UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

SCUOLA DI DOTTORATO DI RICERCA IN: Ingegneria dell'Informazione

INDIRIZZO: Scienza e Tecnologia dell'Informazione

CICLO: XXXII

3D FEATURE REPRESENTATIONS FOR VISUAL PERCEPTION AND GEOMETRIC SHAPE UNDERSTANDING

Direttore della Scuola: *Ch.mo Prof. Andrea Neviani*
Supervisore: *Ch.mo Prof. Emanuele Menegatti*

Dottorando:
Yongheng Zhao

"Look inside yourself, you are more than what you have become."
~Mufasa's ghost (The Lion King)

Acknowledgments

The majority of the work in this thesis is done between two beautiful cities: Padova and Munich. With many long late nights and missing vacations, I have gain both knowledge and joy. I feel fortunate to experience this research journey and there are many people and institutions I would like to thank for helping me go through this journey.

First, I would like to thank the China Scholarship Council (CSC) for the financial support of my Ph.D. period. I always feel proud of having such a powerful country behind. Then I would like to thank my supervisor Prof. Emanuele Menegatti, my mentors Dr. Federico Tombari and Dr. Tolga Birdal for their continuous support. They have always granted me the freedom to pursue my own ideas, but also provided valuable guidance and new impulses when needed. Furthermore, I would like to thank Prof. Enrico Pagello and Prof. Nassir Navab for their effort in founding such good research facilities.

A particular thank you goes to my exceptional colleagues at Computer Vision Team at the CAMP (Computer Aided Medical Procedures & Augmented Reality), TUM, who make the group the stimulating and very enjoyable working environment it is. Thanks to all of you for the many inspiring discussions.

I would like to thank a number of colleagues who I was fortunate to learn from or/and collaborate with to produce the research in this thesis. In alphabetical order: David Joseph Tan, Haowen Deng, Jan Eric Lenssen, Marco Carraro, Matteo Munaro, Morris Antonello, Shun-Cheng Wu, Yida Wang.

Finally, I would like to thank my girlfriend Yanfang Zhang for the support of the whole Ph.D. period. I could not make it without you.

Abstract

In this thesis, we first present a unified look to several well known 3D feature representations, ranging from hand-crafted design to learning based ones. Then, we propose three kinds of feature representations from both RGB-D data and point cloud, addressing different problems and aiming for different functionality.

With RGB-D data, we address the existing problems of 2D feature representation in visual perception by integrating with the 3D information. We propose an RGB-D data based feature representation which fuses object's statistical color model and depth information in a probabilistic manner. The depth information is able to not only enhance the discriminative power of the model toward clutters with a different range but also can be used as a constraint to properly update the model and reduce model drifting. The proposed representation is then evaluated in our proposed object tracking algorithm (named MS3D) on a public RGB-D object tracking dataset. It runs in real-time and produces the best results compared against the other state-of-the-art RGB-D trackers. Furthermore, we integrate MS3D tracker in an RGB-D camera network in order to handle long-term and full occlusion. The accuracy and robustness of our algorithm are evaluated in our presented dataset and the results suggest our algorithm is able to track multiple objects accurately and continuously in the long term.

For 3D point cloud, the current deep learning based feature representations often discard spatial arrangements in data, hence falling short of respecting the parts-to-whole relationship, which is critical to explain and describe 3D shapes. Addressing this problem, we propose 3D point-capsule networks, an autoencoder designed for unsupervised learning of feature representations from sparse 3D point clouds while preserving spatial arrangements of the input data into different feature attentions. 3D capsule networks arise as a direct consequence of our unified formulation of the common 3D autoencoders. The dynamic routing scheme [1] and the peculiar 2D latent feature representation deployed by our capsule networks bring in improvements for several common point cloud-related tasks, such as object classification, object reconstruction and part segmentation as substantiated by our extensive evaluations. Moreover, it enables new applications such as part interpolation and replacement.

Finally, towards rotation equivariance of the 3D feature representation, we present a 3D capsule architecture for processing of point clouds that is equivariant with respect to the $SO(3)$ rotation group, translation, and permutation of the unordered input sets. The network operates on a sparse set of local reference frames, computed from an input point cloud and establishes end-to-end equivariance through a novel 3D quaternion group capsule layer, including an equivariant dynamic routing procedure. The capsule layer enables us to disentangle geometry from the pose, paving the way for more informative descriptions and structured latent space. In the process, we theoretically connect the process of dynamic routing between capsules to the well-known Weiszfeld algorithm, a scheme for solving *iterative re-weighted least squares (IRLS)* problems with provable convergence properties, enabling robust pose estimation between capsule layers. Due to the sparse equivariant quaternion capsules, our architecture allows joint object classification and orientation estimation, which we validate empirically on common benchmark datasets.

Contents

1. Introduction

1.1. Motivation

Fueled by recent developments in 3d sensing, robotics, autonomous driving and augmented/mixed reality, 3D visual perception and geometric shape processing & understanding have become major research trends in computer vision. Differently from RGB cameras, the sensors used for 3D capture provide rich geometric structure, rather than high-fidelity appearance information. This is proved advantageous for those applications where color and texture are insufficient to accomplish the given tasks, such as reconstruction/detection of texture-less objects. Moreover, 3D data has the born nature to handle the illumination and scale variation properly.

Unlike the RGB camera case, 3D data is represented in a variety of forms: range maps, fused RGB-D images, volumetric data, meshes, and point clouds. Each data representation has its characteristics and application scenario. Thanks to their capability of representing a sparse 3D structure accurately while being acquired directly with the ubiquitous sensors, RGB-D images, and point clouds have been widespread chosen for 3D processing. This thesis will focus on discussing the feature representations based on these two forms of 3D data.

RGB-D feature representations As a lightweight 3D data representation, RGB-D images have a high potential in real-time applications, *e.g.*, object tracking, object detection. The depth range maps of RGB-D images have well-organized data structure and there are plenty of studies of well-investigated RGB features can be transferred to depth range maps, *e.g.*, Histogram of Oriented Depths (HOD) feature [2] transferred from Histogram of Oriented Gradients (HOG) of RGB images. Although the depth range map only covers 2.5D views, it has been demonstrated that RGB features can be enhanced by approaches that fuse color and depth [3, 4]. Inspired by this, we proposed a 3D feature representation that fuses RGB and depth information in a probabilistic manner and evaluated its performance both in accuracy and efficiency.

Learned representations based on point cloud With the highly prosper of the deep learning technology, the 3D feature representations based point clouds have been shifted from hand-craft based to learning based and recent years have seen great progress regarding this field. Deep learning architectures for consuming 3D points have been proposed for sparse 3D representations [5, 6]. These architectures overcame many challenges brought in by 3D data, such as permutation invariance, complexity due to the added data dimension and local density variations.

Representations with spatial attention Unfortunately, the existed 3D deep learning architectures often discard spatial arrangements in data, hence falling short of respecting the parts-to-whole relationship, which is critical to explain and describe 3D shapes; maybe even more severe than in the 2D domain due to the increased dimensionality. Therefore, we propose *3D-PointCapsNet* to learn feature representations from sparse 3D point cloud while preserving spatial arrangements of the input data into different feature attentions. The proposed feature representations improve numerous 3D tasks while enabling interesting applications such as latent space part interpolation or complete part modification, an application where simple cut-and-paste results in inconsistent outputs.

Representations with $SO(3)$ equivariance Although the current learned feature representations from 3D point clouds have achieved equivariance to permutations and translations, they are still limited in other groups (*e.g.*, $SO(3), SE(3)$) which are also practically important. There are related works that augment either the input data or the feature kernels in order to obtain features with rotation invariance. But those features can be considered as extra embedding for equivariant ones in the lower level and they are limited to specific tasks, *e.g.*, shape classification. Besides, the strict/truly equivariance will not maintain in such embedding which makes the feature representation less compact, generative and fail to other tasks like pose estimation. Therefore, we propose a quaternion equivariant point capsule network that is suited to process point clouds and learn feature representation which is equivariant to $SO(3)$ rotations, in addition to preserved translation and permutation equivariance. There are no data or kernels augmentations during the whole training process. The network architecture maintains truly equivariance that results in explicit pose parameterizations as well as invariant latent bottleneck features. Such properties allow us to perform orientation estimation jointly with object classification for the shapes in arbitrary rotations.

1.2. Contributions

The contribution of this thesis is three-fold.

RGB-D statistical feature representations

1. We first propose the object representation by fusing color and depth information in a statistical manner and apply it in mean-shift tracking algorithm. The proposed tracker runs in real-time and achieves the best results compared against the other state-of-the-art RGB-D trackers in a public dataset.

2. We build an RGB-D camera network and manage to integrate the proposed tracker in order to deal with serious occlusion problem and perform accurate tracking performance in the long term.

3. We present a method to obtain object tracking ground truth (3D position) automatically with a robot manipulator.

4. We first create a multiple object tracking dataset in an RGB-D camera network.

3D Point Capsule Networks

1. Motivated by a unified perspective of the common point cloud autoencoders(AE), we propose capsule networks for the realm of 3D data processing as a powerful and effective tool.

2. Our proposed point-capsule AE respects the geometric relationships between the parts and learns the representation with spatial attention which enables shape disentanglement and generation in latent space. This property also enables training part-segmentation with very limited annotated data.

3. We show that our point-capsule AE can surpass the current art in reconstruction quality, local 3D feature extraction and transfer learning for 3D object recognition.

Quaternion Equivariant Capsule Networks

1. We propose a novel, fully $SO(3)$-equivariant capsule architecture that is tailored for simultaneous classification and pose estimation of 3D point clouds. This network produces invariant latent representations while explicitly decoupling the orientation

into capsules, thus attaining equivariance. Note that equivariance results have not been previously achieved regarding the quaternion parameterization of the 3D special orthogonal group.

2. By utilizing LRFs on points, we reduce the space of orientations that we consider and hence can work sparsely on a subset of the group elements.

3. We theoretically prove the equivariance properties of our 3D network regarding the quaternion group. Moreover, to the best of our knowledge, for the first time we establish a connection between the dynamic routing of [1] and Generalized Weiszfeld iterations [7]. By that, we theoretically argue for the convergence of the employed dynamic routing.

4. We experimentally demonstrate the capabilities of our network on classification and orientation estimation of 3D shapes.

1.3. Related Work

1.3.1. RGB-D data based representations for object perception

It has been demonstrated that RGB-D data fusion based features can enhance the performance in object perception tasks, *e.g.*, object tracking [8, 3, 9, 10, 11, 12, 4]. The feature representations of RGB-D data usually is based on and improved upon the 2D features. In general, there are two kinds of 2D features: *template-based* and *color statistics-based*. The *template-based* ones have achieved excellent performance with both high processing speed and robustness to challenging situations, such as fast motion and illumination variation. Therefore, many state-of-the-art RGB-D representations are *template-based* features [10, 11, 3]. However, the *template-based* features are sensitive to shape deformation because the models (like HOG) depend on the spatial configuration which can change rapidly during fast shape variation. On the contrary, *color statistics-based* features are more robust to shape variation while they are insufficiently discriminative and tend to drift towards nearby regions with similar appearance [13]. Compared with exploiting depth information to deal with shape deformation in *template-based* features, it is more desirable to apply depth to enhance the discriminative capability towards background clutter in *color statistics-based* representations. Moreover, although the object representations have shifted from color statistics to complex features based templates [14, 15], *color statistics-based* ones are still able to achieve the sate-of-the-art

performance [16, 17, 18]. Therefore, with the strong potential of depth information on discriminating the object from the background, we apply the fusion of color and depth as target representation which has the capability on dealing with both shape deformation and background clutter issues.

1.3.2. Learning representation from point cloud

End-to-end supervised learning

Thanks to their generic capability of efficiently explaining 3D data without making assumptions on the modality, point clouds are the preferred containers for many 3D applications [19, 20]. Due to this widespread use, recent works such as PointNet [5], spherical convolutions [21] and Monte Carlo convolutions [22] have all devised point cloud-specific architectures that exploited the sparsity and permutation-invariant properties of 3D point sets. Thanks to the point-wise convolutions and the permutation invariant pooling functions of PointNet, many works have extended it primarily to increase the local receptive field size [6, 23, 24, 25, 26]. Point-clouds are generally thought of assets. This makes any permutation-invariant network that can operate on sets an amenable choice for processing points [27, 28]. Unfortunately, common neural network operators in this category are solely equivariant to permutations and translations but to no other groups, *e.g.*, $SO(3), SE(3)$.

Unsupervised learning with 3D autoencoder

Recently, unsupervised architectures followed up on their supervised counterparts. PU-Net [29] proposed better upsampling schemes to be used in decoding. FoldingNet [30] introduced the idea of deforming a 2D grid to decode a 3D surface as a point set. PPF-FoldNet [31] improved upon the supervised PPFNet [32] in local feature extraction by benefiting from FoldingNet's decoder [30]. AtlasNet [33] can be seen as an extension of FoldingNet to multiple grid patches and provided extended capabilities in data representation. PointGrow [34] devised an auto-regressive model for both unconditional and conditional point cloud generation leading to effective unsupervised feature learning. Achlioptas *et al.* [35] adapted GANs to 3D point sets, paving the way to enhanced generative learning. All those 3D autoencoders are embedding the shape into latent features representd by a single vector which is hard for disentanglement. We propose a CapsuleNet based approach to enable feature control as well as shape parsing.

1.3.3. Equivariance in neural networks

The early attempts to achieve invariant feature representations usually involved data augmentation techniques to accomplish tolerance to input transformations [36, 37, 5]. Motivated by the difficulty associated with augmentation efforts and acknowledging the importance of theoretically equivariant or invariant representations, the recent years have witnessed a leap in theory and practice of equivariant neural networks [38, 39].

While laying out the fundamentals of the group convolution, G-CNNs [40] guaranteed equivariance with respect to finite symmetry groups. Similarly, Steerable CNNs [41] and its extension to 3D voxels [42] considered discrete symmetries only. Other works opted for designing filters as a linear combination of harmonic basis functions, leading to frequency domain filters [43, 44]. Apart from suffering from the dense coverage of the group using group convolution, filters living in the frequency space are less interpretable and less expressive than their spatial counterparts, as the basis does not span the full space of spatial filters.

Achieving equivariance in 3D is possible by simply generalizing the ideas of the 2D domain to 3D by voxelized 3D data. However, methods using dense grids [45, 41] suffer from increased storage costs, eventually rendering the implementations infeasible. An extensive line of work generalizes the harmonic basis filters to $SO(3)$ by using *e.g.*, a spherical harmonic basis instead of circular harmonics [46, 47, 48]. In addition to the same downsides as their 2D, these approaches have in common that they require their input to be projected to the unit sphere [49], which poses additional problems for unstructured point clouds. A related line of research are methods which define a regular structure on the sphere to propose equivariant convolution operators [50, 51].

To learn a rotation equivariant representation of a 3D shape, one can either act on the input data or on the network. In the former case, one either presents augmented data to the network [5, 36] or ensures rotation-invariance in the input [52, 32, 53]. In the latter case, one can enforce equivariance in the bottleneck so as to achieve an invariant latent representation of the input [54, 55, 56]. Further, equivariant networks for discrete sets of views [57] and cross-domain views [58] have been proposed. Here, we aim for a different way of embedding equivariance in the network by means of an explicit latent rotation parametrization in addition to the invariant feature. [59] developed *Vector Field Networks*, which was followed by the 3D *Tensor Field Networks* (TFN) [55] that are closest to our work. Based upon a geometric algebra framework, the authors did achieve localized filters that are equivariant to rotations, translations, and permutations. Moreover, they are able to cover the continuous groups. However, TFN is designed for

physics applications, is memory consuming and a typical implementation is neither likely to handle the datasets we consider nor can provide orientations in an explicit manner.

1.3.4. Capsule networks

2D Capsule Networks

The idea of capsule networks (CNs) was first mentioned by [60], before [1] proposed the *dynamic routing by agreement*, which started the recent line of work investigating the topic. Thanks to their general applicability, CNs have found tremendous use in 2D deep learning. LaLonde and Bagci [61] developed a deconvolutional capsule network, called *SegCaps*, tackling object segmentation. Durate *et al*. [62] extended CNs to action segmentation and classification by introducing *capsule-pooling*. Jaiswal *et al*. [63, 64], Saqur *et al*. [65] and Upadhyay *et al*. [66] proposed Capsule-GANs, *i.e*.capsule network variants of the standard generative adversarial networks (GAN) [67]. These have shown better 2D image generation performance. Lin *et al*. [68] showed that capsule representations learn more meaningful 2D manifold embeddings than neurons in a standard CNN do. Further, capsule networks have been applied for specific kinds of input data, *e.g*.graphs [69], or medical images [70].

There have also been significant improvements upon the initial CN proposal. Hinton *et al*.improved the routing by EM algorithm [71]. Wang and Liu saw the routing as an optimization minimizing a combination of clustering-like loss and a KL regularization term [72]. Chen and Crandall [73] suggested *trainable routing* for better clustering of capsules. Zhang *et al*. [74] unified the existing routing methods under one umbrella and proposed weighted kernel density estimation based routing methods. Zhang *et al*. [75] chose to use the norm to explain the existence of an entity and proposed to learn a group of capsule subspaces onto which an input feature vector is projected. Lenssen *et al*. [76] introduced guaranteed equivariance and invariance properties to capsule networks by the use of group convolutions.

3D Capsule Networks

Up until now, the use of the capsule idea in the 3D domain has been a rather uncharted territory. Weiler *et al*. [77] rigorously formalized the convolutional capsules and presented a convolutional neural network (CNN) equivariant to rigid motions. Jimenez *et al*. [78] as well as Mobniy and Nguyen [79] extended capsules to deal with volumetric medical data. VideoCapsuleNet [62] also used a volumetric representation to handle temporal

frames of the video. Yet, to the best of our knowledge, we are the first to devise a capsule network specifically for 3D point clouds, exploiting their sparse and unstructured nature for representing 3D surfaces. Our work connects to this line of work by extending the concept to 3D point cloud, include 3D auto encoder [80] and 3D rotation group $SO(3)$ equivariance.

1.4. List of Publications

The works we will describe in this thesis have been presented in the following publications:

- **Y. Zhao**, M. Carraro, M. Munaro and E. Menegatti, Robust Multiple Object Tracking in RGB-D Camera Networks, in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, IEEE*, 2017. [8]
- **Y. Zhao**, and E. Menegatti, MS3D: mean-shift object tracking boosted by joint back projection of color and depth, in *The 15th International Conference on Intelligent Autonomous Systems (IAS-15)*, 2018. [81]
- **Y. Zhao**, T. Birdal, H. Deng. and F. Tombari, 3D Point Capsule Networks, in *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. [80]

1.5. Outline

The remainder of this thesis is organized as follows:

We present three kinds of 3D feature representations in parallel, aiming various of feature properties with both hand-crafted design and deep neural networks. More specific background and the evaluations are also introduced respectively in each chapter.

In **Chapter 2**, we propose an RGB-D feature representation based on color and depth statistical model fusion. We also present an object tracking algorithm based on the proposed representation in order to evaluate its performance.

In **Chapter 3** and **Chapter 4**, two kinds of 3D deep learning architectures are presented to learn feature representations from 3D point cloud. In **Chapter 3**, we propose the unsupervised *3D point-capsule networks*, an autoencoder that respects the geometric relationships between the parts and learns the representation with spatial attention. In

Chapter 4, we propose the quaternion equivariant point capsule network or *QE-Network* that learns feature representation which is equivariant to $SO(3)$ rotations compactly parameterized by quaternions, in addition to preserved translation and permutation equivariance.

Finally, we present the conclusions achieved in this thesis and provide both limitations and possible directions for further developments in this field in **Chapter 5**. More experiment details, results and proofs are presented in **Appendix A** and **Appendix B**.

2. RGB-D Statistical Representation with Distance Discrimination

2.1. Introduction

Recently, with the disruptive advent of consumer RGB-D cameras which facilitate the real-time acquisition of aligned color and depth information at no extra computational cost, studying object representations based on RGB-D data are becoming popular. As a light-weighted 3D representation, its real-time capability makes it be widely used in object perception tasks, *e.g.*object detection, tracking and pose estimation

Most of the RGB-D data based object representation is based on and improved upon the well-investigated 2D representation with the depth information. The 2D features based on object color statistics are generative and robust to appearance or shape changing. But they are insufficient to discriminate the target from the background or the foreground occlusion in challenging and cluttered situations.

Therefore, there are related works that try to improve their discriminant power by utilizing surrounding color distribution to weigh the original color model of the target[16, 17]. Since the appearance of the target and its surrounding background can vary significantly during a sequence, this kind of algorithm requires continuous and precise model updating. However, there are potentially model drift problems caused by tracking error accumulation, especially when there is occlusion or fast shape changes.

In order to address those issues, we propose our object representation which combines the depth information with the color statistics model to improve its discriminant power. With the distance discrimination power from the depth information, the model is sufficient to discriminate the target from the background and foreground occlusions. Besides, the depth information can stabilize the model updating even when the object changes appearance rapidly.

In order to evaluate the proposed RGB-D statistical representation, we proposed a short-term object tracker, namely MS3D which exploit the proposed representation in the mean-shift object tracking algorithm. Mean-shift provides an accurate localization

without an exhaustive search by maximizing the Bhattacharyya coefficient between the object and the candidates distributions[82]. Additionally, we propose a Bhattacharyya coefficient based occlusion handling mechanism which actively detects occlusion and re-localizes the object after short-term occlusion.

We have evaluated MS3D tracker on an RGB-D object tracking dataset created by Xiao et al. [3]. The results shows that our proposed tracker outperforms the base-line tracker of the dataset[3], the other state-of-the-art RGB-D trackers[10, 11, 9] and another mean-shift based RGB tracker[17].

Additionally, we built an RGB-D camera network and integrate with the MS3D tracker in order to evaluate the proposed representation in long-term object tracking which is required in real applications. We created a dataset of multiple objects in multiple RGB-D cameras in the evaluation. The quantitative evaluation result suggests that our algorithm is able to track multiple objects precisely and continuously in many challenge situations, *e.g.*, heavy occlusion, fast motion and illuminate variation.

2.2. Joint RGB and Depth statistical representation

2.2.1. Discriminative object color representation

Foreground extraction

Most perception algorithms build object models from an ROI (region of interest) containing the object to be tracked, see Fig. 2.1(a). This ROI commonly is a rectangle consisting of both the target pixels and a small number of background pixels, see Fig. 2.1(b). In order to obtain an object representation which is able to discriminate the target object from the background, the combination of mean-shift segmentation and region growing was proposed in [83]. Instead, we use the depth distribution analysis to extract the pixels in the ROI belonging to the target. The depth distribution of the ROI in Fig. 2.1(b) is displayed as a histogram in Fig. 2.1(c). We exploit a peak searching method to find the peaks of depth histogram. Here, we make an assumption that the target object is not occluded in the initial frame of the sequence. This makes the pixels belonging to the target object closer to the camera than other pixels in the ROI. With this assumption, the depth histogram can be divided into two parts: foreground peak with the minimum depth and background peaks with higher values of depth. Then we estimate the target depth interval around the foreground peak. This depth interval can be used to create a mask on the ROI to filter out the ROI pixels belonging to the background. As shown in Fig. 2.1(c),

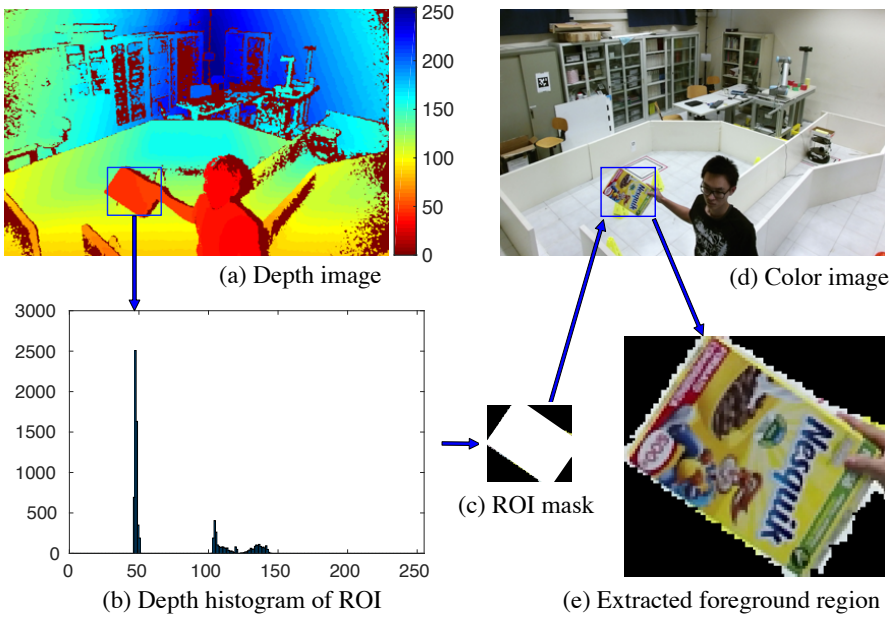


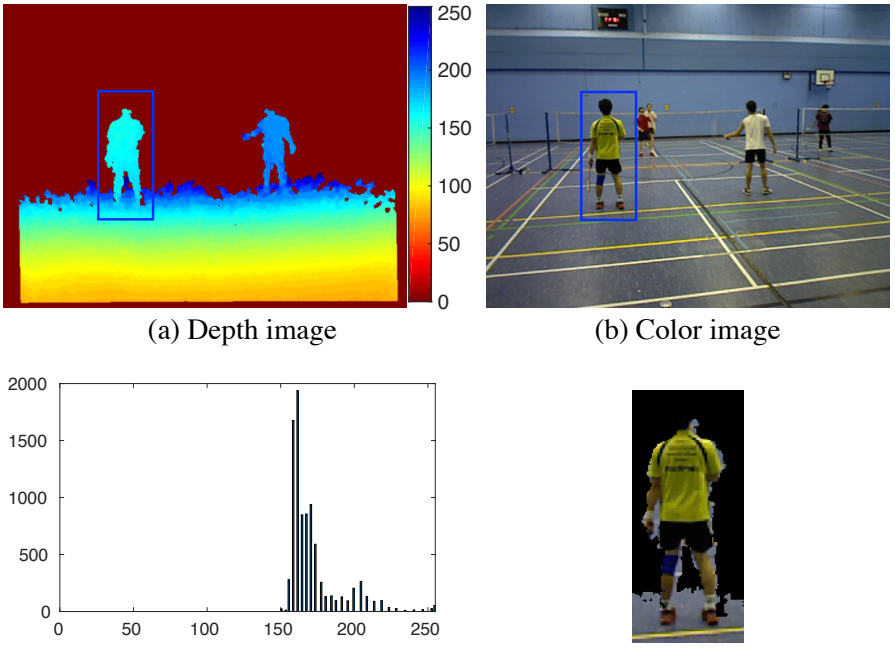
Figure 2.1.: Target Region Extraction. The depth distribution of the ROI in (a) is displayed as histogram in (b). The ROI mask in (c) is generated by the estimated foreground depth interval. The extracted target region is shown in (e) with background masked in black.

the pixels in the ROI belonging to the background have been masked in black.

This target extraction method performs well when the object and background are distant (in depth), but can perform poorly when the object is close to (or even touching) the background, as shown in Fig. 2.2 where the athlete stands on the floor. In this case, the calculated foreground can easily include pixels of the background, failing to mask them out. To prevent this, we apply the so-called *s*urrounding background weighting to filter out pixels erroneously assigned to the foreground.

Background weighting

If the pixels of the background cannot be masked out using the depth information, like in Fig. 2.2(d). We propose to use the probabilistic description of the color of the background to filter them out. The intuition is to use the color histogram of a larger ROI surrounding the target ROI, and thus likely comprising a large portion of the background, to weight the color histogram of the ROI. The higher is the frequency (amount of pixels) of one color that is present in the surrounding ROI, the lower weight will be used to weigh the frequency of this color in the ROI. After this weighting, we can obtain a discriminative



(a) Depth image (b) Color image

(c) Depth histogram of ROI (d) Extracted foreground region

Figure 2.2.: Inaccurate Target Region Extraction. The depth histogram showed in (c) represent the depth distribution of athlete and floor. The depth distribution is insufficient to filter all the background (d).

color model to represent the target object with a foreground color likelihood. An example of the surrounding region is shown in Fig. 2.3(a) with a larger rectangle around the ROI.

Denoting $\hat{q}_F$,$\hat{q}_S$ as the normalized color histogram calculated over region F and S where F is the extracted foreground region mentioned in Section 2.2.1 while S is its surroundings. The surrounding weighted color histogram $\mathbf{q}_{rgb}$ with m_1-bin is calculated as:

$$q_{\text{rgb}}(u_1)=\begin{cases} \hat{q}_F(u_1)\cdot\frac{\min(\hat{q}_S)}{\hat{q}_S(u_1)} & \text{if } \hat{q}_S(u_1)>0 \\ \hat{q}_F(u_1) & \text{if } \hat{q}_S(u_1)=0 \end{cases} \tag{2.1}$$

where u_1 is the index of bins in histogram and $u_1 = 1...m_1$. $\min(\hat{q}_S)$ is the lowest frequency in the surrounding histogram and it is used to normalize the weight.

The weighted color histogram is then normalized into 2.2

$$\hat{\mathbf{q}}_{\text{rgb}} = \left\{\hat{q}_{\text{rgb}}(u_1)\right\}_{u_1=1...m_1} \quad \sum_{u_1=1}^{m_1}\hat{q}_{\text{rgb}}(u_1) = 1 \tag{2.2}$$

(a) Color image

(b) Probability distribution image back-projected by original color histogram

(c) Probability distribution image back-projected by our proposed color model

Figure 2.3.: Probability distribution images back-projected by different color model. The object color model (histogram) obtained from color image can be back-projected to image in order to obtain probability distribution images which represents the object model likelihood in pixels.

The object color histogram obtained from the color image can be back-projected to the image in order to obtain the probability distribution images which represents the object model likelihood in pixels. See for example in Fig. 2.3(a) the original color image with the bounding boxes of the target and surrounding region. The discriminative capability of color model is clearly shown by the probability distribution images. The probability distribution image obtained with the original color histogram is reported in Fig. 2.3(b) and the one obtained with our proposed weighted histogram is shown in Fig. 2.3(c). With our proposed color model, the probability of background pixels to be included in the foreground is effectively decreased which makes the color model more robust to background interference.

2.2.2. Object depth statistical representation

The proposed discriminative color model has enhanced the capability of basic color statistics on dealing with background clutter. However, the surrounding background can vary significantly during a sequence while the proposed discriminative model is efficient solely if the background is similar to the one shown in the initial frame. Therefore, we fused this color statistics model with another depth model in order to deal with the interference caused by continuously changing the background and other objects with a similar appearance.

Usually, to filter the background and the occluding objects in the search window, the depth interval constraint is widely used in RGB-D perception algorithm. In this work, we propose a two-joint terms likelihood based contextual peaks matching algorithm to

predict the interval in a non-parametric way. If the target object is not fully occluded, the depth distribution of the search window can be ideally considered into three components: target, background, and occluder[4]. Each component may have a relevant peak on the depth histogram. We try to match the target peak in contextual depth histogram in order to obtain the center of the current depth constraint interval. The "range-invariant" model[3] is applied to represent the depth interval since the depth range of the target in adjacent frames can be considered as invariant. In summary, the center and range of the depth interval can be obtained by the matched target peak and target depth range in the previous frames.

The likelihood of two-joint terms, depth index and its frequency (amount of pixels) of the peaks in contextual depth histogram are applied in matching. The Standardized Euclidean Distance is used to measure the likelihood of the peaks which are extracted after morphological pre-processing. The depth histogram of the current search window with $m_2 -$ bin can be represented with:

$$\hat{\mathbf{q}}_{\mathrm{d}} = \{\hat{q}_d(u_2)\}_{u_2=1\ldots m_2} \quad \sum_{u_2=1}^{m_2} \hat{q}_d(u_2) = 1 \tag{2.3}$$

where $\hat{q}_d(u_2)$ denotes the frequency of depth index u_2. Let $P_{k-1}^T : (d_{k-1}^T, f_{k-1}^T)$ denote the peak that represents the target in depth histogram of frame $k-1$ and let $P_{k,i}^C : (d_{k,i}^C, f_{k,i}^C), i \in [1,n]$ be the n candidate peaks in frame k. (d,f) represents the depth index of the peak and its frequency in histogram respectively. The Standardized Euclidean Distance between two peaks is calculated with

$$D(P_{k,i}^C, P_{k-1}^T) = \sqrt{\left(\frac{d_{k,i}^C - d_{k-1}^T}{\sigma_d}\right)^2 + \left(\frac{f_{k,i}^C - f_{k-1}^T}{\sigma_f}\right)^2} \tag{2.4}$$

where (σ_d, σ_f) denotes the depth index and frequency variance of the peak set composed of the target and all candidate peaks. Then, the center of the depth constraint interval is the depth index of the matched peak $P_k^T : (d_k^T, f_k^T)$ with the minimized distance. The range of interval is equal to the target depth range $[d_{k-1}^T - r_{l,k-1}, d_{k-1}^T + r_{r,k-1}]$ in frame $k-1$. $[r_{l,k-1}, r_{r,k-1}]$ is the range between the center of depth interval and its left/right edges in frame $k-1$.

However, the depth interval prediction is sensitive even to a very little amount of outliers caused by the shape and pose deformation, leading to a wider depth interval than the real target depth range. Therefore, we back-project the predicted interval on the current depth histogram of the search window in order to obtain a constrained depth

histogram as the final depth model that is more robust to outliers. The constrained depth histogram $\hat{\mathbf{q}}'_d$ can be calculated with 2.5.

$$\hat{\mathbf{q}}'_{\mathrm{d}} = \left\{\hat{q}'_{\mathrm{d}}(u_2)\right\}_{u_2=1\ldots m_2} \quad \sum_{u_2=1}^{m_2} \hat{q}'_{\mathrm{d}}(u_2) = 1 \tag{2.5}$$

and

$$\hat{q}'_{\mathrm{d}}(u_2) = C \cdot \mathrm{H}[(r_{l,k-1} - d_k^T + u_2)(r_{r,k-1} + d_k^T - u_2)] \cdot \hat{q}_d(u_2) \tag{2.6}$$

where H is the Heaviside step function and C is the normalization constant which is derived by imposing the condition $\sum_{u_2=1}^{m_2} \hat{q}'_d(u_2) = 1$.

2.2.3. Model fusion

The object is finally represented by the joint color and depth pdf which denotes the joint probability of both object color and depth. Since the color and depth are independent, the joint PDF is the product of the marginals

$$P(rgb,d) = P(rgb)P(d) \tag{2.7}$$

The marginals which are color PDF and depth PDF estimated separately by discriminative color histogram $\hat{\mathbf{q}}_{rgb}$ and constraint depth histogram $\hat{\mathbf{q}}_{\mathrm{d}}$. The target is modeled as an $(m1 \cdot m2)$-bin kernel-estimated histogram in a feature space located at the origin in

$$\hat{\mathbf{q}}_{\mathrm{rgb,d}} = \hat{\mathbf{q}}_{\mathrm{rgb}}\hat{\mathbf{q}}'_{\mathrm{d}} = \{\hat{q}_{\mathrm{rgb,d}}(u)\}_{u=1..m_1\cdot m_2} \sum_{u=1}^{m_1\cdot m_2} \hat{q}_{\mathrm{rgb,d}}(u) = 1 \tag{2.8}$$

where

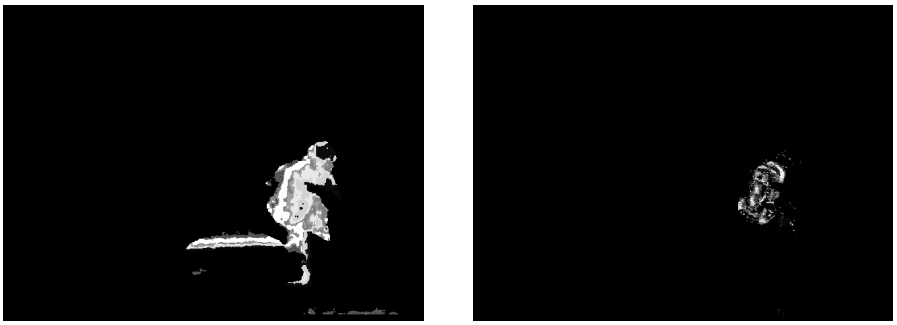
$$\hat{q}_{\mathrm{rgb,d}}(u\,|u_1,u_2) = C' \cdot \hat{q}_{\mathrm{rgb}}(u_1)\hat{q}'_{\mathrm{d}}(u_2) \tag{2.9}$$

and C' is the normalization constant.

Similar to Fig. 2.3, we back-project the joint PDF to current color and depth image in order to get the probability distribution image which represents the object model likelihood in pixels. Fig. 2.4 shows an example of the discriminative power of the joint PDF. In Fig. 2.4(a), one can notice that there is a colored object in the background with the similar appearance as the target object, which could easily cause model drift if only the color information is considered (see the Probability distribution image projected by color model only in Fig. 2.4(b)). At the same time, the depth between the target and the person is similar which will also lead to model drift if only exploiting depth distribution (see

(a) Color image

(b) Probability distribution image with color back-projection



(c) Probability distribution image with depth back-projection

(d) Probability distribution image with joint back-projection

Figure 2.4.: Probability distribution image by different back-projection. In (d), the proposed joint back-projection shows the discriminative power against both objects with similar color and objects with similar depth.

the probability distribution image projected by depth PDF only in Fig. 2.4.c). However, the object model is more discriminative when the proposed joint color and depth PDF is applied, see Fig. 2.4(d).

2.3. Object tracking with proposed representation

2.3.1. Mean-shift based object localization

The robust model of the object to be tracked we created with the color and depth joint PDF is exploited in a mean-shift tracking framework to discriminate the object to be tracked from similar objects. A target candidate which has the same size of the last tracked target is located at y in the current search window and described by its histogram.

$$\mathbf{p}_{\mathrm{rgb,d}}(y) = \{\hat{p}_{\mathrm{rgb,d}}(u,y)\}_{u=1\ldots m_1\cdot m_2} \sum_{u=1}^{m_1\cdot m_2} \hat{p}_{\mathrm{rgb,d}}(u) = 1 \tag{2.10}$$

The candidate joint PDF is also the product of the original color PDF and the depth PDF. While the depth PDF is estimated from the depth histogram of the subregion located at y.

Similar to [14], the distance between two probability distributions can be measured by

$$d(y) = \sqrt{1 - \rho[\hat{\mathbf{p}}_{\mathrm{rgb,d}}(y), \hat{\mathbf{q}}_{\mathrm{rgb,d}}]} \tag{2.11}$$

where the Bhattacharyya coefficient between p and q are

$$\hat{\rho}(y) = \rho[\hat{\mathbf{q}}_{\mathrm{rgb,d}}(y), \hat{\mathbf{q}}_{\mathrm{rgb,d}}] = \sum_{u=1}^{m_1\cdot m_2} \sqrt{\hat{p}_{\mathrm{rgb,d}}(u,y)\hat{q}_{\mathrm{rgb,d}}(u)} \tag{2.12}$$

By maximizing the Bhattacharyya coefficient, we can localize the target from all the candidates. In this work, we use the same localization method as [14]. The localizer starts from the center of last tracked target y_0 and moves to a new location step by step by utilizing gradient optimization with the mean-shift algorithm. Let $\{\mathbf{x_i}\}_{i=1\ldots n_h}$ denote the pixel locations of target candidate in location y_0 of current frame and h the bandwidth, then the new location y_1 is

$$y_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x_i} w_i g(\left\| \frac{y_0 - \mathbf{x_i}}{h} \right\|^2)}{\sum_{i=1}^{n_h} w_i g(\left\| \frac{y_0 - \mathbf{x_i}}{h} \right\|^2)} \tag{2.13}$$

where

$$w_i = \sum_{u=1}^{m_1 m_2} \sqrt{\frac{\hat{q}_{\mathrm{rgb,d}}(u)}{\hat{p}_{\mathrm{rgb,d}}(u, y_0)}} \delta[b(\mathbf{x_i}) - u] \tag{2.14}$$

and $g(x) = -k'(x)$. Here, the function $k(x)$ is a convex and monotonic decreasing kernel profile. Function b: $2R^2 \rightarrow 1\ldots m_1 \cdot m_2$ associates to the pixel at location $\mathbf{x_i}$ of both color and depth image to index $b(\mathbf{x_i})$ of its bin in the quantized feature space. σ is the Kronecker delta. Searching will finally converge to a new location by considering the length of step and the iteration times.

2.3.2. Occlusion handling and scale estimation

Occlusion handling

Being able to detect occlusions is of great importance to enhance the overall tracking quality. With mean-shift localizer mentioned in Sec.2.3.1, we can get the optimized location by maximizing the Bhattacharyya coefficient. However, the Bhattacharyya coefficient could decrease significantly during occlusion so it can be utilized to detect occlusion. We apply a threshold on the ratio of current and initial maximized Bhattacharyya coefficient to detect its great variation in order to recognize occlusion. We assign y_m as the optimized location in the current frame. The ratio can be calculated as

$$r = \hat{\rho}^i / \hat{\rho}^c(y_m) \tag{2.15}$$

where $\hat{\rho}^i$ is the sum of frequency in the initial joint PDF. It can be considered as the initial maximized Bhattacharyya coefficient.

$$\hat{\rho}^i = \sum_{u=1}^{m_1 \cdot m_2} \hat{q}^i_{\mathrm{rgb,d}}(u) = 1 \tag{2.16}$$

and

$$\hat{\rho}^c(y_m) = \sum_{u=1}^{m_1 \cdot m_2} \sqrt{\hat{p}^c_{\mathrm{rgb,d}}(u, y_m) \hat{q}^c_{\mathrm{rgb,d}}(u)} \tag{2.17}$$

represents the current maximized Bhattacharyya coefficient at location y_m.

When the occlusion is detected, the global target re-localization procedure will generate an expanding search window centered in the last tracked window to re-localize the target object. The depth PDF will stop updating during occlusion since the changed depth data of the object can not be fetched. By maximizing color PDF based Bhattacharyya coefficient with mean-shift searching, a strict restriction will be conducted on the optimized location in order to ensure an accurate re-localization. This restriction is composed of two joint terms: the ratio of the initial and the current maximized color Bhattacharyya coefficient and its density over the converged search window. Let $\hat{\rho}_{rgb}$ denotes the maximized color Bhattacharyya coefficient, r_1, r_2 represent the two joint terms:

$$\begin{cases} r_1 = \hat{\rho}^i_{\mathrm{rgb}} \Big/ \hat{\rho}^c_{\mathrm{rgb}}(y_m) \\ r_2 = n_c \hat{\rho}^i_{\mathrm{rgb}} \Big/ n_i \hat{\rho}^c_{\mathrm{rgb}}(y_m) \end{cases} \tag{2.18}$$

where n_i is the number of pixels in the initial ROI while n_c is the size of the expanded search window at the optimized location y_m. $\hat{\rho}^i_{rgb} = 1$ can be considered as initial maximized color Bhattacharyya coefficient.

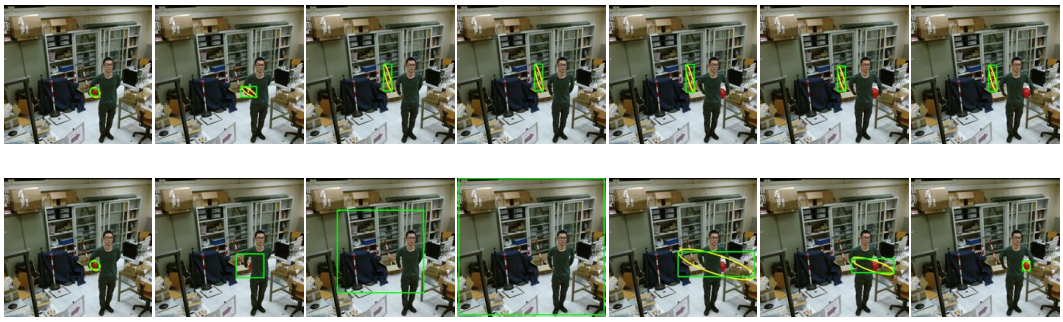


Figure 2.5.: Tracking without occlusion handling (top) and with occlusion handling (bottom).

The tracking result with and without handling occlusion during occlusion is shown in Fig. 2.5. The target is occluded from the third image. In the top row, without occlusion handling, the tracker drifts to the background where there is a self-similar colored object. However, in the bottom row, the occlusion is well-detected and the global re-localizer start to expand the search window in the third frame. When the object is partially recovered from full occlusion in the fifth frame, the re-localizer start to rescale the search window until the object is fully recovered in the previous frame.

Scale estimation

In order to deal with the scale variation problem, we apply the scale estimation mechanism in Cam-shift[84] to adjust the size of the converged search window. By back-projecting joint PDF on color and depth image, we can get the joint probability distribution image. After obtaining the optimized location by mean-shift searching algorithm, we can re-scale the search window by invariant moments. Letting (x_c, y_c) to denote the optimized location, then the estimated size for re-scaling (l, w) is calculated as 2.19:

$$\begin{cases} l = \sqrt{\dfrac{(a+c)+\sqrt{b^2+(a-c)^2}}{2}} \\ w = \sqrt{\dfrac{(a+c)-\sqrt{b^2+(a-c)^2}}{2}} \end{cases}, \begin{cases} a = \dfrac{M_{20}}{M_{00}} - x_c^2 \\ b = 2 \times (\dfrac{M_{11}}{M_{00}} - x_c \times y_c) \\ c = \dfrac{M_{02}}{M_{00}} - y_c^2 \end{cases} \tag{2.19}$$

where M_{pq} is the $max(p,q)$ order moment which is given as

$$M_{pq} = \sum_x \sum_y x^p y^q I(x,y) \tag{2.20}$$

and

$$I(x,y) = \hat{q}_{\text{rgb,d}}(b(x,y)) \tag{2.21}$$

denotes the the joint probability distribution image. Here, b is the same index function mentioned in 2.14. The tracking result is the re-sized converged search window which will be expanded as the ROI of the subsequent frame.

2.4. Evaluation

2.4.1. Short-term tracking evaluation

As mentioned above, we evaluated the proposed single object tracker on the new public RGB-D object tracking dataset built by Xiao et al. [3]. We preferred this new dataset to the more famous PTB benchmark[4] because the latter is affected by some issues. Notably, the authors of [3] outlined 4 problems of PTB: (i) it contains some sequences where the RGB and depth images pairs are not synchronous; (ii) over half of the PTB image sequences are devoted to people tracking which introduce a bias in the results of evaluating tracking of generic targets; (iii) the majority of the benchmark videos are captured by a stationary camera; (iv) many videos in PTB are imaging the same scene, thus there is not enough variability in the background.

We tested our tracker on this new dataset with the evaluation methodology from OTB [85]. This methodology does not allow re-initialization. If the tracker loses the object during the sequence, no information on the ground-truth is provided to the tracker to locate again the object.

We compared our tracker proposed in this work with:

- the RGB-D trackers which have best performances in PTB benchmark: PT[4], DS-KCF1[11], DS-KCF2[10], OAPF[9]
- the base-line tracker of the new dataset (STC [3])
- a mean-shift based tracker (ASMS [17]) which achieved state-of-the-art performance.

The tracking result of DSKCF, OAPF, STC are taken by what is reported in [3] (see [3] for a detailed description of the labels).

We exploit the AUC (area-under-the-curve) of region overlap ratio to evaluate some state-of-the-art RGB-D trackers. The trade-off curves of all sequences are shown in Fig. 2.6.

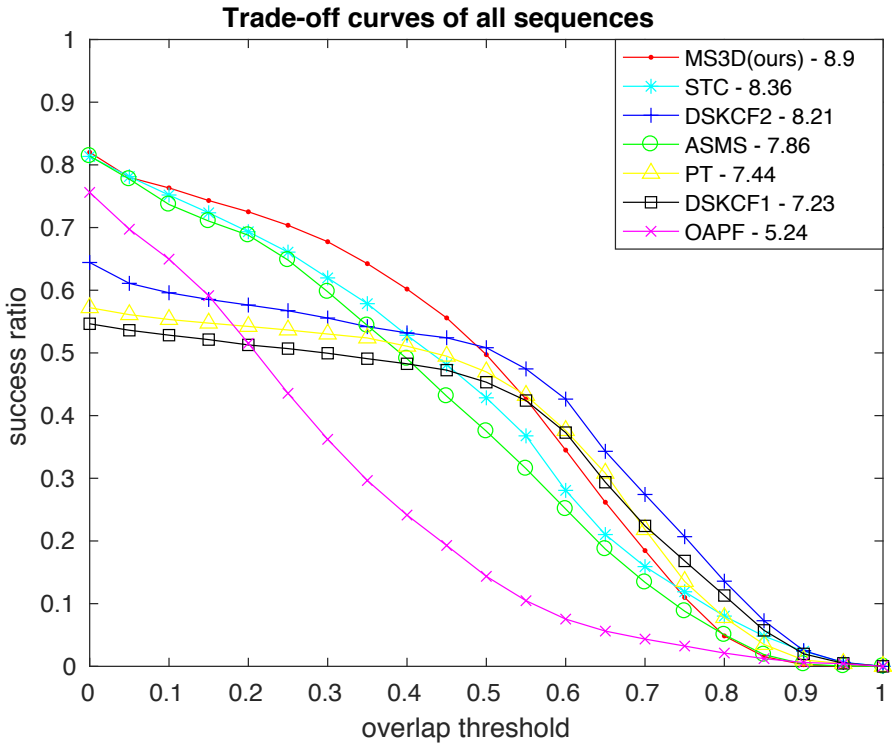


Figure 2.6.: Trade-off curves of all sequences

It is clear to see from the Fig. 2.6 that our algorithm outperforms others. The overall performance has been improved by 0.44 by comparing MS3D with STC which is the baseline algorithm of the new dataset.

Tab. 2.1 shows the AUC performance of trackers in different attributes. Our tracker outperforms the others in both stationary, moving cameras as well as most of the challenging situations. The improvement in DV (Depth variation), DDV (Depth distribution variation) of our tracker successfully proved the functionality of the proposed depth model. However, it does not perform well in IV (Illumination variation), CDV (Color distribution variation), SCC (Surrounding color clutter) since the color varies significantly between the initial frame and the rest in these sequences (*e.g.*, turn off lights, change clothes) and MS3D doesn't exploit color model update strategy in order to avoid model drift problem. However, in our opinion, the great change of appearance should belong to the detection problem rather than tracking.

The running performance of MS3D is tested on a desktop computer (Intel Core i7-3820 @3.60GHz) with the image resolution of 640×480. Since the proposed tracker depends

on mean-shift localizer rather than sliding window, it has a high frame rate of over 100 FPS when there is no occlusion. During full occlusion, the frame rate will decrease to 30-50 FPS because of the global searching of the re-localizer.

Qualitative evaluation

As shown in Fig. 2.7, the proposed tracker (MS3D) outperforms the other trackers in various situations. From top to bottom, the challenge factors are scale variation and outdoor, fast motion, shape deformation, self-similar colored objects interference and outdoor in dark, shape deformation with the self-similar background. In several of these challenging scenes, the objects are held by persons which can lead to serious background clutters in depth image. However, MS3D survived in all of them because of its capability to discriminate both color and depth difference.

2.4.2. Long-term tracking evaluation

Long-term object tracking

The long-term object tracking algorithm is required in practical scenarios. The main difference between the long-term and short-term tracking algorithm is that the former has the capability of dealing with the object disappearance for a long period of full occlusion. This capability demands an efficient object re-detection or re-localization mechanism[86].

The occlusion handling mechanism proposed in short-term MS3D tracker is able to re-localize the object after partial occlusion and after a short-term heavy occlusion, but it can not handle long-term full occlusion. Therefore, the MS3D tracker belongs to the short-term trackers and it needs an efficient re-detection algorithm when applied to the real application which requires long-term tracking.

During long-term full occlusion, the appearance and shape of the objects could vary significantly and the detection model might not be able to be correctly updated. Moreover, the motion of the object is also hard to predict during this period which might lead to fragmentary tracks. Furthermore, full occlusion could lead to serious ID switch problems in the context of multiple object tracking.

Object tracking in RGB-D camera network

In order to deal with long-term occlusion, ID switch problem and obtain a continuous, large-scale tracking, we apply the proposed MS3D tracker in RGB-D camera network

Table 2.1.: AUC of bounding box overlap, RED demotes best performing tracker.

	Overall	Stationary	Moving	IV	DV	SV	CDV	DDV	SDC	SCC	BCC	BSC	PO
MS3D(ours)	**8.9**	**9.89**	**7.98**	5.88	**7.86**	**5.98**	3.03	**8.57**	**8.67**	9.28	**7.95**	**7.9**	**8.26**
STC	8.36	9.57	7.18	5.78	7.56	5.07	5.12	7.66	8.01	9.53	6.67	7.17	7.73
DSKCF2	8.21	9.36	7.13	6.10	**7.88**	4.39	0.94	5.26	7.93	**9.81**	5.66	6.50	7.76
ASMS	7.86	8.48	7.28	**6.99**	7.48	5.66	**6.58**	6.96	7.37	7.41	6.67	7.28	7.75
PT	7.44	8.54	6.43	4.15	6.65	2.81	0.43	3.49	6.80	8.23	5.73	5.76	6.20
DSKCF1	7.23	7.52	6.85	5.50	7.06	3.36	1.43	4.16	7.90	8.25	4.82	5.16	6.02
OAPF	5.24	6.0	4.54	3.18	4.45	3.07	3.22	3.71	5.00	6.13	3.79	4.82	5.82

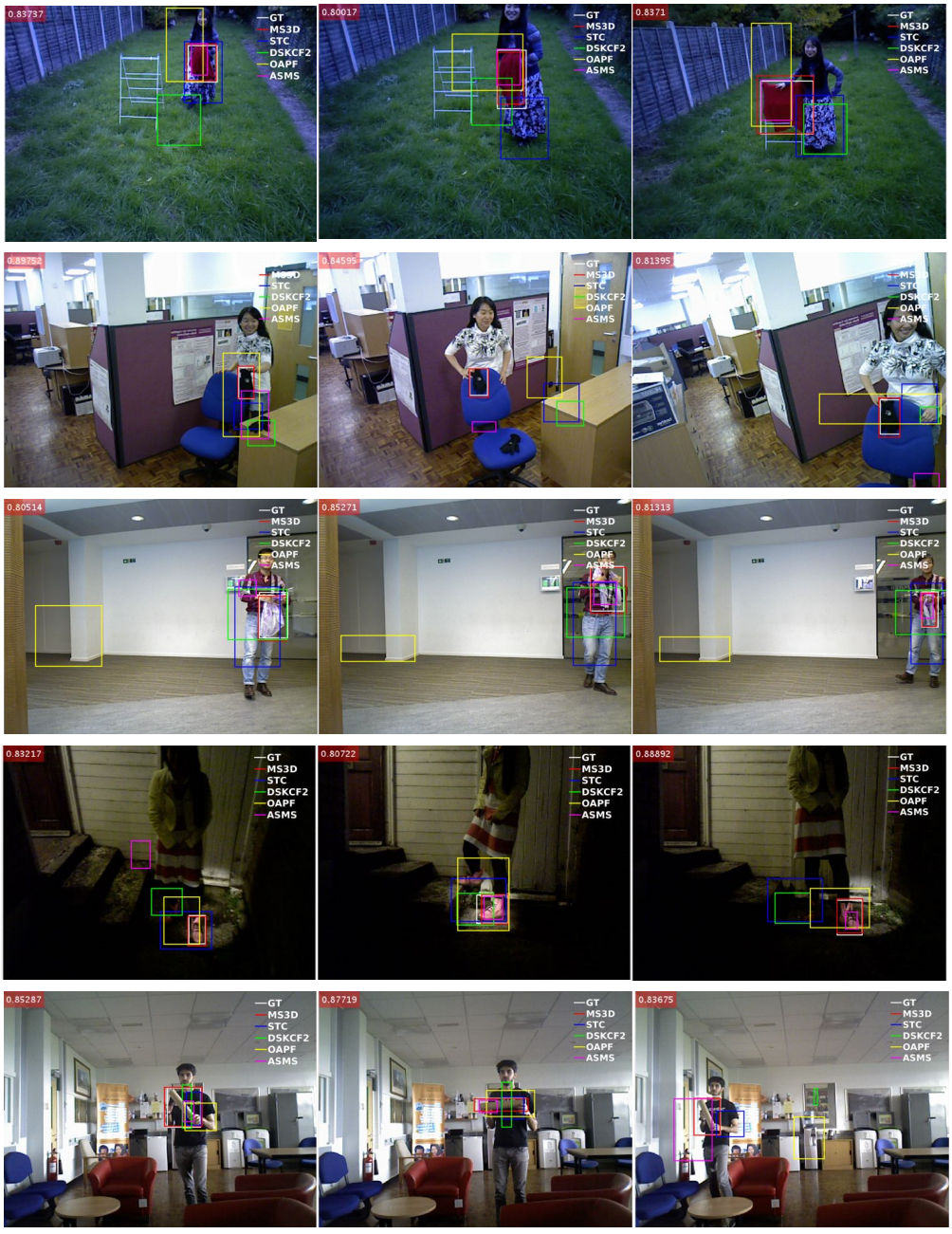


Figure 2.7.: A visual review of the trackers' performance in sequences with various of challenges. The tracking results are shown in different colored bounding box with a demonstration legend on the top-right corner. The overlap ratio between MS3D tracking result and ground truth is shown at the top-left corner.

Figure 2.8.: Images captured from multiple view points of the RGB-D camera network.

which is first introduced in OpenPTrack[87, 88] for people tracking. In the camera network, there are multiple RGB-D cameras which are calibrated and set up in proper positions in order to obtain multiple viewpoints. In Fig. 2.8, there are four cameras with an overlapped field of view. The objects may occlude each other in one of the camera views, but they can be well detected in another.

In each node of the camera network, an MS3D tracker is generated for each object to be tracked. So, each RGB-D camera is running multiple instances of the MS3D algorithm. We design a trackers fusion algorithm to handle these distributed trackers from all the nodes. The schematic diagram and procedure is shown in Fig. 2.9 and Alg. 1. Firstly, the fusion algorithm initializes a Kalman filter for each object with its identity and the initial tracking results. Then it associates coming tracking results(a set of 3D centers from one camera) to the Kalman filters by considering the likelihood between the 3D center and the state estimation of Kalman filter. The Kalman filters will be appropriately updated after association. In the meantime, the trackers fusion algorithm back-projects the current state estimation to the specific tracker in order to re-localize the target once the occlusion in this tracker is detected.

Distributed trackers association The tracking results from the MS3D tracking algorithm is a 2D bounding box. By projecting the center points of the 2D box to the world coordinate, we can get the targets' predicted 3D centers. Then, the trackers fusion algorithm associates each center to one of the Kalman filters by motion likelihood every time a new set of centers arrives. The Mahalanobis distance between the 3D center and estimated motion distribution of the Kalman filter is exploited to measure the motion likelihood.

We compute the Mahalanobis distance between the estimated motion distribution of i-*th* Kalman filter and j-*th* predicted 3D center as:

$$D_M(I_i, t_j) = (t_j - \mu_{I_i}) \cdot \Sigma_{I_i} \cdot (t_j - \mu_{I_i})^T \tag{2.22}$$

where the measured motion vector:

$$t_j = [x_j, y_j, z_j, vx_j, vy_j, vz_j]$$

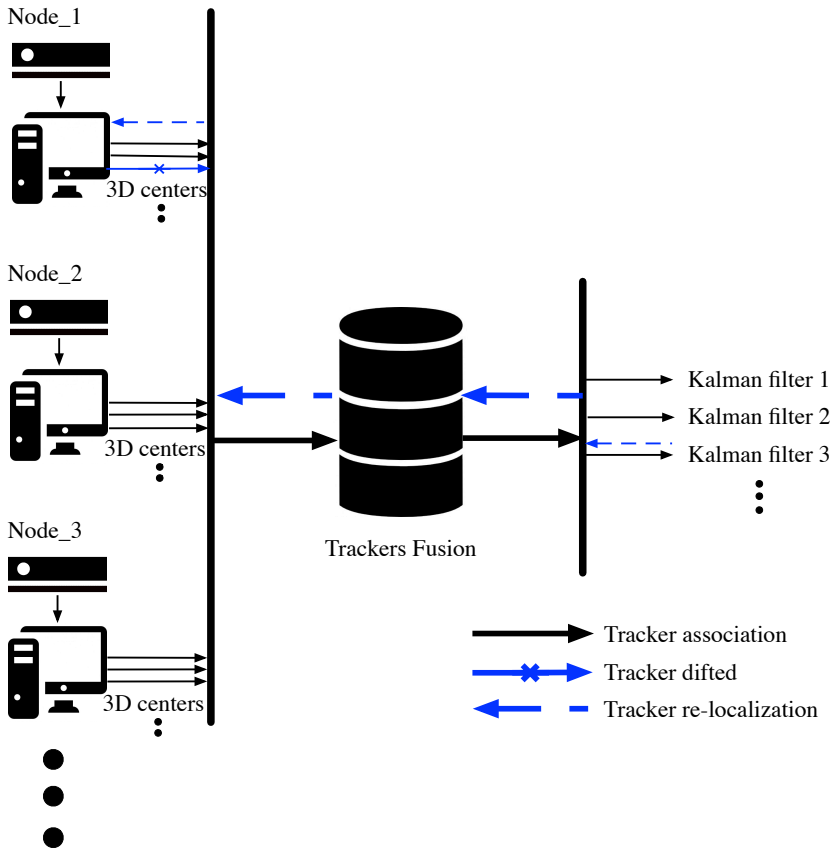


Figure 2.9.: Distributed Trackers Fusion. The 3D centers exported by the trackers are associated to the Kalman filters one node by one node. The drifted tracker is detected when it fails the association (line with blue color). Then the current state estimation of the recent associated Kalman filter will be used to back-project to this drifted tracker in order to optimize the re-localization.

Algorithm 1: Distributed Trackers Fusion

Input: Predicted 3D centers of trackers in camera reference of all the objects in all the camreas.
Output: Current state estimation of Kalman filters: 3D centers of all the objects in 3D world reference.
if *Callback with a set of predicted 3D centers: t* **then**
    /* *n postions from camera k* */
    if *Kalman filters are not initialized* **then**
        Initialze the motion distribution of Kalman filter K with t;
    else
        for $i = 0$ **to** n **do**
            Find the i-th Kalman filter's nearest predicted 3D center t_j in t;
            /* $j \in \{1,2...n\}$, *measurment in mahalanobis distance* */
            if $D_M(I_i, t_j) < \Theta$ **then**
                /* *Association success* */
                Update i-th Kalman filter;
            else
                /* *Association failed* */
                Tracker j in camera k drifted;
                Project 3D center of Kalman filter x which is recentlly associated to tracker j, to camera k in order to re-localize the drifted tracker.

is composed of the j-th predicted 3D center position and the velocity that i-th instance would have if j-th predicted 3D center was associated to it. $I_i \sim (\mu_{I_i}, \Sigma_{I_i})$ is the estimated motion distribution of i-th Kalman Filter. As for the motion model, we chose a constant velocity model because it is good at managing full occlusions, as described in [89]. Given that the Mahalanobis distance for multi-normal distributions is distributed as a chi-square, we use this distribution for defining a gating function for updating the instance motion with possible associations and filtering the fault predictions. With continuous and frequent instance motion updating by the distributed trackers all over the camera network, the object can be tracked most of the time, even in presence of long-term and heavy occlusion in some of the cameras.

Distributed trackers re-localization with camera network feedback. In the last section, the fault prediction of drifted trackers can be filtered in the camera network association. However, these trackers will no longer be able to contribute to the fusion algorithm without an appropriate object re-localization. Although the proposed occlusion handling mechanism mentioned in Sec.2.3.2 can re-localize the target by color distribution likelihood, it has inferior performance when there are heavy occlusion and self-similar object interference at the same time.

Therefore, the trackers fusion algorithm is also designed to back-project the current state estimation (3D location of the object in the world reference) of Kalman filter to specific cameras in which there are drifted trackers. The identity of the Kalman filter is well-affiliated with the identity of the object in drifted tracker by considering the contextual association. After this back-projection, the drifted tracker will receive an updated 2D center of the target object which will be used to supervise the re-localization in occlusion handling. The object will be only be considered as successfully re-localized when the current 3D prediction is capable of associating the affiliated Kalman filter. This camera network based feedback mechanism can appropriately supervise and optimize tracking re-localization in harsh situations.

Dataset

In this thesis, we create a novel dataset to evaluate the performance of our multiple object tracking algorithm in an RGB-D camera network. In order to automatically register the ground truth (objects' 3D center positions in world reference), we exploited a robot manipulator to move the objects and we calculated their ground-truth positions from the recorded robot kinematics.

Camera network configuration The camera network consists of 4 Microsoft Kinect V2 sensors. The extrinsic parameters of the sensors are calculated with the routine described in [88] to achieve a precise calibration. The RGB and depth images captured from these sensors are well aligned due to accurate intrinsic calibration exploiting the routines described in [90, 91]. Each RGB-D camera (*i.e.* MS Kinect) is connected to a single computer and all computers are precisely time-synchronized by NTP (Network Time Protocol). The whole experiment is conducted on the Robot Operating System (ROS[1]) middleware and the images are acquired with an open-source library named IAI Kinect2 [92].

[1] `http://www.ros.org/`

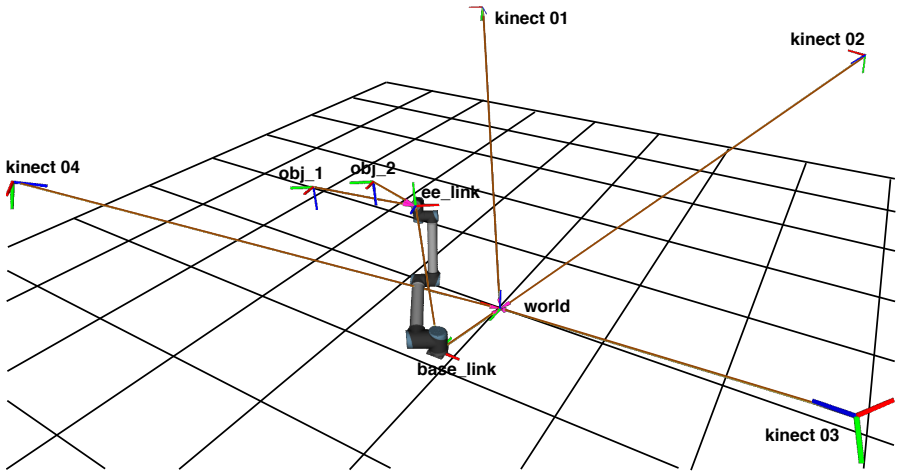


Figure 2.10.: Robot TF based ground truth annotation. There are two objects (obj_1, obj_2) mounted on the end effector *ee_link*. The 4 kinects and robot (*base_link*) are calibrated to the same *world* coordinate. The 3D ground-truth positions of each object with respect to the world reference can be calculated from the transform: $world \rightarrow base\_link \rightarrow ee\_link \rightarrow obj\_1, obj\_2$.

Robot TF based ground truth annotation. In this experiment, we mount the objects on an unbendable stick which is fixed to the end effector of the robot named UR 10. Firstly, we calibrated the robot with respect to the camera network with an automatic hand-eye calibration algorithms [93], in order to obtain the transform (TF) from world reference to the robot end-effector. Then, the distance from every object to the robot end-effector is measured to obtain the TF from the object to the robot end effector. During the experiment, we only need to record the motion of end effector with respect to the robot base reference and all the objects' motion can be easily calculated. Let T_{X-Y} and $\hat{T}_{X-Y}$ denote the dynamic and static transform from X to Y reference. The dynamic transform from world to object reference is calculated as

$$T_{W-O} = \hat{T}_{W-B} \cdot T_{B-E} \cdot \hat{T}_{E-O} \tag{2.23}$$

where W, O, B, E represent *world*, *object*, *robot_base* and *robot_end_effector* reference. $\hat{T}_{W-B}$ is obtained from "robot-camera network" calibration while $\hat{T}_{E-O}$ is measured manually. T_{B-E} is recorded during the whole sequence by subscribing to the robot state. The configuration of the cameras and of the UR10 robot used in the experiments is depicted in Fig. 2.10. The *base_link* represents the origin of the robot-base frame of reference, while the *ee_link* denotes the frame of reference of the end effector. There are two objects (obj_1, obj_2) mounted on the end effector. Both the cameras (kinect_01, 02, 03, 04) and the robot (*base_link*) are calibrated to the *world* coordinate.

Figure 2.11.: This figure depicts three frames of a sequence with very different illumination, namely: *n*atural light only, *a*ll lamps ON + natural light, *h*alf lamps ON + natural light. One can see strong shadows on the left image, strong highlights in the middle image, and uniform illumination on the right image.

Dataset organization The dataset is recorded into *Rosbags* consisting of color images, depth images, camera calibrations, and objects' 3D ground truth. All *Rosbags* have the frame rate of 30 FPS and each frame has a specific timestamp. There are 25 sequences (with more than 7.000 frames for each camera) with multiple objects moving in challenging paths.

Challenging factors in dataset The robot moved the objects on complex paths (one of them is shown in Fig. 2.12). A first complexity arises from the fact that due to perspective the objects could appear in the images as overlapping or even fully occluded by the others. A second complexity is due to the high speed of the motion of the objects (up to 2m/s). A third one is that in some sequences we used two indistinguishable objects.

Moreover, the dataset is recorded both during daytime with sunshine and in the evening with lighting from fluorescent lamps. As an additional complexity, in some sequences, the lamps were switched on and off one by one in order to generate abrupt and strong illumination variation.

Tracking evaluation

We evaluate our algorithm on 25 challenging sequences recorded in the daytime with sunlight, during the night with artificial illumination, and in the daytime with sunlight and artificial illumination. The lamps are turned on and off during the image acquisition in order to create abrupt illumination changes. Fig 2.11 depicts three frames of a sequence with very different illumination, namely: *n*atural light only, *a*ll lamps ON + natural light, *h*alf lamps ON + natural light. One can see strong shadows on the left image, strong highlights in the middle image, and uniform illumination on the right image.

An example of a **qualitative evaluation** of the performance of our tracking algorithm

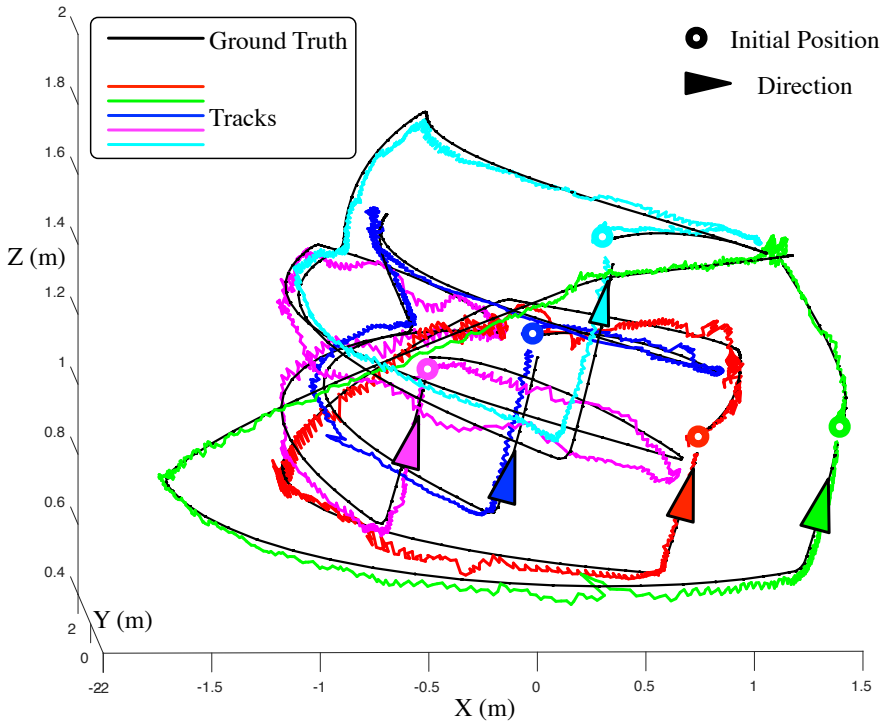


Figure 2.12.: Objects 3D center trajectories of proposed tracking algorithm and ground truth. Different colored trajectories represent 5 object tracks. In this sequence, the objects are moving under a fast and strong illumination variation. Some examples of frames are shown in Fig. 2.11 (Best viewed in color).

in one sequence with strong illumination variation is shown in Fig. 2.12. Fig. 2.12 the ground truth 3D trajectories of Center of mass (COM) of the five objects with a solid line. The 3D trajectories of the COMs estimated by our multi-camera tracking algorithm are depicted for every object in a different color. One can see, from Fig. 2.12, that the 3D estimated path and the ground truth are aligned well most of the time even when the objects are moving on a path with sudden changes of directions and that also during full occlusion even by objects with very similar appearance there are no ID switches. The 3D estimated path for each object shows a zig-zag pattern especially if the size of the object is large. This is because each camera doesn't know the 3D shape of the object it is tracking, thus it can estimate only the 3D position of the surface of the object that it can see. This surface has an unknown offset with respect to the COM of the object. Thus, to calculate the COM of the object, we calculate the COM of the 3D detection of the surfaces of the same object by the different cameras.

We also performs **q**uantitative evaluation of the tracking accuracy and robustness on our proposed dataset. The **Accuracy** is represented by the successful ratio which denotes the percentage of frames in which the objects are successfully tracked. This is done by evaluating the center error which is the distance between the target's predicted center from the tracker and ground truth, following the definitions in [94, 95]. It is worth to mention that the accuracy evaluation is only applied before all the trackers are drifted. This is because the center error could be very massively increased by a few drifted frames which will cause inaccurate evaluation.

Moreover, we apply a **Robustness** evaluation which measures the length of the period before the tracking drift. This measurement calculates the percentage of this period over the whole sequence in the timeline.

$$L_{track} = \sum_{i=1}^{I} \sum_{n=1}^{N_i} \hat{l}(i,n) \Bigg/ \sum_{i=1}^{I} \sum_{n=1}^{N_i} l(i) \tag{2.24}$$

where $\hat{l}(i,n)$ denotes the length of sequence i before all the distributed trackers of object n drifted. $l(i)$ represents the length of the whole sequence i. In sequence i, there are N_i objects to be tracked.

The tracking results of the proposed algorithm is not uniformly distributed in the timeline. If f_r denotes the frame rate (fps) while n_c the number of cameras, theoretically, there should be $(f_r \times n_c)$ tracking results per second for each object by associating all the target predicted centers from all the cameras. However, the distributed trackers may drift sometimes which can lower the number of tracking results. Therefore, we average center error of every constant time interval $(1/f_r)$ second to make the tracking results uniform distributed in the timeline. With the proposed process, there will be always constant numbers of tracking results in each second. This uniform process makes the successful ratio based accuracy evaluation more impartial and convincing.

The quantitative evaluation results are reported in Table 2.2. The accuracy is calculated under the threshold of 0.1m. **A / R** represents the result of **Accuracy / Robustness** (%/%). Our overall performance on the robustness is 92.89% that indicates we are able to keep tracking the object for such a long period and tracking drift seldom happens during the whole sequence. Among those 92.89% of the sequences, the tracking accuracy is 87.22%. This denotes that the 87.22% of those frames has the center error less than 0.1m. We separate the sequences into two parts by considering the illumination variations. The sequences which have heavy illumination variations is easier to have tracker failed problem and less accurate. In order to show the long-term tracking performance of our proposed algorithm: **Cam-Net (AS + RE)**, we also include the tracking results of single

Table 2.2.: Quantitative Evaluation of Tracking performance. **A / R** represents the result of **Accuracy / Robustness** (%/%). (IV: Illumination Variation, AS: Trackers Association, RE: Tracers Re-localization).

Sequence.	overall	with IV	without IV
Single Camera	65.41 / 61.95	64.54 / 57.85	63.93 / 69.57
Cam-Net (AS)[8]	80.25 / 84.16	79.70 / 80.79	81.20 / 90.16
Cam-Net (AS + RE)	**87.22 / 92.89**	**83.98 / 86.27**	**88.33 / 95.23**

camera based tracker: **Single-Camera** and camera network based tracking algorithm with solely trackers association: **Cam-Net (AS)** [8]. Note that in all those long-term tracking algorithms, we use the same base tracker (MS3D) proposed in this thesis. It is obvious to see that the tracking performance is gradually increased from up to bottom. Our proposed trackers fusion algorithm outperforms other baseline both in accuracy and robustness. With multiple camera from different view points, distributed trackers association and trackers re-localization, the object can be well tracked in a long term.

The proposed algorithm is tested on 4 desktop computers (Intel Core i7-7700 @3.60-GHz) based network with 4 kinect V2 which have the image resolution of 960×540. During the sequences without occlusion, our algorithm conducts computing on the search window rather than the whole image. Therefore, the time cost of the main tracking procedure is 1-2ms for single object tracking while it takes 7-8ms for tracking 5 objects. During occlusion, the algorithm runs slower in order to re-localize the target in an enlarged search window.

2.5. Conclusion

In this chapter, we presented an RGB-D object representation based on the fusion of the color and depth information in a probabilistic manner. With the color information, we proposed a discriminative color model based on foreground extraction and background weighting to enhance the discriminative capability of the object representation against background clutter.

With the depth information, we proposed a depth statistic model which is updated with predicted depth constraint interval and it works robustly with respect to fast depth variation. Then we applied the proposed representation in mean-shift object tracking algorithm to evaluate its performance. Moreover, an occlusion handling mechanism

based on the Bhattacharyya coefficient is designed to actively detect occlusion and to re-localize the object after short-term occlusion.

The evaluation result of the RGB-D tracker on the public dataset suggests that it out-performs other state-of-the-art RGB-D trackers in both efficiency and robustness. It is able to handle background clutter, shape deformation and partial occlusion in such challenging situations. The tracker can work efficiently both in static and moving cameras.

Moreover, we integrate the proposed tracker to an RGB-D camera network to evaluate its performance in long-term tracking. The proposed distributed trackers management with the functionalities of tracking association and feedback in RGB-D camera network can handle long-term and full occlusions. The distributed trackers association is able to continuously track the objects even if they are fully occluded in some of the cameras. The fault tracking results can be filtered in the camera network due to the Unscented Kalman Filter. The camera network feedback makes the drifted trackers be able to re-localize effectively and maintain their functionality. It is worth to mention that the proposed RGB-D camera network based tracker requires both accurate intrinsic and extrinsic camera calibrations. The evaluation results suggest the high potential of our proposed algorithm in real applications, *e.g.*, sports analysis, human-robot interaction. The dataset has been made public on the website.

3. Unsupervised Learning of Representation with Spatial Attention

Although the RGB-D data can offer both rich high-fidelity appearance information and geometric structure, there are still problems that are hard to address in some tasks. For instance, the illumination variation problem in object tracking, the ambiguity of the appearance in symmetric object 6D pose estimation and the inaccurate RGB and Depth registration, etc. Thus, there is a lot of research which applies solely geometric structure information to obtain the 3D object representation. Other than the depth image which can be used to represent the geometric structure, there are also meshes and point clouds, volumetric data. Thanks to their capability of representing a sparse 3D structure accurately while being agnostic to the sensing modality, point clouds have been a widespread choice for 3D processing.

3.1. Introduction

The proliferation of deep learning has recently leaped into the 3D domain and architectures for consuming 3D points have been proposed either for volumetric [96] or sparse [5] 3D representations. These architectures overcame many challenges brought in by 3D data, such as order-invariance, complexity due to the added data dimension and local density variations. Unfortunately, they often discard spatial arrangements in data, hence falling short of respecting the parts-to-whole relationship, which is critical to explain and describe 3D shapes; maybe even more severe than in the 2D domain due to the increased dimensionality [97].

In this chapter, we first present a unified look to some well-known 3D point decoders. Within this view, and based on the renowned 2D capsule networks (CN) [1], we propose the unsupervised *3D point-capsule networks* (3D-PointCapsNet), an autoencoder

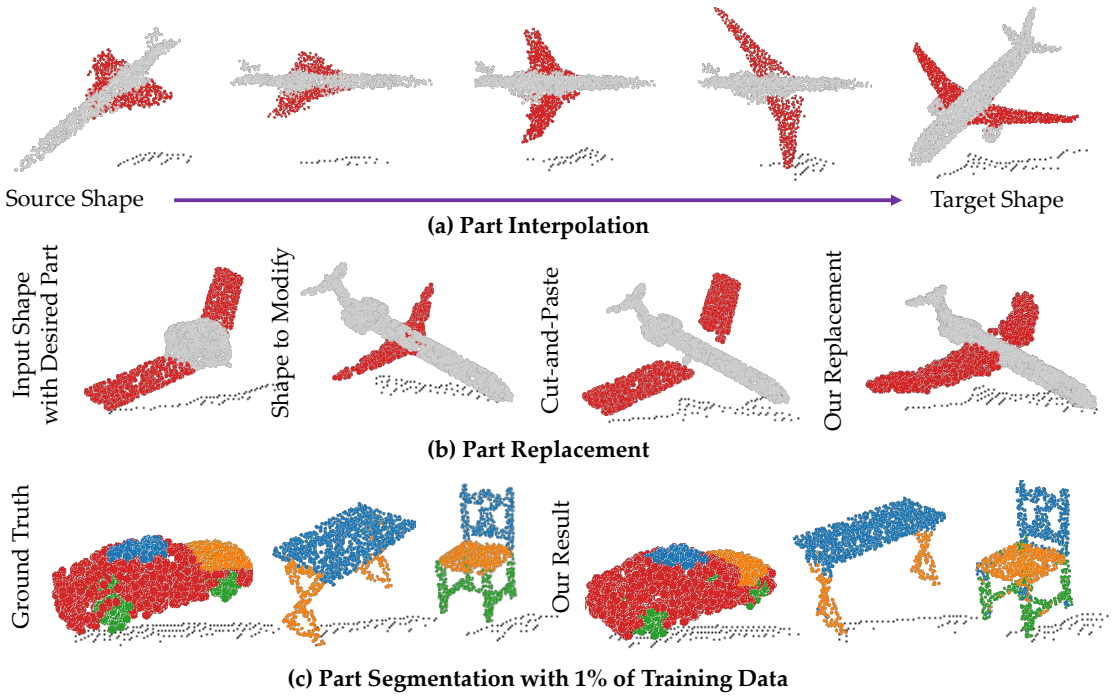


Figure 3.1.: Our *3D-PointCapsNet* improves numerous 3D tasks while enabling interesting applications such as latent space part interpolation or complete part modification, an application where a simple cut-and-paste results in inconsistent outputs.

for generic representation learning in unstructured 3D data. Powered by the built-in routing-by-agreement algorithm [1], our network respects the geometric relationships between the parts and learns the representation with spatial attention. We design our 3D-PointCapsNet architecture to take into account the sparsity of point clouds by employing PointNet-like input layers [5]. Through an unsupervised dynamic routing, we organize the outcome of multiple max-pooled feature maps into a powerful latent representation. This intermediary latent space is parameterized by *latent capsules* - stacked latent activation vectors specifying the features of the shapes and their likelihood.

Latent capsules obtained from point clouds alleviate the restriction of parameterizing the latent space by a single, low dimensional vector; instead, they give explicit control on the basis functions that get composed into 3D shapes. We further propose a novel 3D point-set decoder operating on these capsules, leading to better reconstructions with increased operational capabilities as shown in Fig. 3.1. These new abilities stem from the latent capsules instantiating as various shape parameters and concentrating not spatially but semantically across the shape under consideration, even when trained in

an unsupervised fashion. We also propose to supply a limited amount of task-specific supervision such that the individual capsules can excel at solving individual sub-problems, *e.g.*, if the task is part-based segmentation, they specialize in different meaningful parts of each shape.

Our extensive quantitative and qualitative evaluation demonstrates the superiority of our architecture. First, we advance the state of the art by a significant margin on multiple frontiers such as 3D local feature extraction, point cloud reconstruction, and transfer learning. Next, we show that the distinct attention mechanism of the capsules, driven by dynamic routing, allows a wider range of 3D applications compared to the state of the art autoencoders: a) part replacement, b) part-by-part animation via interpolation. Note that both of these tasks are non-trivial for standard architectures that rely on 1D latent vectors. Finally, we present improved generalization to unseen data, reaching accuracy levels up to 85% even when using 1% of training data.

Our source code is publicly available under: `https://tinyurl.com/yxq2tmv3`.

3.2. Encode 3D shapes with spatial attention

3.2.1. Formulation

We first follow the AtlasNet convention [33] and present a unified view of some of the common 3D autoencoders. Then, we explain our *3D-PointCapsNet* within this geometric perspective and justify its superiority compared to its ancestors. We will start by recalling the basic concepts:

Definition 1 (Surface and Point Cloud)
A 3D surface (shape) is a differentiable 2-manifold embedded in the ambient 3D Euclidean space: $\mathcal{M}^2 \in \mathbb{R}^3$. *We approximate a* ***point cloud*** *as a sampled discrete subset of the surface* $\mathbf{X} = \{\mathbf{x}_i \in \mathcal{M}^2 \cap \mathbb{R}^3\}$.

Definition 2 (Diffeomorphism)
A diffeomorphism is a continuous, invertible, structure-preserving map between two differentiable surfaces.

Definition 3 (Chart and Parametrization)
We admit an open set $U \in \mathbb{R}^2$ *and a diffeomorphism* $C : \mathcal{M}^2 \mapsto U \in \mathbb{R}^2$ *mapping an open neighborhood in 3D to its 2D embedding.* C *is called a* ***chart****. Its inverse,* $\Psi \equiv C^{-1} : \mathbb{R}^2 \mapsto \mathcal{M}^2$ *is called a* ***parameterization****.*

Definition 4 (Atlas)
*A set of charts with images covering the 2-manifold is called an **atlas**:* $\mathcal{A} = \cup_i C_i(\mathbf{x}_i)$.

A 3D autoencoder learns to generate a 3D surface $\mathbf{X} \in \mathcal{M}^2 \cap \mathbb{R}^{N\times 3}$. By virtue of Dfn. 3 Ψ deforms a 2D point set to a surface. The goal of the generative models that are of interest here is to learn Ψ to best reconstruct $\hat{\mathbf{X}} \approx \mathbf{X}$:

Definition 5 (Problem)
Learning to generate the 2-manifolds is defined as finding function(s) $\Psi(U \mid \boldsymbol{\theta}) : \Psi(U \mid \boldsymbol{\theta}) \approx \mathbf{X}$ *[33].* $\boldsymbol{\theta}$ *is a lower dimensional parameterization of these functions:* $|\boldsymbol{\theta}| < |\mathbf{X}|$.

Theorem 1
Given that C^{-1} *exists,* Ψ*, chosen to be a 3-layer MLP, can reconstruct arbitrary 3D surfaces.*

Proof. The proof is given in [30] and follows from the universal approximation theorem (UAT). □

Theorem 2
There exists an integer K s.t. an MLP with K hidden units universally reconstruct $\mathbf{X}$ *up to a precision* ε.

Proof. The proof follows trivially from Thm. 1 and UAT [33]. □

Given these definitions, some of the typical 3D point decoders differentiate by making four choices [5, 33, 30]:

1. An open set U or discrete grid $\mathbf{U} \equiv \mathbf{P} = \{\mathbf{p}_i \in \mathbb{R}^2\}$.
2. Distance function $d(\mathbf{X}, \hat{\mathbf{X}})$ between the reconstruction $\hat{\mathbf{X}}$ and the input shape $\mathbf{X}$.
3. Parameterization function(s) Ψ.
4. Parameters $(\boldsymbol{\theta})$ of Ψ: $\Psi(U \mid \boldsymbol{\theta})$.

One of the first works in this field, PointNet [5] is extended naturally to an AE by [35] making arguably the simplest choice. We will refer to this variant as *PointNet*. It lacks the grid structure $U = \emptyset$ and functions Ψ only depend upon a single latent feature: $\Psi(U \mid \boldsymbol{\theta}) = \Psi(\boldsymbol{\theta}) = \text{MLP}(\cdot \mid \mathbf{f} \in \mathbb{R}^k)$. FoldingNet uses a two-stage MLP as Ψ to warp a fixed grid $\mathbf{P}$ onto $\mathbf{X}$. A transition from FoldingNet to AtlasNet requires having multiple MLP networks operating on multiple 2D sets $\{\mathbf{P}_i\}$ constructed randomly on the domain $]0,1[^2$: $\mathcal{U}(0,1)$. These explain the better learning capacity of AtlasNet: different MLPs learn to reconstruct distinct local surface patches by learning different charts.

Unfortunately, while numerous charts can be defined in the case of AtlasNet, all of the methods above still rely on a single latent feature vector, replicated and concatenated with U to create the input to the decoders. However, point clouds are found to consist of multiple basis functions [98] and having a single representation governing them all is not optimal. We opt to go beyond this restriction and choose to have a set of latent features $\{\mathbf{f}_i\}$ to capture different, meaningful basis functions.

With the aforementioned observations we can now re-write the well known 3D autoencoders and introduce a new decoder formulation:

PointNet [5]	AtlasNet [33]	
$\mathbf{U} = \mathbf{P} = \emptyset$	$\mathbf{U} = \{\mathbf{P}_i\} : \mathbf{P}_i \in \mathcal{U}(0,1)$	(3.1)
$\Psi(\boldsymbol{\theta}) := \text{MLP}(\cdot)$	$\Psi(\boldsymbol{\theta}) := \{\text{MLP}_i(\cdot)\}$	(3.2)
$\boldsymbol{\theta} := \mathbf{f}$	$\boldsymbol{\theta} := \{\mathbf{f}, \{\mathbf{P}_i\}\}$	(3.3)
$d(\mathbf{X}, \hat{\mathbf{X}}) := d_{\text{EMD}}(\mathbf{X}, \hat{\mathbf{X}})$	$d(\mathbf{X}, \hat{\mathbf{X}}) := d_{\text{CH}}(\mathbf{X}, \hat{\mathbf{X}})$	(3.4)
FoldingNet [30]	Ours	
$\mathbf{U} = \mathbf{P} = \mathbf{G}^{M \times M}$	$\mathbf{U} = \{\mathbf{P}_i\} : \mathbf{P}_i \in \mathcal{U}(0,1)$	(3.5)
$\Psi(\boldsymbol{\theta}) :=\text{MLP}(\text{MLP}(\cdot))$	$\Psi(\boldsymbol{\theta}) := \{\text{MLP}_i(\cdot)\}$	(3.6)
$\boldsymbol{\theta} := \{\mathbf{f}, \mathbf{P}\}$	$\boldsymbol{\theta} := \{\mathbf{F} \triangleq \{\mathbf{f}_i\}, \{\mathbf{P}_i\}\}$	(3.7)
$d(\mathbf{X}, \hat{\mathbf{X}}) := d_{\text{CH}}(\mathbf{X}, \hat{\mathbf{X}})$	$d(\mathbf{X}, \hat{\mathbf{X}}) := d_{\text{CH}}(\mathbf{X}, \hat{\mathbf{X}})$	(3.8)

where d_{EMD} is the Earth Mover [99] and d_{CH} is the Chamfer distance. $\mathbf{G}^{M \times M} = \{(i \otimes j) : \forall i, j \in [0, \ldots, \frac{M-1}{M}]\}$ is a 2D uniform grid. $\mathbf{f} \in \mathbb{R}^k$ represents a k-dimensional latent vector. $\mathcal{U}(a,b)$ depicts an open set defined by a uniform random distribution in the interval $]a,b[$[2].

Note that it is possible to easily mix these choices to create variations[4]. Though, many interesting architectures only optimize for a single latent feature $\mathbf{f}$. To the best of our knowledge, one promising direction is taken by the capsule networks [60], where multitudes of convolutional filters enable the learning of a collection of *capsules* $\{\mathbf{f}_i\}$ thanks to the dynamic routing [1]. Hence, we learn our parameters $\{\boldsymbol{\theta}_i\}$ by devising a new point cloud *capsule decoder* that we coin *3D-PointCapsNet*. We illustrate the choices made by four AEs under this unifying umbrella in Fig. 3.2.

[4]FoldingNet presents evaluations with random grids in their appendix.

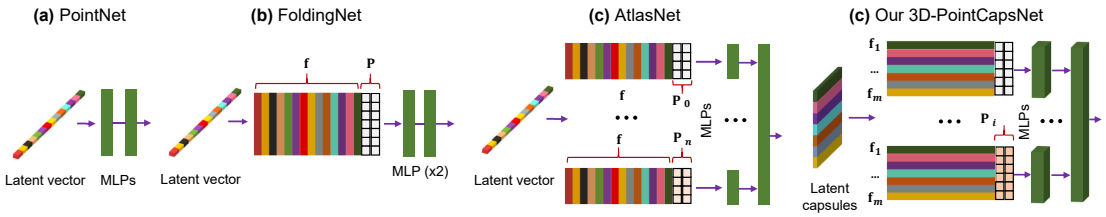


Figure 3.2.: Comparison of four different state-of-the-art 3D point decoders. PointNet uses a single latent vector, and no surface assumption. Thus, $\boldsymbol{\theta}_{\text{pointnet}} = \mathbf{f}$. FoldingNet [30] learns a 1D latent vector along with a fixed 2D grid $\boldsymbol{\theta}_{\text{folding}} = \{\mathbf{f}, \mathbf{P}\}$. The advanced AtlasNet [33] learns to deform multiple 2D configurations onto local 2-manifolds: $\boldsymbol{\theta}_{\text{atlas}} = \{\mathbf{f}, \{\mathbf{P}_i\}\}$. Our point-capsule-network is capable of learning multiple latent representations each of which can fold a distinct 2D grid onto a specific local patch, $\boldsymbol{\theta}_{\text{ours}} = \{\{\mathbf{f}_i\}, \{\mathbf{P}_i\}\}$

3.2.2. 3D-PointCapsNet Architecture

We now describe the architecture of the proposed *3D-PointCapsNet* as a deep 3D point cloud autoencoder, whose structure is depicted in Fig. 3.3.

Encoder The Input to our network is an $N \times d$ point cloud, where we fix $N = 2048$ and for typical point sets $d = 3$. Similarly to PointNet [5], we use a point-wise Multi-Layer Perceptron (MLP) $(3-64-128-1024)$ to extract individual local feature maps. In order to diversify the learning as suggested by capsule networks, we feed these feature maps into multiple independent convolutional layers with different weights, each with a distinct summary of the input shape with diversified attention. We then max-pool their responses to obtain a global latent representation. These descriptors are then concatenated into a set of vectors named *primary point capsules*, $\mathbf{F}$. Size of $\mathbf{F}$ depends upon the size $S_c := 1024$ and the number $K := 16$ of independent kernels at the last layer of MLP. We then use the dynamic routing [1] to embed the primary point capsules into higher-level *latent capsules*. Each capsule is independent and can be considered as a *cluster centroid* (codeword) of the primary point capsules. The total size of the latent capsules is fixed to 64×64 (*i.e.*, 64 vectors each sized 64).

Decoder Our decoder treats the latent capsules as a feature map and uses MLP$(64-64-32-16-3)$ to reconstruct a patch of points $\hat{\mathbf{X}}_i$, where $|\hat{\mathbf{X}}_i| = 64$. At this point, instead of replicating a single vector as done in [30, 33], we replicate the entire capsule m times and to each replica, we append a unique randomly synthesized grid $\mathbf{P}_i$ specializing it to a local area. This further stimulates diversity. We arrive at the final shape $\hat{\mathbf{X}}_i$ by

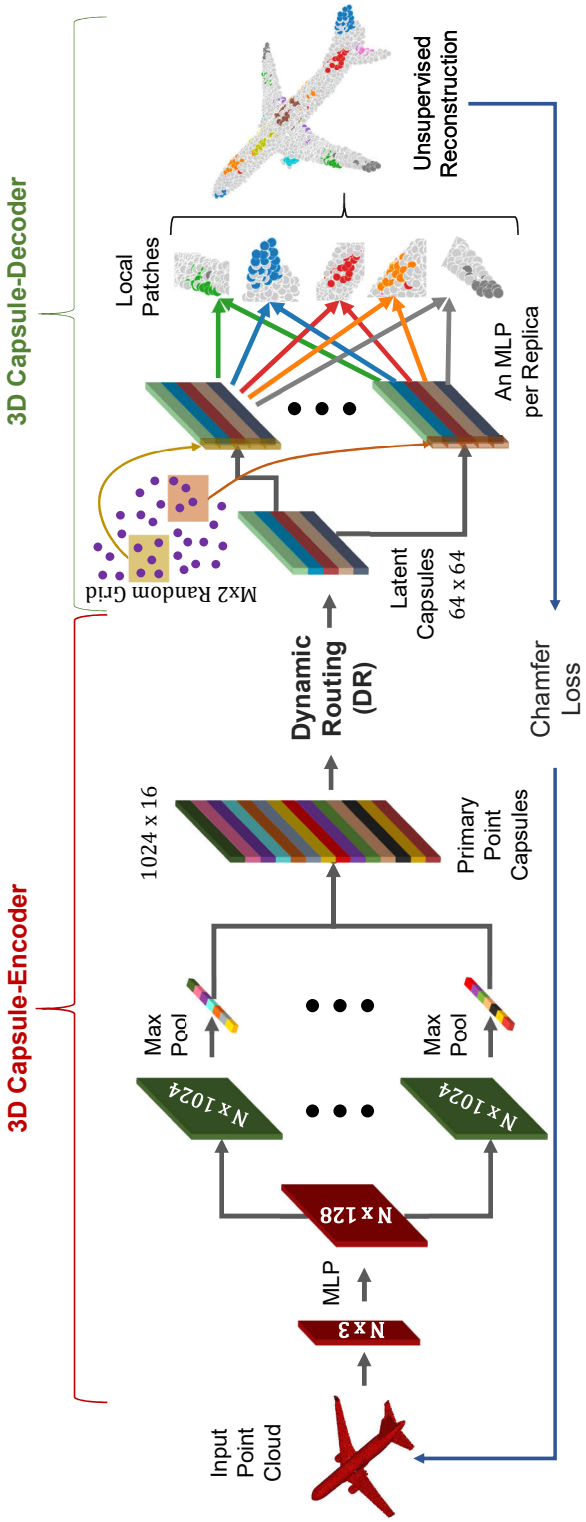


Figure 3.3.: 3D Point Capsule Networks. Our capsule-encoder accepts a $N \times 3$ point cloud as input and uses an MLP to extract $N \times 128$ features from it. These features are then sent into multiple independent convolutional-layers with different weights, each of which is max-pooled to a size of 1024. The pooled features are then concatenated to form the *primary point capsules* (PPC) (1024×16). A subsequent dynamic routing clusters the PPC into the final *latent capsules*. Our decoder, responsible for reconstructing point sets given the latent features, endows the latent capsules with random 2D grids and applies MLPs ($64-64-32-16-3$) to generate multiple point patches. These point patches target different regions of the shape thanks to the DR [1]. Finally, we collect all the patches into a final point cloud and measure the Chamfer distance to the input to guide the network to find the optimal reconstruction. In figure, part-colors encode capsules.

propagating the replicas through a final MLP for each patch and gluing the output patches together. We choose $m = 32$ to reconstruct $|\hat{\mathbf{X}}| = 32 \times 64 = 2048$ points, the same amount as the input. Similar to other AEs, we approximate the loss over 2-manifolds by the discrete Chamfer metric:

$$d_{CH}(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} \min_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2 + \frac{1}{|\hat{\mathbf{X}}|} \sum_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}} \min_{\mathbf{x} \in \mathbf{X}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2 \tag{3.9}$$

However, this time $\hat{\mathbf{X}}$ follows from the capsules: $\hat{\mathbf{X}} = \cup_i \Psi_i(\mathbf{P}_i | \{\mathbf{f}_i\})$.

Incorporating Optional Supervision Motivated by the regularity of capsule distribution over the 2-manifold, we created a capsule-part network that spatially segments the object by associating capsules to parts. The goal here is to assign each capsule to a single part of the object. Hence, we treat this part-segmentation task as a per-capsule classification problem, rather than a per-point one as done in various preceding algorithms [5, 100]. This is only possible due to the spatial attention of the capsule networks.

The input of capsule-part network is the latent-capsules obtained from the pre-trained encoder. The output is the part label for each capsule. The ground truth (GT) capsule labeling is obtained from the ShapeNet-Part dataset [101] in three steps: 1) reconstructing the local part given the capsule and a pre-trained decoder, 2) retrieving the label of the nearest neighbor (NN) GT point for each reconstructed point, 3) computing the most frequent one (mode) among the retrieved labels.

To associate a part to a capsule, we use a shared MLP with a cross-entropy loss to classify the latent capsules into parts. This network is trained independently from the 3D-PointCapsNet AE for part supervision. We provide additional architectural details in (Appendix A).

3.3. Evaluation

We evaluate our method first quantitatively and then qualitatively on numerous challenging 3D tasks such as local feature extraction, point cloud classification, reconstruction, part segmentation, and shape interpolation. We also include a more specific application of *latent space part-interpolation* that is made possible by the use of capsules. For evaluation regarding these tasks, we use multiple benchmark datasets: ShapeNet-Core [102], Shapenet-Part [101], ModelNet40 [103] and 3DMatch benchmark [104].

Table 3.1.: Descriptor matching results (recall) on the standard 3DMatch benchmark [104, 31].

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
3DMatch [104]	0.5751	0.7372	0.7067	0.5708	0.4423	0.6296	0.5616	0.5455	0.5961
CGF [105]	0.4605	0.6154	0.5625	0.4469	0.3846	0.5926	0.4075	0.3506	0.4776
PPFNet [32]	0.8972	0.5577	0.5913	0.5796	0.5769	0.6111	0.5342	0.6364	0.6231
FoldNet [30]	0.5949	0.7179	0.6058	0.6549	0.4231	0.6111	0.7123	0.5844	0.6130
PPF-FoldNet-2K [31]	0.7352	0.7564	0.625	0.6593	0.6058	0.8889	0.5753	0.5974	0.6804
PPF-FoldNet-5K [31]	0.7866	0.7628	0.6154	0.6814	0.7115	**0.9444**	0.6199	0.6234	0.7182
Ours-2K	**0.8518**	**0.8333**	**0.7740**	**0.7699**	**0.7308**	**0.9444**	**0.7397**	**0.6494**	**0.7867**

Implementation Details Prior to training, the input point clouds are aligned to a common reference frame and size normalized. To train our network we use an ADAM optimizer with an initial learning rate of 0.0001 and a batch size of 8. We also employ batch normalization (BN) and RELU activation units at the point of feature extraction to generate primary capsules. Similarly, the multi-stage MLP of the decoder also uses a BN and RELU units except for the last layer, where the activations are scaled by a $tanh(\cdot)$. During dynamic routing operation, we use the squash activation function mentioned in [1, 60].

3.3.1. Quantitative Evaluations

3D Local Feature Extraction We first evaluate 3D Point-Capsule Networks on the challenging task of local feature extraction from point cloud data. In this domain, learning methods have already outperformed their handcrafted counterparts by a large margin and hence, we compare only against those, namely 3DMatch [104], PPFNet [32], CGF [105] and PPF-FoldNet [31]. PPF-FoldNet is completely unsupervised and yet is still the top performer, thanks to the FoldingNet [30] encoder-decoder. It is thus intriguing to see how its performance is affected if one simply replaces its FoldingNet autoencoder with 3D-PointCapsNet. In an identical setting as [31], we learn to reconstruct the 4-dimensional point pair features [106, 107] of a local patch, instead of the 3D space of points, and use the latent capsule (codeword) as a 3D descriptor. To restrict the feature vector to a reasonable size of 512, we limit ourselves only to 16×32 capsules. We then run the matching evaluation on the 3DMatch Benchmark dataset [104] as detailed in [31], and report the recall of correctly founded matches after 21 epochs in Tab. 3.1.

We note that our point-capsule networks exhibit an advanced capacity for learning local features, surpassing the state of the art by 10% on the average, even when using $2K$ points unlike the $5K$ of PPF-FoldNet. It is also noteworthy that, except for the

Table 3.2.: Descriptor matching results (recall) on the rotated 3DMatch benchmark [104, 31].

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
3DMatch [104]	0.0040	0.0128	0.0337	0.0044	0.0000	0.0096	0.0000	0.0260	0.0113
CGF [105]	0.4466	0.6667	0.5288	0.4425	0.4423	0.6296	0.4178	0.4156	0.4987
PPFNet [32]	0.0020	0.0000	0.0144	0.0044	0.0000	0.0000	0.0000	0.0000	0.0026
FoldNet [30]	0.0178	0.0321	0.0337	0.0133	0.0096	0.0370	0.0171	0.0260	0.0233
PPF-FoldNet-2K [31]	0.7352	0.7692	0.6202	0.6637	0.6058	0.9259	0.5616	0.6104	0.6865
PPF-FoldNet-5K [31]	0.7885	0.7821	0.6442	0.6770	0.6923	**0.9630**	0.6267	0.6753	0.7311
Ours-2K	**0.8498**	**0.8525**	**0.7692**	**0.8141**	**0.7596**	0.9259	**0.7602**	**0.7272**	**0.8074**

Kitchen sequence where PPFNet shows remarkable performance, the recall attained by our network consistently remains above all others. We believe that such dramatic improvement is related to the robustness of capsules towards slight deformations in the input data, as well as to our effective decoder.

Do Our Features Also Perform Well Under Rotation? PPF local encoding of PPF-FoldNet is rotation-invariant. Being based on the same representation, our local feature network should enjoy similar properties. It is of interest to see whether the good performance attained on the standard 3DMatch benchmark transfers to more challenging scenes demanding rotation invariance. To this aim, we repeat the previous assessment on the Rotated-3DMatch benchmark [31], a dataset that introduces arbitrary rotations to the scenes of [104]. Since this dataset contains 6DoF scene transformations, many methods that lack theoretical invariance, *e.g.*, 3DMatch, PPFNet, and FoldingNet simply fail. Our unsupervised capsule AE, however, is once again the top performer, surpassing the state of the art by $\sim 12\%$ on $2K$-point case as shown in Tab. 3.2. This significant gain justifies that our encoder manages to operate also on the space of 4D PPFs, holding on the theoretical invariances.

3D Reconstruction In a further experiment, we evaluate the quality of our architecture in point generation. We assess the reconstruction performance by the standard Chamfer metric and base our comparisons on the state of the art autoencoder AtlasNet and its baselines (point-MLP) [33]. We rely on the ShapeNet Core v2 dataset [102], using the same training and test splits as well as the same evaluation metric as those in AtlasNet's [33]. We show in Tab. 3.3 the Chamfer distances averaged overall categories and for $N > 2K$ points. It is observed that our capsule AE results in lower reconstruction error even when a large number of patches (125) is used in favor of AtlasNet. This

Table 3.3.: Evaluating reconstruction quality. Oracle refers to a random sampling of the input 3D shape and constitutes an lower bound on what is achievable. The Chamfer Distance is multiplied by 10^3 for better viewing. CD denotes *Chamfer distance* and PB refers to *Point Baseline*.

	Oracle	PB	AtlasNet-25	AtlasNet-125	Ours
CD	0.85	1.91	1.56	1.51	**1.46**

justifies that the proposed network has a better summarization capability and can result in higher fidelity reconstructions.

Transfer Learning for 3D Object Classification In this section, we demonstrate the efficiency of learned representation by evaluating the classification accuracy obtained by performing transfer learning. Identical to [108, 35, 30], we train a linear SVM classifier so as to regress the shape class given the latent features. To do that, we reshape our latent capsules into a one-dimensional feature and train the classifier on Modelnet40 [103]. We use the same train/test split sets as [30] and obtain the latent capsules by training 3D-PointCapsNet on a different dataset, the ShapeNet-Parts [101]. The training data has 14,000 models subdivided into 16 classes. The evaluation result is shown in Tab. 3.4, where our AE, trained on a smaller dataset compared to the ShapeNet55 of [35, 30] is capable of performing at least on par or better. This shows that learned latent capsules can handle smaller datasets and generalize better to new tasks. We also evaluated our classification performance when the training data is scarce and obtained the similar result as the FoldingNet, $\sim 85\%$ on $\sim 20\%$ of training data.

Table 3.4.: Accuracy of classification by transfer learning on the ModelNet40 dataset. Networks are trained out ShapeNet55, except *Ours-Parts* that is trained on smaller ShapeNet-Parts dataset.

	Latent-GAN[35]	FoldingNet[30]	Ours-Parts	Ours
Acc.	85.7	88.4	88.9	**89.3**

3.3.2. Qualitative Results

3D Object Part Segmentation with Limited Data We now demonstrate the regional attention of our latent capsule and their capacity to learn with limited data. To this end, we trained 3D-PointCapsNet on the ShapeNet-Part dataset [102] for part segmentation as explained in Sec. 3.2, with the supervision of only $1-5\%$ part labeled training data. We

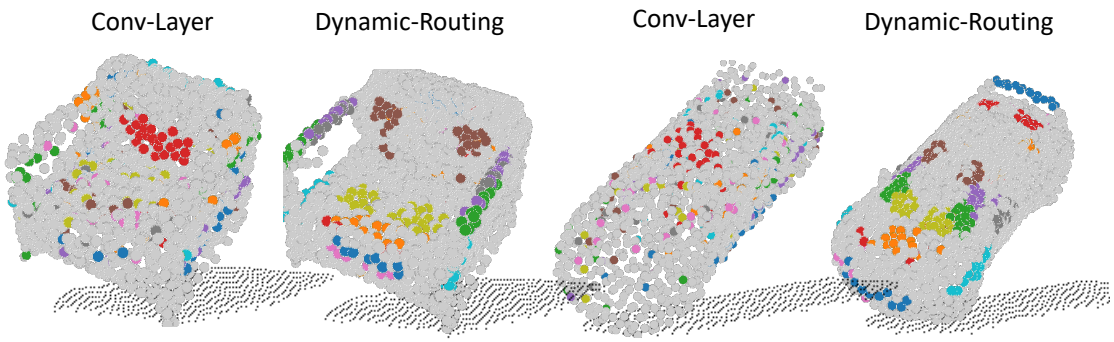


Figure 3.4.: Distribution of 10 randomly selected capsules on the reconstructed shape after unsupervised autoencoder training **a**) with dynamic routing, **b**) with a simple convolutional layer.

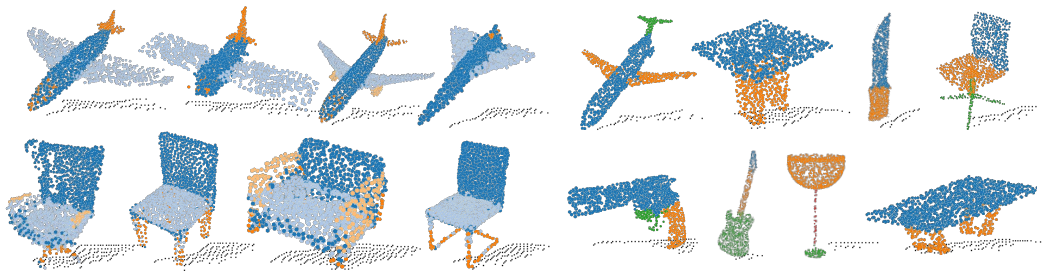
tested our network on all of the available test data. To specialize the capsules to distinct parts, we select as many capsules as the part labels and let the per-capsule classification coincide to part predictions. Predicted capsule labels are propagated to the related points. For the sake of space, we compared our results only with the state of the art on this dataset, the SO-Net [68]. We use identical evaluation metrics as SO-Net [68]: *Accuracy* and *IoU* (Intersection over Union), and report our findings in Tab. 3.5. Note that, when trained on 1% of input data, we perform 7% better than SO-Net. When the amount of training data is increased to 5%, the gap closes but we still surpass SO-Net by 2%, albeit training a smaller network to classify latent-capsules rather than 3D points.

Table 3.5.: Part segmentation on ShapeNet-Part by learning only on the $x\%$ of the training data.

Metric	SONet-1%	Ours-1%	SONet-5%	Ours-5%
Accuracy	0.78	**0.85**	0.84	**0.86**
IoU	0.64	**0.67**	0.69	**0.70**

Does unsupervised training lead to specialized capsules? It is of interest to see whether the original argument of the capsule networks [1, 60] claiming to better capture the intrinsic geometric properties of the object still holds in the case of our unsupervised 3D-AE. To this aim, we first show in Fig. 3.4 that even with lack of supervision the capsules specialize in local parts of the model. While these parts may sometimes not correspond to the human-annotated part segmentation of the model, we still expect them to concentrate on semantically similar regions of the 2-manifold. Fig. 3.4 visualizes the distribution of 10 capsules by coloring them individually and validates our argument.

To validate our second hypothesis, stating that such clustering arises thanks to the



(a) Unprocessed part segmentation on same class

(b) Part segmentation of multiple objects of different class

Figure 3.5.: Part segmentation by capsule association. Having pre-trained the autoencoder, we append a final part-supervision layer and use a limited amount of data to specialize the capsules on object parts. (**a**) across the shapes of the same class capsules capture semantic regions. (**b**) inter-class part segmentation. Colors indicate different capsule groups and (**b**) uses only a simple median filter to smooth the results.

dynamic routing, we replace the DR part of the AE with standard PointNet-like layers projecting the 1024×64 PPC to 64^2 capsules and repeat the experiment. Fig. 3.4 depicts the spread of the latent vectors over the point set when such layer is employed as opposed to DR. Note that using this simple layer instead of DR both harms the reconstruction quality and yields an undesired spread of the capsules across the shape. We leave it as future work to study the DR energy theoretically and provide more details on this experiment in Appendix A.

Semi-supervision guides the capsules to meaningful parts. We now consider the effect of training in steering the capsules towards the optimal solution in the task of supervised part segmentation. First, we show in Fig. 3.5 the results obtained by the proposed part segmentation: (**a**) shows part segmentation across multiple shapes of the same class. These results are also unfiltered and the raw outcome of our network. (**b**) depicts part segmentation across a set of object classes from Shapenet-Part. It also shows that some minor confusions present in (**a**) can be corrected with a simple median filter. This is contrary and computationally preferable to costly CRFs smoothing the results [109].

Next, we observe that, as training iterations progress, the randomly initialized capsules specialize to parts, achieving a good part segmentation at the point of convergence. We visualize this phenomenon in Fig. 3.6, where the capsules that have captured the wings of the airplane are monitored throughout the optimization procedure. Even though the initial random distribution is spatially spread out, the resulting configuration is still part specific. This is a natural consequence of our capsule-wise part semi-supervision.

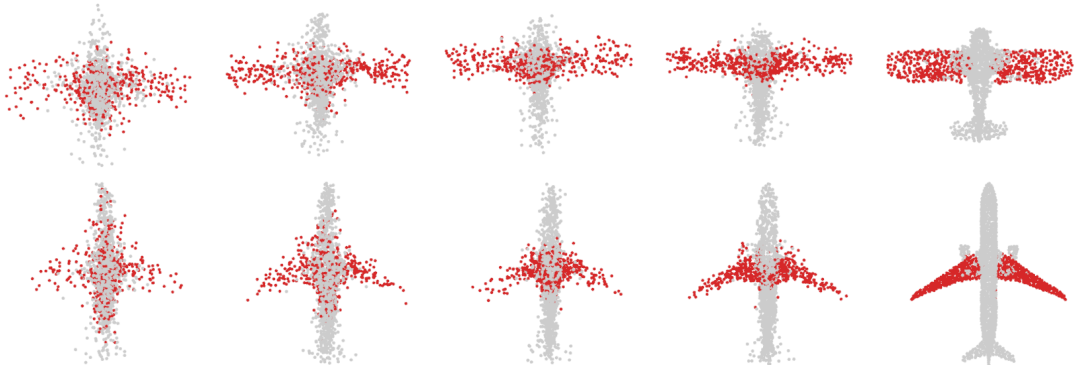


Figure 3.6.: Visualizing the iterations of unsupervised AE training on the airplane object. For clear visualization, we fetch the colors belonging to the ∼20 capsules of the wing-part from our part predictions trained with part supervision.

Part Interpolation / Replacement Finally, we explore the rather uncommon but particularly interesting application of interpolating, exchanging or switching object parts via latent-space manipulation. Thanks to the fact that 3D-PointCapsNet discovers multiple latent vectors specific to object attributes/shape parts, our network is capable of performing per-part processing in the latent space. To do that, we first spot a set of latent capsule pairs belonging to the same parts of two 3D point shapes and intersect them. Because these capsules explain the same part in multiple shapes, we assume that they are specific to the part under consideration and nothing else. We then interpolate linearly in the latent space between the selected capsules. As shown in Fig. 3.7 the reconstruction of intermediate shapes vary only at a single part, the one being interpolated. When the interpolator reaches the target shape it *replaces* the source part with the target one, enabling *part-replacement*. Fig. 3.8 further shows this in action. Given two shapes and latent capsules of the related parts, we perform a part exchange by simply switching some of the latent capsules and reconstructing. Conducting a part exchange directly on the input space by a cut-and-place would yield inconsistent shapes as the replaced parts would have no global coherence.

3.4. Conclusion

We have presented 3D Point-Capsule Network, a flexible and effective tool for unsupervised learning representations with spatial attention for 3D shape processing and understanding.

We first presented a broad look to the common point cloud autoencoders. With the

observation that a one-dimensional latent embedding, the choice of the most preceding autoencoders, is potentially sub-optimal, we opted to summarize the point clouds as a union of disjoint latent basis functions. We have shown that such a choice can be implemented by learning the embedded *latent capsules* via dynamic routing. Our algorithm proved successful on an extensive evaluation on many 3D shape processing tasks such as 3D reconstruction, local feature extraction, and part segmentation. Having a latent capsule set rather than a single vector also enabled us to address new applications such as part interpolation and replacement. In the future, we plan to deploy our network for pose estimation and object detection from 3D data, currently two of the key challenges in 3D computer vision.

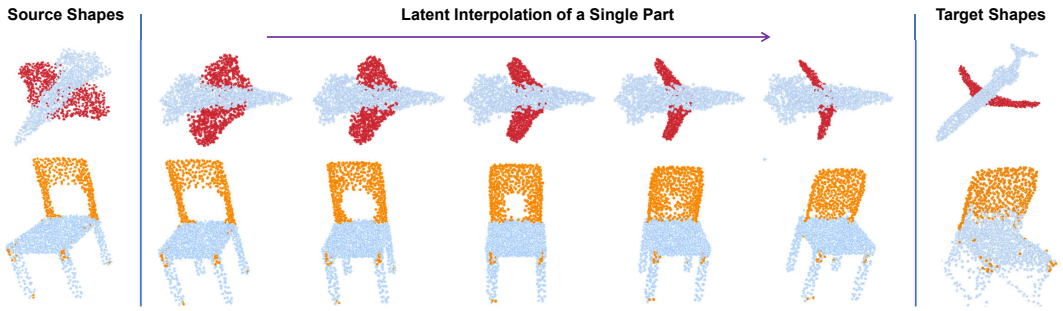


Figure 3.7.: Part interpolation on the Shapenet-Part [101] dataset. (**left**) The source point cloud. (**right**) Target shape. (**middle**) Part interpolation. Fixed part is marked in light blue and the interpolated part is highlighted. Capsules are capable of performing part interpolation purely via latent space arithmetic.

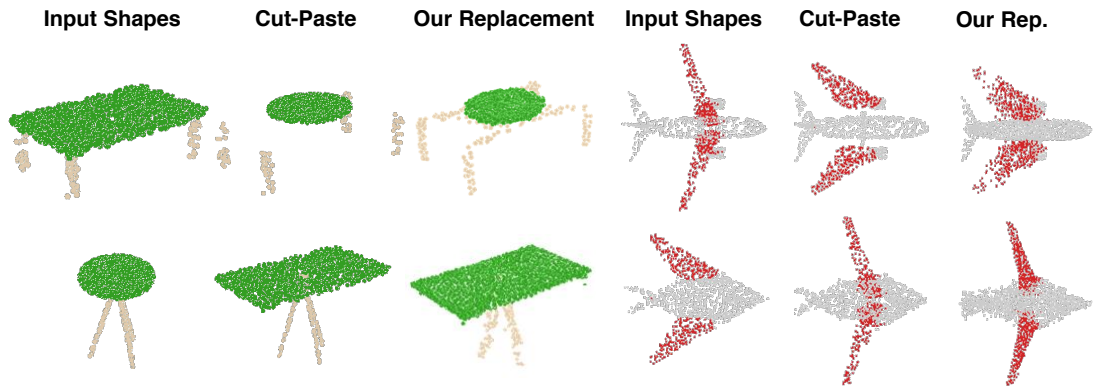


Figure 3.8.: Part replacement. Performing replacement in the latent space rather than Euclidean space of 3D points yields geometrically consistent outcome.

4. Learning Features with SO(3) Rotation Invariance and Equivariance

4.1. Introduction

It is now well understood that in order to learn a compact representation of the input data, one needs to respect the symmetries in the problem domain [110, 77]. Arguably, one of the primary reasons of the success of 2D convolutional neural networks (CNN) is the *translation-invariance* of the 2D convolution acting on the image grid [111, 112]. Recent trends aim to transfer this success into the 3D domain in order to support many applications such as shape retrieval, shape manipulation, pose estimation, 3D object modeling and detection. There, the data is naturally represented as sets of 3D points or a *point cloud* [5, 6]. Unfortunately, extension of CNN architectures to 3D point clouds is non-trivial due to two reasons: 1) point clouds are irregular and unstructured, 2) the group of transformations that we are interested in is more complex as 3D data is often observed under arbitrary non-commutative $SO(3)$ rotations. As a result, achieving appropriate embeddings requires 3D networks that work on points to be *equivariant* to these transformations, while also being invariant to the permutations of the point set.

In order to fill this important gap, we propose the quaternion equivariant point capsule network or *QE-Network* that is suited to process point clouds and is equivariant to $SO(3)$ rotations compactly parameterized by quaternions, in addition to preserved translation and permutation equivariance. Inspired by the local group equivariance [76, 110], we efficiently cover $SO(3)$ by restricting ourselves to the sparse set of local reference frames (LRF) that collectively characterize the object orientation. The proposed capsule layers [60] deduces equivariant latent representations by robustly combining those local LRFs using the proposed *Weiszfeld dynamic routing*. Hence, our latent features specify to local orientations disentangling the pose from object existence. Such explicit storage is unique to our work and allows us to perform rotation estimation jointly with

object classification. Our final architecture is a hierarchy of QE-networks, where we use classification error as the only training cue and adapt a Siamese version when the relative rotation is to be regressed. We neither explicitly supervise the network with pose annotations nor train by augmenting rotations.

4.2. Preliminaries and Technical Background

4.2.1. Equivariance

Definition 6 (Equivariant Map)
For a $\mathcal{G}$-space acting on $\mathcal{X}$, the map $\Phi : \mathcal{G} \times \mathcal{X} \mapsto \mathcal{X}$ is said to be equivariant if its domain and co-domain are acted on by the same symmetry group [40, 113]:

$$\Phi(\mathbf{g}_1 \circ \mathbf{x}) = \mathbf{g}_2 \circ \Phi(\mathbf{x}) \tag{4.1}$$

where $\mathbf{g}_1 \in \mathcal{G}$ and $\mathbf{g}_2 \in \mathcal{G}$. Equivalently $\Phi(T(\mathbf{g}_1)\,\mathbf{x}) = T(\mathbf{g}_2)\,\Phi(\mathbf{x})$, where $T(\cdot)$ is a linear representation of the group $\mathcal{G}$. Note that $T(\cdot)$ does not have to commute. It suffices for $T(\cdot)$ to be a homomorphism: $T(\mathbf{g}_1 \circ \mathbf{g}_2) = T(\mathbf{g}_1) \circ T(\mathbf{g}_2)$. In this work we use a stricter form of equivariance and consider $\mathbf{g}_2 = \mathbf{g}_1$.

Definition 7 (Equivariant Network)
An architecture or network is said to be equivariant if all of its layers are equivariant maps. Due to the transitivity of the equivariance, stacking up equivariant layers will result in globally equivariant networks e.g., rotating the input will produce output vectors which are transformed by the same rotation [76, 39].

4.2.2. The Quaternion Group $\mathbb{H}_1$

Definition 8 (Quaternion)
A quaternion $\mathbf{q}$ *is an element of Hamilton algebra $\mathbb{H}_1$, extending the complex numbers with three imaginary units $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ in the form:*

$$\mathbf{q} = q_1\boldsymbol{1} + q_2\boldsymbol{i} + q_3\boldsymbol{j} + q_4\boldsymbol{k} = (q_1, q_2, q_3, q_4)^T, \tag{4.2}$$

with $(q_1, q_2, q_3, q_4)^T \in \mathbb{R}^4$ and $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -\boldsymbol{1}$. $q_1 \in \mathbb{R}$ denotes the scalar part and $\boldsymbol{v} = (q_2, q_3, q_4)^T \in \mathbb{R}^3$, the vector part. The conjugate $\bar{\mathbf{q}}$ *of the quaternion $\mathbf{q}$ is given by $\bar{\mathbf{q}} := q_1 - q_2\boldsymbol{i} - q_3\boldsymbol{j} - q_4\boldsymbol{k}$. A* unit quaternion $\mathbf{q} \in \mathbb{H}_1$ *with $1 \stackrel{!}{=} \|\mathbf{q}\| := \mathbf{q} \cdot \bar{\mathbf{q}}$ and $\mathbf{q}^{-1} = \bar{\mathbf{q}}$, gives a compact and numerically stable parametrization to represent*

orientation of objects on the unit sphere $\mathcal{S}^3$, avoiding gimbal lock and singularities [114]. Identifying antipodal points $\mathbf{q}$ *and* $-\mathbf{q}$ *with the same element, the unit quaternions form a double covering group of SO*(3). $\mathbb{H}_1$ *is closed under the non-commutative multiplication or the Hamilton product:*

$$(\mathbf{p} \in \mathbb{H}_1) \circ (\boldsymbol{r} \in \mathbb{H}_1) = [p_1 r_1 - \mathbf{v}_p \cdot \mathbf{v}_r ; p_1 \mathbf{v}_r + r_1 \mathbf{v}_p + \mathbf{v}_p \times \mathbf{v}_r]. \tag{4.3}$$

Definition 9 (Linear Representation of $\mathbb{H}_1$)
We follow [115] and use the parallelizable ($d = 1, 2, 4$ *or* 8*) nature of unit quaternions to define* $T : \mathbb{H}_1 \mapsto \mathbb{R}^{4\times 4}$ *as:*

$$\mathbf{T}(\mathbf{q}) \triangleq \begin{bmatrix} q_1 & -q_2 & -q_3 & -q_4 \\ q_2 & q_1 & -q_4 & q_3 \\ q_3 & q_4 & q_1 & -q_2 \\ q_4 & -q_3 & q_2 & q_1 \end{bmatrix}.$$

To be concise we will use capital letters to refer to the matrix representation of quaternions e.g.$\mathbf{Q} \equiv T(\mathbf{q})$, $\mathbf{G} \equiv T(\mathbf{g})$. *Note that* $T(\cdot)$*, the injective homomorphism to the orthonormal matrix ring, by construction satisfies the condition in Dfn. 6 [116]:* $\det(\mathbf{Q}) = 1, \mathbf{Q}^\top = \mathbf{Q}^{-1}, \|\mathbf{Q}\| = \|\mathbf{Q}_{i,:}\| = \|\mathbf{Q}_{:,i}\| = 1$ *and* $\mathbf{Q} - q_1\mathbf{I}$ *is skew symmetric:* $\mathbf{Q} + \mathbf{Q}^\top = 2q_1\mathbf{I}$. *It is easy to verify these properties.* T *linearizes the Hamilton product or the group composition:* $\mathbf{g} \circ \mathbf{q} \triangleq T(\mathbf{g})\mathbf{q} \triangleq \mathbf{G}\mathbf{q}$.

4.2.3. 3D Point Clouds

Definition 10 (Point Cloud)
We define a 3D surface to be a differentiable 2-manifold embedded in the ambient 3D Euclidean space: $\mathcal{M}^2 \in \mathbb{R}^3$ *and a point cloud to be a discrete subset sampled on* $\mathcal{M}^2$*:* $\mathbf{X} \in \{\mathbf{x}_i \in \mathcal{M}^2 \cap \mathbb{R}^3\}$.

Definition 11 (Local Geometry)
For a smooth point cloud $\{\mathbf{x}_i\} \in \mathcal{M}^2 \subset \mathbb{R}^{N\times 3}$*, a* local reference frame (LRF) *is defined as an ordered basis of the tangent space at* $\mathbf{x}$*,* $\mathbf{T}_\mathbf{x}\mathcal{M}$*, consisting of orthonormal vectors:*

$$\mathcal{L}(\mathbf{x}) = [\boldsymbol{\partial}_1, \boldsymbol{\partial}_2, \boldsymbol{\partial}_3 \equiv \boldsymbol{\partial}_1 \times \boldsymbol{\partial}_2]. \tag{4.4}$$

Usually the first component is defined to be the surface normal $\boldsymbol{\partial}_1 \triangleq \mathbf{n} \in \mathcal{S}^2 : \|\mathbf{n}\| = 1$ *and the second one is picked according to a modality dependent heuristic.*

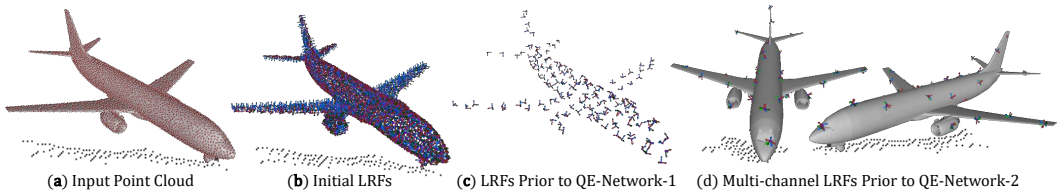


Figure 4.1.: Our network operates by processing local reference frames (LRF) on the object. Initial LRFs (**b**) are obtained by computing normal & tangent vectors on the point set in (**a**). (**c**) shows the LRFs randomly sampled from (a) and these are inputs to the first layer of our network. Subsequently, we obtain a multi-channel LRF that is a set of reference frames per pooling center (**d**). Holistically, our network aggregates the LRFs to arrive at rotation equivariant capsules.

Note that recent trends such as [110] acknowledge the ambiguity and either employ a *gauge* (tangent frame) equivariant design or propagate the determination of a certain direction until the last layer [117]. Here, we will assume that $\boldsymbol{\partial}_2$ can be uniquely and repeatably computed, a reasonable assumption for the point sets we consider [118]. For the cases where this does not hold, we will rely on the network's robustness. We will explain our method of choice in Sec. 4.3.2 and visualize LRFs of an airplane object in Fig. 4.1.

4.3. $SO(3)$-Equivariant 3D Capsule Networks

Disentangling orientation from representations requires guaranteed equivariances and invariances. Yet, the original capsule networks of [1] cannot achieve equivariance to general groups. To this end, [76] proposed to use a manifold-mean and a special aggregation that makes sure that the trainable transformations get pose-aligned points as input. We will extend this idea to the non-abelian $SO(3)$ and design capsule networks sparsely operating on a set of LRFs computed on local neighborhoods of points, parameterized by quaternions. In the following, we first explain our novel capsule layers, the main building block of our architecture. We then show how to stack those layers via a simple aggregation resulting in an $SO(3)$-equivariant 3D capsule network that yields invariant representations (or activations) as well as equivariant rotations (latent capsules).

4.3.1. Quaternion Equivariant Capsule Layers

To construct equivariant layers on the group of rotations, we are required to define a left-equivariant averaging operator $\mathcal{A}$ that is invariant under permutations of the group

elements, as well as a distance metric δ that remains unchanged under the action of the group. For these, we make the following choices:

Definition 12 (Geodesic Distance)
The Riemannian distance in the manifold of rotations lead to the following geodesic distance $\delta(\cdot) \equiv d_{quat}(\cdot)$:

$$\begin{aligned} d_{Riemann}(\mathbf{R}_1 \in SO(3), \mathbf{R}_2 \in SO(3)) &= \|\log(\mathbf{R}_1 \mathbf{R}_2^\top)\| \\ d_{quat}(\mathbf{q}_1, \mathbf{q}_2) &= 2\cos^{-1}(|\langle \mathbf{q}_1, \mathbf{q}_2 \rangle|) \end{aligned} \tag{4.5}$$

Definition 13 (Quaternion Mean)
For a set of Q rotations $\mathbf{S} = \{\mathbf{q}_i\}$ and associated weights $\mathbf{w} = \{w_i\}$, the weighted mean operator $\mathcal{A}(\mathbf{S}, \mathbf{w}) : {\mathbb{H}_1}^n \times \mathbb{R}^n \mapsto {\mathbb{H}_1}^n$ is defined through the following maximization procedure [119]:

$$\bar{\mathbf{q}} = \underset{\mathbf{q} \in \mathbb{S}^3}{\arg\max}\, \mathbf{q}^\top \mathbf{M} \mathbf{q} \tag{4.6}$$

where $\mathbf{M} \in \mathbb{R}^{4\times 4}$ is defined as: $\mathbf{M} \triangleq \sum_{i=1}^{Q} w_i \mathbf{q}_i \mathbf{q}_i^\top$. The average quaternion is the eigenvector of $\mathbf{M}$ corresponding to the maximum eigenvalue. This operation lends itself to both analytic [120] and automatic differentiation [121].

Theorem 3
Quaternions, the employed mean $\mathcal{A}(\mathbf{S}, \mathbf{w})$ and geodesic distance $\delta(\cdot)$ enjoy the following properties:

1. *$\mathcal{A}(\mathbf{g} \circ \mathbf{S}, \mathbf{w})$ is left-equivariant: $\mathcal{A}(\mathbf{g} \circ \mathbf{S}, \mathbf{w}) = \mathbf{g} \circ \mathcal{A}(\mathbf{S}, \mathbf{w})$.*
2. *Operator $\mathcal{A}$ is invariant under permutations:*
   $\mathcal{A}(\{\mathbf{q}_{\sigma(1)}, \ldots, \mathbf{q}_{\sigma(Q)}\}, \mathbf{w}_\sigma) = \mathcal{A}(\{\mathbf{q}_1, \ldots, \mathbf{q}_Q\}, \mathbf{w})$.
3. *The transformations $\mathbf{g} \in \mathbb{H}_1$ preserve the geodesic distance $\delta(\cdot)$ given in Dfn. 12.*

Proof. The proofs are given in (Appendix B). □

We also note that the above mean is closed form, differentiable and can be implemented batchwise. We are now ready to construct the group dynamic routing (DR) by agreement that is equivariant thanks to Thm. 3. The core idea is to *route* from or assign the *primary capsules* that constitute the input LRF set, to the *latent capsules* by an iterative clustering which respects the group structure. At each step, we assign the weighted group mean to

Algorithm 2: Quaternion Equivariant Dynamic Routing

input : Input points $\{\mathbf{x}_1, ..., \mathbf{x}_K\} \in \mathbb{R}^{K\times 3}$, input capsules (LRFs) $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_L\} \in {\mathbb{H}_1}^L$, with $L = N^c \cdot K$, N^c is the number of capsules per point, activations $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_L)^T$, trainable transformations $\mathbf{T} = \{\mathbf{t}_{i,j}\}_{i,j} \in {\mathbb{H}_1}^{L\times M}$

output : Updated frames $\hat{\mathcal{Q}} = \{\hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_M\} \in {\mathbb{H}_1}^M$, updated activations $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_M)^T$

for *All primary (input) capsules i* **do**
  for *All latent (output) capsules j* **do**
    $\mathbf{v}_{i,j} \leftarrow \mathbf{q}_i \circ \mathbf{t}_{i,j}$ `// compute votes`
for *All latent (output) capsules j* **do**
  $\hat{\mathbf{q}}_j \leftarrow \mathcal{A}\big(\{\mathbf{v}_{1,j} \ldots \mathbf{v}_{K,j}\}, \boldsymbol{\alpha}\big)$ `// initialize output capsules`
  for *k iterations* **do**
    for *All primary (input) capsules i* **do**
      $w_{i,j} \leftarrow \alpha_i \cdot \text{sigmoid}\big(-\delta(\hat{\mathbf{q}}_j, \mathbf{v}_{i,j})\big)$ `// compute the current weight`
    $\hat{\mathbf{q}}_j \leftarrow \mathcal{A}\big(\{\mathbf{v}_{1,j} \ldots \mathbf{v}_{L,j}\}, \mathbf{w}_{:,j}\big)$ `// see Eq (4.6)`
  $\hat{\alpha}_j \leftarrow \text{sigmoid}\big(-\frac{1}{K}\sum_{1}^{L}\delta(\hat{\mathbf{q}}_j, \mathbf{v}_{i,j})\big)$ `// recompute activations`

each output capsule. The weights $w \leftarrow \sigma(\mathbf{x}, \mathbf{y})$ are inversely proportional to the distance between the vote quaternion and the new quaternion (cluster center). See Alg. 2 for details. In the following, we analyze our variant of routing as an interesting case of the affine, Riemannian Weiszfeld algorithm [7, 122].

Lemma 1
For $\sigma(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x}, \mathbf{y})^{q-2}$ the equivariant routing procedure given in Alg. 2 is a variant of the affine subspace Wieszfeld algorithm [7, 122] that is a robust algorithm for computing the L_q geometric median.

Proof Sketch. The proof follows from the definition of Weiszfeld iteration [122] and the mean and distance operators defined in Sec. 4.3.1. We first show that computing the weighted mean is equivalent to solving the normal equations in the iteratively reweighted least squares (IRLS) scheme [123]. Then, the inner-most loop correspond to the IRLS or Weiszfeld iterations. We provide the detailed proof in Appendix B. □

Note that, in practice one is quite free to choose the weighting function $\sigma(\cdot)$ as long

as it is inversely proportional to the geodesic distance and concave [124]. We leave the analyses of the variants of these algorithms as future work. The original dynamic routing can also be formulated as a clustering procedure with a KL divergence regularization. This holistic view paves the way to better routing algorithms [125]. Our perspective is akin yet more geometric due to the group structure of the parameter space. Thanks to the connection to the Weiszfeld algorithm, the convergence behavior of our dynamic routing can be directly analyzed within the theoretical framework presented by [122, 7].

Theorem 4
Under mild assumptions provided in Appendix B, the sequence of the DR-iterates generated by the inner-most loop almost surely converges to a critical point.

Proof Sketch. Proof, given in the appendix, is a direct consequence of Lemma 1 and directly exploits the connection to the Weiszfeld algorithm. □

4.3.2. Equivariant 3D Point Capsule Network Architecture

The essential ingredient of our architecture, QE-Network, is shown in Fig. 4.2. We also provide a corresponding pseudocode in Alg. 4 of (Appendix B). The input of the QE-Network are a local patch of points with coordinates $\mathbf{x}_i \subset \mathbb{R}^{K\times 3}$, rotations parametrized as quaternions $\mathbf{q}_i \subset {\mathbb{H}_1}^{K\times N^c}$ and activations $\boldsymbol{\alpha}_i \subset \mathbb{R}^{K\times N^c}$. $\mathbf{q}_i$ also represents input primary capsules and local reference frames. N^c is the number of input capsule channels per point and it is equal to the number of output capsules (M) from the last layer. In the initial layer, $\mathbf{q}_i$ represents the pre-computed LRFs and N^c is equal to 1. Given points $\mathbf{x}_i$ and rotations $\mathbf{q}_i$, we compute the quaternion average μ_i in channel-wise as the initial pose candidates: $\mu_i \subset {\mathbb{H}_1}^{N^c}$. These candidates are used to bring the receptive field in multiple canonical orientations by rotating the points: $\mathbf{x}'_i = (\mu_i{}^{-1} \circ \mathbf{x}_i) \subset \mathbb{R}^{K\times N^c\times 3}$. Since the points in the local receptive field lie in continuous $\mathbb{R}^3$, training a discrete set of pose transformations $\mathbf{t}_{i,j}$ based on local coordinates is not possible. Instead, we employ a point-to-transform network $t(\cdot) : \mathbb{R}^{N^c\times 3} \rightarrow \mathbb{R}^{M\times N^c\times 4}$ that maps the point in multiple canonical orientations to transformations. The network is shared over all points to compute the transformations $\mathbf{t}_{i,j} = (t(\mathbf{x}'_1), ..., t(\mathbf{x}'_K))_{i,j} \subset \mathbb{R}^{K\times M\times N^c\times 4}$, which are used to calculate the votes for dynamic routing as $\mathbf{v}_{i,j} = \mathbf{q}_i \circ \mathbf{t}_{i,j}$. The network $t(\cdot)$ consists of fully-connected layers that regresses the transformations, similar to common operators for continuous convolutions [126, 127, 128]. It is the continuous alternative to directly optimizing transformations lying in a grid kernel, as it is done in the original dynamic routing by [1]. Note that $t(\cdot)$ predicts quaternions by unit-normalizing the regressed

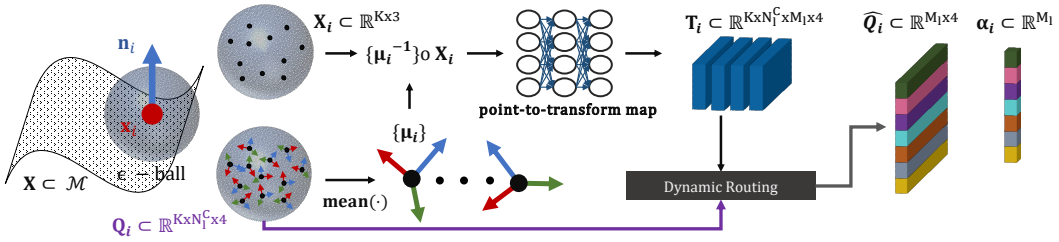


Figure 4.2.: Our **q**uaternion **e**quivariant (QE) network for processing local patches: Our input is a 3D point set $\mathbf{X}$ on which we query local neighborhoods $\{\mathbf{x}_i\}$ with precomputed LRFs $\{\mathbf{q}_i\}$. Essentially, we learn the parameters of a fully connected network that continuously maps the canonicalized local point set to transformations $\mathbf{t}_i$, which are used to compute hypotheses (votes) from input poses. By a special dynamic routing procedure that uses the activations determined in a previous layer, we arrive at latent capsules that are composed of a set of orientations $\hat{\mathbf{q}}_i$ and new activations $\hat{\boldsymbol{\alpha}}_i$. Thanks to the decoupling of local reference frames, $\hat{\boldsymbol{\alpha}}_i$ is invariant and orientations $\hat{\mathbf{q}}_i$ are equivariant to input rotations. All the operations and hence the entire QE-network are equivariant achieving a guaranteed disentanglement of the rotation parameters. *Hat symbol ($\hat{\mathbf{q}}$) refers to 'estimated'.*

output: $\mathbf{t}_{i,j} \subset {\mathbb{H}_1}^{K\times M\times N^c}$. Although Riemannian layers of [129] or spherical predictions of [130] can improve the performance, the simple strategy works reasonably for our case. After computing the votes, we utilize the input activation $\boldsymbol{\alpha}_i$ to iteratively refine the output capsules (weighted average of votes) $\hat{\mathbf{q}}_i$ and activations $\hat{\boldsymbol{\alpha}}_i$ by routing by agreement as shown in Alg. 2.

In order to gradually increase the receptive field, we stack QE-networks creating a deep hierarchy, pooling the points and the LRFs before each layer. Note that we are allowed to do so thanks to the properties of equivariance. In particular, we input $N = 64$ patches to our architecture that is composed of two QE-networks. We call the centers of these patches *pooling centers*. In the first layer, each of those centers is linked to their immediate vicinity leading to $K = 9$-star local connectivity from which we compute the $64 \times 64 \times 4$ intermediary capsules. The input LRFs of the first layer are sampled from pre-calculated LRF-set and the input activation is set to 1. The LRFs in the second layer $l = 2$ are the output capsules of the first layer, $l = 1$ and are routed to the output capsules that are as many as the number of classes C, $M_2 = C$. The activation of the second layer is updated by the output of the first layer as well. This construction is shown in Fig. 4.3. Specifically, for $l = 1$, we use $K = 9, N_l{}^c = 1, M_l = 64$ and for $l = 2$, $K = 64, N_l{}^c = 64, M_l = C = 40$. This way, in this last layer all the pooling centers act as a single patch ($K = 64$). A single QE-network acts on this patch to create the final $C \times 4$

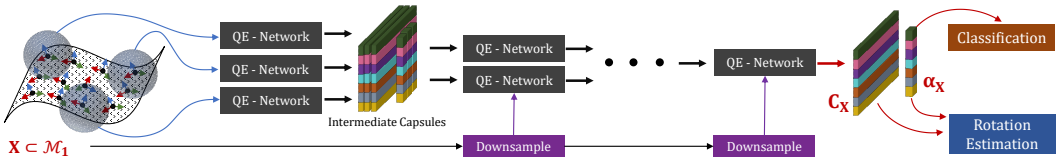


Figure 4.3.: Our entire capsule architecture. We hierarchically send all the local patches to our Q-network as shown in Fig. 4.2. At each level the points are pooled in order to increase the receptive field, gradually reducing the LRFs into a single capsule per class. We use classification and pose estimation (in the siamese case) as supervision cues to train the point-to-transform maps.

capsules and C activations. More details are reported in Alg. 4 of Appendix B.

Implementation Details We implement our network in PyTorch and use the ADAM optimizer [131] with a learning rate of 0.001. The point-transformation mapping network is implemented by two FC-layers composed of 64 hidden units. We set the initial activation of the input LRF to 1.0. In each layer, we use 3 iterations of DR. For classification we use the spread loss [132] and the rotation loss is identical to $\delta(\cdot)$.

Surface normals are computed by local plane fits [133]. We compute the second axis of the LRF, $\boldsymbol{\partial}_2$, by FLARE [134], that uses the normalized projection of the point within the periphery of the support showing the largest distance, onto the tangent plane of the center: $\boldsymbol{\partial}_2 = \frac{\mathbf{p}_{\max} - \mathbf{p}}{\|\mathbf{p}_{\max} - \mathbf{p}\|}$. Note that using other LRFs such as SHOT [135] or the more modern GFrames of [136] is possible. We found FLARE to be sufficient for our experiments. Prior to all operations, we flip all the LRF quaternions such that they lie on the northern hemisphere: $\{\mathbf{q}_i \in \mathbb{S}^3 : q_i^w > 0\}$.

3D Shape Classification. We use ModelNet40 dataset of [103, 6] to assess our classification performance. Each shape is composed by $10K$ points. We assign the LRFs to a subset of the uniformly sampled points, $N = 512$ [107]. We train the networks without any rotation augmentation (NR) and put them to test under arbitrary $SO(3)$ rotations (AR). Our results are shown in Tab. 4.1 along with that of PointNet (PN) [5], PointNet++ (PN++) [5], KD-treeNet [26], Point2Seq [137], Spherical CNNs [47], PRIN [138] and PPF-FoldNet (PPF) [52]. We also present a version of our algorithm (*Var*) that avoids the canonicalization within the QE-network. This is a non-equivariant network that we still train without data augmentation. While this version gets comparable results to the state of the art for the NR/NR case, it cannot handle random $SO(3)$ variations (AR). Note that PPF uses the point-pair-feature [106] encoding and hence creates invariant

Table 4.1.: Classification accuracy on ModelNet40 dataset [103] for different methods as well as ours. We also report the number of parameters optimized for each method. Right hand side of the table denotes the symmetric objects, which we include for completeness. **X/Y** means that we train with **X** and test with **Y**.

	PN	PN++	KD-treeNet	Point2Seq	Sph.CNNs	PRIN	PPF	Ours (Var.)	Ours
NR/NR	88.45	89.82	86.20	**92.60**	-	80.13	70.16	85.27	74.43
NR/AR	12.47	21.35	8.49	10.53	43.92	68.85	70.16	11.75	**74.07**
# Params	3.5M	1.5M	3.6M	1.8M	0.5M	1.5M	3.5M	0.4M	**0.4M**

input representations. For the scenario of NR/AR, our equivariant version outperforms all the other methods, including equivariant spherical CNNs [47] by a significant gap of at least 5% even when [47] uses the mesh. The object symmetries in this dataset are responsible for a significant portion of the errors we make. It is worth mentioning that we also trained TFNs [55] for that task, but their memory demand made it infeasible to scale to this application.

Number of Parameters. Use of LRFs helps us to restrict the rotation group to certain elements and thus we can use networks with significantly fewer parameters (as low as $0.44M$) compared to others as shown in Tab. 4.1. The number of parameters in our network depends upon the number of classes, *e.g.*for ModelNet10 we have $0.047M$ parameters.

Rotation estimation in 3D point clouds. Our network can estimate both the canonical and relative object rotations without pose-supervision. To evaluate this desired property, we used the well-classified shapes on ModelNet10 dataset, a sub-dataset of Modelnet40 [103]. We generate multiple instances per shape by transforming the instance with five arbitrary $SO(3)$ rotations. As we are also affected by the sampling of the point cloud, we resample the mesh five times and generate different pooling graphs across all the instances of the same shape. Our QE-architecture can estimate the pose in two ways: 1) by directly using the output capsule with the highest activation, 2) by a siamese architecture that computes the relative quaternion between the capsules that are maximally activated as shown in Fig. 4.4. Both modes of operation are free of the data augmentation. Our results against the baselines including a naive averaging of the LRFs (Mean LRF) and principal axis alignment (PCA) are reported in Tab. 4.2

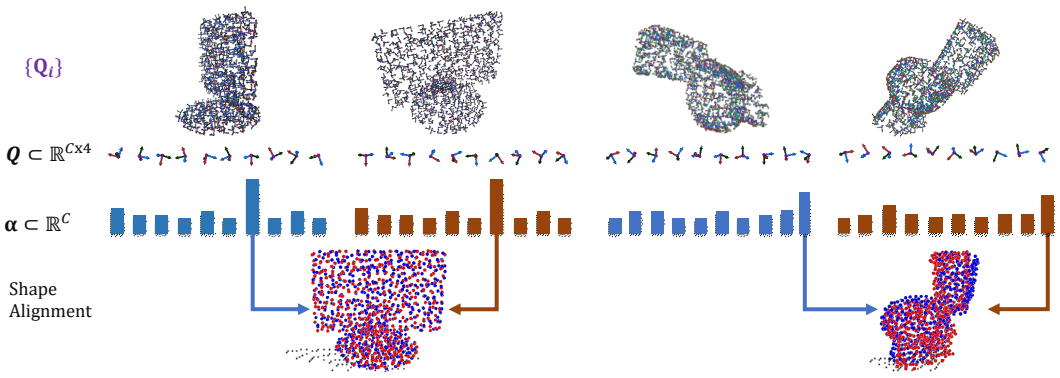


Figure 4.4.: Shape alignment on the **monitor** (left) and **toilet** (right) objects via our siamese equivariant capsule architecture. The shapes are assigned to the the maximally activated class. The corresponding pose capsule provides the rotation estimate.

as the relative angular error (RAE). We further include results of PointNetLK [139] and IT-Net [140], two state of the art 3D networks that iteratively aligns two point sets. These methods are in nature similar to iterative closest point (ICP) algorithm [141] but 1) do not require an initialization (first iteration estimates the pose), 2) learn data driven updates. Methods that use mesh inputs such as Spherical CNNs [47] cannot be included here as the random sampling of the same surface would not affect those. We also avoid methods that are just invariant to rotations (and hence cannot estimate the pose) such as Tensorfield Networks [55]. Finally, note that , IT-net [140] and PointLK need to train a lot of epoches (e.g. 500) with random $SO(3)$ rotation augmentation in order to get the models that cover the full $SO(3)$, whereas we train only for $\sim$ 100 epochs. We include more details about the baselines in Appendix B under Fig. B.4. RAE between the ground truth and the prediction is computed as the relative angle in degrees: $d(\mathbf{q}_1, \mathbf{q}_2)/\pi$.

Note that resampling and random rotations render the job of all methods difficult. However, both our version that tries to find a canonical alignment and the siamese variant which seeks a relative rotation is better than the baselines. As unsupervised pose estimation of symmetric objects is a challenging task we also report results on the non-symmetric subset (No_Sym).

Robustness against point resampling. Density changes in the local neighborhoods of the shape are an important cause of error for our network. Hence, we ablate by applying random resamplings (patch-wise dropout) to the objects in ModelNet10 dataset and repeating the pose estimation and classification as described above. The first part (LRF-10K) of Tab. 4.3 shows our findings against gradual increases of the number of

Table 4.2.: Error of rotation estimation in different categories of ModelNet10. Right side of the table denotes the objects with rotational symmetry, which we include for completeness. PCA-S refers to running PCA only on a resampled instance, while PCA-SR applies both rotations and resampling.

Method	Avg.	No_Sym	Chair	Bed	Sofa	Toilet	Monitor	Table	Desk	Dresser	NS	Bathtub
Mean LRF	0.41	0.35	0.32	0.36	0.34	0.41	0.34	0.45	0.60	0.50	0.46	0.32
PCA-S	0.40	0.42	0.60	0.53	0.46	0.32	0.12	0.47	**0.23**	**0.33**	0.43	0.55
PCA-SR	0.67	0.67	0.69	0.70	0.67	0.68	0.61	0.67	0.67	0.67	0.66	0.70
PointNetLK	0.37	0.38	0.43	0.31	0.40	0.40	0.31	0.40	0.33	0.39	0.38	0.34
IT-net	0.27	0.19	0.10	0.22	0.17	0.20	0.28	**0.31**	0.41	0.44	0.40	0.39
Ours	0.27	0.17	0.11	0.20	0.16	0.18	0.19	0.43	0.40	0.48	0.33	0.31
Ours (siamese)	**0.20**	**0.09**	**0.08**	**0.10**	**0.08**	**0.11**	**0.08**	0.40	0.35	0.34	**0.32**	**0.30**

patches. Here, we sample 2K LRFs from the 10K LRFs computed on an input point 10K. 100% dropout corresponds to 2K points in all columns. On second ablation, we reduce the amount of points on which we compute the LRFs, to 2K and 1K respectively. As we can see from the table, our network is robust towards the changes in the LRFs as well as the density of the points.

Table 4.3.: Ablation study on point density.

LRF Input	LRF-10K				LRF-2K	LRF-1K
Dropout	50%	66%	75%	100%	100%	100%
Class. Err	77.8	83.3	83.4	87.8	85.46	79.74
Angle. Err	0.34	0.27	0.25	0.09	0.10	0.12

4.4. Conclusion and Discussion

In this chapter, we have presented a new framework for achieving permutation invariant and $SO(3)$ equivariant representations on 3D point clouds. Proposing a variant of the capsule networks, we operate on a sparse set of rotations specified by the input LRFs thereby circumventing the effort to cover the entire $SO(3)$. Our network natively consumes a compact representation of the group of 3D rotations - quaternions, and we have theoretically shown its equivariance. We have also established convergence results for our Weiszfeld dynamic routing by making connections to the literature of robust

optimization. Our network is among the few for having an explicit group-valued latent space and thus naturally estimates the orientation of the input shape, even without a supervision signal.

Limitations. In the current form our performance is severely affected by the shape symmetries. The length of the activation vector depends on the number of classes and for achieving sufficiently descriptive latent vectors we need to have a significant number of classes. On the other side, this allows us to perform with merit on problems where the number of classes is large. Although we have reported robustness to those, the computation of LRFs is still sensitive to the point density changes and resampling. LRFs themselves are also ambiguous and sometimes non-unique.

Future work. Inspired by [110] and [117] our feature work will involve establishing invariance to the direction in the tangent plane. We also plan to apply our network in the broader context of 3D object detection under arbitrary rotations and look for equivariances among point resampling.

5. Conclusions

Conclusions In this thesis, three kinds of 3D features representations are developed towards different properties *e.g.*, discrimination, spatial attention, rotation invariance or equivariance. Thanks to the capability of representing a sparse 3D structure accurately while being acquired directly with the ubiquitous sensors, RGB-D images, and point clouds are the main 3D data formats used in this thesis.

Based on RGB-D images, we proposed a feature representation by fusing the color and depth model in a probabilistic manner. Thanks to the range information from the depth image, the feature shows strong discrimination power towards background clutters and foreground occlusions which are the main drawbacks of the previous color statistical models of the object. We also present an object tracking algorithm based on the proposed representation to evaluate its performance. The evaluation result on a public RGB-D object tracking dataset suggests that our tracker out-performs other state-of-the-art trackers in both efficiency and robustness. Moreover, we integrate the proposed tracker to an RGB-D camera network to evaluate its performance in long-term tracking. We also are the first to present a multiple object tracking dataset with RGB-D sequences from well-calibrated RGB-D cameras in different viewpoints. It is worth to mention that the ground truth is automatically obtained from the movements of a calibrated robot. The evaluation results suggest the high potential of our proposed algorithm in real applications that require long-term tracking, *e.g.*, sports analysis, human-robot interaction. The dataset has been made public on the website.

The other two kinds of feature representations are based on 3D point cloud and learned with 3D deep neural networks.

One of them, namely *3D-PointCapsNet* is a flexible and effective tool for unsupervised learning representations with spatial attention for 3D shape processing and understanding. With the proposed network architecture, we can encode 3D shapes into multiple vectors (capsules) and each capsule is embedded with the information of one part of the shape with specific spatial attention. With this property, we manage to specify different parts of the shape in latent feature space which offers us the possibility for parsing/disentangling the shape to achieve better understanding and shape generation. This property also

enables training part-segmentation with very limited annotated data. Moreover, our proposed representation can surpass the current art in reconstruction quality, local 3D feature extraction and transfer learning for 3D object recognition.

The other feature representation is designed towards the rotation invariance/equivariance property. We operate on a sparse set of rotations specified by the input LRFs thereby circumventing the effort to cover the entire $SO(3)$. The network natively consumes a compact representation of the group of 3D rotations - quaternions, and we have theoretically shown its equivariance. We have also established convergence results for our Weiszfeld dynamic routing by making connections to the literature of robust optimization. The proposed network is among the few for having an explicit group-valued latent space and thus naturally estimates the orientation of the input shape, even without a supervision signal.

Future Work Inspired by [110] and [117] our feature work will involve establishing invariance to the direction in the tangent plane. We also plan to apply our network in the broader context of 3D object detection under arbitrary rotations and look for equivariances among point resampling.

Appendices

A. Appendix for Chapter 3

A.1. Semi-supervised Classification

We begin by showing semi-supervised classification results in Tab. A.1. Note that our network can generate predictions that are on par with or better than FoldingNet [30].

Table A.1.: Part segmentation on ShapeNet-Part by learning on limited training data. The table shows the accuracies obtained by FoldingNet [30] and our approach for different amount of training data.

	1%	2%	5%	20%	100%
FoldingNet	56.15	67.05	75.97	84.06	88.41
Ours	59.24	67.67	76.49	84.48	88.91

A.2. Part Segmentation

We first give a small summary of the part association network for optional supervision. The input to this one-layer architecture is the latent capsules combined with one-hot vector of the object category. The output is the part prediction of each capsule. We use the cross entropy loss as our loss function and Adam as the optimizer with the learning rate of 0.01. The network structure is shown in Fig. A.1.

Then we utilize the pre-trained decoder to reconstruct the object with the labeled capsules. Fig. A.3 depicts further visualizations for different objects from the ShapeNet-Part dataset [101]. Our results are also qualitatively comparable to ground truth.

A.3. Part Interpolation

We first show an overview of how we perform part interpolation. While this part has been thoroughly explained in Sec. 3.3, we have omitted this architecture illustration due

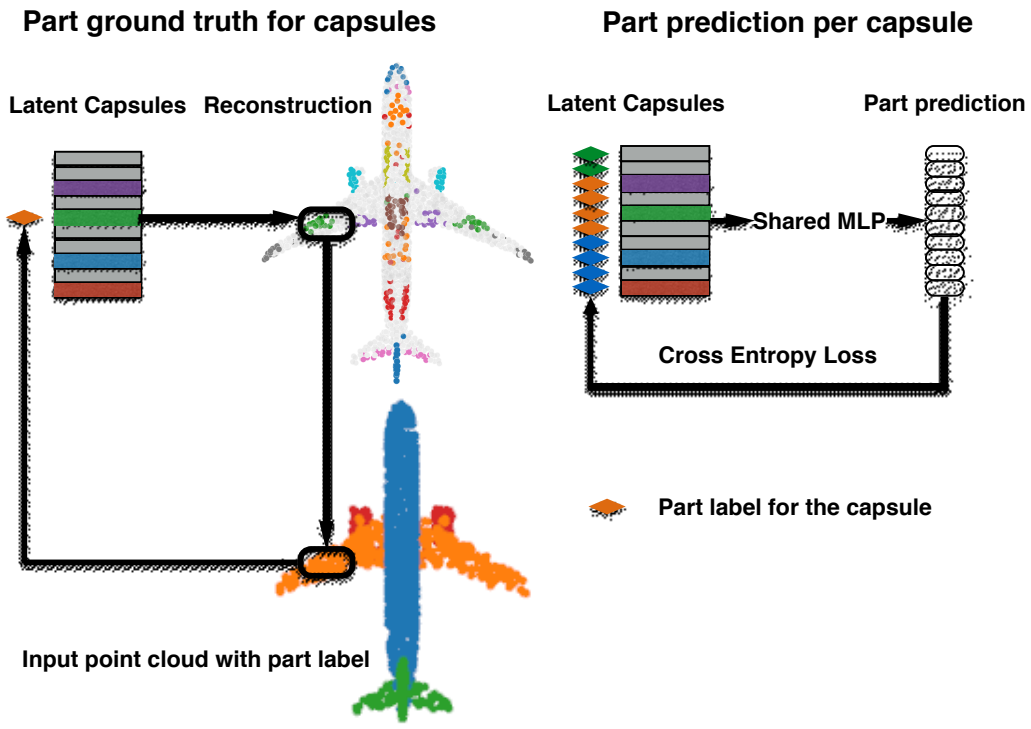


Figure A.1.: Supervising the 3d point capsule networks for part prediction. Instead of performing a point-wise part labeling, we use a capsule-wise association requiring less data annotation efforts.

to space considerations. We now provide this in Fig. A.4.

Next we show, the part interpolation results on different objects. In this qualitative evaluation, we are given two shapes and the goal is to interpolate the source part towards the target. To do that we find the matching capsules that represent the part of interest in both shapes. We then linearly interpolate from the capsule(s) of the source to the one(s) of the target. This generates visually pleasing intermediate shapes, which our network has never seen before. Here we see that the learned embedding resemble a Euclidean space where linear latent space arithmetic is possible. It is also visible that such interpolation scheme can handle topological changes such as merging or branching legs. In the end of interpolation a new shape is generated in which the part is replaced completely with the target's. That brings us to our second and interesting application, part replacement.

A.4. Part Replacement

We now supplement our thesis by presenting additional qualitative results on the task of part replacement. Fig. A.6 shows numerous object pairs where a part-of-interest is selected in both and exchanged by the help of latent space capsule arithmetic. Analogous to the ones in Sec. 3.3 we also show a cut-and-paste operation that is a mere exchange of the parts in 3D space, obviously resulting in undesired disconnected shapes. Thanks to our decoder's capability in generating high fidelity shapes, our capsule-replacement respects the overall coherence of the resulting point cloud.

A.5. Ablation Study

In order to show the prosperity of the dynamic routing, we compare the reconstruction result by replacing the DR with PointNet-like set of convolutional layers. In this ablation study, the primary point capsules (1024×16) are considered as 1024 point-features and each point has the feature dimension of 16. We utilize a shared MLP to increase the feature dimension from 16 to 64. After conducting max pooling, we can obtain a vector of length 64. With multiple MLPs and max-pooling, we are able to generate 64 vectors which have the same dimensions as the latent capsules produced by dynamic routing. The structure of this comparison module is shown in Fig. A.2. To carry out our fair evaluation, we re-train the whole AE with this module. The result of the reconstruction is shown in Fig. 3.4 of Sec. 3.3.

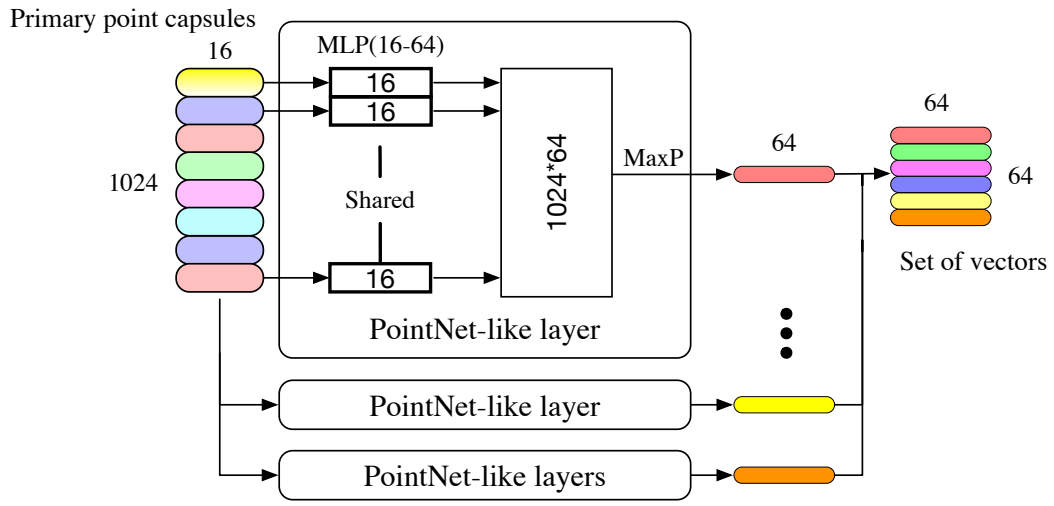


Figure A.2.: The structure of the comparison module that operates on the primary point capsules and generates a set of vectors having the same dimensionality as the latent capsule output of DR.

A.6. A Discussion on the Local Spatial Attention

Our network consists of multiple MLPs acting on a single capsule. It encodes the part information inside that capsule rather than the MLPs themselves. For that reason, the local attention stems from both the organization of primary point capsules (in our case obtained by dynamic routing) and potentially the decoder (see Fig. 3.4). Thus, we are able to control and represent the shape instantiation in the latent space as shown in *part interpolation/replacement* evaluations. Contrarily, AtlasNet reconstructs different local patches with different MLPs from the same latent vector. This embeds the part knowledge into the MLPs, making it challenging to control the shape properties.

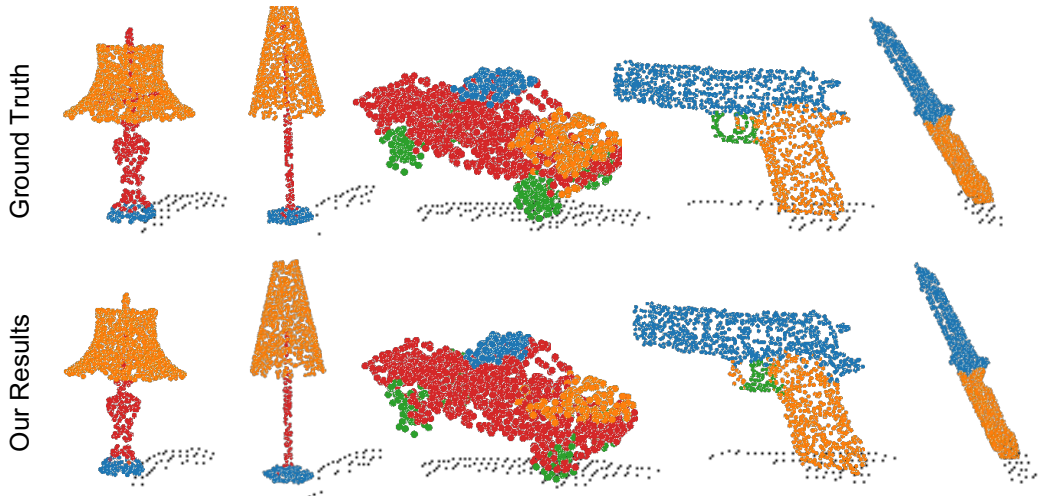


Figure A.3.: Part segmentation on limited amount of training data.

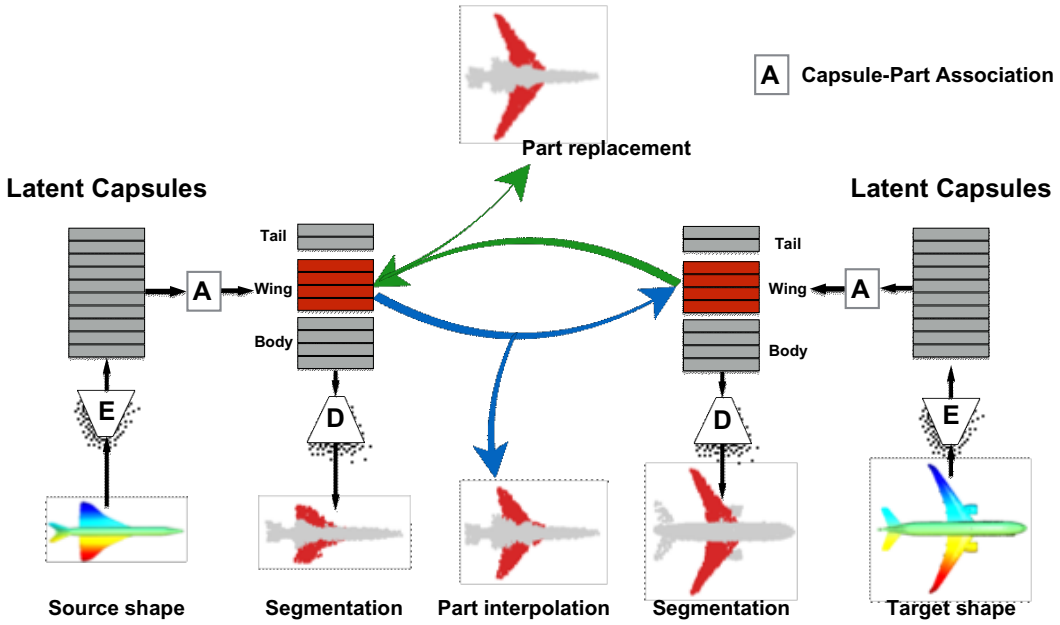


Figure A.4.: Our interpolation / replacement pipeline.

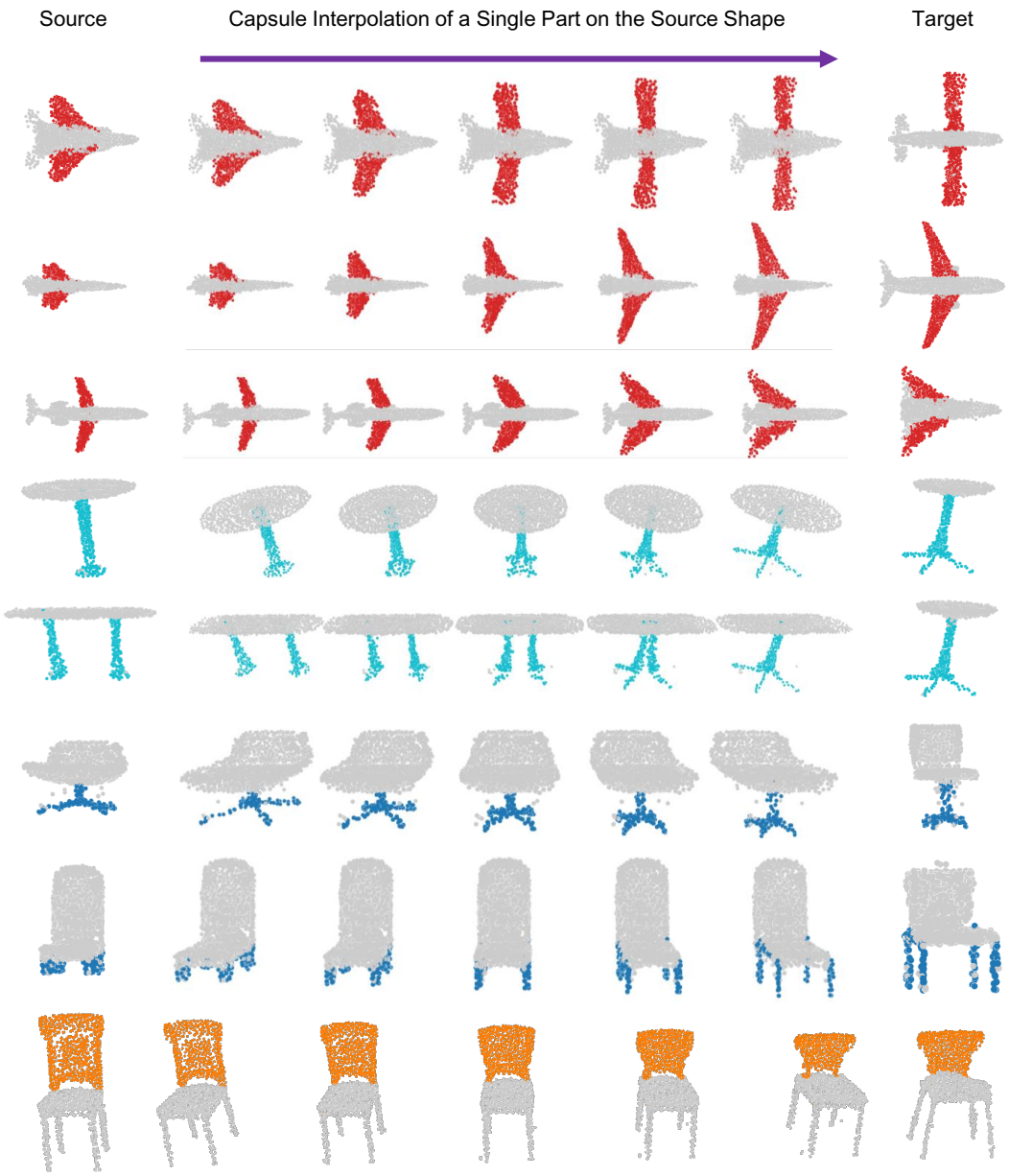


Figure A.5.: Visualization of part interpolation from source shape part to target. By simple linear interpolation on the correspondent capsule(s), smooth intermediate topologies could be generated.

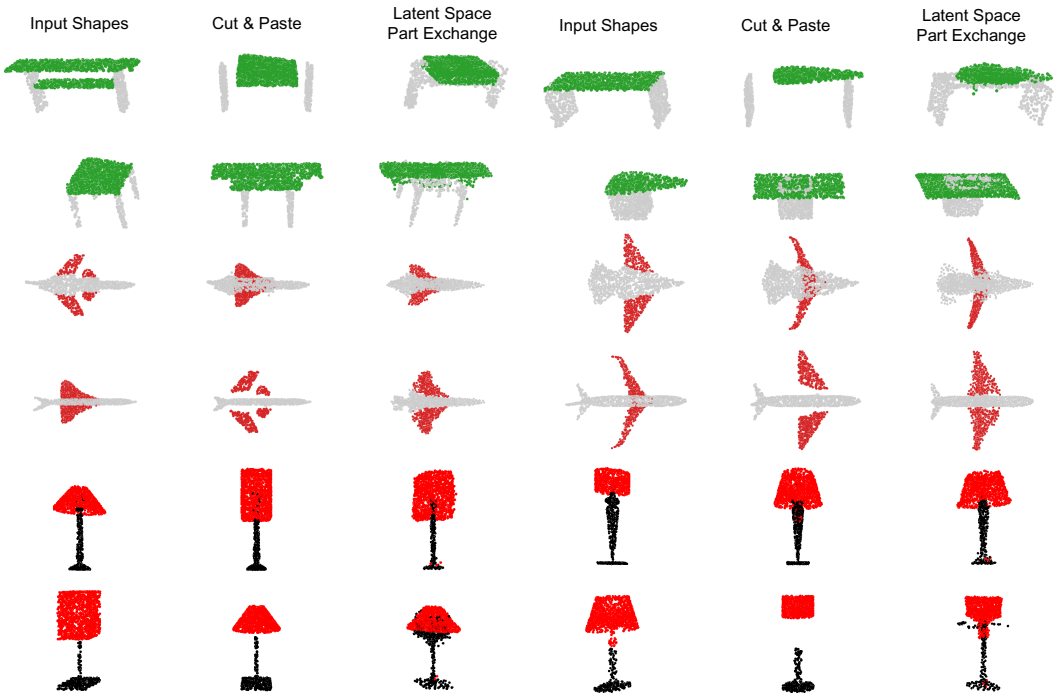


Figure A.6.: Part replacement visualization and comparison. By operating in the latent space, more natural replacement results could be obtained, without suffering from the detachment problems as with simple Cut & Paste method.

B. Appendix for Chapter 4

B.1. Proof of Proposition 1

Before presenting the proof we recall the three individual statements contained in Prop. 1:

1. $\mathcal{A}(\mathbf{g} \circ \mathbf{S}, \mathbf{w})$ is left-equivariant: $\mathcal{A}(\mathbf{g} \circ \mathbf{S}, \mathbf{w}) = \mathbf{g} \circ \mathcal{A}(\mathbf{S}, \mathbf{w})$.
2. Operator $\mathcal{A}$ is invariant under permutations: $\mathcal{A}(\{\mathbf{q}_{\sigma(1)}, \ldots, \mathbf{q}_{\sigma(Q)}\}, \mathbf{w}_\sigma) = \mathcal{A}(\{\mathbf{q}_1, \ldots, \mathbf{q}_Q\}, \mathbf{w})$.
3. The transformations $\mathbf{g} \in \mathbb{H}_1$ preserve the geodesic distance $\delta(\cdot)$.

Proof. We will prove the propositions in order.

1. We start by transforming each element and replace $\mathbf{q}_i$ by $(\mathbf{g} \circ \mathbf{q}_i)$ of the cost in Eq (4.6):

$$\begin{aligned}
\mathbf{q}^\top \mathbf{M} \mathbf{q} &= \mathbf{q}^\top \Big(\sum_{i=1}^{Q} w_i \mathbf{q}_i \mathbf{q}_i^\top \Big) \mathbf{q} && \text{(B.1)} \\
&= \mathbf{q}^\top \Big(\sum_{i=1}^{Q} w_i (\mathbf{g} \circ \mathbf{q}_i)(\mathbf{g} \circ \mathbf{q}_i)^\top \Big) \mathbf{q} && \text{(B.2)} \\
&= \mathbf{q}^\top \Big(\sum_{i=1}^{Q} w_i \mathbf{G} \mathbf{q}_i \mathbf{q}_i^\top \mathbf{G}^\top \Big) \mathbf{q} && \text{(B.3)} \\
&= \mathbf{q}^\top \Big(\mathbf{G} \mathbf{M}_1 \mathbf{G}^\top + \cdots + \mathbf{G} \mathbf{M}_Q \mathbf{G}^\top \Big) \mathbf{q} \\
&= \mathbf{q}^\top \mathbf{G} \Big(\mathbf{M}_1 \mathbf{G}^\top + \cdots + \mathbf{M}_Q \mathbf{G}^\top \Big) \mathbf{q} && \text{(B.4)} \\
&= \mathbf{q}^\top \mathbf{G} \Big(\mathbf{M}_1 + \cdots + \mathbf{M}_Q \Big) \mathbf{G}^\top \mathbf{q} && \text{(B.5)} \\
&= \mathbf{q}^\top \mathbf{G} \mathbf{M} \mathbf{G}^\top \mathbf{q} && \text{(B.6)} \\
&= \mathbf{p}^\top \mathbf{M} \mathbf{p}, && \text{(B.7)}
\end{aligned}$$

where $\mathbf{M}_i = w_i \mathbf{q}_i \mathbf{q}_i^\top$ and $\mathbf{p} = \mathbf{G}^\top \mathbf{q}$. From orthogonallity of $\mathbf{G}$ it follows $\mathbf{p} = \mathbf{G}^{-1}\mathbf{q} \implies \mathbf{g} \circ \mathbf{p} = \mathbf{q}$ and hence $\mathbf{g} \circ \mathcal{A}(\mathbf{S}, \mathbf{w}) = \mathcal{A}(\mathbf{g} \circ \mathbf{S}, \mathbf{w})$.

2. The proof follows trivially from the permutation invariance of the symmetric summation operator over the outer products in Eq (B.4).

3. It is sufficient to show that $|\mathbf{q}_1^\top \mathbf{q}_2| = |(\mathbf{g} \circ \mathbf{q}_1)^\top (\mathbf{g} \circ \mathbf{q}_2)|$ for any $\mathbf{g} \in \mathbb{H}_1$:

$$|(\mathbf{g} \circ \mathbf{q}_1)^\top (\mathbf{g} \circ \mathbf{q}_2)| = |\mathbf{q}_1^\top \mathbf{G}^\top \mathbf{G} \mathbf{q}_2| \tag{B.8}$$
$$= |\mathbf{q}_1^\top \mathbf{I} \mathbf{q}_2| \tag{B.9}$$
$$= |\mathbf{q}_1^\top \mathbf{q}_2|, \tag{B.10}$$

where $\mathbf{g} \circ \mathbf{q} \equiv \mathbf{G}\mathbf{q}$. The result is a direct consequence of the orthonormality of $\mathbf{G}$.

□

B.2. Proof of Lemma 1

We will begin by recalling some preliminary definitions and results that aid us to construct the connection between the dynamic routing and the Weiszfeld algorithm.

Definition 14 (Affine Subspace)
A d-dimensional affine subspace of R^N is obtained by a translation of a d-dimensional linear subspace $V \subset \mathbb{R}^N$ such that the origin is included in S:

$$S = \Big\{ \sum_{i=1}^{d+1} \alpha_i \mathbf{x}_i \,|\, \sum_{i=1}^{d+1} \alpha_i = 1 \Big\}. \tag{B.11}$$

Simplest choices for S involve points, lines and planes of the Euclidean space.

Definition 15 (Orthogonal Projection onto an Affine Subspace)
An orthogonal projection of a point $\mathbf{x} \in \mathbb{R}^N$ onto an affine subspace explained by the pair $(\mathbf{A}, \mathbf{c})$ is defined as:

$$\Pi_i(\mathbf{x}) \triangleq proj_S(\mathbf{x}) = \mathbf{c} + \mathbf{A}(\mathbf{x} - \mathbf{c}). \tag{B.12}$$

c *denotes the translation to make origin inclusive and* **A** *is a projection matrix typically defined via the orthonormal bases of the subspace.*

Definition 16 (Distance to Affine Subspaces)
Distance from a given point $\mathbf{x}$ *to a set of affine subspaces* $\{S_1, S_2 \dots S_k\}$ *can be written as [7]:*

$$C(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, S_i) = \sum_{i=1}^{k} \|\mathbf{x} - proj_{S_i}(\mathbf{x})\|^2. \tag{B.13}$$

Lemma 2
Given that all the antipodal counterparts are mapped to the northern hemisphere, we will now think of the unit quaternion or versor as the unit normal of a four dimensional hyperplane h, passing through the origin:

$$h_i(\mathbf{x}) = \mathbf{q}_i^\top \mathbf{x} + q_d := 0. \tag{B.14}$$

q_d *is an added term to compensate for the shift. When* $q_d = 0$ *the origin is incident to the hyperplane. With this perspective, quaternion* $\mathbf{q}_i$ *forms an affine subspace with* $d = 4$*, for which the projection operator takes the form:*

$$proj_{S_i}(\mathbf{p}) = (\mathbf{I} - \mathbf{q}_i \mathbf{q}_i^\top)\mathbf{p} \tag{B.15}$$

Proof. We consider Eq (B.15) for the case where $\mathbf{c} = \mathbf{0}$ and $\mathbf{A} = (\mathbf{I} - \mathbf{q}\mathbf{q}^\top)$. The former follows from the fact that our subspaces by construction pass through the origin. Thus, we only need to show that the matrix $\mathbf{A} = \mathbf{I} - \mathbf{q}\mathbf{q}^\top$ is an orthogonal projection matrix onto the affine subspace spanned by $\mathbf{q}$. To this end, it is sufficient to validate that $\mathbf{A}$ is symmetric and idempotent: $\mathbf{A}^\top\mathbf{A} = \mathbf{A}\mathbf{A} = \mathbf{A}^2 = \mathbf{A}$. Note that by construction $\mathbf{q}^\top\mathbf{q}$ is a symmetric matrix and hence $\mathbf{A}$ itself. Using this property and the unit-ness of the quaternion, we arrive at the proof:

$$\begin{aligned}
\mathbf{A}^\top\mathbf{A} &= (\mathbf{I} - \mathbf{q}\mathbf{q}^\top)^\top(\mathbf{I} - \mathbf{q}\mathbf{q}^\top) && \text{(B.16)}\\
&= (\mathbf{I} - \mathbf{q}\mathbf{q}^\top)(\mathbf{I} - \mathbf{q}\mathbf{q}^\top) && \text{(B.17)}\\
&= \mathbf{I} - 2\mathbf{q}\mathbf{q}^\top + \mathbf{q}\mathbf{q}^\top\mathbf{q}\mathbf{q}^\top && \text{(B.18)}\\
&= \mathbf{I} - 2\mathbf{q}\mathbf{q}^\top + \mathbf{q}\mathbf{q}^\top && \text{(B.19)}\\
&= \mathbf{I} - \mathbf{q}\mathbf{q}^\top \triangleq \mathbf{A} && \text{(B.20)}
\end{aligned}$$

It is easy to verify that the projections are orthogonal to the quaternion that defines the subspace by showing $\text{proj}_S(\mathbf{q})^\top\mathbf{q} = 0$:

$$\mathbf{q}^\top \text{proj}_S(\mathbf{q}) = \mathbf{q}^\top\mathbf{A}\mathbf{q} = \mathbf{q}^\top(\mathbf{I} - \mathbf{q}\mathbf{q}^\top)\mathbf{q} = \mathbf{q}^\top(\mathbf{q} - \mathbf{q}\mathbf{q}^\top\mathbf{q}) = \mathbf{q}^\top(\mathbf{q} - \mathbf{q}) = 0. \tag{B.21}$$
$$\tag{B.22}$$

Also note that this choice corresponds to $\text{tr}(\mathbf{q}\mathbf{q}^\top) = \sum_{i=1}^{d+1} \alpha_i = 1$. □

Lemma 3
The quaternion mean we suggest to use [119] is equivalent to the Euclidean Weiszfeld mean on the affine quaternion sub-spaces.

Proof. We now recall and summarize the L_q-Weiszfeld Algorithm on affine subspaces [7], which minimizes a q-norm variant of the cost defined in Eq (B.13):

$$C_q(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, S_i) = \sum_{i=1}^{k} \|\mathbf{x} - \text{proj}_{S_i}(\mathbf{x})\|^q. \tag{B.23}$$

Defining $\mathbf{M}_i = \mathbf{I} - \mathbf{A}_i$, Alg. 3 summarizes the iterative procedure.

Algorithm 3: L_q Weiszfeld Algorithm on Affine Subspaces [7].

input : An initial guess $\mathbf{x}_0$ that does not lie any of the subspaces $\{S_i\}$, Projection operators Π_i, the norm parameter q

$\mathbf{x}^t \leftarrow \mathbf{x}_0$

while *not converged* **do**

Compute the weights $\mathbf{w}^t = \{w_i^t\}$:

$$w_i^t = \|\mathbf{M}_i(\mathbf{x}^t - \mathbf{c}_i)\|^{q-2} \quad \forall i = 1 \dots k \tag{B.24}$$

Solve:

$$\mathbf{x}^{t+1} = \underset{\mathbf{x} \in \mathbb{R}^N}{\arg\min} \sum_{i=1}^{k} w_i^t \|\mathbf{M}_i(\mathbf{x} - \mathbf{c}_i)\|^2 \tag{B.25}$$

Note that when $q = 2$, the algorithm reduces to the computation of a non-weighted mean ($w_i = 1 \, \forall i$), and a closed form solution exists for Eq (B.25) and is given by the normal equations:

$$\mathbf{x} = \Big(\sum_{i=1}^{k} w_i \mathbf{M}_i \Big)^{-1} \Big(\sum_{i=1}^{k} w_i \mathbf{M}_i \mathbf{c}_i \Big) \tag{B.26}$$

For the case of our quaternionic subspaces $\mathbf{c} = \mathbf{0}$ and we seek the solution that satisfies:

$$\Big(\sum_{i=1}^{k} \mathbf{M}_i \Big) \mathbf{x} = \Big(\frac{1}{k} \sum_{i=1}^{k} \mathbf{M}_i \Big) \mathbf{x} = \mathbf{0}. \tag{B.27}$$

It is well known that the solution to this equation under the constraint $\|\mathbf{x}\| = 1$ lies in nullspace of $\mathbf{M} = \frac{1}{k}\sum_{i=1}^{k}\mathbf{M}_i$ and can be obtained by taking the singular vector of $\mathbf{M}$ that corresponds to the largest singular value. Since $\mathbf{M}_i$ is idempotent, the same result can also be obtained through the eigendecomposition:

$$\mathbf{q}^\star = \arg\max_{\mathbf{q}\in\mathcal{S}^3} \mathbf{q}\mathbf{M}\mathbf{q} \tag{B.28}$$

which gives us the unweighted Quaternion mean [119]. □

B.3. Proof of Theorem 1

Once the Lemma 1 is proven, we only need to apply the direct convergence results from the literature. Consider a set of points $\mathbf{Y} = \{\mathbf{y}_1 \dots \mathbf{y}_K\}$ where $K > 2$ and $\mathbf{y}_i \in \mathbb{H}_1$. Due to the compactness, we can speak of a ball $\mathcal{B}(\mathbf{o}, \rho)$ encapsulating all $\mathbf{y}_i$. We also define the $\mathcal{D} = \{\mathbf{x} \in \mathbb{H}_1 \,|\, C_q(\mathbf{x}) < C_q(\mathbf{o})\}$, the region where the loss decreases.

We first state the assumptions that permit our theoretical result. These assumptions are required by the works that establish the convergence of such Weiszfeld algorithms [142, 124, 122] :

H1. $\mathbf{y}_1 \dots \mathbf{y}_K$ should not lie on a single geodesic of the quaternion manifold.
H2. $\mathcal{D}$ is bounded and compact. The topological structure of $SO(3)$ imposes a bounded convexity radius of $\rho < \pi/2$.
H3. The minimizer in Eq (B.25) is continuous.
H4. The weighting function $\sigma(\cdot)$ is concave and differentiable.
H5. Initial quaternion (in our network chosen randomly) does not belong to any of the subspaces.

Note that **H5** is not a strict requirement as there are multiple ways to circumvent (simplest being a re-initialization). Under these assumptions, the sequence produced by Eq (B.25) will converge to a critical point unless $\mathbf{x}^t = \mathbf{y}_i$ for any t and i [122]. For $q = 1$, this critical point is on one of the subspaces specified in Eq (B.14) and thus is a geometric median. □

Note that due to the assumption **H2**, we cannot converge from any given point. For randomly initialized networks this is indeed a problem and does not guarantee practical convergence. Yet, in our experiments we have not observed any issue with the convergence of our dynamic routing. As our result is one of the few ones related to the analysis of DR, we still find this to be an important first step.

For different choices of $q : 1 \leq q \leq 2$, the weights take different forms. In fact, this IRLS type of algorithm is shown to converge for a larger class of weighting choices as long as the aforementioned conditions are met. That is why in practice we use a simple sigmoid function.

B.4. Our Siamese Architecture and The Algorithm

For estimation of the relative pose with supervision, we benefit from a Siamese variation of our network. In this case, latent capsule representations of two point sets $\mathbf{X}$ and $\mathbf{Y}$ jointly contribute to the pose regression as shown in Fig. B.1.

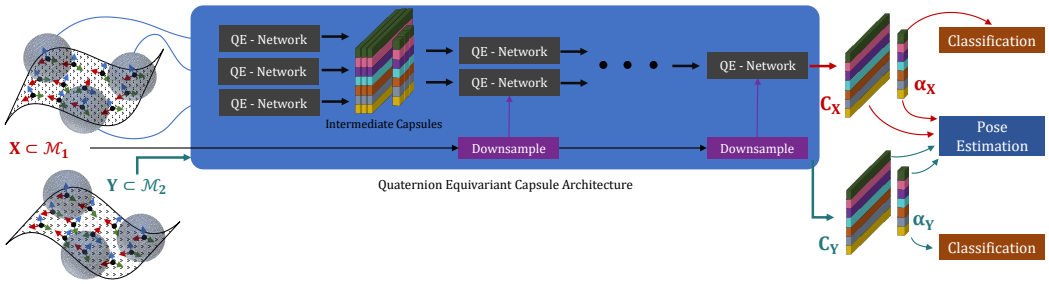


Figure B.1.: Our siamese architecture used in the estimation of relative poses. We use a shared network to process two distinct point clouds $(\mathbf{X}, \mathbf{Y})$ to arrive at the latent representations $(\mathbf{C}_X, \boldsymbol{\alpha}_X)$ and $(\mathbf{C}_Y, \boldsymbol{\alpha}_Y)$ respectively. We then look for the highest activated capsules in both point sets and compute the rotation from the corresponding capsules. Thanks to the rotations disentangled into capsules, this final step simplifies to a relative quaternion calculation.

We show additional results from the computation of local reference frames and the multi-channel capsules deduced from our network in Fig. B.2.

Finally, the overall algorithm of our network is summarized under Alg. 4.

Alg. 4 summarizes the overall pipeline of our QE-net depicted in Fig. 4.3. We use multiple layers in a hierarchical architecture. In the first layer, the input primary capsules are represented by LRFs computed with FLARE algorithm [134]. Therefore, the number of input capsule channels N^c in the first layer is equal to 1. Its activation is also defaulted to 1. The output of a former layer is propagated to the input of the latter, creating the hierarchy.

Algorithm 4: Quaternion Equivariant Network

input : Input points of one patch $\{\mathbf{x}_1, ..., \mathbf{x}_K\} \in \mathbb{R}^{K\times 3}$, input capsules (LRFs) $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_L\} \in {\mathbb{H}_1}^L$, with $L = N^c \cdot K$, N^c is the number of capsules per point, activations $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_L)^T$

output : Updated frames $\hat{\mathcal{Q}} = \{\hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_M\} \in {\mathbb{H}_1}^M$, updated activations $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_M)^T$

for *Each input channel* n^c *of all the primary capsules channels* N^c **do**
  $\mu(n^c) \leftarrow \mathcal{A}(\mathcal{Q}(n^c))$ `// Input quaternion average, see Eq (4.6)`
  for *Each point* $\mathbf{x}_i$ *of this patch* **do**
    $\mathbf{x}'_i \leftarrow \mu(n^c)^{-1} \circ \mathbf{x}_i$ `// rotate point in a canonical orientation`
$\{\mathbf{x}'_i\} \in \mathbb{R}^{K\times N^c\times 3}$ `// Points in multiple(`N^c`) canonical frames`
for *Each point* $\mathbf{x}'_i$ *of this patch* **do**
  $\mathbf{t} \leftarrow t(\mathbf{x}'_i)$ `// Point to Transform,` $t(\cdot) : \mathbb{R}^{N^c\times 3} \rightarrow \mathbb{R}^{N^c\times M\times 4}$
$\mathbf{T} \equiv \{\mathbf{t}_i\} \in {\mathbb{H}_1}^{K\times N_i^c\times M} \leftarrow \{\mathbf{t}\} \in {\mathbb{H}_1}^{L\times M}$
$(\hat{\mathcal{Q}}, \hat{\boldsymbol{\alpha}}) \leftarrow \text{DynamicRouting}(X, \mathcal{Q}, \boldsymbol{\alpha}, \mathbf{T})$ `// see Alg. 2`

B.5. Additional Details on Evaluations

Details on the evaluation protocol. For Modelnet40 dataset used in Tab. 4.1, we used the official split with 9,843 shapes for training and 2,468 different shapes for testing. For rotation estimation in Tab. 4.2, we used the official Modelenet10 dataset split with 3991 for training and 908 shapes for testing. 3D point clouds (10K points) are randomly sampled from the mesh surfaces of each shape [5, 6]. The objects in training and testing dataset are different, but they are from the same categories so that they can be oriented meaningfully. During training, we did not augment the dataset with random rotations. All the shapes are trained with single orientation (well-aligned). We call this *trained with NR*. During testing, we randomly generate multiple arbitrary $SO(3)$ rotations for each shape and evaluate the average performance for all the rotations. This is called *test with AR*. This protocol is used in both our algorithms and the baselines.

Confusion of classification in ModelNet. We now report the confusion matrix in the task of classification on the all the objects of ModelNet10. The classification and rotation estimation affects one another. As we can see from Fig. B.3, the first five categories that exhibit less rotational symmetry has the higher classification accuracy than their

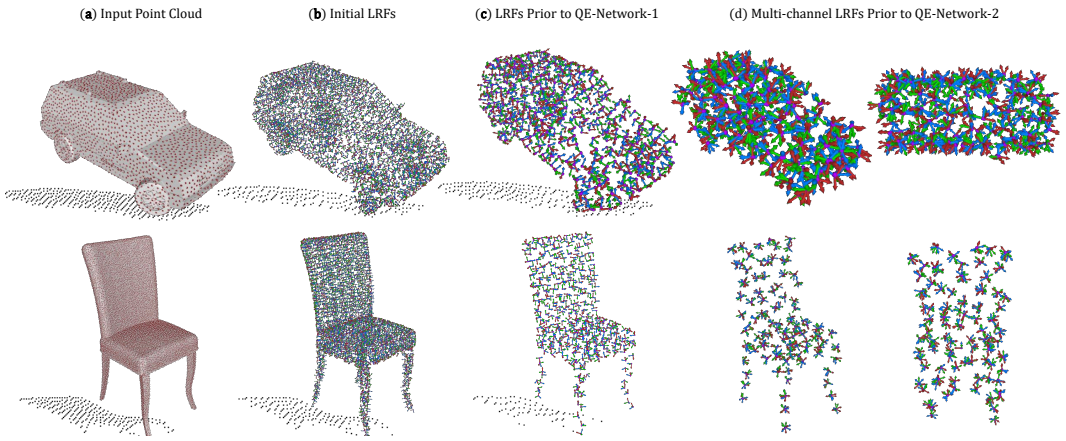


Figure B.2.: Additional intermediate results on car (first row) and chair (second row) objects. This figure supplements Fig. 4.1.

rotationally symmetric counterparts.

Distribution of errors reported in Tab. 4.2. We now provide more details on the errors attained by our algorithm as well as the state of the art. To this end, we report, in Fig. B.4 the histogram of errors that fall within quantized ranges of orientation errors. It is noticeable that our Siamese architecture behaves best in terms of estimating the objects rotation. For completeness, we also included the results of the variants presented in our ablation studies: Ours-2kLRF, Ours-1kLRF. They evaluate the model on the re-calculated LRFs in order to show the robustness towards to various point densities. We have also modified IT-Net and PointNetLK only to predict rotation because the original works predict both rotations and translations. Finally, note here that we do not use data augmentation for training our networks (see AR), while both for PointNetLK and for IT-Net we do use augmentation.

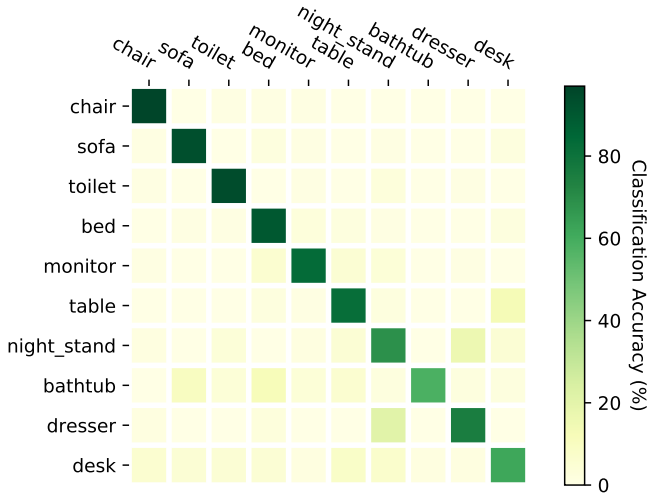
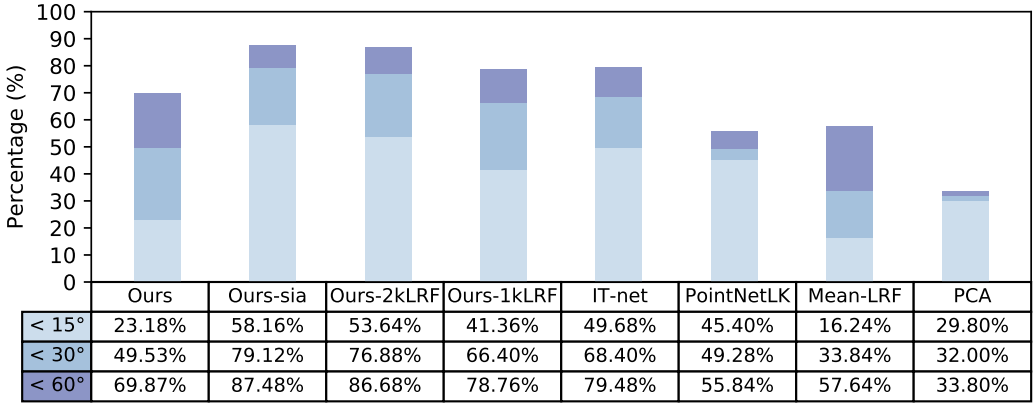


Figure B.3.: Confusion matrix on ModelNet10 for classification.



	Ours	Ours-sia	Ours-2kLRF	Ours-1kLRF	IT-net	PointNetLK	Mean-LRF	PCA
< 15°	23.18%	58.16%	53.64%	41.36%	49.68%	45.40%	16.24%	29.80%
< 30°	49.53%	79.12%	76.88%	66.40%	68.40%	49.28%	33.84%	32.00%
< 60°	69.87%	87.48%	86.68%	78.76%	79.48%	55.84%	57.64%	33.80%

Figure B.4.: Cumulative error histograms of rotation estimation on ModelNet10. Each row ($< \theta^\circ$) of this extended table shows the percentage of shapes that have rotation error less than θ. The colors of the bars correspond to the rows they reside in. The higher the errors are contained in the first bins (light blue) the better. Vice versa, the more the errors are clustered toward the 60° the worse the performance of the method.

List of Figures

List of Tables

Bibliography

[1] S. Sabour, N. Frosst, and G. E. Hinton. “Dynamic routing between capsules”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 3856–3866.

[2] L. Spinello and K. O. Arras. “People detection in RGB-D data”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 3838–3843.

[3] J. Xiao, R. Stolkin, Y. Gao, and A. Leonardis. “Robust Fusion of Color and Depth Data for RGB-D Target Tracking Using Adaptive Range-Invariant Depth Models and Spatio-Temporal Consistency Constraints”. In: *IEEE transactions on cybernetics* (2017).

[4] S. Song and J. Xiao. “Tracking revisited using RGBD camera: Unified benchmark and baselines”. In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE. 2013, pp. 233–240.

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660.

[6] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in neural information processing systems*. 2017, pp. 5099–5108.

[7] K. Aftab, R. Hartley, and J. Trumpf. “L_q Closest-Point to Affine Subspaces Using the Generalized Weiszfeld Algorithm”. In: *International Journal of Computer Vision* 114.1 (2015), pp. 1–15.

[8] Y. Zhao, M. Carraro, M. Munaro, and E. Menegatti. “Robust multiple object tracking in RGB-D camera networks”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 6625–6632.

[9] K. Meshgi, S.-i. Maeda, S. Oba, H. Skibbe, Y.-z. Li, and S. Ishii. “An occlusion-aware particle filter tracker to handle complex and persistent occlusions”. In: *Computer Vision and Image Understanding* 150 (2016), pp. 81–94.

[10] S. Hannuna, M. Camplani, J. Hall, M. Mirmehdi, D. Damen, T. Burghardt, A. Paiement, and L. Tao. "DS-KCF: a real-time tracker for RGB-D data". In: *Journal of Real-Time Image Processing* (2016), pp. 1–20.

[11] M. Camplani, S. L. Hannuna, M. Mirmehdi, D. Damen, A. Paiement, L. Tao, and T. Burghardt. "Real-time RGB-D Tracking with Depth Scaling Kernelised Correlation Filters and Occlusion Handling." In: *BMVC*. 2015, pp. 145–1.

[12] Q. Wang, J. Fang, and Y. Yuan. "Multi-cue based tracking". In: *Neurocomputing* 131 (2014), pp. 227–236.

[13] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr. "Staple: Complementary learners for real-time tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1401–1409.

[14] D. Comaniciu, V. Ramesh, and P. Meer. "Kernel-based object tracking". In: *IEEE Transactions on pattern analysis and machine intelligence* 25.5 (2003), pp. 564–577.

[15] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr. "Struck: Structured output tracking with kernels". In: *IEEE transactions on pattern analysis and machine intelligence* 38.10 (2016), pp. 2096–2109.

[16] H. Possegger, T. Mauthner, and H. Bischof. "In defense of color-based model-free tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2113–2120.

[17] T. Vojir, J. Noskova, and J. Matas. "Robust scale-adaptive mean-shift for tracking". In: *Pattern Recognition Letters* 49 (2014), pp. 250–258.

[18] F. LIRIS. "The Visual Object Tracking VOT2014 challenge results". In: ().

[19] Y. Zhou and O. Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[20] M. Naseer, S. Khan, and F. Porikli. "Indoor Scene Understanding in 2.5/3D for Autonomous Agents: A Survey". In: *IEEE Access* 7 (2019), pp. 1859–1887.

[21] H. Lei, N. Akhtar, and A. Mian. "Spherical Convolutional Neural Network for 3D Point Clouds". In: *arXiv preprint arXiv:1805.07872* (2018).

[22] P. Hermosilla, T. Ritschel, P.-P. Vázquez, À. Vinacua, and T. Ropinski. "Monte Carlo convolution for learning on non-uniformly sampled point clouds". In: *SIGGRAPH Asia 2018 Technical Papers*. ACM. 2018, p. 235.

[23] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. “Pointcnn: Convolution on x-transformed points”. In: *Advances in Neural Information Processing Systems*. 2018.

[24] Y. Shen, C. Feng, Y. Yang, and D. Tian. “Mining point cloud local structures by kernel correlation and graph pooling”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4548–4557.

[25] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. “Dynamic graph cnn for learning on point clouds”. In: *arXiv preprint arXiv:1801.07829* (2018).

[26] J. Li, B. M. Chen, and G. H. Lee. “SO-Net: Self-Organizing Network for Point Cloud Analysis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9397–9406.

[27] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. “Deep Sets”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. 2017.

[28] S. H. Rezatofighi, A. Milan, E. Abbasnejad, A. Dick, I. Reid, et al. “DeepSetNet: Predicting sets with deep neural networks”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 5257–5266.

[29] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. “Pu-net: Point cloud upsampling network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2790–2799.

[30] Y. Yang, C. Feng, Y. Shen, and D. Tian. “FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[31] H. Deng, T. Birdal, and S. Ilic. “PPF-FoldNet: Unsupervised Learning of Rotation Invariant 3D Local Descriptors”. In: *The European Conference on Computer Vision (ECCV)*. Sept. 2018.

[32] H. Deng, T. Birdal, and S. Ilic. “Ppfnet: Global context aware local features for robust 3d point matching”. In: *Conference on Computer Vision and Pattern Recognition*. 2018.

[33] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry. "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation". In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[34] Y. Sun, Y. Wang, Z. Liu, J. E. Siegel, and S. E. Sarma. "PointGrow: Autoregressively Learned Point Cloud Generation with Self-Attention". In: *arXiv preprint arXiv:1810.05591* (2018).

[35] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. "Learning Representations and Generative Models for 3D Point Clouds". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 40–49.

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

[37] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. "Volumetric and multi-view cnns for object classification on 3d data". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5648–5656.

[38] E. Bao and L. Song. "Equivariant neural networks and equivarification". In: *arXiv preprint arXiv:1906.07172* (2019).

[39] R. Kondor and S. Trivedi. "On the generalization of equivariance and convolution in neural networks to the action of compact groups". In: *arXiv preprint arXiv:1802.03690* (2018).

[40] T. Cohen and M. Welling. "Group equivariant convolutional networks". In: *International conference on machine learning*. 2016, pp. 2990–2999.

[41] T. S. Cohen and M. Welling. "Steerable cnns". In: *International Conference on Learning Representations (ICLR)* (2017).

[42] D. Worrall and G. Brostow. "CubeNet: Equivariance to 3D Rotation and Translation". In: *The European Conference on Computer Vision (ECCV)*. Sept. 2018.

[43] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. "Harmonic Networks: Deep Translation and Rotation Equivariance". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[44] M. Weiler, F. A. Hamprecht, and M. Storath. "Learning Steerable Filters for Rotation Equivariant CNNs". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[45] R. Chakraborty, M. Banerjee, and B. C. Vemuri. "H-CNNs: Convolutional neural networks for riemannian homogeneous spaces". In: *arXiv preprint arXiv:1805.05487* (2018).

[46] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. "Spherical CNNs". In: (2018).

[47] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis. "Learning so (3) equivariant representations with spherical cnns". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 52–68.

[48] J. Cruz-Mota, I. Bogdanova, B. Paquier, M. Bierlaire, and J.-P. Thiran. "Scale Invariant Feature Transform on the Sphere: Theory and Applications". In: *International Journal of Computer Vision* 98.2 (June 2012), pp. 217–241.

[49] C. M. Jiang, J. Huang, K. Kashinath, Prabhat, P. Marcus, and M. Niessner. "Spherical CNNs on Unstructured Grids". In: *International Conference on Learning Representations*. 2019.

[50] M. Liu, F. Yao, C. Choi, S. Ayan, and K. Ramani. "Deep Learning 3D Shapes Using Alt-az Anisotropic 2-Sphere Convolution". In: *International Conference on Learning Representations (ICLR)*. 2019.

[51] W. Boomsma and J. Frellsen. "Spherical convolutions and their application in molecular modelling". In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 3433–3443.

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

[53] M. Khoury, Q.-Y. Zhou, and V. Koltun. "Learning compact geometric features". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 153–161.

[54] E. Mehr, A. Lieutier, F. Sanchez Bermudez, V. Guitteny, N. Thome, and M. Cord. "Manifold learning in quotient spaces". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9165–9174.

[55] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. "Tensor field networks: Rotation-and translation-equivariant neural networks for 3D point clouds". In: *arXiv preprint arXiv:1802.08219* (2018).

[56] R. Spezialetti, S. Salti, and L. Di Stefano. "Learning an Effective Equivariant 3D Descriptor Without Supervision". In: *arXiv preprint arXiv:1909.06887* (2019).

[57] C. Esteves, Y. Xu, C. Allen-Blanchette, and K. Daniilidis. “Equivariant Multi-View Networks”. In: *arXiv preprint arXiv:1904.00993* (2019).

[58] C. Esteves, A. Sud, Z. Luo, K. Daniilidis, and A. Makadia. “Cross-Domain 3D Equivariant Image Embeddings”. In: *International Conference on Machine Learning (ICML)*. 2019.

[59] D. Marcos, M. Volpi, N. Komodakis, and D. Tuia. “Rotation Equivariant Vector Field Networks”. In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[60] G. E. Hinton, A. Krizhevsky, and S. D. Wang. “Transforming auto-encoders”. In: *International Conference on Artificial Neural Networks*. Springer. 2011, pp. 44–51.

[61] R. LaLonde and U. Bagci. “Capsules for Object Segmentation”. In: *arXiv preprint arXiv:1804.04241* (2018).

[62] K. Duarte, Y. Rawat, and M. Shah. “Videocapsulenet: A simplified network for action detection”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 7621–7630.

[63] A. Jaiswal, W. AbdAlmageed, Y. Wu, and P. Natarajan. “Capsulegan: Generative adversarial capsule network”. In: *European Conference on Computer Vision*. Springer. 2018, pp. 526–535.

[64] A. Jaiswal, W. AbdAlmageed, Y. Wu, and P. Natarajan. “CapsuleGAN: Generative Adversarial Capsule Network”. In: *Computer Vision – ECCV 2018 Workshops*. Springer International Publishing, 2019, pp. 526–535.

[65] R. Saqur and S. Vivona. “CapsGAN: Using Dynamic Routing for Generative Adversarial Networks”. In: *arXiv preprint arXiv:1806.03968* (2018).

[66] Y. Upadhyay and P. Schrater. “Generative Adversarial Network Architectures For Image Synthesis Using Capsule Networks”. In: *arXiv preprint arXiv:1806.03796* (2018).

[67] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[68] A. Lin, J. Li, and Z. Ma. “On Learning and Learned Representation with Dynamic Routing in Capsule Networks”. In: *arXiv preprint arXiv:1810.04041* (2018).

[69] Z. Xinyi and L. Chen. "Capsule Graph Neural Network". In: *International Conference on Learning Representations (ICLR)*. 2019. URL: https://openreview.net/forum?id=Byl8BnRcYm.

[70] P. Afshar, A. Mohammadi, and K. N. Plataniotis. "Brain Tumor Type Classification via Capsule Networks". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. 2018.

[71] G. Hinton, S. Sabour, and N. Frosst. "Matrix capsules with EM routing". In: *ICLR 2018 Conference Blind Submission*. 2018. URL: https://openreview.net/pdf?id=HJWLfGWRb.

[72] D. Wang and Q. Liu. "An optimization view on dynamic routing between capsules". In: *ICLR Workshop Submission* (2018).

[73] Z. Chen and D. Crandall. "Generalized Capsule Networks with Trainable Routing Procedure". In: *arXiv preprint arXiv:1808.08692* (2018).

[74] S. Zhang, Q. Zhou, and X. Wu. "Fast Dynamic Routing Based on Weighted Kernel Density Estimation". In: *Cognitive Internet of Things: Frameworks, Tools and Applications*. Springer International Publishing, 2020.

[75] L. Zhang, M. Edraki, and G.-J. Qi. "CapProNet: Deep feature learning via orthogonal projections onto capsule subspaces". In: *Advances in Neural Information Processing Systems*. 2018, pp. 5819–5828.

[76] J. E. Lenssen, M. Fey, and P. Libuschewski. "Group equivariant capsule networks". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8858–8867.

[77] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen. "3d steerable cnns: Learning rotationally equivariant features in volumetric data". In: *Advances in Neural Information Processing Systems*. 2018, pp. 10402–10413.

[78] A. Jiménez-Sánchez, S. Albarqouni, and D. Mateus. "Capsule networks against medical imaging data challenges". In: *Intravascular Imaging and Computer Assisted Stenting and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*. Springer, 2018, pp. 150–160.

[79] A. Mobiny and H. Van Nguyen. "Fast capsNet for lung cancer screening". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 741–749.

[80] Y. Zhao, T. Birdal, H. Deng, and F. Tombari. "3D Point Capsule Networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[81] Y. Zhao and E. Menegatti. "MS3D: Mean-Shift Object Tracking Boosted by Joint Back Projection of Color and Depth". In: *International Conference on Intelligent Autonomous Systems*. Springer. 2018, pp. 222–236.

[82] D. Comaniciu, V. Ramesh, and P. Meer. "Real-time tracking of non-rigid objects using mean shift". In: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. Vol. 2. IEEE. 2000, pp. 142–149.

[83] P. Hidayatullah and H. Konik. "CAMSHIFT improvement on multi-hue object and multi-object tracking". In: *Visual Information Processing (EUVIP), 2011 3rd European Workshop on*. IEEE. 2011, pp. 143–148.

[84] G. R. Bradski. "Real time face and object tracking as a component of a perceptual user interface". In: *Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on*. IEEE. 1998, pp. 214–219.

[85] Y. Wu, J. Lim, and M.-H. Yang. "Object tracking benchmark". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1834–1848.

[86] A. Lukežič, L. Č. Zajc, T. Vojíř, J. Matas, and M. Kristan. "Now you see me: evaluating performance in long-term visual tracking". In: *arXiv preprint arXiv:1804.07056* (2018).

[87] M. Munaro, A. Horn, R. Illum, J. Burke, and R. B. Rusu. "OpenPTrack: People tracking for heterogeneous networks of color-depth cameras". In: *IAS-13 Workshop Proceedings: 1st Intl. Workshop on 3D Robot Perception with Point Cloud Library*. 2014, pp. 235–247.

[88] M. Munaro, F. Basso, and E. Menegatti. "OpenPTrack: Open source multi-camera calibration and people tracking for RGB-D camera networks". In: *Robotics and Autonomous Systems* 75 (2016), pp. 525–538.

[89] N. Bellotto and H. Hu. "Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of bayesian filters". In: *Autonomous Robots* 28.4 (2010), pp. 425–438.

[90] F. Basso, A. Pretto, and E. Menegatti. "Unsupervised intrinsic and extrinsic calibration of a camera-depth sensor couple". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 6244–6249.

[91] F. Basso, E. Menegatti, and A. Pretto. “Robust intrinsic and extrinsic calibration of rgb-d cameras”. In: *IEEE Transactions on Robotics* 99 (2018), pp. 01–18.

[92] T. Wiedemeyer. *IAI Kinect2*. `https://github.com/code-iai/iai_kinect2`. Accessed June 12, 2015. University Bremen: Institute for Artificial Intelligence, 2014 – 2015.

[93] M. Antonello, A. Gobbi, S. Michieletto, S. Ghidoni, and E. Menegatti. “A fully automatic hand-eye calibration system”. In: *2017 European Conference on Mobile Robots (ECMR)*. IEEE. 2017, pp. 1–6.

[94] B. Babenko, M.-H. Yang, and S. Belongie. “Robust object tracking with online multiple instance learning”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.8 (2011), pp. 1619–1632.

[95] Y. Wu, J. Lim, and M.-H. Yang. “Online object tracking: A benchmark”. In: *Computer vision and pattern recognition (CVPR), 2013 IEEE Conference on*. Ieee. 2013, pp. 2411–2418.

[96] G. Riegler, O. Ulusoy, and A. Geiger. “OctNet: Learning Deep 3D Representations at High Resolutions”. In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[97] R. Bellman. *Dynamic programming*. Courier Corporation, 2013.

[98] M. Sung, H. Su, R. Yu, and L. Guibas. “Deep Functional Dictionaries: Learning Consistent Semantic Structures on 3D Models from Functions”. In: *NIPS*. 2018.

[99] Y. Rubner, C. Tomasi, and L. J. Guibas. “The earth mover’s distance as a metric for image retrieval”. In: *International journal of computer vision* 40.2 (2000), pp. 99–121.

[100] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *NIPS*. 2017.

[101] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, L. Guibas, et al. “A scalable active framework for region annotation in 3d shape collections”. In: *ACM Transactions on Graphics (TOG)* (2016).

[102] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv preprint arXiv:1512.03012* (2015).

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

[104] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. "3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions". In: *CVPR*. 2017.

[105] M. Khoury, Q.-Y. Zhou, and V. Koltun. "Learning Compact Geometric Features". In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[106] T. Birdal and S. Ilic. "Point pair features based object detection and pose estimation revisited". In: *2015 International Conference on 3D Vision*. IEEE. 2015, pp. 527–535.

[107] T. Birdal and S. Ilic. "A point sampling algorithm for 3d matching of irregular geometries". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 6871–6878.

[108] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". In: *Advances in Neural Information Processing Systems*. 2016, pp. 82–90.

[109] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. "O-cnn: Octree-based convolutional neural networks for 3d shape analysis". In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), p. 72.

[110] T. S. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling. "Gauge equivariant convolutional networks and the icosahedral cnn". In: *arXiv preprint arXiv:1902.04615* (2019).

[111] C. L. Giles and T. Maxwell. "Learning, invariance, and generalization in high-order neural networks". In: *Applied optics* 26.23 (1987), pp. 4972–4978.

[112] R. Kondor, Z. Lin, and S. Trivedi. "Clebsch–gordan nets: a fully fourier space spherical convolutional neural network". In: *Advances in Neural Information Processing Systems*. 2018.

[113] T. Cohen, M. Geiger, and M. Weiler. "A General Theory of Equivariant CNNs on Homogeneous Spaces". In: *arXiv preprint arXiv:1811.02017* (2018).

[114] B. Busam, T. Birdal, and N. Navab. "Camera Pose Filtering with Local Regression Geodesics on the Riemannian Manifold of Dual Quaternions". In: *IEEE International Conference on Computer Vision Workshop (ICCVW)*. Oct. 2017.

[115] T. Birdal, U. Simsekli, M. O. Eken, and S. Ilic. "Bayesian Pose Graph Optimization via Bingham Distributions and Tempered Geodesic MCMC". In: *Advances in Neural Information Processing Systems*. 2018, pp. 308–319.

[116] N. E. Steenrod. *The topology of fibre bundles*. Vol. 14. Princeton University Press, 1951.

[117] A. Poulenard and M. Ovsjanikov. "Multi-directional geodesic neural networks via equivariant convolution". In: *SIGGRAPH Asia 2018 Technical Papers*. ACM. 2018, p. 236.

[118] A. Petrelli and L. Di Stefano. "On the repeatability of the local reference frame for partial shape matching". In: *2011 International Conference on Computer Vision*. IEEE. 2011.

[119] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman. "Averaging quaternions". In: *Journal of Guidance, Control, and Dynamics* 30.4 (2007), pp. 1193–1197.

[120] J. R. Magnus. "On differentiating eigenvalues and eigenvectors". In: *Econometric Theory* 1.2 (1985).

[121] S. Laue, M. Mitterreiter, and J. Giesen. "Computing Higher Order Derivatives of Matrix and Tensor Expressions". In: *Advances in Neural Information Processing Systems*. 2018.

[122] K. Aftab, R. Hartley, and J. Trumpf. "Generalized weiszfeld algorithms for lq optimization". In: *IEEE transactions on pattern analysis and machine intelligence* 37.4 (2014).

[123] C. S. Burrus. "Iterative reweighted least squares". In: *OpenStax CNX. Available online: http://cnx. org/contents/92b90377-2b34-49e4-b26f-7fe572db78a1* 12 (2012).

[124] K. Aftab and R. Hartley. "Convergence of iteratively re-weighted least squares to robust M-estimators". In: *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE. 2015.

[125] D. Wang and Q. Liu. *An Optimization View on Dynamic Routing Between Capsules*. 2018. URL: https://openreview.net/forum?id=HJjtFYJDf.

[126] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller. "SchNet: A continuous-filter convolutional neural network for modeling quantum interactions". In: *Advances in Neural Information Processing Systems*. 2017.

[127] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun. "Deep Parametric Continuous Convolutional Neural Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[128] M. Fey, J. Eric Lenssen, F. Weichert, and H. Müller. "SplineCNN: Fast Geometric Deep Learning With Continuous B-Spline Kernels". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[129] G. Bécigneul and O.-E. Ganea. "Riemannian adaptive optimization methods". In: *arXiv preprint arXiv:1810.00760* (2018).

[130] S. Liao, E. Gavves, and C. G. Snoek. "Spherical Regression: Learning Viewpoints, Surface Normals and 3D Rotations on n-Spheres". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9759–9767.

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

[132] S. Sabour, N. Frosst, and G. Hinton. "Matrix capsules with EM routing". In: *6th International Conference on Learning Representations, ICLR*. 2018.

[133] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. *Surface reconstruction from unorganized points*. Vol. 26.2. ACM, 1992.

[134] A. Petrelli and L. Di Stefano. "A repeatable and efficient canonical reference for surface matching". In: *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. IEEE. 2012, pp. 403–410.

[135] F. Tombari, S. Salti, and L. Di Stefano. "Unique signatures of histograms for local surface description". In: *European conference on computer vision*. Springer. 2010, pp. 356–369.

[136] S. Melzi, R. Spezialetti, F. Tombari, M. M. Bronstein, L. D. Stefano, and E. Rodola. "GFrames: Gradient-Based Local Reference Frame for 3D Shape Matching". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[137] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker. "Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 8778–8785.

[138] Y. You, Y. Lou, Q. Liu, Y.-W. Tai, W. Wang, L. Ma, and C. Lu. "PRIN: Pointwise Rotation-Invariant Network". In: *arXiv preprint arXiv:1811.09361* (2018).

[139] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey. "PointNetLK: Robust & Efficient Point Cloud Registration using PointNet". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7163–7172.

[140] W. Yuan, D. Held, C. Mertz, and M. Hebert. "Iterative Transformer Network for 3D Point Cloud". In: *arXiv preprint arXiv:1811.11209* (2018).

[141] P. J. Besl and N. D. McKay. "Method for registration of 3-D shapes". In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. International Society for Optics and Photonics. 1992, pp. 586–606.

[142] D. G. Luenberger, Y. Ye, et al. *Linear and nonlinear programming*. Springer, 1984.