal{H}(\overline{\mathbf{P}}) \tag{3.33}$$

where$\langle \cdot, \cdot \rangle$ is the Frobenius dot product, $\langle \overline{\mathbf{S}}, \overline{\mathbf{P}} \rangle = \sum_{i,j} \overline{\mathbf{S}}_{i,j} \overline{\mathbf{P}}_{i,j}$ and $\mathcal{H}(\cdot)$ is an entropy regularization term defined as $\mathcal{H}(\overline{\mathbf{P}}) = -\sum_{i,j} \overline{\mathbf{P}}_{i,j} \log \left(\overline{\mathbf{P}}_{i,j}\right)$.

Sinkhorn's algorithm According to Sinkhorn [Sinkhorn, 1964], Equation 3.33 constrained by Equation 3.32 can be solved using the method of Lagrangian multipliers, the final solution is given in an iterative form[13]. In practice, $\overline{\mathbf{P}}$ is found by alternating between row normalisation and column normalisation of $\overline{S}$. However, in a deep learning model, this is found to suffer from numerical overflow and is unstable [Dang et al., 2020]. Instead, we follow [Sarlin et al., 2020, Yew and Lee, 2020] and apply such normalisation in log-space. The algorithm is in Algo. 1, where $\text{logsumexp}(A)_i = \log \sum_j (\exp(A_{i,j}))$.

Algorithm 1: Log-domain Sinkhorn's algorithm.

Input: score matrix $\mathbf{S} \in \mathbb{R}^{K \times K}$, iterations n, dustbin value z, count $j = 0$
Output: $\overline{\mathbf{P}} \in \mathbb{R}^{K+1,K+1}$

Initialise $\overline{\mathbf{S}} \in \mathbb{R}^{K+1,K+1}$ by padding z to $\mathbf{S}$;
Initialise $\mathbf{t} = [-\log(2K), -\log(2K), ..., -\log(2K), \log(K) - \log(2K)]^T \in \mathbb{R}^{K+1,1}$;
Initialise $\mathbf{u} = \mathbf{v} = \mathbf{0}_{K+1}$;
while *j<n* **do**
  $\mathbf{u} = \mathbf{t} - \log \text{sumexp}\left(\overline{\mathbf{S}} + \mathbf{1}_{K+1}\mathbf{v}^T\right)$
  $\mathbf{v} = \mathbf{t} - \log \text{sumexp}\left(\overline{\mathbf{S}} + \mathbf{u}\mathbf{1}_{K+1}^T\right)$
  $j = j + 1$
end
$\overline{\mathbf{P}} = \overline{\mathbf{S}} + \mathbf{u} + \mathbf{v} + \log(2K)$
$\overline{\mathbf{P}} = \exp(\overline{\mathbf{P}})$

With Sinkhorn's algorithm, we obtain a doubly stochastic matrix $\overline{\mathbf{P}}$[14]. This enforces reciprocity of the matches, it is similar to mutual check but in a soft manner.

Loss function Sinkhorn's algorithm is deterministic and has no learnable parameters. To supervise our set aggregation model and the dustbin parameter z, inspired by [Sarlin et al., 2020], we minimise the negative log-likelihood loss:

$$\mathcal{L} = -w \sum_{(i,j)\in\mathcal{M}} \log \overline{\mathbf{P}}_{i,j} - (1-w)(\sum_{i\in\mathcal{I}} \log \overline{\mathbf{P}}_{i,K+1} - \sum_{j\in\mathcal{J}} \log \overline{\mathbf{P}}_{K+1,j}) \tag{3.34}$$

where $\mathcal{M}$ is the ground truth matches set, $\mathcal{I}$ and $\mathcal{J}$ are unmatched sets, w is a weight to balance positive and negative samplers. This supervision aims at simultaneously maximizing the precision and recall of the matching.

We build the ground truth $\mathcal{M}, \mathcal{I}$ and $\mathcal{J}$ in the following ways. We first compute the overlap ratios between two sets of patches, and denote such overlap ratio matrix as $\mathbf{O} \in \mathbb{R}^{K \times K}$. Pair $(i, j) \in \mathcal{M}$ if and only if i). $\mathbf{O}_{i,j} > 50\%$, ii). $\mathbf{O}_{i,j}$ is the maximum along both row and column. If none of $\{(i, j)\}, \forall i$ is assigned to $\mathcal{M}$, then i is assigned to $\mathcal{I}$. $\mathcal{J}$ is constructed in the same manner.

[13] We provide detailed derivatives in Appendix A.1

[14] In probability and combinatorics, a doubly stochastic matrix (also called bistochastic), is a square matrix A of non-negative real numbers, each of whose rows and columns sums to 1. This is actually an abuse of doubly stochastic matrix, as is shown in Equation 3.32, only first K rows and columns sum 1

3.7. Our complete model

Our complete model proposed for registration of 3D point clouds with low overlap is shown in Figure 3.14. Consider two point clouds $\mathbf{P}$ and $\mathbf{Q}$, they are first voxelised into 3D grids, then we run feature extraction model to get FCGF features [Choy et al., 2019b]. Next, we sample N points and associated features from each point cloud for efficient batch processing. Afterwards, patch-based pruning is applied to sample *topk* patch pairs. Each patch pair is processed independently by our filtering network to determine inliers and outliers. Finally, we use the *inlier ratio* as in indicator to select *one* patch pair from in total *topk* pairs and then run Kabsch algorithm over it to give final pose estimation.

We find inlier ratio to be a reliable indicator to determine the most overlapping patch pair. The intuition is that a patch pair with high overlap has more ground truth inliers in an over-complete correspondence set, our filtering network is thus less influenced by outliers and is more *confident* when predicting inliers. We measure such *confidence* by thresholding the predicted logits and term this inlier ratio.

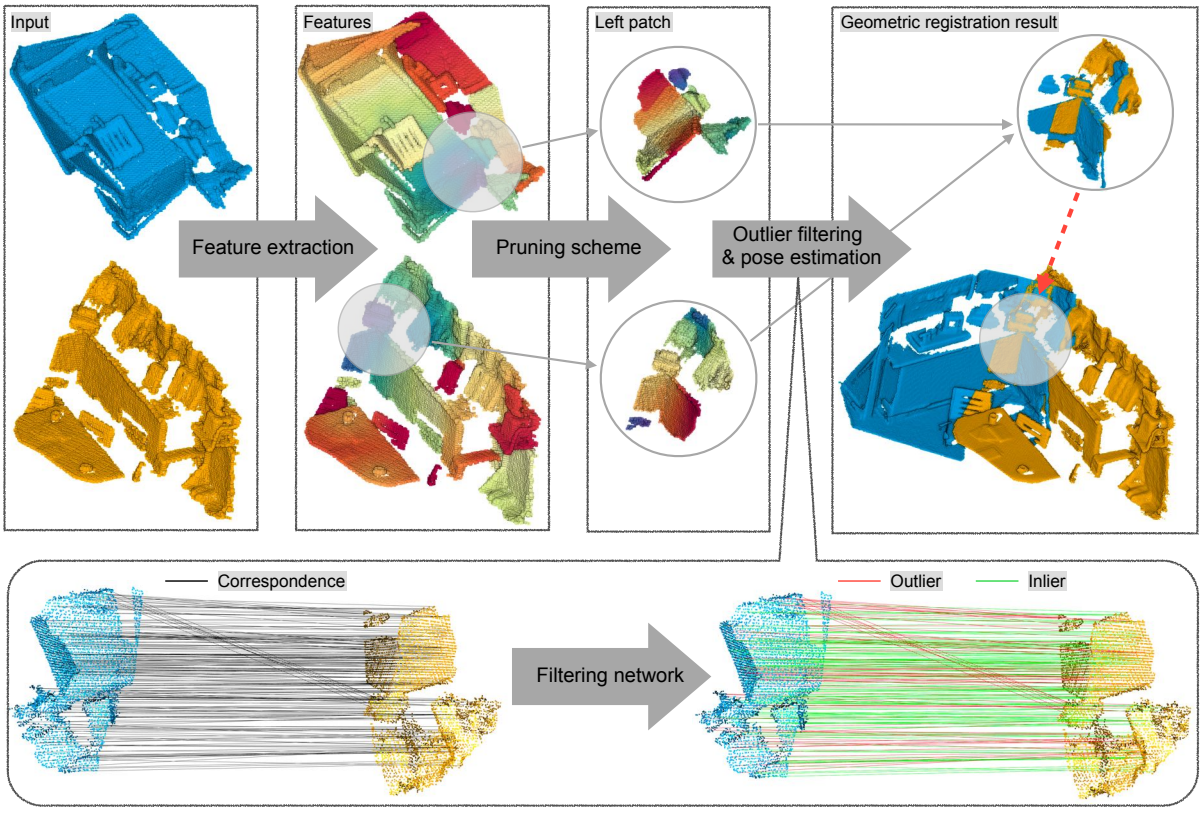

Figure 3.14.: *Proposed pipeline for registration of point clouds with low overlap. It consists of four steps: feature extraction, patch-based pruning, outlier filtering and pose estimation.*

4

Experiments

In this chapter, we first talk about several evaluation metrics; then we introduce 3DMatch benchmark [Zeng et al., 2017] and our new LowOverlap benchmark built on 3DMatch dataset; next, we present results from baseline methods and our proposed model;

4.1. Evaluation metrics

Overlap ratio Intuitively, overlap is the intersection between two point clouds after ground truth transformation. Formally, it's defined as follows: consider a set of point clouds $\{\mathbf{P}_i\}$, for any two point clouds $\mathbf{P}_i, \mathbf{P}_j \in \mathbb{R}^{3\times N}$, we use a ground truth transformation $(\mathbf{R}_{ij}, \mathbf{t}_{ij})$ to align them into a common coordinate frame. Consider a point $\mathbf{p} \in \mathbf{P}_i$, let $nn(\mathbf{p}, \mathbf{P}_i)$ denotes the nearest neighbor of $\mathbf{p}$ in $\mathbf{P}_i \backslash \mathbf{p}$. Let ε_i be the median of the set of distances $\{\|\mathbf{p} - \mathrm{nn}(\mathbf{p}, \mathbf{P}_i)\| : \mathbf{p} \in \mathbf{P}_i\}$ and define $\varepsilon = \max_i \varepsilon_i$. Now for a pair of point cloud $(\mathbf{P}_i, \mathbf{P}_j)$, for each point $\mathbf{p} \in \mathbf{R}_{ij}\mathbf{P}_i + \mathbf{t}_{ij}$, we can compute nearest neighbor $nn(\mathbf{p}, \mathbf{P}_j)$. Consider the fraction of such pairs that are within distance ε. Specifically, define α_{ij}

$$\alpha_{ij} = \frac{|\{\mathbf{p} \in \mathbf{R}_{ij}\mathbf{P}_i + \mathbf{t}_{ij} : \|\mathbf{p} - \mathrm{nn}(\mathbf{p}, \mathbf{P}_j)\| \leq \varepsilon\}|}{|\mathcal{P}_i|} \tag{4.1}$$

and similarly for α_{ji}. We say that $\mathcal{P}_i$ and $\mathcal{P}_j$ overlap if $\min(\alpha_{ij}, \alpha_{ji}) \geq 0.3$ [Khoury et al., 2017].

On 3DMatch benchmark [Zeng et al., 2017], after voxel down-sampling(voxel size=2.5cm) the point clouds, ϵ is 2.03cm.

Feature match recall Feature match recall shows how good the feature extraction model is at finding a reasonable number of good correspondences for final pose estimation [1]. Formally, it measures the percentage of fragment pairs that can recover the pose with high confidence(the more inliers, the more likely our model is to recover the true pose):

$$R = \frac{1}{M}\sum_{s=1}^{M}\mathbb{1}\left(\left[\frac{1}{|\mathcal{K}_{ij}|}\sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}_{ij}}\mathbb{1}(\|\mathbf{R}_{ij}\mathbf{p}+\mathbf{t}_{ij}-\mathbf{q}\|<\tau_1)\right]>\tau_2\right) \tag{4.2}$$

where M is the number of fragment pairs, τ_1 is a threshold on Euclidean distance between the correspondence $(\mathbf{p},\mathbf{q})$ found in the feature space. τ_2 is a threshold on the inlier ratio over the correspondence set and could be determined by theoretical analysis of the number of iterations k needed by RANSAC. To find at least $n = 3$ corresponding points with the probability of success $p = 99.9\%$, the number of iterations equals $k \approx 55000$ when $\tau_2 = 0.05$ and could be greatly reduced if τ_2 can be increased(e.g. $k = 860$ if $\tau_2 = 0.2$).

Feature match recall is preferred over precision because the latter can be arbitrarily increased by pruning fragment pairs [Deng et al., 2018a]. As is shown in Figure 4.1, FCGF features [Choy et al., 2019b] can keep high recall with both strict inlier ratio and inlier distance thresholds, this is a promising feature for efficient and precise pose estimation.

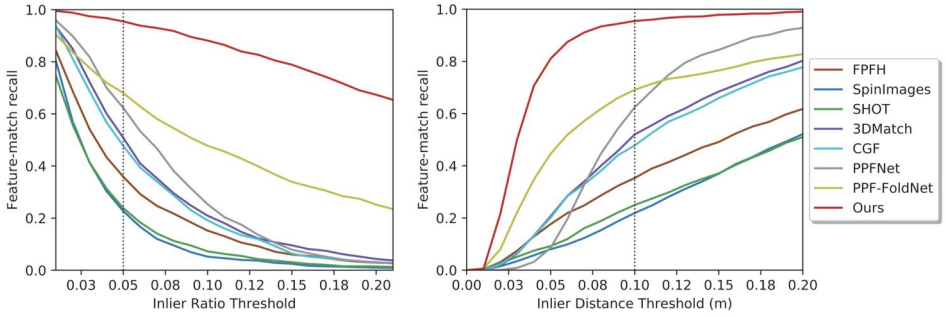

Figure 4.1.: *Feature-match recall with respect to inlier ratio threshold τ_2(left) and inlier distance accuracy tolerance τ_1(right). Figure is taken from [Choy et al., 2019b].*

Relative translation and rotation error We follow the definition used in [Yew and Lee, 2018], the relative translation error (RTE) and relative rotation error (RRE) measure the deviations from the ground truth pose:

$$\begin{aligned}\text{RTE} &= \|\hat{\mathbf{t}}-\mathbf{t}\| \\ \text{RRE} &= \arccos\left(\frac{\text{Tr}(\hat{\mathbf{R}}^T\mathbf{R})-1}{2}\right)\end{aligned} \tag{4.3}$$

where $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ represent the estimated rotation and translation, $\text{Tr}(\mathbf{X})$ represents the *trace* of matrix $\mathbf{X}$.

[1] Note that feature match recall here is defined over a set of fragment pairs, this is different from that over a fragment pair, which measures the percentage of retrieved ground truth correspondences.

Geometric registration recall To evaluate the final geometric registration on two fragments with over 30% overlap, Choi *et al.* [Choi et al., 2015] propose to directly measure the effect of estimation on the ground-truth correspondences $\mathcal{K}_{ij}$. A transformation is accepted if it brings these ground-truth correspondence pairs into alignment. Specifically, $(\mathbf{R}_{ij}, \mathbf{t}_{ij})$ is considered a true positive if the root-mean-squares-error(RMSE) of the ground-truth correspondences is below a threshold τ^2:

$$\frac{1}{|\mathcal{K}_{ij}^*|} \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}_{ij}^*} \|\mathbf{R}_{ij}\mathbf{p} + \mathbf{t}_{ij} - \mathbf{q}\|^2 < \tau^2 \tag{4.4}$$

τ is defined as 0.2 meter in [Choi et al., 2015]. This calculation requires to pre-compute the ground truth correspondences, instead, we directly compare with the ground truth transformation, a transformation is accepted if the relative rotation error is below 10° and the relative translation error is below 30 centimeters [Yang et al., 2020][2]. Similar to feature match recall, registration recall is considered as the primary measure as precision can be raised by pruning false positives using robust global optimisation [Choi et al., 2015].

4.2. Dataset and benchmark

We perform experiments on two benchmarks, 3DMatch benchmark [Zeng et al., 2017] that focuses on high overlap regions and LowOverlap benchmark which goes to extremely low overlap regions.

4.2.1. 3DMatch benchmark

Zeng *et al.* [Zeng et al., 2017] provide a dataset of in total 62 scenes collected from Analysis-by-Synthesis [Valentin et al., 2016], 7Scenes [Shotton et al., 2013], SUN3D [Xiao et al., 2013], RGB-D Scenes v.2 [Lai et al., 2014], and Halber *et al.* [Halber and Funkhouser, 2016]. 54 scenes are used for training and 8 scenes are used for testing[3]. These scenes are captured in different indoor spaces include bedrooms, offices, living rooms, and restrooms, by different depth sensors include KMicrosoft Kinect, Structure Sensor, Asus Xtion Pro Live, and Intel RealSense, at different scales and viewpoints with diverse local geometries. It provides great diversities and allows our models to generalize across different indoor space. 3DMatch benchmark [Zeng et al., 2017] provides eight sets of scene fragments(50-60 fragments for each scene), each fragment is integrated from 50 depth frames using TSDF volumetric fusion [Curless and Levoy, 1996] and represents part of the scene with 3D point cloud, they are. The visualizations are in Figure 4.2 and Figure A.1.

[2]Note that these hyper-parameters should adapt to the nature of the dataset.

[3]The detailed train/test split can be found in the project webpage:`https://3dmatch.cs.princeton.edu`

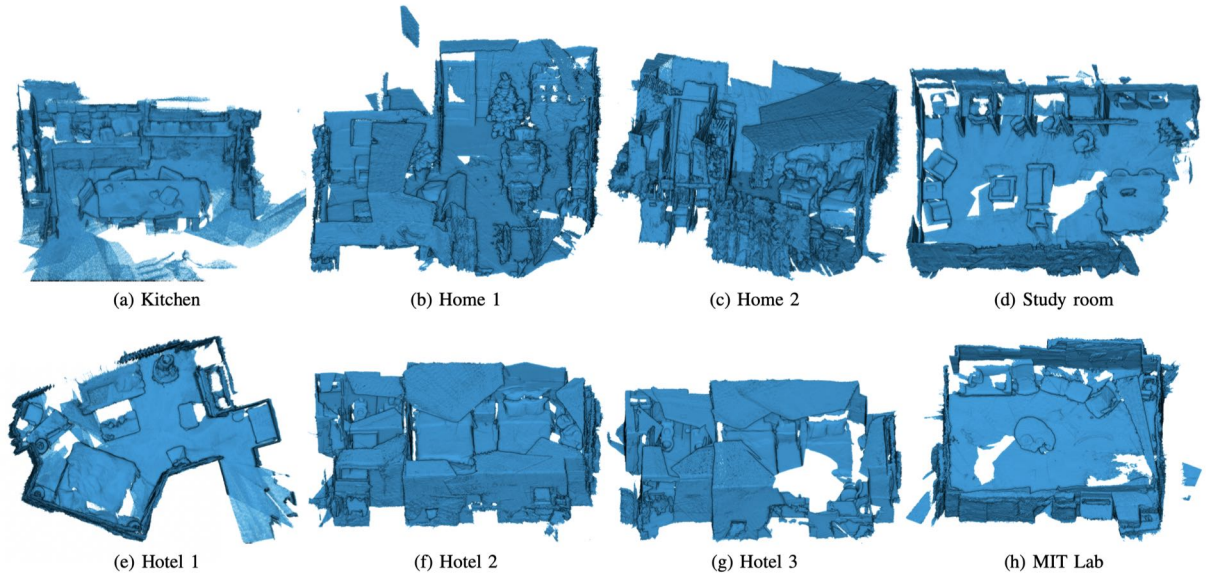

Figure 4.2.: *Visualizing geometric reconstructions of 8 test scenes.*

4.2.2. LowOverlap benchmark

The official 3DMatch benchmark [Zeng et al., 2017] only considers registration of point clouds with over 30% overlap [4], however, this thesis aims at low overlap region. For all fragment sets of 8 test scenes, we compute the overlap ratios between each fragment pair and construct a new benchmark, we term it *LowOverlap benchmark*. Based on the overlap ratio, it's divided into 4 levels: 5%-10%,10%-20%, 20%-30%, and 30%-40%, each has 1591, 2315, 1477 and 989 fragment pairs respectively.

4.3. Results

In this section, we present detailed registration results. We fix FCGF [Choy et al., 2019b] as our default feature extraction model[5] for all the methods and the voxel size is 2.5 centimeters. We run all the experiments on a workstation with Intel(R) Core(TM) i7-7700K CPU@4.20GHz, 32GB of RAM and one NVIDIA GeForce GTX 1080Ti GPU.

4.3.1. Baseline methods

We first present geometric registration results from existing methods, including hand-crafted filtering methods and deep point cloud registration models.

[4]Note that from our inspection, there are fragment pairs of low overlap(way lower than 30%) listed as ground truth to be registered.

[5]The pre-trained model can be found here`https://github.com/chrischoy/FCGF`.

Hand-crafted filtering methods RANSAC [Fischler and Bolles, 1981] has been a classic for filtering outliers for 40 years, together with heuristics(Lowe's ratio test [Lowe, 2004] and mutual check) and careful hyper-parameters tuning, even today, they are still difficult to beat [Jin et al., 2020]. Truncated least squares Estimation And SEmidefinite Relaxation(TEASER++) [Yang et al., 2020] is a quite recent work that allows for a *certifiable* registration. It provides an indicator of the reliability of the final solution and allows to reject the registration, this is particularly useful to loop closure detection in a Simultaneous-Localisation-and-Mapping(SLAM) system.

We take RANSAC implementation from Open3D [Zhou et al., 2018], maximum correspondence distance is 5 centimeters, two convergence criteria are 50000 and 1000. For TEASER++, we take the hyper-parameters tailed to 3DMatch dataset from the author [6].

We sample 5000 points and associated features for each point cloud[7], and report results with and without mutual check as a preliminary outlier filter. Mutual check turns out to be an efficient preliminary step that it can filter out around 90% correspondences. Detailed results are in Table 4.1 and Table 4.2, * indicates method with mutual check.

We can see that RANSAC benefits from mutual check on both benchmarks while TEASER++ suffers from it. On one hand, mutual check increases the inlier ratio τ_2 in Equation 4.2, allows RANSAC to sample better hypothesis set given fixed iterations. On the other hand, it also decreases the real number of inliers, thus harming the performance of TEASER++. RANSAC and TEASER++ have similar performances in high overlap regions, but the gap becomes big in low overlap regions(see Table 4.2).

Table 4.1.: *Registration recall on 3DMatch benchmark with RANSAC and TEASER++.*

	RANSAC	RANSAC*	TEASER++	TEASER++*	# Samples
Kitchen	0.878	0.953	0.967	0.962	449
Home 1	0.868	0.934	0.962	0.925	106
Home 2	0.623	0.742	0.792	0.736	159
Hotel 1	0.863	0.962	0.940	0.940	182
Hotel 2	0.872	0.872	0.897	0.859	78
Hotel 3	0.808	0.808	0.846	0.808	26
Study	0.752	0.889	0.850	0.855	234
MIT Lab	0.667	0.714	0.733	0.711	45
Mean recall	0.791	0.859	0.874	0.849	

[6] `https://github.com/MIT-SPARK/TEASER-plusplus/blob/master/examples/teaser_python_3dsmooth/teaser_python_3dsmooth.py`

[7] For TEASER++ without mutual check, we only sample 2000 points, as it becomes super slow with more samples.

Table 4.2.: *Registration recall on LowOverlap benchmark with RANSAC and TEASER++.*

	RANSAC	RANSAC*	TEASER++	TEASER++*	# Samples
5%-10%	0.041	0.214	0.306	0.315	1591
10% - 20%	0.308	0.582	0.672	0.662	2315
20% - 30%	0.706	0.848	0.871	0.866	1477
30% - 40%	0.902	0.930	0.942	0.930	989
Mean recall	0.489	0.643	0.697	0.693	

Deep point cloud registration In spite of many end-to-end point cloud registration models [Aoki et al., 2019, Sarode et al., 2019, Wang and Solomon, 2019a], most of them only deal with point clouds of almost 100% overlap. Figure 4.3 shows examples of such registration. To the best of our knowledge, PRNet [Wang and Solomon, 2019b] and RPM-Net [Yew and Lee, 2020] are the only two open-source models that aim for partial-to-partial registration. However, they are still limited to the simulated ModelNet40 dataset [Wu et al., 2015] and are trained with roughly 50% overlaps. We train DCP [Wang and Solomon, 2019a] and PRNet [Wang and Solomon, 2019b] on 3DMatch dataset and can not get reasonable results, this is also confirmed in Table 1 of DGR [Choy et al., 2020a]. For RPM-Net[Yew and Lee, 2020], we replace the feature extraction model with FCGF [Choy et al., 2019b] and final registration recall on 3DMatch benchmark is 0.549. Besides, we also replace the front-end of SuperGlue [Sarlin et al., 2020] with FCGF and extend it for 3D point cloud registration, the recall increases to 0.605, still way lower than that from RANSAC or TEASER++. The main reason is that the Sinkhorn layer in both models can not provide enough confident correspondences. We attribute this to the domain gap between ModelNet40 and 3DMatch dataset. It's easier for deep models to learn the canonical representation of most shapes in ModelNet40 dataset. In addition, 3DMatch dataset has more ambiguities in feature matching due to repetitive and planar local geometries.

4.3.2. Our methods

We test both modified LTFGC [Moo Yi et al., 2018] and OANet [Zhang et al., 2019] as filtering networks in our model. Filtering networks are trained from scratch on 3DMatch training datasets. We take all the fragment pairs with over 20% overlaps, randomly sample 2048 points for each point cloud and build correspondences, then train them with classification loss only using Adam optimiser [Kingma and Ba, 2014] for 100 epochs, the learning rate initialises at 0.0001 and is divided by 10 at 20th, 50th and 80th epochs. Pruning scheme is also trained separately, it's an extremely light network, which can be trained within 10 epochs using negative log-likelihood loss. The patch number K is set to be 10, each patch has 512 points, covering a quarter of the whole point cloud. We sample 10 patch pairs from 100 possibilities after removing the dustbin rows and columns from the partial assignment matrix.

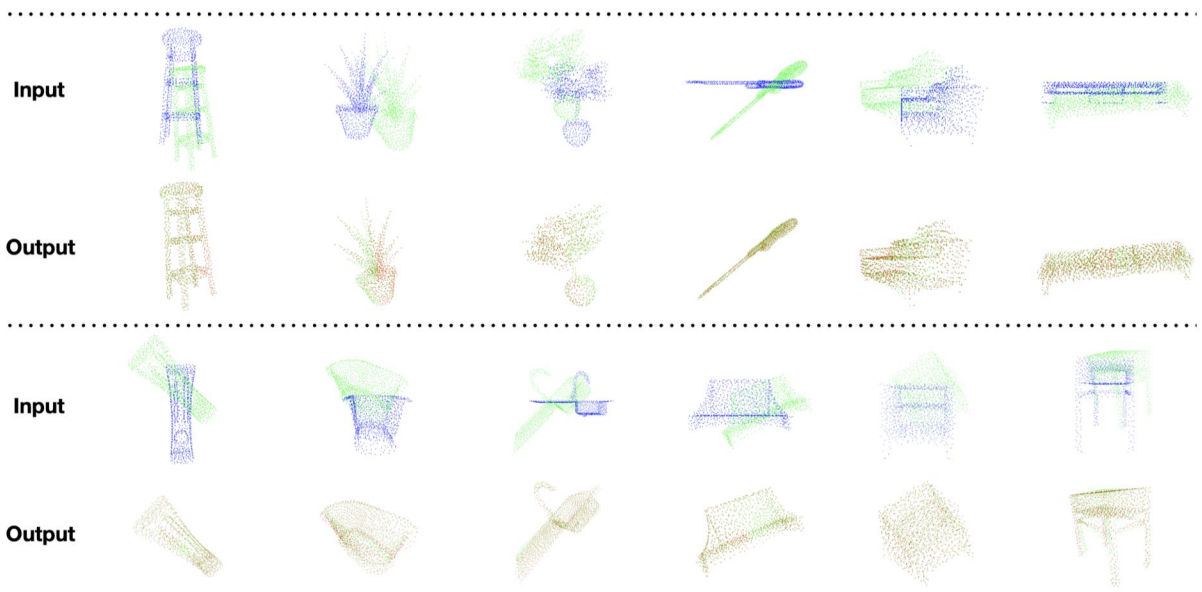

Figure 4.3.: *Geometric registration results on ModelNet40 [Wu et al., 2015]. Figure is taken from [Wang and Solomon, 2019a].*

Pruning results In Figure A.2, we show the intermediate results from our pruning scheme. We can see that for most fragment pairs, our pruning scheme can find highly overlapping regions. This can efficiently reduce the search space of feature matching, thus increasing the inlier ratio of the over-complete correspondence set.

Quantitative results The detailed results are in Table 4.3 and Table 4.4, * indicates model with our patch-based pruning scheme. We can see that our pruning scheme increases the geometric registration in both high and low overlap regions, the improvement is significant in low overlap regions. On standard 3DMatch benchmark [Zhang et al., 2019], there are in total 68 fragment pairs saved by our pruning scheme, their median overlap ratio is 30.8%, some of them only have 14% overlaps. On LowOverlap benchmark, 1009 fragment pairs are saved, the median overlap ratio is 12.8%.

Qualitative results We present samples that fail to be registered by modified OANet [Zhang et al., 2019] but saved by our pruning scheme. They are shown in Figure A.3 and Figure A.4, we can see that by mining the most overlapping patch pairs, we can register quite challenging fragment pairs with extreme low overlap ratios(5%).

Failed cases In Figure A.5, we show some failed cases on LowOverlap benchmark. There are two sources for these failures, the **first** is that our pruning scheme can not always find most overlapping patch pairs. In Figure 4.4 we plot patch-based pruning recall with respect to different *topk*, we can see that it requires big *topk* to achieve a high recall. The **second** source is our filtering network. Although it can get high precision and recall(above 95%) most of the time, but it still could go wrong when there are too many outliers.

Table 4.3.: *Registration recall on 3DMatch benchmark with filtering networks.*

	LTFGC	LTFGC*	OANet	OANet*	# Samples
Kitchen	0.898	0.949	0.931	0.960	449
Home 1	0.896	0.906	0.925	0.906	106
Home 2	0.686	0.742	0.704	0.698	159
Hotel 1	0.890	0.929	0.923	0.956	182
Hotel 2	0.859	0.923	0.872	0.872	78
Hotel 3	0.769	0.846	0.846	0.808	26
Study	0.701	0.795	0.782	0.833	234
MIT Lab	0.689	0.644	0.689	0.689	45
Mean recall	0.816	0.842	0.834	0.842	

Table 4.4.: *Registration recall on LowOverlap benchmark with filtering networks.*

	LTFGC	LTFGC*	OANet	OANet*	# Samples
5%-10%	0.069	0.255	0.1	0.309	1591
10% - 20%	0.375	0.6	0.444	0.641	2315
20% - 30%	0.717	0.823	0.778	0.847	1477
30% - 40%	0.886	0.906	0.901	0.926	989
Mean recall	0.512	0.646	0.556	0.681	

4.4. Discussion

In this section, we compare our method with hand-crafted filters.

4.4.1. Hand-crafted filters vs. learning based filters

On standard 3DMatch benchmark, we can see that TEASER++ [Yang et al., 2020] has the strongest performance, followed by RANSAC [Fischler and Bolles, 1981] with mutual check, then is filtering networks with our pruning scheme. If we only look at the scene *Kitchen* which has the most fragment pairs and the result is less biased by randomness compared to mean recall, our model is quite close to TEASER++ and is better than RANSAC. The same trend is also shown on LowOverlap benchmark.

We don't compare their run-time in our study, the main reason is that learning-based meth-

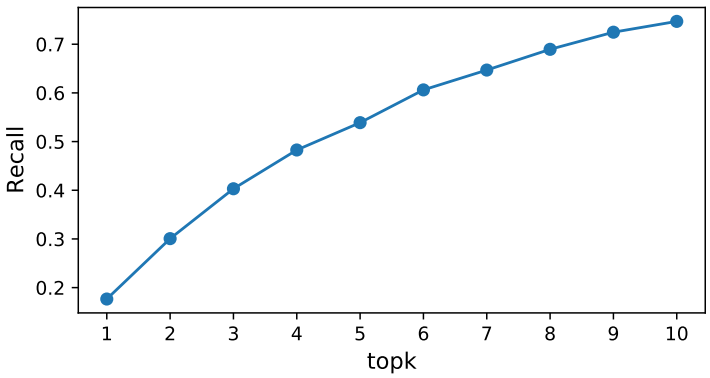

Figure 4.4.: *Patch-based pruning recall@k. A patch-based pruning is accepted if the most overlapping patch pair is within the topk patch pair set.*

ods are highly optimised by batch processing on GPU, while hand-crafted filters are mostly implemented on CPU and don't benefit from parallelization. In [Gojcic et al., 2020], they report that filtering network is around 14x faster than RANSAC on the whole scene, while [Cavalli et al., 2020] reports the contrary result(14ms(hand-crafts) vs. 21ms(learning-based)), we attribute this to different implementation details.

Deep learning models are data-driven models, while hand-crafted models are mostly data-agnostic and can not benefit from training samples. The most recent success of GPT-3 [Brown et al., 2020] has shown that by scaling up language models to have 175 billion parameters and train on 45TB data, it can bring substantial gains on many NLP tasks and benchmarks. We have reasons to believe that our performances can also be further improved by scaling up the training samples and using more powerful models.

4.4.2. Pruning vs. heuristics

Despite the fact that both increase the final inlier ratio, our pruning scheme is different from heuristics(mutual check). Pruning scheme is applied before feature matching, it aims to find the most overlapping region and reduce search space for feature matching, while heuristics work after feature matching, aiming to increase the inlier ratio. To some degree, our pruning scheme works by directly manipulating the input, it is flexible to be plugged into any deep learning models. In addition, it can also be combined with heuristics, for example adding heuristics in between feature matching and filtering networks in our model. In theory, this could improve our filtering network.

5

Conclusion and future work

5.1. Conclusion

In this thesis, we study the task of registration of 3D point clouds with low overlap, and show improved results from pair-wise registration [Gojcic et al., 2020] with proposed pruning scheme.

We first show that the performances on standard 3DMatch benchmark [Zeng et al., 2017] are almost saturated, by analysing the failed cases, we find most of these fragment pairs have low overlap. Then we report that most end-to-end deep point cloud registration models fail to achieve reasonable performances on real scans, this is attributed to low overlap on real scans and the domain gap between ModelNet40 [Su et al., 2015] and 3DMatch [Zeng et al., 2017], the latter is more difficult as feature matching has more ambiguities due to repetitive and planar local geometries. Next, we analyse the learning-based pair-wise registration pipeline proposed by Gojcic *et al.* [Gojcic et al., 2020], and find it could still break at outlier filtering step for low overlapping fragment pairs. Having observed that two patches intersect in the overlap region share similar geometric appearance which is robust to Euclidean transformations, we propose a pruning scheme to mine the most overlapping patch pairs and use them for further registration.

In detail, our pruning scheme starts by building several patches from input point clouds, then a light attentive pooling model aggregates the local features and predicts a global patch descriptor. Finally a differentiable Sinkhorn layer [Cuturi, 2013] solves the partial assignment problem from the score matrix extended by dustbin rows and columns. The proposed pruning scheme is light and flexible that can be easily plugged into deep point cloud registration models. We hope it could serve as a good start to explore this challenging task in deep learning era.

5.2. Future work

In this section, we present several future works to improve our model.

Co-contextual learning Co-contextual learning has shown strong performances for feature matching in 2D [Sarlin et al., 2020]. The high level idea is that feature extraction model works on each image independently, rendering the extracted features unaware of the other image to be matched and causing many ambiguities at feature matching step. Co-contextual learning works by allowing two feature sets to communicate with each other, features are then updated with information aggregated through message passing formulation [Gilmer et al., 2017] over a complete graph. We find such scheme powerful for image matching that it almost doubles the inlier ratios from original features. In addition, it greatly decreases the ratio between second highest similarity score and highest score from 92% to 70%, making feature matching less ambiguous. We've conducted extensive experiments to extend this idea to point cloud registration, but merely get any improvements. One possible breakthrough would be the rotation invariant positional encoding.

Soft attention scores At current stage, we prune points in a *hard* manner by simply removing them, this is problematic if we fail to find the most overlapping patch pairs at first step. A *soft* pruning scheme would rely on point-wise attention scores to guide feature matching and correspondence weighting. This is more robust as all components of our model have access to complete input regardless of their orders, attention scores are updated such that all components finally reach a consensus.

Hierarchical structure Our model prunes points in an one-shot manner, the final patch covers 25% of the input, regardless of the real size of overlap region. Ideally, the model follows a coarse-to-fine strategy, it first takes the sparse point clouds and roughly determines the overlap regions, then it consumes dense point clouds and attends to the overlap region more precisely. We hope above-mentioned soft attention scores could make such hierarchical structure possible.

Fine-tuning the whole model Currently we train each part of our model separately and finally put them together. In theory, our model could still benefit from fine-tuning them altogether.

A

Appendix

A.1. Full derivatives of Sinkhorn iterations

Consider two discrete probability distribution P and Q defined on two finite sets $X = \{x_1, x_2, ..., x_k\}$ and $Y = \{y_1, y_2, ..., y_k\}$ respectively. Stochastic transportation maps Γ between P and Q are conditional distributions that fulfill the following marginalization property:

$$\sum_k \Gamma\left(y_n|x_k\right) P\left(x_k\right) = Q\left(y_n\right) \tag{A.1}$$

Denote the set of all possible stochastic transportation maps as G, the cost of sampling (x_n, y_k) as $c(x_n, y_k)$, then the expected cost of the least expensive transportation map is:

$$OT_c[P, Q] = \min_{\Gamma \in G} \sum_{n,k} \left[c\left(x_n, y_k\right) \Gamma\left(y_k|x_n\right) P\left(x_n\right)\right] \tag{A.2}$$

The optimal conditional probability $\Gamma(y|x)$ is non-zero only for few values of y. The reason is, given an element $\hat{x}$, there is usually a $\hat{y}$ such that the transportation cost is the smallest:

$$c(\hat{x}, \hat{y}) < c(\hat{x}, y), \quad \forall y \in Y \tag{A.3}$$

The optimal solution is transporting all the probability mass from $\hat{x}$ to $\hat{y}$. However, this is not always possible because the probability mass $P(\hat{x})$ could be larger than $P(\hat{y})$. In this case, then the best thing to do is allocating as much mass as possible to $\hat{y}$ and the rest to the second cheapest element of Y. In the ideal case, for each element x in X, there is a unique and distinct y in Y that minimizes the transportation cost and shares the same probability mass, then the transportation map Γ is fully deterministic. Entropic regularization is a way to counteract the tendency of optimal transport to produce nearly deterministic transportation maps by adding a

term that favors randomness, which is the entropy of the transportation maps, then the result is a regularized transportation problem:

$$OT_c^{\epsilon}[P,Q] = \min_{\Gamma \in G} \left[\sum_{n,k} \left[c\left(x_n, y_k\right) \Gamma\left(y_k | x_n\right) P\left(x_n\right) \right] - \epsilon \mathcal{H}[\Gamma(y,x)] \right] \tag{A.4}$$

The biggest advantage of including entropic regularization in an optimal transport problem is that the regularized solution can be found very efficiently using a simple iterative algorithm. Denote the joint distribution $\Gamma(y_n \| x_k) P(x_k)$ as $\Gamma(x_k)$, the probability $P(x_k)$ as P_k and $Q(y_k)$ as Q_k. Then we need to optimize the objective function:

$$E[\Gamma] = \sum_{n,k} \left[C_{nk} \Gamma_{nk} \right] + \epsilon \sum_{n,k} \log\left(\Gamma_{nk}\right) \Gamma_{nk} \tag{A.5}$$

under the following set of constraints:

$$\begin{aligned} \textstyle\sum_k \Gamma_{nk} &= Q_n \\ \textstyle\sum_n \Gamma_{nk} &= P_k \end{aligned} \tag{A.6}$$

These two constraints can be integrated into the objective function using two sets of Lagrangian multipliers and will result in the following functions:

$$\mathcal{L}[\Gamma] = E[\Gamma] - \sum_n \lambda_n \left(Q_n - \sum_k \Gamma_{nk} \right) - \sum_k \chi_k \left(P_k - \sum_n \Gamma_{nk} \right) \tag{A.7}$$

This function is smooth and convex, which implies that we can find the global minimum by differentiating the loss and setting the gradients to be zero:

$$\nabla_{n,k} \mathcal{L}[\Gamma] = C_{kn} + \epsilon \log\left(\Gamma_{nk}\right) + \epsilon + \lambda_k + \chi_n \tag{A.8}$$

After setting the gradients with respect to Γ_{nk} to zero, we get:

$$\begin{aligned} \Gamma_{nk} &= \exp\left(-\lambda_k/\epsilon\right) \exp\left(-C_{kn}/\epsilon\right) \exp\left(-\chi_n/\epsilon - 1\right) \\ &= v_k K_{kn} u_n \end{aligned} \tag{A.9}$$

where $K_{kn} = \exp\left(-C_{kn}/\epsilon\right)$,$v_k = \exp\left(-\lambda_k/\epsilon\right)$, $u_n = \exp\left(-\chi_n/\epsilon - 1\right)$. together with the constraints in A.6, we get:

$$\begin{aligned} \textstyle\sum_k \Gamma_{nk} &= v_n \left(\sum_k K_{n,k} u_k \right) = Q_n \\ \textstyle\sum_n \Gamma_{nk} &= u_k \left(\sum_n K_{n,k} v_n \right) = P_n \end{aligned} \tag{A.10}$$

To satisfy the two constraints, one strategy is to iteratively update each set of variables while keeping the other set fixed:

$$\begin{aligned} v_n^{(t+1)} &= \frac{Q_n}{\left(\sum_k K_{nk} u_k^{(t)} \right)} \\ u_k^{(t+1)} &= \frac{P_k}{\left(\sum_n K_{nk} v_n^{(t+1)} \right)} \end{aligned} \tag{A.11}$$

Then we get the final sinkhorn iterations[1].

[1]Refer to this post for more details: https://mindcodec.ai/2018/10/01/an-intuitive-guide-to-optimal-transport-part-iii-entropic-regularization-and-the-sinkhc

A.2. 3DMatch training dataset

Figure A.1 shows RGB-D reconstruction of 3DMatch training dataset.

A.3. Geometric registration with pruning scheme

Figure A.2 shows the intermediate pruning results. Figure A.3 and Figure A.4 show the successful geometric registration results with our pruning scheme, Figure A.5 shows some examples of failed cases.

Figure A.1.: *Visualizing several RGB-D reconstructions of training scenes. Figure is taken from [Zeng et al., 2017].*

Figure A.2.: *Visualisation of intermediate pruning results on LowOverlap benchmark. We randomly sample 12 fragment pairs, the first row shows the input, the second row shows the pruning results. Note that each patch actually only has 512 points, for better visualisation, we apply a radius search to the raw point clouds to get dense representation.*

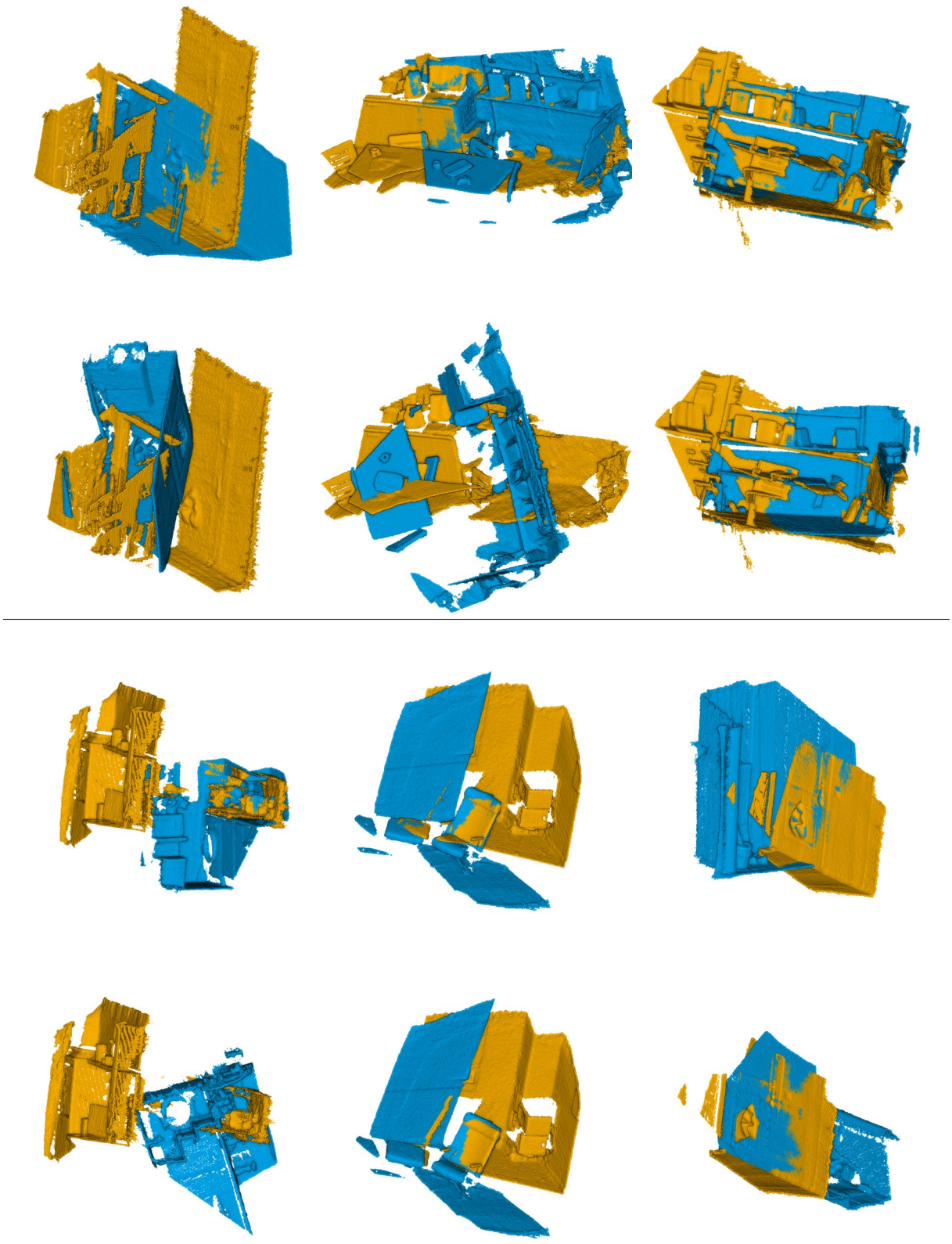

Figure A.3.: *Visualisation of successful registrations on 3DMatch benchmark. The first row shows successful registrations from our model, the second row shows failed registrations from modified OANet.*

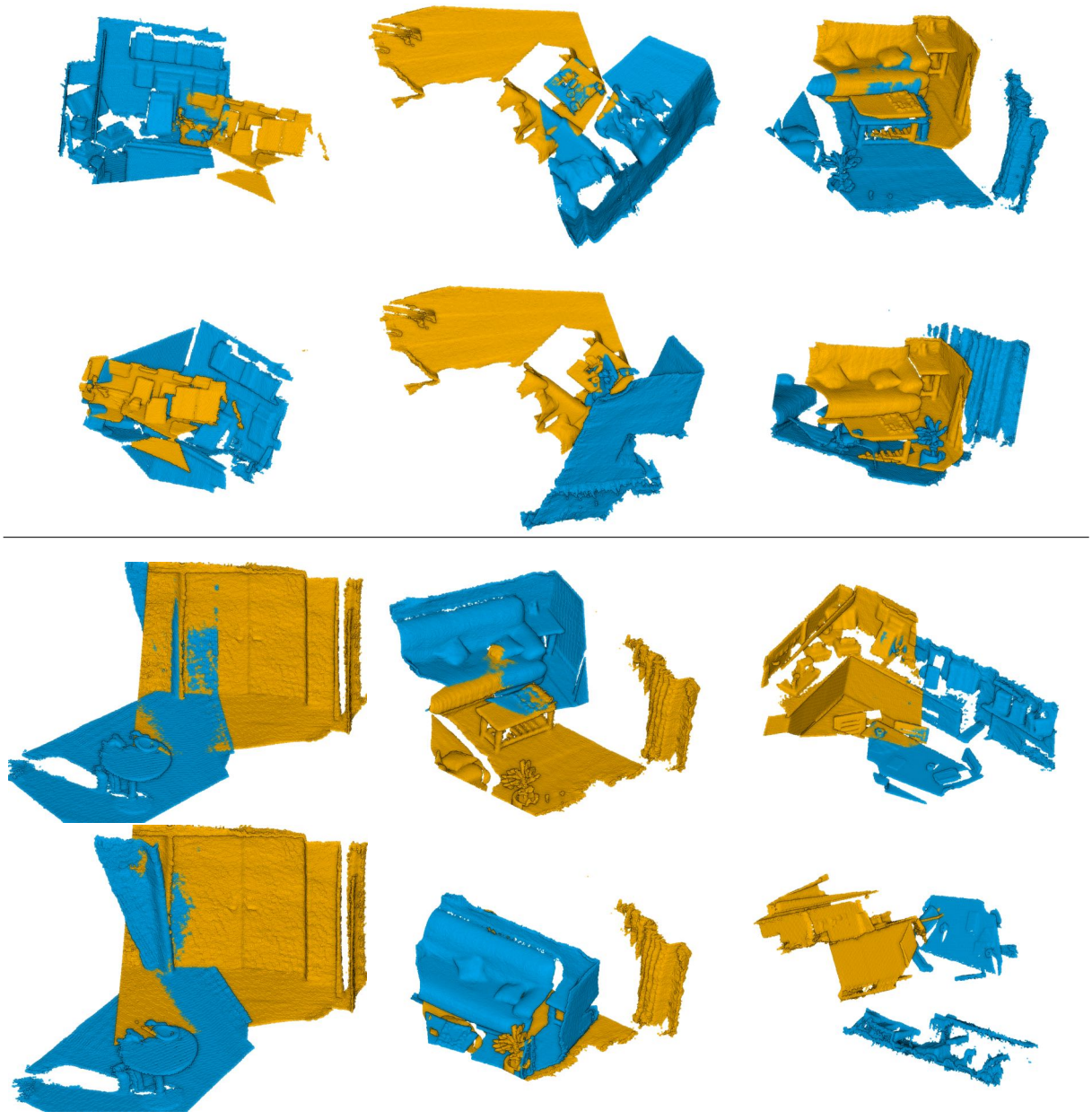

Figure A.4.: *Visualisation of successful registrations on LowOverlap benchmark. The first row shows successful registrations from our model, the second row shows failed registrations from modified OANet.*

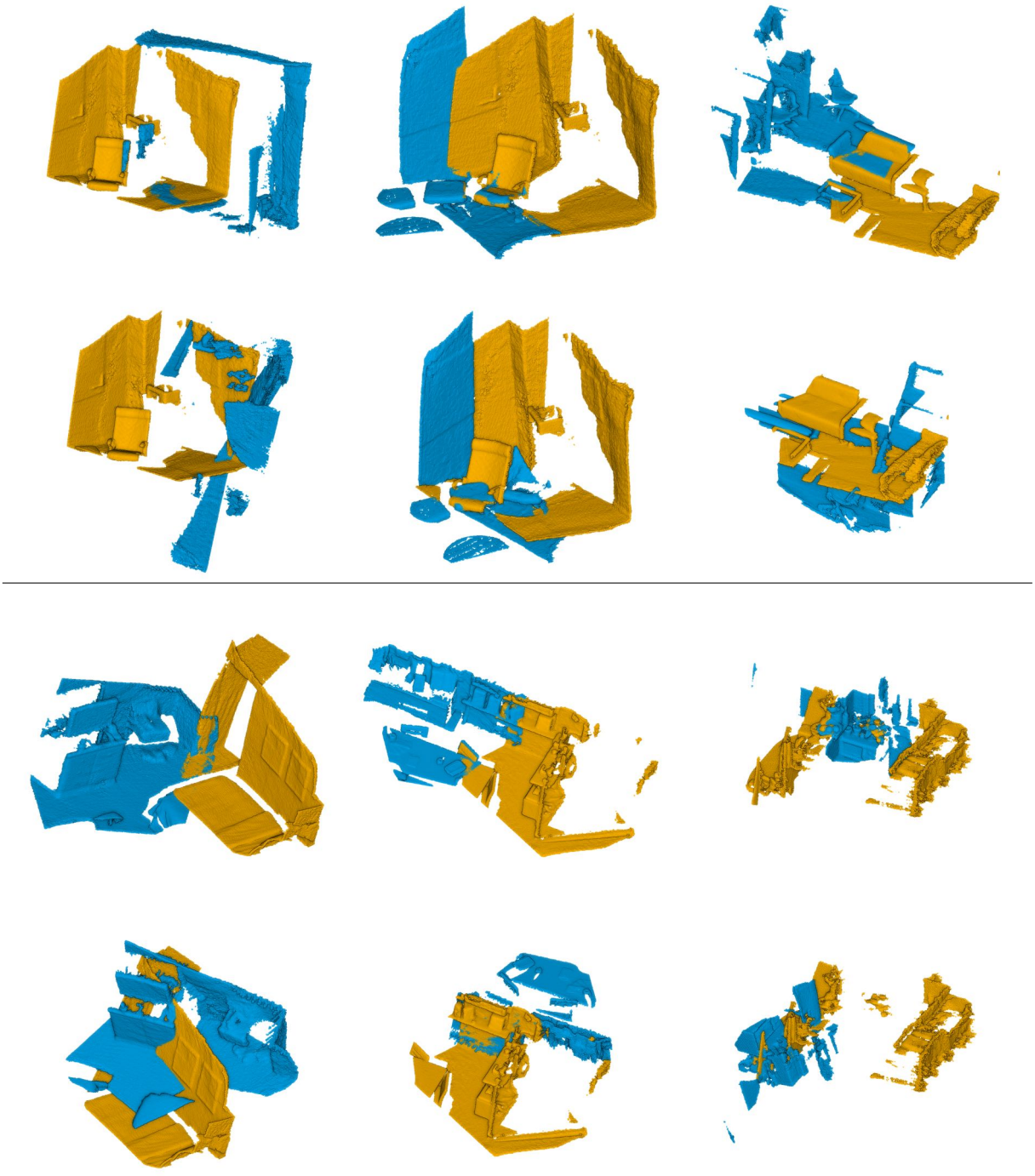

Figure A.5.: *Visualisation of failed registration on LowOverlap benchmark. The first row shows the ground truth, the second row shows that from our model.*

Bibliography

[Anastasiya Mishchuk, 2017] Anastasiya Mishchuk, Dmytro Mishkin, F. R. J. M. (2017). Working hard to know your neighbor's margins: Local descriptor learning loss.

[Aoki et al., 2019] Aoki, Y., Goforth, H., Srivatsan, R. A., and Lucey, S. (2019). Pointnetlk: Robust & efficient point cloud registration using pointnet. In *CVPR*, pages 7163–7172.

[Arandjelovic et al., 2016] Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, pages 5297–5307.

[Arun et al., 1987] Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *TPAMI*, (5):698–700.

[Atzmon et al., 2018] Atzmon, M., Maron, H., and Lipman, Y. (2018). Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*.

[Bai et al., 2020] Bai, X., Luo, Z., Zhou, L., Fu, H., Quan, L., and Tai, C.-L. (2020). D3feat: Joint learning of dense detection and description of 3d local features. *CVPR*.

[Barath and Matas, 2018] Barath, D. and Matas, J. (2018). Graph-cut ransac. In *CVPR*, pages 6733–6741.

[Barath et al., 2019] Barath, D., Matas, J., and Noskova, J. (2019). Magsac: marginalizing sample consensus. In *CVPR*, pages 10197–10205.

[Bengio et al., 2013] Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

[Bian et al., 2017] Bian, J., Lin, W.-Y., Matsushita, Y., Yeung, S.-K., Nguyen, T.-D., and Cheng, M.-M. (2017). Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence. In *CVPR*, pages 4181–4190.

[Brachmann et al., 2017] Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., and Rother, C. (2017). Dsac-differentiable ransac for camera localization. In *CVPR*, pages 6684–6692.

[Brachmann and Rother, 2018] Brachmann, E. and Rother, C. (2018). Learning less is more-6d camera localization via 3d surface regression. In *CVPR*, pages 4654–4662.

[Brachmann and Rother, 2019] Brachmann, E. and Rother, C. (2019). Neural-guided ransac: Learning where to sample model hypotheses. In *ICCV*, pages 4322–4331.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

[Cavalli et al., 2020] Cavalli, L., Larsson, V., Oswald, M. R., Sattler, T., and Pollefeys, M. (2020). Adalam: Revisiting handcrafted outlier detection. *CVPR*.

[Choi et al., 2015] Choi, S., Zhou, Q.-Y., and Koltun, V. (2015). Robust reconstruction of indoor scenes. In *CVPR*, pages 5556–5565.

[Choy et al., 2020a] Choy, C., Dong, W., and Koltun, V. (2020a). Deep global registration. In *CVPR*.

[Choy et al., 2019a] Choy, C., Gwak, J., and Savarese, S. (2019a). 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*.

[Choy et al., 2020b] Choy, C., Lee, J., Ranftl, R., Park, J., and Koltun, V. (2020b). High-dimensional convolutional networks for geometric pattern recognition. In *CVPR*.

[Choy et al., 2019b] Choy, C., Park, J., and Koltun, V. (2019b). Fully convolutional geometric features. In *ICCV*, pages 8958–8966.

[Chum et al., 2005] Chum, O., Werner, T., and Matas, J. (2005). Two-view geometry estimation unaffected by a dominant plane. In *CVPR*, volume 1, pages 772–779. IEEE.

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

[Cuturi, 2013] Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300.

[Dang et al., 2020] Dang, Z., Wang, F., and Salzmann, M. (2020). Learning 3d-3d correspondences for one-shot partial-to-partial registration. *arXiv preprint arXiv:2006.04523*.

[Defferrard et al., 2016] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*.

[Deng et al., 2018a] Deng, H., Birdal, T., and Ilic, S. (2018a). Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *ECCV*, pages 602–618.

[Deng et al., 2018b] Deng, H., Birdal, T., and Ilic, S. (2018b). Ppfnet: Global context aware

local features for robust 3d point matching. In *CVPR*, pages 195–205.

[DeTone et al., 2018] DeTone, D., Malisiewicz, T., and Rabinovich, A. (2018). Superpoint: Self-supervised interest point detection and description. In *CVPR Workshops*, pages 224–236.

[Dusmanu et al., 2019] Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., and Sattler, T. (2019). D2-net: A trainable cnn for joint description and detection of local features. In *CVPR*, pages 8092–8101.

[Elbaz et al., 2017] Elbaz, G., Avraham, T., and Fischer, A. (2017). 3d point cloud registration for localization using a deep neural network auto-encoder. In *CVPR*, pages 4631–4640.

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

[Gilmer et al., 2017] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. *ICML*.

[Gkioxari et al., 2019] Gkioxari, G., Malik, J., and Johnson, J. (2019). Mesh R-CNN. In *CVPR*.

[Gojcic et al., 2020] Gojcic, Z., Zhou, C., Wegner, J. D., Guibas, L. J., and Birdal, T. (2020). Learning multiview 3d point cloud registration. *CVPR*.

[Gojcic et al., 2019a] Gojcic, Z., Zhou, C., Wegner, J. D., and Wieser, A. (2019a). The perfect match: 3d point cloud matching with smoothed densities. In *CVPR*, pages 5545–5554.

[Gojcic et al., 2019b] Gojcic, Z., Zhou, C., and Wieser, A. (2019b). Robust pointwise correspondences for point cloud based deformation monitoring of natural scenes. In *Joint International Symposium on Deformation Monitoring (JISDM)*.

[Graham, 2014] Graham, B. (2014). Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*.

[Graham and van der Maaten, 2017] Graham, B. and van der Maaten, L. (2017). Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*.

[Guo et al., 2016] Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., and Kwok, N. M. (2016). A comprehensive performance evaluation of 3d local feature descriptors. *IJCV*, 116(1):66–89.

[Guo et al., 2013] Guo, Y., Sohel, F., Bennamoun, M., Lu, M., and Wan, J. (2013). Rotational projection statistics for 3d local surface description and object recognition. *IJCV*, 105(1):63–86.

[Gwak et al., 2020] Gwak, J., Choy, C., and Savarese, S. (2020). Generative sparse detection networks for 3d single-shot object detection. *ECCV*.

[Hackel et al., 2018] Hackel, T., Usvyatsov, M., Galliani, S., Wegner, J. D., and Schindler, K. (2018). Inference, learning and attention mechanisms that exploit and preserve sparsity in CNNs. In *IJCV*.

[Hahnloser et al., 2000] Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J.,

and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951.

[Halber and Funkhouser, 2016] Halber, M. and Funkhouser, T. A. (2016). Structured global registration of rgb-d scans in indoor environments. *ArXiv*, abs/1607.08539.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778.

[Hermans et al., 2017] Hermans, A., Beyer, L., and Leibe, B. (2017). In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*.

[Hu et al., 2020] Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., and Markham, A. (2020). Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, pages 11108–11117.

[Huang et al., 2020] Huang, S., Usvyatsov, M., and Schindler, K. (2020). Indoor scene recognition in 3d. *IROS*.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[Jang et al., 2016] Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

[Jin et al., 2020] Jin, Y., Mishkin, D., Mishchuk, A., Matas, J., Fua, P., Yi, K. M., and Trulls, E. (2020). Image matching across wide baselines: From paper to practice. *arXiv preprint arXiv:2003.01587*.

[Kabsch, 1976] Kabsch, W. (1976). A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923.

[Kalogerakis et al., 2017] Kalogerakis, E., Averkiou, M., Maji, S., and Chaudhuri, S. (2017). 3d shape segmentation with projective convolutional networks. In *CVPR*.

[Khoury et al., 2017] Khoury, M., Zhou, Q.-Y., and Koltun, V. (2017). Learning compact geometric features. In *ICCV*, pages 153–161.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105.

[Kurobe et al., 2020] Kurobe, A., Sekikawa, Y., Ishikawa, K., and Saito, H. (2020). Corsnet: 3d point cloud registration by deep neural network. *IEEE Robotics and Automation Letters*.

[Lai et al., 2014] Lai, K., Bo, L., and Fox, D. (2014). Unsupervised feature learning for 3d scene labeling. In *ICRA*, pages 3050–3057. IEEE.

[Landrieu and Simonovsky, 2018] Landrieu, L. and Simonovsky, M. (2018). Large-scale point

cloud semantic segmentation with superpoint graphs. In *CVPR*, pages 4558–4567.

[Li and Lee, 2019] Li, J. and Lee, G. H. (2019). Usip: Unsupervised stable interest point detection from 3d point clouds. In *ICCV*, pages 361–370.

[Li et al., 2020] Li, L., Zhu, S., Fu, H., Tan, P., and Tai, C.-L. (2020). End-to-end learning local multi-view descriptors for 3d point clouds. *arXiv preprint arXiv:2003.05855*.

[Li et al., 2018] Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. (2018). Pointcnn: Convolution on x-transformed points. In *NeurIPS*, pages 820–830.

[Lin et al., 2015] Lin, J., Morere, O., Chandrasekhar, V., Veillard, A., and Goh, H. (2015). Deephash: Getting regularization, depth and fine-tuning right. *arXiv preprint arXiv:1501.04711*.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110.

[Lu et al., 2019] Lu, W., Wan, G., Zhou, Y., Fu, X., Yuan, P., and Song, S. (2019). Deepicp: An end-to-end deep neural network for 3d point cloud registration. *arXiv preprint arXiv:1905.04153*.

[Lucas et al., 1981] Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision.

[Maturana and Scherer, 2015] Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*.

[Mescheder et al., 2019] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, pages 4460–4470.

[Moo Yi et al., 2018] Moo Yi, K., Trulls, E., Ono, Y., Lepetit, V., Salzmann, M., and Fua, P. (2018). Learning to find good correspondences. In *CVPR*.

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163.

[Newcombe et al., 2011] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE.

[Park et al., 2019] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174.

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

[Plötz and Roth, 2018] Plötz, T. and Roth, S. (2018). Neural nearest neighbors networks. In *NeurIPS*, pages 1087–1098.

[Qi et al., 2017a] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660.

[Qi et al., 2016] Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J. (2016). Volumetric and multi-view CNNs for object classification on 3d data. In *CVPR*.

[Qi et al., 2017b] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108.

[Ranftl and Koltun, 2018] Ranftl, R. and Koltun, V. (2018). Deep fundamental matrix estimation. In *ECCV*, pages 284–299.

[Rocco et al., 2020] Rocco, I., Arandjelović, R., and Sivic, J. (2020). Efficient neighbourhood consensus networks via submanifold sparse convolutions. *ECCV*.

[Rocco et al., 2018] Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., and Sivic, J. (2018). Neighbourhood consensus networks. In *NeurIPS*, pages 1651–1662.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Rusu et al., 2009] Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *ICRA*, pages 3212–3217. IEEE.

[Rusu et al., 2008] Rusu, R. B., Marton, Z. C., Blodow, N., and Beetz, M. (2008). Persistent point feature histograms for 3d point clouds. In *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany*, pages 119–128.

[Sarlin et al., 2020] Sarlin, P.-E., DeTone, D., Malisiewicz, T., and Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks. In *CVPR*.

[Sarode et al., 2019] Sarode, V., Li, X., Goforth, H., Aoki, Y., Srivatsan, R. A., Lucey, S., and Choset, H. (2019). Pcrnet: Point cloud registration network using pointnet encoding. *arXiv preprint arXiv:1908.07906*.

[Shotton et al., 2013] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., and Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*, pages 2930–2937.

[Sinkhorn, 1964] Sinkhorn, R. (1964). A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879.

[Sinkhorn and Knopp, 1967] Sinkhorn, R. and Knopp, P. (1967). Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348.

[Su et al., 2015] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *CVPR*.

[Sun et al., 2019] Sun, W., Jiang, W., Trulls, E., Tagliasacchi, A., and Yi, K. M. (2019). Attentive context normalization for robust permutation-equivariant learning. *arXiv preprint*

arXiv:1907.02545.

[Thomas et al., 2019] Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, pages 6411–6420.

[Tombari et al., 2010a] Tombari, F., Salti, S., and Di Stefano, L. (2010a). Unique shape context for 3d data description. In *Proceedings of the ACM workshop on 3D object retrieval*, pages 57–62.

[Tombari et al., 2010b] Tombari, F., Salti, S., and Di Stefano, L. (2010b). Unique signatures of histograms for local surface description. In *ECCV*, pages 356–369. Springer.

[Ulyanov et al., 2016] Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

[Valentin et al., 2016] Valentin, J., Dai, A., Nießner, M., Kohli, P., Torr, P., Izadi, S., and Keskin, C. (2016). Learning to navigate the energy landscape. In *3DV*, pages 323–332. IEEE.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *NeurIPS*, pages 5998–6008.

[Wang and Solomon, 2019a] Wang, Y. and Solomon, J. M. (2019a). Deep closest point: Learning representations for point cloud registration. In *ICCV*, pages 3523–3532.

[Wang and Solomon, 2019b] Wang, Y. and Solomon, J. M. (2019b). Prnet: Self-supervised learning for partial-to-partial registration. In *NeurIPS*, pages 8812–8824.

[Wang et al., 2019] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):1–12.

[Wu et al., 2015] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920.

[Xiao et al., 2013] Xiao, J., Owens, A., and Torralba, A. (2013). Sun3d: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, pages 1625–1632.

[Yang et al., 2020] Yang, H., Shi, J., and Carlone, L. (2020). Teaser: Fast and certifiable point cloud registration. *arXiv preprint arXiv:2001.07715*.

[Yew and Lee, 2018] Yew, Z. J. and Lee, G. H. (2018). 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *ECCV*, pages 630–646. Springer.

[Yew and Lee, 2020] Yew, Z. J. and Lee, G. H. (2020). Rpm-net: Robust point matching using learned features. In *CVPR*.

[Zeng et al., 2017] Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., and Funkhouser, T. (2017). 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, pages 1802–1811.

[Zhang and Singh, 2015] Zhang, J. and Singh, S. (2015). Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *ICRA*, pages 2174–2181. IEEE.

[Zhang et al., 2019] Zhang, J., Sun, D., Luo, Z., Yao, A., Zhou, L., Shen, T., Chen, Y., Quan, L., and Liao, H. (2019). Learning two-view correspondences and geometry using order-aware network. In *ICCV*, pages 5845–5854.

[Zhou et al., 2018] Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.