TECHNISCHE UNIVERSITÄT MÜNCHEN
TUM School of Engineering and Design

Perception of vehicles and place recognition in urban environment based on MLS point clouds

Yan Xia

Dissertation

2022

TECHNISCHE UNIVERSITÄT MÜNCHEN
TUM School of Engineering and Design

Perception of vehicles and place recognition in urban environment based on MLS point clouds

Yan Xia

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:	Prof. Dr. Marco Körner
Prüfer*innen der Dissertation:	1. Prof. Dr.-Ing. Uwe Stilla
	2. Prof. Dr. Daniel Cremers
	3. Prof. Dr.-Ing. habil. Yusheng Xu Tongji Universität / Shanghai / China

Die Dissertation wurde am 14.10.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 14.03.2023 angenommen.

Abstract

As a cornerstone of achieving autonomous driving, recognition of urban street environments has drawn attention increasingly in the fields of photogrammetry, computer vision, and robotics. Point clouds, acquired via mobile laser scanning, can provide detailed 3D information during driving for recognizing the 3D urban street environments.

In this thesis, point cloud based place recognition, 3D vehicle detection and tracking, and 3D shape completion methods are developed to achieve localization of the current scene within an existing 3D map and status analysis of the dynamic vehicles during the driving. Hereby, the thesis provides contributions on the following three core aspects: (i) global descriptors for localizing the current scene, (ii) a vehicle detection and tracking method for avoiding obstacles, and (iii) shape completion for visualizing the complete shape of the object.

To design robust and discriminative global descriptors of point clouds, the local orientations among the raw 3D points and the long-range context information are encoded. The point-wise features and voxel-wise features are also fused via a transformer network to enhance the representation ability of global descriptors. In the 3D vehicle detection and tracking method, both a detector and a lightweight tracker are proposed. The spatial-temporal correlations are explicitly leveraged via a voting unit for improving tracking speed and accuracy. To complete 3D partial point clouds robustly and accurately, improving the global features, refining the coarse results, and learning prior information are investigated. The global features can be enhanced by fusing both low-level and high-level feature information. A refinement unit is introduced to retain the information of incomplete inputs. In addition, learning prior information can be achieved by designing an asymmetrical Siamese auto-encoder network.

All proposed methods were evaluated by extensive experiments with different benchmark datasets, like the Oxford RobotCar and TUM City Campus datasets. For the point cloud based place recognition, the performance on the Oxford RobotCar and TUM datasets reaches a recall of 95% and 86%, respectively. For vehicle detection and tracking, an overall detection precision of about 68% can finally be achieved on the TUM dataset and a tracking precision can be achieved up to 93% on the KITTI dataset. Notably, the tracking performance on the NuScenes dataset reaches ∼10% improvement on average compared with previous state-of-the-art methods. As for the 3D shape completion, an average Chamfer Distance of only 6.68×10^{-4} on the Completion3D benchmark is achieved.

Kurzfassung

Als Eckpfeiler für das autonome Fahren hat die Erkennung von städtischen Straßenumgebungen in den Bereichen Photogrammetrie, Computer Vision und Robotik zunehmend an Bedeutung gewonnen. Punktwolken, die durch mobiles Laserscanning erfasst werden, können während des Fahrens detaillierte 3D-Informationen für die Erkennung der 3D-Umgebung städtischer Straßen liefern.

In dieser Arbeit werden Methoden zur punktwolkenbasierten Ortserkennung, 3D-Fahrzeugerkennung und -verfolgung sowie zur 3D-Formvervollständigung entwickelt, um eine Lokalisierung der aktuellen Szene innerhalb einer bestehenden 3D-Karte und eine Statusanalyse der dynamischen Fahrzeuge während der Fahrt zu erreichen. Dabei liefert die Arbeit Beiträge zu den folgenden drei Kernaspekten: (i) globale Deskriptoren zur Lokalisierung der aktuellen Szene, (ii) ein Fahrzeugerkennungs- und -verfolgungsverfahren zur Vermeidung von Hindernissen und (iii) Formvervollständigung zur Visualisierung der vollständigen Form des Objekts.

Um robuste und aussagekräftige globale Deskriptoren für Punktwolken zu entwickeln, werden die lokalen Orientierungen zwischen den rohen 3D-Punkten und die weitreichenden Kontextinformationen erfasst. Die punktweisen Merkmale und voxelweisen Merkmale werden auch Ã¼ber ein Transformatorennetzwerk fusioniert, um die Darstellungsfähigkeit der globalen Deskriptoren zu verbessern. Für die 3D-Fahrzeugerkennung und -verfolgung werden sowohl ein Detektor als auch ein Tracker vorgeschlagen. Die räumlichen und zeitlichen Korrelationen werden über einer expliziten Zustimmeinheit, um die Verfolgungsgeschwindigkeit und -genauigkeit zu verbessern. Um 3D-Teilpunktwolken robust und genau zu vervollständigen, werden die Verbesserung der globalen Merkmale, die Verfeinerung der groben Ergebnisse und das Lernen von Vorinformationen untersucht. Die globalen Merkmale können verbessert werden, indem sowohl Low-Level- als auch High-Level-Merkmalsinformationen zusammengeführt werden. Eine Verfeinerungseinheit wird eingeführt, um Informationen aus unvollständigen Eingaben zu erhalten. Darüber hinaus kann das Erlernen von Vorwissen durch die Entwicklung eines asymmetrischen siamesischen Autoencodernetzwerks erreicht werden.

Alle vorgeschlagenen Methoden wurden durch umfangreiche Experimente mit verschiedenen Benchmark-Datensätzen, wie Oxford RobotCar und TUM City Campus Dataset, evaluiert. Bei der punktwolkenbasierten Ortserkennung erreicht die Leistung auf dem Oxford RobotCar- und dem TUM-Datensatz einen Recall von 95% bzw. 86%. Für die Erkennung und Verfolgung von Fahrzeugen kann schließlich eine Erkennungsgenauigkeit von etwa 68% im TUM-Datensatz und für die Verfolgung eine Genauigkeit bis zu 93% im KITTI-Datensatz erreicht werden. Insbesondere die Verfolgungsleistung auf den NuScenes-Datensatz erreicht eine durchschnittliche Verbesserung von 10% im Vergleich zu früheren State-of-the-Art-Methoden. Bei der Vervollständigung von 3D-Formen wird eine durchschnittliche Chamfer Distance von nur 6.68×10^{-4} im Completion3D-Benchmark erreicht.

Contents

List of Abbreviations

Abbreviation	Description	Page
RADAR	Radio Detection and Ranging	15
LiDAR	Light Detection and Ranging	15
MLS	Mobile Laser Scanning	15
GPS	Global Positioning System	16
SLAM	Simultaneous Localization and Mapping	16
DTMO	Detection and Tracking of Moving Objects	17
BEV	Bird's Eye View	22
VFE	Voxel Feature Encoder	22
RPN	Region Proposal Network	22
MLP	Multilayer Perceptions	27
SOT	Single Object Tracking	27
STN	Spatial Transform Network	27
PFE	Point Feature Enhancement	28
FPS	farthest point sampling	30
FFN	Feed-Forward Network	37
MHSA	Multi-Head Self-Attention	39
PointOE	Point Orientation Encoding	41
UTM	Universal Transverse Mercator	41
FC	Fully Connected	42
OE	Orientation Encoding	42
S8N	Stacked 8-neighborhood Search	43
HPHN	Hard Positive Hard Negative	42
HCAT	Hierarchical Cross-Attention Transformer	46
LSA	Lightweight Self-Attention	50
GeM	Generalized Mean	11
FPN	Feature Pyramid Network	46
TConv	Transposed Convolution	46
CAT	Cross Attention Transformer	47
DPSA	Dot-Product Self-Attention	49
NMS	Non Maximum Suppression	54
IoU	Intersection over Union	87
BAFF	Box-aware Feature Fusion	58
MPM	Motion Prediction Module	59
LSTM	Long Short-Term Memory	60
EVM	Explicit Voting Module	60
CD	Chamfer Distance	70
EMD	Earth Mover's Distance	70
PN	PointNet	73
INS	Inertial Navigation System	79
RTK	Real-time Kinematic	79
ECEF	Earth-centered, Earth-fixed	80
LLA	Latitude, Longitude and Altitude	80
AP	Average Precision	87
AUC	Area Under the Curve	87

Abbreviation	Description	Page
OPE	One Pass Evaluation	88
IOU	Intersection Over Union	88
FLOP	floating point operations	99
FD	Fidelity Distance	105
MMD	Minimal Matching Distance	105
ICP	Iterative Closest Point	119

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Traffic accidents are a leading source of disability and mortality worldwide. Every year, 1.2 million people are killed and up to 50 million people are injured [Stavens, 2011]. Designing autonomous or highly aware vehicles have the potential to reduce these numbers dramatically. In the past decades, working on autonomous driving has drawn many researchers and engineers. Dickmanns et al. [1994] designed a passenger car Mercedes 500 SEL equipped with a vision system for detecting and tracking obstacles. Pomerleau & Jochem [1996] explored a Ralph vision system to help automobile drivers steer. Broggi et al. [1999] developed an active safety system and an automatic pilot for a standard road vehicle. Recently, many companies (e.g. Tesla, Waymo, Mercedes-Benz, Baidu) possessed permits to test autonomous vehicles on the roads in different counties, including Germany, America, and China. However, there are still gaps between the current development of autonomous driving and the final goal of a self-driving car that outperforms human driving performance.

Typically, an autonomous system consists of a series of complicated modules: sensing, high-definition (HD) map creation, localization, perception, prediction, motion planning, and control [Chen et al., 2020]. Firstly, an HD map is created offline. At run-time, the autonomous online system will localize itself to the HD map, perceive the surrounding environment and predict the corresponding motion. Next, the motion planner makes a safe route for this self-driving car. Finally, the controller executes this route to the destination. Therefore, for navigating a self-driving car in an urban street environment, it is essential to recognize the location of the current scene and achieve object recognition in the 3D urban street environment for a moving vehicle, as shown in Fig. 1.1. To accomplish this mission, a series of sensors, such as optical cameras, radio detection and ranging (RADAR), light detection and ranging (LiDAR), and ultrasonic sensors, are commonly equipped with a mobile laser scanning (MLS) system for various recognition tasks, including the place recognition, obstacle detection, and tracking, and shape completion. All of them can be very useful for a self-driving car in order to understand its surroundings. Each of the sensors has its own pros and cons, given by physical law. To exemplify, many researchers have studied various recognition techniques via optical cameras, such as 2D image-based place recognition, 2D object detection, tracking, and image completion. The reason is those optical cameras can obtain objects' texture information and are cheap. However, when the light is poor, such as at night, cameras struggle to function because they rely on light from their surroundings. In addition, illumination and seasonal changes bring more challenges to achieving visual recognition tasks based on images.

Thus, LiDAR mounted on MLS has been considered an important technique to perceive the 3D environment. A point cloud acquired from MLS delivers detailed 3D information of road scenes during driving with a high measuring frequency, including scattered 3D point compositions, and accurate (millimeter-level) 3D positions. Moreover, geometric information of point clouds is

invariant to drastic illumination changes, making it more robust for different seasons and times of the day compared with images.

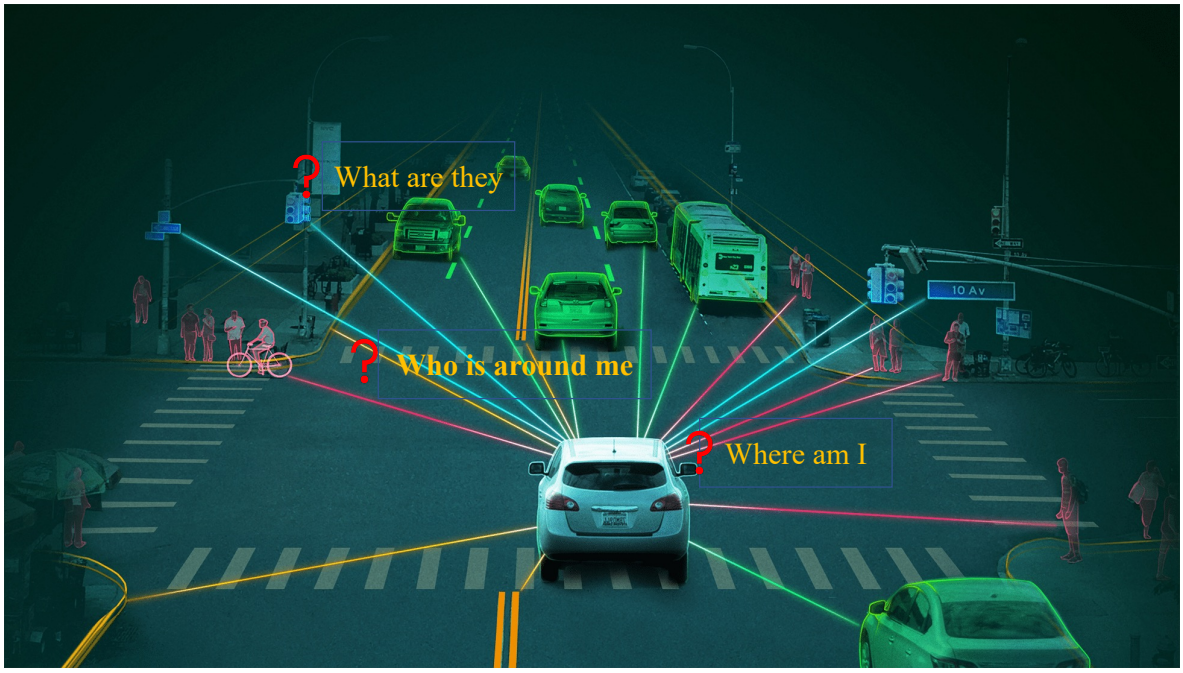


Figure 1.1: Illustration of general tasks for navigating an autonomous vehicle in an urban area [Boric et al., 2021].

'Where am I?' is the first important question for autonomous navigation. Localization is a critical capability for self-driving vehicles, allowing them to pinpoint their location on a map. Accurate localization allows a self-driving car to understand its surroundings and form a sense of the road and lane structures. The well-known technique for localization is to use positioning systems, such as the global positioning system (GPS), which is one of the most popular and powerful sensors. However, the signal of satellites can be very poor in a cluttered environment, like city canyons. Besides, due to the error sources such as receiver noise, atmospheric delays, multi-path, and satellite clock timing, the GPS solution is limited in accuracy and reliability. The positioning-based system reduces the degrees of autonomy since it requires information about the external infrastructure. As a result, sensors with greater degrees of independence from infrastructure are preferred [Panphattarasap, 2019]. The scene information processed by sensor-based solutions can be decisive in improving the localization of the features that characterize a location through place recognition techniques. Place recognition can be complementary to GPS-based proposals, or even supplementary in environments where GPS signal is not completely available or denied [Arroyo Contera et al., 2017]. Another popular localization technique is based on simultaneous localization and mapping (SLAM) [Durrant-Whyte & Bailey, 2006; Bailey & Durrant-Whyte, 2006]. However, accumulated drift error over time is inevitably introduced in SLAM systems because they are unable to rely on an external reference such as a previously provided map or explicitly known sensor poses. To address this issue, the place recognition technique assists the SLAM system in recognizing that it is currently located at a previously visited location, which corrects the drift in the map and thus improves sensor localization. In addition, place recognition techniques can be used to provide an accurate initial position for better convergence in map-based localization methods [Schlichting & Brenner, 2014].

'Who is around me?' is the next general question for autonomous navigation. Knowing where the objects lie in the currently scanned scene and how to keep track of the trajectories of these objects is helpful to avoid obstacles for self-driving vehicles. Detection and Tracking of Moving Objects (DTMO) [Moosmann & Stiller, 2013] is a typical solution, which has been widely explored to avoid collisions and traffic accidents in autonomous driving. Imaging that a self-driving car is trying to cross an intersection in heavy traffic, the car should first detect individual moving objects in its vicinity, and then predict the individual movement of the objects over time for navigating through the intersection efficiently and safely [Söderlund, 2019]. Vehicles are the most concerned investigation objects.

'What are they?' is the third general question for autonomous navigation. Although MLS point clouds provide detailed 3D information of urban scenes with high accuracy and precision, they are usually partial due to occlusion and self-occlusion. To exemplify, an object may only be visible from one side while driving. As a result, real-world 3D scans often contain large missing regions, resulting in a significant loss in geometric and semantic information. Knowing the complete shape of the object is beneficial in the case of autonomous driving: A closed surface mesh of an obstacle provides more information about its nature, allowing a self-driving car to better decide which action to take next. Many researchers have demonstrated that the complete geometric shapes of objects can provide more robust features for many 3D perceptual tasks, including the above-mentioned 3D object detection and tracking. Giancola et al. [2019] demonstrates the shape completion improves the 3D tracking performance [Geiger et al., 2013]. Xu et al. [2022] estimates the complete object shapes to help a 3D object detection model.

From the aforementioned research, three major tasks should be addressed in the recognition process using MLS point clouds (see Fig. 1.2): (1) point cloud based place recognition, (2) 3D object detection and tracking, and (3) 3D shape completion. To be specific, the location of a given 3D scan in a GPS-denied environment should be determined by querying the locations of scans belonging to the same location in a large geo-tagged database. Meanwhile, 3D detection and tracking should be conducted to recognize the locations of moving objects (e.g. vehicles) in the surrounding scenes and keep track of these objects in successive frames. Furthermore, obtaining the full shapes of the detected objects should be implemented by 3D shape completion methods.

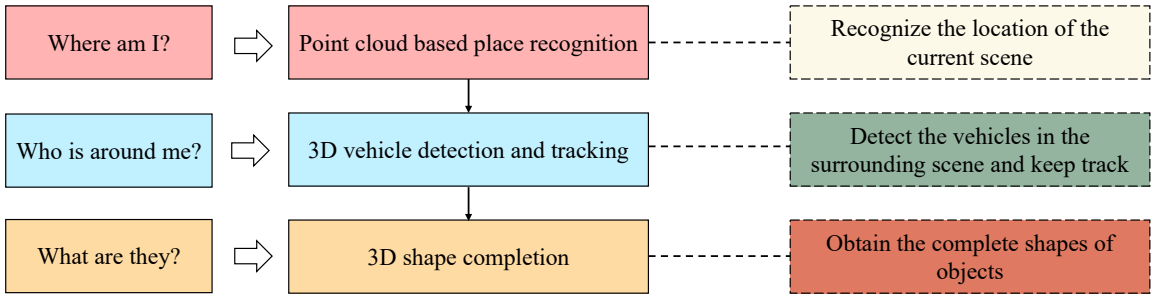


Figure 1.2: Three-step workflow for recognition of 3D urban street environment from MLS point clouds.

However, achieving the above three tasks is non-trivial (see Fig. 1.3). First, designing a robust and discriminative global descriptor for a scene point cloud in 3D urban environments is challenging since the temporal changes and incompleteness are caused by occlusions or self-occlusions. Furthermore, some locations are extremely similar, making it difficult to locate the correct location via a feature-based retrieve strategy. Second, the density of 3D point clouds in urban scenes collected by LiDAR is uneven, where the area closer to the scanner location has a much greater density than the farther-away region. That makes the extracted features

ineffectual and leads to difficulties in recognizing and tracking objects in the complex 3D scene. Besides, the sparsity of point clouds and the outliers in point clouds burden feature extraction. For example, the point cloud of a vehicle in a single scan could only have 12 points [Xia et al., 2021b]. Third, MLS point clouds are always partial due to occlusions and self-occlusions. Last but not least, completing the proper topology of missing shapes and keeping the fine-grained details is challenging only based on incomplete geometric information.

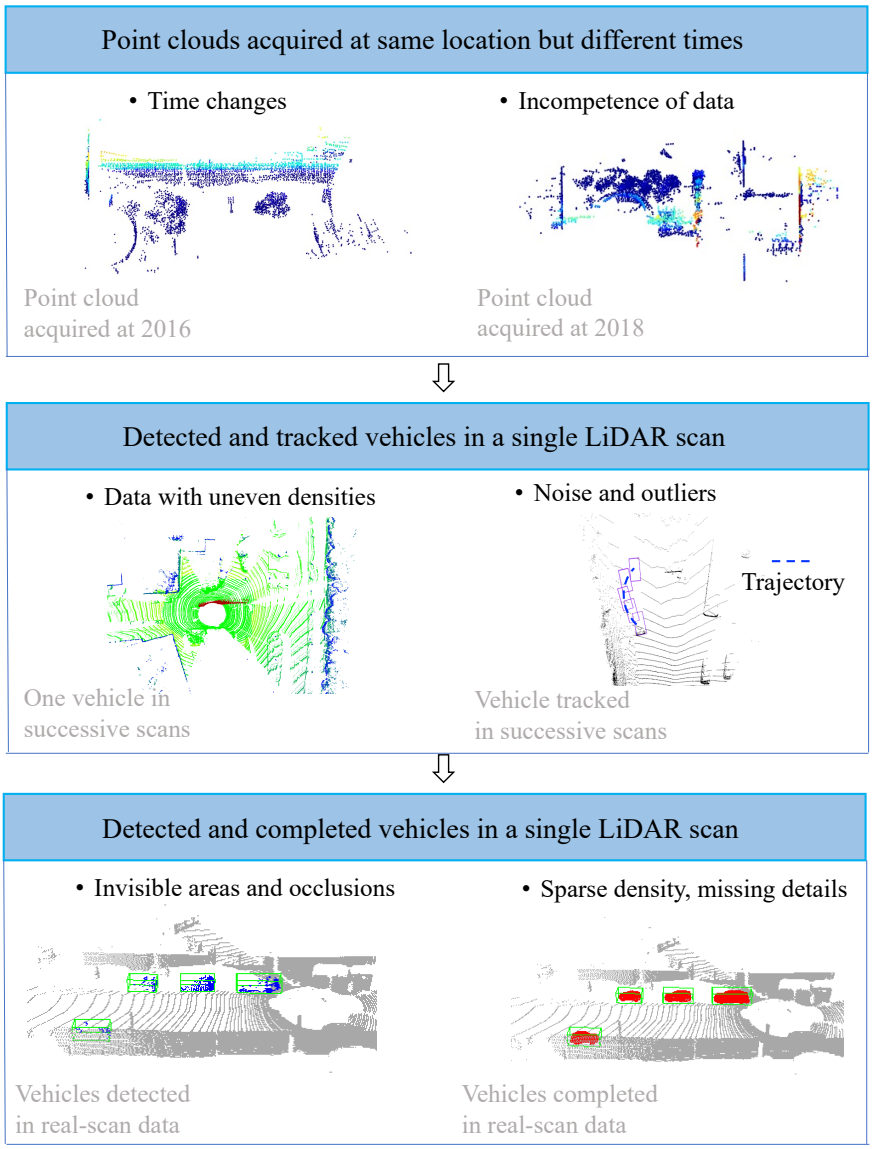


Figure 1.3: Challenges for recognition of 3D urban street environment from MLS point clouds.

Before introducing the objectives and contributions of this thesis, a detailed survey and reviews of related work in the following section are provided in order to find answers to the aforementioned tasks.

1.2 Related work

Recently, many researchers have studied the recognition of 3D urban street environments using MLS point clouds. According to the three-step workflow mentioned in the last section, detailed

reviews relating to point cloud based place recognition, vehicle detection and tracking, and 3D shape completion are provided in the following section.

1.2.1 Point cloud based place recognition

The implementation of 3D point cloud based place recognition is usually converted to a 3D retrieval task. The community has proposed numerous methods to tackle this task, which can be primarily classified into two categories: based on 3D local descriptors and 3D global descriptors. In addition, some methods based on planes or objects are proposed.

Based on 3D local descriptors

Various attempts have been made to encode robust local geometric structures and then perform matching on the basis of these features. Constructing local descriptors can be achieved via heuristically handcrafted features. With the recent advent of deep learning, a number of works also focus on leveraging data-driven methods to learn local descriptors from large-scale datasets.

Handcrafted local descriptors. In the early years, histograms are representative of extracting local structural information. Johnson & Hebert [1999] propose to deploy the spin image generation for matching 3D points. They create a reference axis in a 2D image by using the normal of the keypoint and then rotate the image around the reference axis. Finally, a spin image is generated by counting the number of neighboring points that fall into each image bin. Yamany & Farag [2002] design surface signatures, which encode the relationships of curvature, normal, and distance between a keypoint and its neighboring points into a 2D image. Frome et al. [2004] explore a novel regional shape context descriptor called Geometry Histogram, which divides the spherical support region into bins and then counted the number of points that fell within each bin to create a histogram. This work improves the 3D object recognition rate on noisy data. Later, Point Feature Histogram (PFH) [Rusu et al., 2008] is proposed to record the normal deviations and distance between any two pairs of points on the local surface. However, PFH is extremely time-consuming. Therefore, the following work Fast Point Feature Histogram (FPFH) [Rusu et al., 2009] is introduced to fast calculate the angular features and surface normals to represent the relationship between a 3D point and its neighbors. To extend the 2D shape context information to 3D meshes, Kokkinos et al. [2012] propose an intrinsic shape context (ISC) by replacing the Euclidean distance with the geodesic distance. Another type of 3D handcrafted descriptor is to incorporate external local reference frames (LRF) into the pipeline, which reduces the effect of rotation variance by performing covariance analysis or searching for salient regions on the local surface. Sun & Abidi [2001] project geodesic circles onto the tangent plane to create a set of 2D contours. These 2D contours and normals are used to create the fingerprint representation of the point. Mian et al. [2006] first define a local reference frame for each pair of vertices that satisfies certain geometrical constraints and then build a 3D grid on this basis. A 3D tensor is created by recording the surface area intersecting each cube of the 3D grid. Malassiotis & Strintzis [2007] adopt a virtual pin-hole camera oriented perpendicularly to the surface. They explore snapshot features by projecting each point onto the XY-plane of the LRF and then calculating the distance on a 2D image. This method is robust to self-occlusions and very efficient. The following work MeshHOG [Zaharescu et al., 2009] first calculates gradient vectors for neighboring vertices and then projects them to the three orthogonal planes of LRF, where each plane is divided into four polar slices and each spatial slice had orientation histograms with eight bins. All histogram values are concatenated to obtain a compact global descriptor. Zhong [2009] propose a novel shape descriptor Intrinsic Shape Signatures (ISS), which performs covariance analysis on the local surface based on LRFs and matches shape patches from different views directly. Signature of Histogram of Orientations (SHOT) [Tombari et al., 2010] introduces a unique and unambiguous LRF for the

keypoint and the keypoint's neighboring points in the local support region are aligned with the LRF. Then, the local support region is divided into multiple spherical volumes. For each volume, a local histogram is generated according to the normal deviations between the keypoint and its neighboring points. Finally, all the local histograms are concatenated to form a SHOT descriptor. However, SHOT is sensitive to mesh resolution variation. [Petrelli & Di Stefano, 2011] propose a novel LRF which aimed to predict a repeatable LRF at the border of a range image. Overall, these handcrafted descriptors are typically tailored to specific tasks and are sensitive to noise and mesh resolution, making them insufficiently descriptive and robust for complex and new scenarios.

Learning-based local descriptors. Recently, learning-based methods for 3D local descriptor extraction have gained significant developments boosted by large-scale 3D datasets. 3DMatch [Zeng et al., 2017] converts 3D points to the volumetric representation and then leverages 3D convolutional neural networks (CNNs) for segment matching. However, this method is not robust to rotation transformation. To alleviate this problem, some researchers are devoted to designing rotation-invariant local descriptors. Khoury et al. [2017] parameterize the input using a spherical histogram centered at each point and mapped the high-dimensional representation into a low-dimensional Euclidean space using a fully-connected network. PPFNet [Deng et al., 2018b] directly uses the raw 3D points as input and learns point pair features from points and normals of local patches. In the follow-up work, they propose PPF-FoldNet [Deng et al., 2018a], which is an unsupervised local descriptor. 3DFeatNet [Yew & Lee, 2018] design a weakly supervised network to learn both the 3D feature detector and descriptor. Zhou et al. [2018] introduce a multi-view local descriptor for the registration of point clouds. They first integrate multiple feature maps from the local patch of each view into a single representation and then adopt the MatchNet [Han et al., 2015] to learn a discriminative 3D descriptor. Spezialetti et al. [2019] convert the unoriented input data into a 3D spherical signals, which is a rotation-equivariant representation. Then, they feed the signals into the Spherical CNNs [Cohen et al., 2018] to learn an invariant descriptor without supervision. RSKDD-Net [Lu et al., 2020] introduces the random sampling concept to efficiently learn the keypoint detector and descriptor from a set of point clusters simultaneously. Although these methods can learn rotation-invariant features from the local surface, they rely on classical handcrafted features or external LRFs at first, which severely limits their effectiveness. Furthermore, new methods for learning dense local descriptors in a single forward pass have recently emerged. Fully Convolutional Geometric Features (FCGF) [Choy et al., 2019b] introduces a compact geometric feature computed by a 3D fully-convolutional network, which is the pioneering work of dense feature description for point cloud registration. Bai et al. [2020] propose to jointly learn both 3D feature detectors and descriptors based on KPConv [Thomas et al., 2019]. The following work, ASLFeat [Luo et al., 2020] focuses on mitigating limitations in the joint learning of 3D feature detectors and descriptors. 3DSmoothNet [Gojcic et al., 2019] and DeepVCP [Lu et al., 2019] learn compact and rotation invariant 3D descriptors relying on 3D CNNs. However, all these methods are not robust to rigid transformation in Euclidian space [Ao et al., 2022]. Some methods explore to compress the dimensions of handcrafted 3D local descriptors utilizing deep learning, such as Compact Geometric Features (CGF) [Khoury et al., 2017] and LORAX [Elbaz et al., 2017].

Based on 3D global descriptors

Different from the place recognition methods based on local descriptors, 3D global descriptors encapsulate comprehensive and global information about the entire scene. A robust and discriminative global descriptor is designed or learned to obtain the unique signature for retrieval.

Handcrafted global descriptors. Most handcrafted global descriptors describe places using LIDAR scan global statistics, which have the advantage of not requiring re-training to adapt to

different environments and sensor types. Magnusson et al. [2009] split the point cloud into overlapping cells and computed shape properties such as spherical, linear, and several types of planar of each cell. Granström et al. [2011] explore the rotation invariant features (e.g. volume, nominal range, and range histogram) to describe 3D point clouds. They first compute the distances between scalar features and cross-correlation for histogram features and then train an AdaBoost classifier to match places. Röhling et al. [2015] propose a fast method of describing places through histograms of point elevation, assuming the sensor had a constant height above the ground plane. M2dp [He et al., 2016b] first projects the source point cloud to multiple 2D planes. The density signature for each of the 2D planes is then used to construct the global descriptor. SegMatch [Dubé et al., 2017] introduces a real-time place recognition method based on 3D segments. It first extracted segments from an input point cloud and then uses a geometric-verification step for recognizing place candidates. Scan-Context [Kim & Kim, 2018] exploits the bird view of the point cloud and encodes the height information of the surrounding objects for place recognition. Recently, Wang et al. [2020c] propose LiDAR Iris, which generates a binary signature image for each point cloud. It firstly expands the bird-eye view of the input scan into an image strip for generating the iris images. Then, the Fourier transform is applied to these iris images by exploiting several filtering and thresholding operations. Finally, spatial place recognition is achieved in the frequency domain by calculating their similarities. However, the discriminative and robust power of such handcrafted global descriptors remains limited.

learning-based global descriptors. With breakthroughs in learning-based image retrieval methods, deep learning on 3D global descriptors for retrieval tasks has drawn growing attention. PointNetVlad [Angelina Uy & Hee Lee, 2018] first tackles 3D place recognition in an end-to-end way, which first uses PointNet [Qi et al., 2017a] to extract local descriptors and then aggregates these descriptors to obtain a global descriptor with NetVlad [Arandjelovic et al., 2016] pooling layer from 3D points. Following this, PCAN [Zhang & Xiao, 2019] proposes an attention mechanism for local features aggregation, discriminating local features that contribute positively. However, both two methods employ PointNet architecture for extracting local features, which does not particularly concern the local geometry. LPD-Net [Liu et al., 2019] enhances the local contextual relationships using graph neural networks but relied on handcrafted features, like changes of curvature and 2D scattering. DH3D [Du et al., 2020] designs a deep hierarchical network to produce more discriminative descriptors, recognizing the places and refining the 3D pose estimation simultaneously. DAGC [Sun et al., 2020] introduces a graph convolution module to encode local neighborhood information. However, they do not consider the spatial relationship between local descriptors. SOE-Net [Xia et al., 2021a] introduces a PointOE module introducing the orientation-encoding into PointNet for generating point-wise local descriptors, and then adopts a self-attention module to encode the spatial relationship of these descriptors. Minkloc3D [Komorowski, 2021] utilizes Feature Pyramid Network [Lin et al., 2017a] (FPN) based on MinkowskiEngine with generalized-mean (GeM) pooling [Radenović et al., 2018] layer to compute a compact global descriptor. Following work Minkloc3D-SI [Żywanowski et al., 2021] explores the place recognition task based on a single 3D LiDAR scan, using non-Cartesian point representation and intensity information.

Based on planes or objects

The approaches based on local descriptors frequently lack descriptive power, and invariance can be a problem for global descriptors. Therefore, some works have proposed using 3D shapes or objects for the 3D place recognition task. Fernández-Moral et al. [2013] propose to detect planes in 3D environments and then accumulate them into a graph. A final geometric consistency test is conducted over the planes in the matched sub-graphs using an interpretation tree. The following work [Fernández-Moral et al., 2016] proposes to utilize the covariance of the plane parameters

instead of the number of points in planes for matching. Finman et al. [2015] present an object-based place recognition in indoor environments based on RGB-D cameras. However, this strategy is only suitable for small, indoor environments.

1.2.2 3D vehicle detection and tracking

3D vehicle detection and tracking are two different but very closely related tasks. The goal of detecting 3D vehicles from point clouds is to recognize the vehicles by drawing an oriented 3D bounding box and assigning a label given a single LiDAR scan, while the goal of 3D vehicle tracking is to locate the vehicle in successive frames given the initial position in the first frame. Thus, numerous object detection methods and object tracking methods are introduced respectively in this section.

3D object detection

With deep neural networks that have been widely applied in computer vision for their capability of exploiting spatially-local correlations, learning-based 3D object detectors have evolved rapidly now. In general, they can be divided into three categories based on different representation learning strategies, voxel-based, point-based, and point-voxel-based detectors.

3D voxel-based detectors first partition the point clouds into discrete voxels by voxelization. Then, the detectors use convolutional neural networks or 3D sparse neural networks to extract features from the voxels. Finally, 3D objects can be detected from the Bird's Eye View (BEV) grid cells. VoxelNet [Zhou & Tuzel, 2018a] is a pioneering network. It first samples a limited number of sparse voxel grids. Then, the stacked voxel feature encoder (VFE) is used to extract features from the points inside a voxel cell. Finally, the 3D region proposal network (RPN) [Ren et al., 2015] is utilized to generate the classification score maps and the 3D bounding boxes. However, the computation overheads and memory footprints in VoxelNet grow cubically. The following work SECOND [Yan et al., 2018] leverages sparse convolutional layers after stacked VFE to get rid of unnecessary computation. PointPillars [Lang et al., 2019] further improves computational efficiency by thoroughly removing the heavy 3D CNNs layers. Specifically, it elongates voxels into pillars in a BEV perspective for creating a 2D pseudo image. However, this way is not good for the learning-based representation since the resolution in the vertical axis is lost. For proposal refinement, Voxel R-CNN aggregates voxel-wise features from 3D convolutional volumes. SE-SSD [Zheng et al., 2021b] jointly supervises a student network under the guidance of its teacher's distilled knowledge. Shi et al. [2020b] incorporates multi-scale voxelization strategies into feature aggregation. Mao et al. [2021] inserts a Transformer architecture to exploit long-range contextual dependencies among voxels. Xu et al. [2022] proposes a shape-learning strategy to solve the occlusion and signal miss problems when detecting objects.

3D point-based detectors implicitly extract local features and fine-grained patterns without any voxelization, which directly operate on raw point clouds. PointRCNN [**?**] is a pioneering work that employs PointNet-like blocks to learn semantic cues and generate 3D proposals from the downsampled point cloud via Furthest Point Sampling (FPS) method. However, the FPS is intrinsically a sequential algorithm and can not become highly parallel [Mao et al., 2022]. The following work 3DSSD [Yang et al., 2020] revisits the sampling strategy and proposes a new sampling method with feature distance. Considering the semantic cues from the neighboring points, Point-GNN designs local neighborhood graphs constructed from point clouds to enhance local and global features. Point-based detectors are generally time-consuming with a ball query complexity.

3D point-voxel-based detectors trend to integrate the merits of both voxel-based and point-based detectors together: Although voxel-based methods benefit from bird-view representation, the loss of fine-grained patterns limits further refinement, whereas point-based methods have relatively higher latency but fully preserve irregularity and locality. STD [Yang et al., 2019] designs a two-stage detector that first applies PointNet++ to obtain spherical anchors from sparse points, then voxelizes and refines them to generate accurate proposals. PV-RCNN [Shi et al., 2020a] proposes a voxel set abstraction layer that deeply integrates both the multi-scale 3D voxel CNN features and the PointNet-based features to capture much richer semantic cues. SA-SSD [He et al., 2020] explores an auxiliary network using two point-level supervisions to guide the features learned in the 3D sparse convolutional network.

3D object tracking

Numerous methods for tracking objects in 3D spaces have been developed, which are divided into two categories based on the number of tracked objects: single object tracking (SOT) and multiple object tracking (MOT).

Early 3D SOT methods [Asvadi et al., 2016; Bibi et al., 2016; Kart et al., 2018, 2019] generally rely on the RGB-D information and employ the 2D siamese tracking architecture. Though these methods are effective in certain situations, they do not fully explore 3D geometric clues. SC3D [Giancola et al., 2019] is a pioneering work for point cloud based tracking, which regularizes the latent spaces of template point cloud and search candidates using a shape completion network. However, this method is time-consuming since it uses Kalman filtering for target proposal generation. Moreover, it ignores the local geometric information of each target proposal. PSN [Cui et al., 2020] leverages a 3D Siamese network for single-person tracking. However, it cannot predict the orientation and size of the target. F-Siamese tracker Zou et al. [2020] explores RGB images to produce 2D region proposals for reducing 3D point cloud searching space. However, its performance depends more on the 2D tracker. 3DSiamRPN [Fang et al., 2020] combines a 3D Siamese network and 3D RPN architecture to track a single object, but the one-stage RPN network limits its performance. Recently, P2B [Qi et al., 2020] fuses the target object information into 3D search space and then adopts a state-of-the-art object detection network (VoteNet) to detect the target. Following this, BAT [Zheng et al., 2021a] proposes to add the bounding box information provided in the first frame as an additional cue. MLVSNet [Wang et al., 2021] performs Hough voting on multi-level features for getting more vote centers. PTT [Shan et al., 2021] introduces the transformer architecture to enhance the target-specific feature extracted in P2B. However, these methods all use the RPN to regress the bounding box of the target, which is inspired by their 2D SOT counterparts [Li et al., 2018; Zhang & Peng, 2019; Li et al., 2019].

Different from SOT where the initial position is given in the first frame, 3D MOT trackers adopt a detector to determine the number of objects in a sequence. The goal of 3D MOT is to associate these detected objects in all frames. Scheidegger et al. [2018] detects and estimates the distance to objects using a single image, and then uses a Poisson multi-Bernoulli mixture tracking filter to achieve 3D tracking. Patil et al. [2019]; Osep et al. [2017]; Weng et al. [2020a] utilize Kalman filter to obtain the 3D motion cue for simplicity and efficiency. With breakthroughs in learning-based 2D MOT methods, deep learning on 3D MOT tasks has drawn growing attention. Weng et al. [2020b]; Zhang et al. [2019b] explore 3D network to learn the 3D geometric characteristics and motion cues. Recently, Wu et al. [2021] adopts feature consistency of objects in consecutive frames to improve tracking accuracy.

1.2.3 3D shape completion

3D shape completion is to recover complete geometric information from partial point clouds, which has long been an attractive research topic in robotics and computer vision for many years [Anguelov et al., 2005; Han et al., 2017]. Generally, the shape completion methods can be primarily classified into three major categories: (i) geometry-based methods, (ii) template-based methods, and (iii) learning-based methods. In the following subsection, these three types of methods will be briefly reviewed.

Geometry-based shape completion

The geometry-based completion methods depend highly on geometric attributes, such as the continuity of local surfaces or volumetric smoothness, which have been applied to retouch small holes on incomplete point clouds successfully [Kazhdan et al., 2006; Tagliasacchi et al., 2011; Wu et al., 2015]. However, these methods are not applicable for completing missing points of larger regions. Thus, some approaches using hand-designed heuristics are proposed to reconstruct surfaces of 3D objects with a large percentage of missing areas. For example, Schnabel et al. [2009] complete 3D shapes with merely partial inputs by combining a series of planes and cylinders. Furthermore, Li et al. [2011] improve the completion performance by learning relations among geometric shapes such as planes and cylinders. For objects with arterial surfaces, Li et al. [2010] propose a novel deformable model named Arterial Snake, which successfully captures the topology and geometry simultaneously from arterial objects with noise and large parts missing.

Additionally, Thrun & Wegbreit [2005]; Zheng et al. [2010]; Pauly et al. [2008]; Tevs et al. [2014]; Harary et al. [2014] find the human-made objects usually have structural regularity, like symmetry. Thus, Thrun & Wegbreit [2005] identify the probable symmetries and apply them to extend the partial 3D model to the occluded space. Pauly et al. [2008] leverage regular structures that form a lattice with discrete rotational, translational, and scaling symmetries to fill missing regions. Zheng et al. [2010] automatically consolidate real-scan data by detecting repeating structures in input 3D models. Tevs et al. [2014] seek to quantify the relationship between shapes based on the regularities of symmetric parts. The shape of objects is firstly decomposed into a set of regions, and a graph is then applied to represent the relations between the regions in terms of symmetric transformations. Harary et al. [2014] utilize context information to synthesize geometry that is similar to the remainder of the input objects. Although these methods are efficient under particular circumstances, they are helpless when the missing area is large or there is no significant symmetry of the object.

Template-based Shape Completion

In addition to the geometry-based methods, some template-based methods are proposed by researchers. They complete 3D surfaces by deforming or reconstructing point clouds according to the most similar templates retrieved from a prepared 3D shape database. Thus, they are also known as retrieval-based methods. As a precondition for the retrieval, Pauly et al. [2005] create a 3D shape database to extract geometric clues for completing missing regions. However, the database retrieval process is time-consuming and labor-intensive since manual interaction is mandatory to constrain the categories of 3D objects in this method. Similarly, Rock et al. [2015] propose to complete any categories of objects automatically. However, this method is based on the usage of additional depth images as auxiliary data. An adequate auxiliary database with sufficient depth images plays a key role in the performance of this method.

To avoid the high dependency on large-scale 3D shape databases, [Schnabel et al., 2009; Nan et al., 2010; Chauve et al., 2010; Li et al., 2011; Shen et al., 2012; Sung et al., 2015] propose to apply

geometric primitives replacing a shape database. Schnabel et al. [2009] reconstruct missing parts with the guidance of a set of detected primitive shapes (e.g. planes and cylinders). Nan et al. [2010] present a novel interactive tool called SmartBoxes to reconstruct structures that are partially missing from inputs. This allows the users to fit polyhedral primitives interactively, avoiding an exhaustive search. Chauve et al. [2010] plausibly complete missing scene parts by decomposing 3D space based on planar primitives. Li et al. [2011] seek to simultaneously recover the local missing parts while using structural relations from man-made objects, but the fundamental primitives must be preserved. Shen et al. [2012] present an assembly method for recovering 3D structures from a small-scale shape dataset using predefined geometric primitives. Sung et al. [2015] use a global optimization method to reconstruct entire surfaces from partial inputs using inference from given geometric information.

However, such methods exhibit several limitations. Firstly, they are not suitable for online operations due to the voluminous computational overhead. Secondly, it is labor-intensive for preparing a 3D shape database since each shape should be labeled and segmented manually. Finally, noise or disturbances (e.g., dynamic changes) affects their performance significantly.

Learning-based shape completion

Compared with geometry-based and template-based methods, completion approaches using learned features can automatically find the feature clues needed for the completion from partial inputs, requiring less prior knowledge and avoiding time-consuming preprocessing. Dictionary learning or deep learning are the typical ways to achieve feature learning. Recently, learning-based methods for 3D shape completion are obtaining significant developments with the help of large-scale 3D synthetic CAD model datasets. And these state-of-the-art methods provide end-to-end solutions that decrease the efforts in feature design and show excellent performance on various representing formats of 3D models. Generally, learning-based methods for 3D shape completion can be categorized into three types: voxel-based methods, mesh-based methods, and point-based methods.

The core idea of voxel-based methods is to structure the 3D space into regular voxel grids and then project these unordered 3D points into these ordered voxels. Thus, 3D convolution and distance field formats can be well suited for processing this kind of discrete and rasterized data. Dai et al. [2017b] propose typical examples of voxel-based methods, which adopt a 3D convolutional network to achieve the excellent performance of completing shapes. Stutz & Geiger [2018] propose a weakly supervised learning-based method to complete a 3D shape, and this method is easier to achieve. However, voxel-based methods are time-consuming and memory-consuming to predict volumes of high spatial resolution. To alleviate this problem, Vo et al. [2015]; Wang & Tseng [2011] propose to adopt the octree structure for increasing the resolution. Following work GRNet [Xie et al., 2020a] propose a gridding network for dense point cloud completion. However, the voxelization representation still has a series of issues. For instance, since grid occupancy is predicted independently, the completion results often miss thin structures or contain flying voxels. Moreover, the volumetric representation obscures natural invariance when it comes to geometric transformations and manipulations.

Recent works [Groueix et al., 2018; Wang et al., 2018; Litany et al., 2018] are proposed to focus on mesh-based reconstruction. Groueix et al. [2018] propose a novel shape reconstruction network called AtlasNet, which regards a 3D shape as a collection of parametric surface elements. Pix2Mesh Wang et al. [2018] introduces a graph-based network to generate 3D manifold shapes. Litany et al. [2018] explore a variational autoencoder using graph convolutional operations to deform meshes. This method focuses on non-rigid deformations of objects such as faces or human

bodies. These methods can reconstruct shape information, however, by deforming a reference mesh to a target mesh. As a result, they are not adaptable to all typologies.

In comparison to 3D mesh, or voxel representations, the point cloud is more suitable during training because of its simple structure. Besides, new points can easily be added or interpolated to a point cloud since all the points are independent and the connectivity information is not needed to update. Thus, more and more works are proposed to process discrete points directly without transferring them into voxels or meshes. PCN [Yuan et al., 2018] is the pioneering work to directly operate on raw point clouds, which is an encoder-decoder network to reconstruct a complete and dense point cloud from a partial point cloud. The folding operation [Yang et al., 2018] is adopted to generate high-resolution outputs. Following that, TopNet [Tchapmi et al., 2019] introduces a hierarchical tree-structure decoder for the point cloud generation. However, they are unable to produce evenly distributed point clouds and reconstruct fine-grained details of objects. In view of this, MSN [Liu et al., 2020] explores preserving the details from partial inputs and refining the geometric information for the missing regions. Another work [Gurumurthy & Agrawal, 2019] adopts a GAN network to implement a denoising optimization algorithm for enhancing the global features extracted from incomplete inputs. PF-Net [Huang et al., 2020] introduces a multi-stage strategy to reconstruct the missing structures of 3D objects. SoftPoolNet [Wang et al., 2020d] designs a soft pooling layer to replace the max-pooling layer, which can keep more feature information when completing the partial inputs. SA-Net [Wen et al., 2020] adopts a self-attention mechanism [Xia et al., 2021a] to effectively exploit the local structure details. Zhang et al. [2020b] propose a feature aggregation strategy for preserving the primitive details of partial inputs. Wang et al. [2020a] design a cascaded module to refine the fine-grained details. However, all these methods only extract global features from partial inputs, resulting in information loss during the encoding process.

Compared with geometry-based and template-based 3D shape completion methods, learning-based methods show their advances in learning high-level feature representation. And they improve the effectiveness of completion using an end-to-end way. Point-based approaches stand out among learning-based methods due to their ability to deal directly with the geometric nature of point clouds. Although many cutting-edge learning-based methods have achieved remarkable performance, there are still some aspects that could be improved. First, how to better utilize the local geometric characteristic of points can be further explored. Second, how to avoid the information loss of global features is worth exploring. Third, the generalization of networks for completing unseen categories of objects could be exploited.

1.3 Objectives and contributions

The top-level goal of this work is to develop a framework for the robust and accurate recognition of 3D urban street environments using mobile laser scanning point clouds, achieving localization of the current scene within an existing 3D map and status analysis of the dynamic objects (e.g. Vehicles). The following research questions should be considered and answered:

- Which accuracy of place recognition can be achieved by MLS point clouds in an urban area collected at different times?
- Which success and precision rate of object detection and tracking (e.g. vehicles) in an urban street environment can be achieved using features learned from 3D sparse point clouds?
- Which completeness and robustness can be achieved for completing point clouds of objects which are only partially scanned?

Based on the analysis of the point cloud based place recognition, vehicle detection and tracking as well as 3D shape completion presented in the previous section, the research questions will be answered by developing the following specific algorithms and methods in this thesis.

1.3.1 Point cloud based place recognition

For point cloud based place recognition tasks, two novel methods are proposed for large-scale point cloud based retrieval. One is an end-to-end network that explores the relationship among the raw 3D points and the different importance of local descriptors for large-scale point cloud based retrieval. Specifically, a novel point orientation encoding module is proposed to effectively extract local descriptors from a given point cloud, considering the relationship between each point and its neighboring points. Besides, a self-attention unit is designed to differentiate the importance of different local descriptors. Furthermore, a new loss function is presented, that is more effective for large-scale point cloud based retrieval. Compared with previous loss functions, the proposed loss function can achieve more versatile global descriptors by relying on the maximum distance of positive pairs and the minimum distance of negative pairs.

Another is a novel Point-Voxel Transformer network, aiming to compensate for quantization loss and introduce long-range spatial relationships. More specifically, a fusion strategy is designed by using points as middle hosts, fusing point-wise features into sparse voxel-wise features. Compared to the previous point-voxel naive fusion strategies, a hierarchical cross-attention transformer is proposed to enhance sparse voxel-wise features in a more flexible and efficient way. The efficiency includes two aspects: Firstly, only simple multilayer perceptions (MLPs) are used on point branches to eliminate inefficient local neighbor searching. Secondly, a lower quantization resolution and sparse convolutions are utilized in the voxel branch. Moreover, the attention mechanisms used in [Zhang & Xiao, 2019; Xia et al., 2021a] are extremely memory-consuming and time-consuming, which limits their employment in large-scale point clouds. Thus a lightweight self-attention unit is introduced to alleviate the low-efficiency problem.

1.3.2 3D vehicle detection and tracking

For achieving object detection and tracking (e.g. vehicles) in an urban street environment, a real-time and effective pipeline is proposed. For vehicle detection, a 3D vehicle detector including data pre-processing, detection network, and post-processing modules is introduced. Furthermore, a novel lightweight and detector-free 3D single object tracking (SOT) network is designed. Specifically, a motion prediction module is first developed to estimate the 3D coordinates of a potential target center in the current frame using previous frames. Although the estimated center is coarse, it can provide strong prior information as guidance. Thus, an explicit voting layer only consisting of several MLPs is further designed to refine the target center with the desired position and rotation. To the best of my knowledge, the usage of complicated 3D detectors or proposal generation in 3D single object trackers is the first to be completely removed. And the proposed 3D tracker can be served as a simple yet strong baseline in the 3D SOT community.

1.3.3 3D shape completion

For completing point clouds of objects which are only partially scanned, two novel 3D shape completion methods are explored. First, a novel end-to-end network for completing point clouds of 3D vehicle shapes, directly operating on partial and sparse point clouds, is proposed. The proposed network can produce uniform, dense, and complete point clouds from partially scanned vehicles in MLS datasets by endorsing an architecture with the encoder, decoder, and refiner. Specifically, a novel encoder module that includes a spatial transformer network (STN) and a point feature

enhancement (PFE) layer are designed to better extract global features from the instance. The STN ensures that the extracted features are not affected by geometric transformations from input point clouds of varying resolutions. To improve feature representation, the PFE layer combines low-level and high-level information. In addition, a refiner module is proposed to preserve the vehicle details from inputs and refine the outputs with fine-grained details. To fully retain the details of the input point cloud, the partial input point cloud and the output generated by the decoder are combined uniformly. A point feature residual network is designed to predict per-wise offsets for every point.

Furthermore, a more general 3D completion neural network is proposed, which can be applied to multiple categories, including unseen categories. It investigates the importance of the shape prior information via a feature-matching strategy. More specifically, an asymmetrical Siamese auto-encoder network is designed to learn shape prior information, which can produce a more informative global feature for incomplete objects. With the guidance of shape priors, an iterative refinement unit is introduced to retain the information of the incomplete inputs and reasonably infer the missing geometric details of the object.

1.4 Structure and organization

This thesis is organized as follows: Chapter 2 presents the theoretical basics of point-based deep learning operations, deep metric learning, and Transformer. Chapters 3-5 describe the core parts of this thesis, namely the methods for answering the aforementioned research questions. Chapter 6 presents the experiments, along with the datasets and the evaluation metrics. Chapter 7 presents the experimental results and analysis. Chapter 8 presents the discussion on the three main tasks involved in the recognition of 3D urban street environment using MLS point clouds. Chapter 9 finalizes this thesis by presenting the conclusions and providing outlooks for future works.

Parts of the thesis have been published or submitted in the following journal articles and conference papers:

(i) Xia Y, Xu Y, Li S, Wang R, Du J, Cremers D, and Stilla U (2021a). Soe-net: A self-attention and orientation encoding network for point cloud based place recognition. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition (CVPR)*, pages 11348-11357.

(ii) Xia Y, Xia Y, Li W, Song R, Cao K, and Stilla U (2021b). Asfm-net: Asymmetrical siamese feature matching network for point completion. In *Proceedings of the 29th ACM International Conference on Multimedia (ACM MM)*, pages 1938-1947.

(iii) Xia Y, Xu Y, Wang C, and Stilla U (2021c). VPC-Net: Completion of 3D vehicles from MLS point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 174:166-181.

(iv) Xia Y, Liu W, Luo Z, Xu Y, and Stilla U (2020). Completion of sparse and partial point clouds of vehicles using a novel end-to-end network. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2, 933-940.

(v) Xia Y, Wu Q, Li W, Chan AB, Stilla U (2023). A Lightweight and Detector-free 3D Single Object Tracker on Point Clouds. *IEEE Transactions on Intelligent Transportation Systems.*

(vi) Xia Y, Gladkova M, Wang R, Q Li, Stilla U, Henriques JF, and Cremers D (2023). CASSPR: Cross Attention Single Scan Place Recognition. In review.

2 Basics

This chapter briefly outlines the basic concepts and specific techniques of point-based deep learning methods, deep metric learning, and Transformers. The proposed methods in this thesis are developed based on similar principles of these techniques. Specifically, for the task of point cloud based place recognition, the proposed methods are developed based on the extension of point-based learning, deep metric learning, and Transformers. The novel solutions are explored for generating robust and discriminative global descriptors considering long-range spatial relations and involving a modified attention mechanism. As for object detection and tracking, point-based learning methods are introduced as the basic architecture of feature extraction. As for the 3D shape completion task, point-based learning methods and the Siamese network provide a solution to close between the partial point cloud feature and the complete point cloud feature. The proposed completion methods make an improvement on global feature extraction based on the closed features.

2.1 Point-based learning methods

In this section, two point-based learning methods which directly apply to 3D point clouds are introduced. The first one is PointNet [Qi et al., 2017a], and another one is PointNet++ [Qi et al., 2017b]. Both of them are utilized as the typical backbone networks for point feature extraction.

2.1.1 PointNet

PointNet is a pioneering work that extracts features directly from 3D point clouds. Qi et al. [2017a] highlights the point feature extraction network should meet three key properties: (1) The network consuming a point set with N points should be insensitive to $N!$ permutations of the point order since the points in the point set are spatially unordered; (2) The network should be able to capture global structures since the points are combined to represent an object surface; (3) The network should be transformation-invariant. This means the global feature of a point cloud should not be changed even though a specific rotation or translation is applied to the point set.

Fig. 2.1 shows the network architecture of PointNet. Given an unordered point cloud with N points $P_{input} = \{(x_i, y_i, z_i)\}_{i=1}^{N}$, a spatial transformation network is first used to transform the input, which is represented by the T-Net in this figure. The aim is to learn the 3×3 affine matrix that enables PointNet to be invariant to the rigid transformation. Then, several multilayer perceptrons are utilized to extract high-dimensional features. Considering the point cloud is a set of points without a specific order, PointNet introduces a symmetric function Max-pooling to obtain a global feature G independent of the order. The reason is the max values are fixed whatever the order changes. The whole process can be formulated as:

$$G = MAX(MLP(x_i, y_i, z_i)) \tag{2.1}$$

Compared to other previous learning-based methods, PointNet is easy to implement and enables the processing of 3D points in a direct manner. However, it regards each point as an independent

individual, ignoring the relationships between points so that it cannot capture the local structure information.

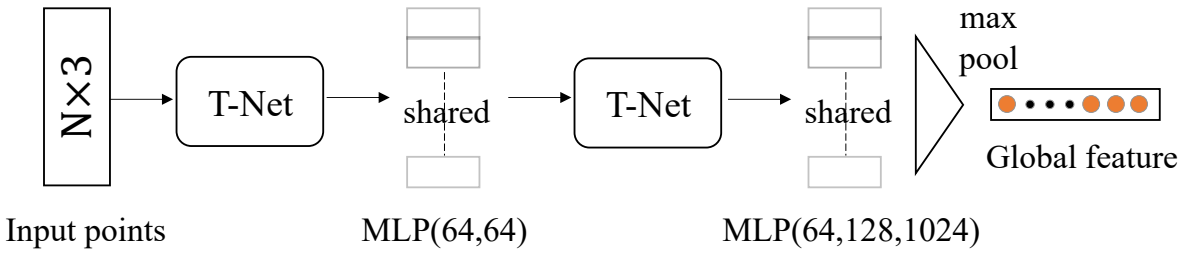


Figure 2.1: The network architecture of PointNet [Qi et al., 2017a]. PointNet takes N points as input, applies T-Net to the input and feature transformations, and then obtains a global feature by aggregating point features. Numbers in brackets are layer sizes.

2.1.2 PointNet++

To tackle the existing problems in PointNet, PointNet++ [Qi et al., 2017b] introduces to add hierarchical structure to extract local features by capturing fine geometric structures from neighborhoods. PointNet++ can be viewed as an extension of PointNet, via building a hierarchical grouping of points and progressively encoding larger and larger local regions along the hierarchy. The network architecture is illustrated in Fig. 2.2.

PointNet++ is made up of three layers: a sampling layer, a grouping layer, and a feature extraction layer. It first employs the farthest point sampling (FPS) algorithm to uniformly sample the key points from the input point cloud. The point cloud is then divided into several spherical spaces of a given radius using a grouping layer that treats the key points as the center point. Each spherical space is considered a subset. Finally, PointNet is used to extract local features from each small subset as a feature extractor. Considering the spatial relations between points in the local area, PointNet++ with the hierarchical propagation strategy extracts a more informative global feature than PointNet.

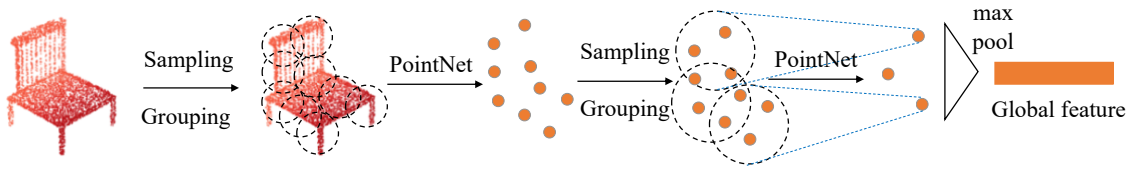


Figure 2.2: The hierarchical architecture of PointNet++.

2.2 Deep metric learning

Metric learning is a method of measuring sample similarity that is directly based on a distance metric, which attempts to reduce the distance between similar samples while increasing the distance between dissimilar samples. A variety of metric learning methods have been proposed and applied in many visual recognition tasks such as one-shot learning [Cai et al., 2018; Fei-Fei et al.,

2006], image retrieval [Huang et al., 2015; Oh Song et al., 2016], person re-identification [Li et al., 2014; Yi et al., 2014], and face recognition [Schroff et al., 2015; Sun et al., 2014].

Metric learning can be divided into two categories: unsupervised and supervised [Kulis et al., 2013]. Unsupervised metric learning seeks to learn a low-dimensional subspace in order to preserve useful geometrical information from samples. Supervised metric learning is the mainstream metric learning technique, which seeks an appropriate metric by formulating an optimization objective function to exploit supervised information from training samples, where the objective functions are designed for different specific tasks.

Most conventional metric learning methods project samples into a new feature space by learning linear mapping. However, it is not efficient when the data points exist the nonlinear relationships. Although some methods explore the kernel trick to tackle this issue, it still suffers from scalability issues since choosing a suitable kernel is typically difficult and empirical. Besides, the expression power of kernel functions is frequently insufficient to capture non-linearity in data. With the popularity of deep learning methods for the non-linearity of samples, deep metric learning methods [Hu et al., 2014; Sun et al., 2014; Wu et al., 2013; Wang et al., 2014] have been proposed.

In deep metric learning, the goal of network optimization is to make the network achieve two distinct goals: (1) minimizing the same class of samples, such as the images/LiDAR scans captured at the same locations but at different times under different postures, lighting, and decoration, where the measured feature distance should be as small as possible; and (2) minimizing non-identical samples, such as the images/LiDAR scans captured at different locations, where the metric distance of their features should be as large as possible, even if they are very similar. The main challenge of deep metric learning is how to learn an efficient function $f(x)$. It maps the samples x to a D-dimensional feature space $\mathbb{R}^D$ in order to obtain more discriminative features, resulting in optimal classifier performance.

In this section, some basic concepts and strategies of deep metric learning, including distance metrics, Siamese networks, and loss functions, are introduced. The common distance metrics (Euclidean distance, Manhattan distance, Chebyshev distance, Minkowski distance, Mahalanobis distance, and Cosine similarity) are first presented in Section 2.2.1. Then, the typical network architecture of Siamese networks used in deep metric learning is introduced in Section 2.2.2. Finally, the training strategy with common loss functions is presented in Section 2.2.3.

2.2.1 Distance metric

To measure similarities among samples, a distance metric must be introduced that determines whether a pair of samples are more similar than another pair of samples. However, there are many different distance metrics, and not all of them can be properly adapted to the samples in special tasks. As a result, choosing the suitable distance metric for metric learning is critical.

In mathematics, a metric space can be defined as a vector set $X = x_1, x_2, \cdots, x_N$ together with a distance metric d on the set. d satisfies the following properties:

$$\begin{aligned} &d(x_i, x_j) = 0 \Leftrightarrow i = j, \quad \text{(identity)} \\ &d(x_i, x_j) \geq 0, \quad \text{(non-negativity)} \\ &d(x_i, x_j) = d(x_j, x_i), \quad \text{(symmetry)} \\ &d(x_i, x_j) + d(x_j, x_p) \geq d(x_i, x_p), \quad \text{(triangle inequality)} \end{aligned} \tag{2.2}$$

where $x_i, x_j, x_p \in X$. Following [Nebel et al., 2017], d is a pseudometric if it only satisfies the first three properties.

There is no negative distance, no sample is separated from itself, and no two samples can occupy the same position. The distance traveling from x_i to x_j is same as traveling from x_j to x_i. The shortest path between two points is always preferred over a detour.

The commonly used distance metrics include Euclidean distance, Manhattan distance, Chebyshev distance, Minkowski distance, Cosine similarity, and Mahalanobis distance.

Definition 2.1 (Euclidean distance) For any $(x_i, x_j) \in \mathbb{R}$, the Euclidean distance is defined as:

$$d(x_i, x_j) = \sqrt{(x_i - x_j)^2 (x_i - x_j)} \tag{2.3}$$

The Euclidean distance between two samples in Euclidean space is the length of a line segment connecting the two samples in mathematics. It is one of the simplest and most widely used metrics for measuring dissimilarity (large distance) and similarity (small distance) between samples.

Definition 2.2 (Manhattan distance) For any $(x_i, x_j) \in \mathbb{R}$, the Manhattan distance is defined as:

$$d(x_i, x_j) = \|x_i - x_j\| = \sum_{p=1}^{n} |x_{ip} - x_{jp}| \tag{2.4}$$

where (x_i, x_j) are n-dimensional vectors $x_i = (x_{i1}, x_{i2}, \cdots, x_{in})$ and $x_j = (x_{j1}, x_{j2}, \cdots, x_{jn})$, respectively. The Manhattan distance between two samples is the sum of the absolute differences of their Cartesian coordinates. The distance is determined by the rotation of the coordinate system, but not by its reflection about a coordinate axis or translation.

Definition 2.3 (Chebyshev distance) For any $(x_i, x_j) \in \mathbb{R}$, the Chebyshev distance is defined as:

$$d(x_i, x_j) = \underset{p}{Max}(|x_{ip} - x_{jp}|) \tag{2.5}$$

where (x_i, x_j) are n-dimensional vectors $x_i = (x_{i1}, x_{i2}, \cdots, x_{in})$ and $x_j = (x_{j1}, x_{j2}, \cdots, x_{jn})$, respectively. The Chebyshev distance between the two samples is the greatest of their differences along the coordinate dimension.

Definition 2.4 (Minkowski distance) For any $(x_i, x_j) \in \mathbb{R}$, the Minkowski distance is defined as:

$$d(x_i, x_j) = (\sum_{p=1}^{n} |x_{ip} - x_{jp}|^q)^{1/q} \tag{2.6}$$

where (x_i, x_j) are n-dimensional vectors $x_i = (x_{i1}, x_{i2}, \cdots, x_{in})$ and $x_j = (x_{j1}, x_{j2}, \cdots, x_{jn})$, respectively. q is an integer. The Minkowski distance can be considered a generalization of both the Euclidean distance and the Manhattan distance. When $q = 1$, it equals to Manhattan distance; when $q = 2$, it equals to Euclidean distance; when $q \to \propto$, it equals Chebyshev distance. However, the above four distances cannot account for data coupling since they treat differences between different attributes of the sample (i.e., each characteristic variable) equally.

Definition 2.5 (Mahalanobis distance) For any $(x_i, x_j) \in \mathbb{R}$, the Mahalanobis distance is defined as:

$$d(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)} \tag{2.7}$$

where $M \in \mathbb{Z}_+$, denoting the cone of symmetric positive semi-definite real-valued matrices.

Mahalanobis distance is an effective multivariate distance metric that measures the distance between a sample and a distribution. Since M is positive semi-definite, it can be decomposed

as $M = L_T L$. L is the matrix of all eigenvectors. Thus, the Mahalanobis distance can be reformulated as:

$$\begin{aligned} d(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)} \\ &= \sqrt{(x_i - x_j)^T L^T L (x_i - x_j)} \\ &= \sqrt{(Lx_i - Lx_j)^T (Lx_i - Lx_j)} \end{aligned} \tag{2.8}$$

Compared to Eq. 2.3 and Eq. 2.8, computing the Mahalanobis distance is the same as computing the Euclidean distance after linearly mapping the data from the original space to transformed space by the transformation matrix L. Both learning the distance matrix M and learning the linear transformation matrix L achieve metric learning, according to the equivalence.

Definition 2.6 (Cosine similarity) For any $(x_i, x_j) \in \mathbb{R}$, the Cosine similarity is defined as:

$$d(x_i, x_j) = \frac{x_i^T x_j}{|x_i| \, |x_j|}. \tag{2.9}$$

Cosine similarity calculates similarity by taking the cosine of the angle between two samples' vectors. A cosine value of 0 indicates that the two vectors are orthogonal to each other and have no match. The smaller the angle and the greater the match between vectors, the closer the cosine value to 1.

2.2.2 Siamese network

Bromley et al. [1993] first propose a Siamese neural network in the early 1990s to solve the signature verification problem. A Siamese neural network is made up of two neural networks that accept different inputs but are linked by an energy function. This function computes a distance metric between the highest level feature representation [Koch et al., 2015]. Notably, a key feature of Siamese networks is that the twin networks share the same architectures and same weights. Fig. 2.3 shows a general Siamese network that contains two identical neural networks G_w. Given a pair data (X_1, X_2) as inputs, the twin neural networks outputs $G_w(X_1)$ and $G_w(X_2)$, respectively. Then, the distance between $G_w(X_1)$ and $G_w(X_2)$ is computed by an energy function E_w:

$$E_w(X_1, X_2) = \|G_w(X_1) - G_w(X_2)\| . \tag{2.10}$$

2.2.3 Loss functions

Unlike other loss functions in traditional metric learning methods, which aim to learn to predict labels, values, or one or more values of a given input directly, the typical loss function in deep metric learning is ranking loss, predicting the relative distance between the inputs. In terms of training data, the ranking loss can be used as long as data points are similar. The outputs could be binary (similar/dissimilar). Assuming an image-based place recognition dataset exists, and it is required to know which images belong to the same location (similar) and which do not (dissimilar). CNNs can be trained using the ranking loss function to determine whether two images belong to the same location.

The goal of ranking loss is to learn a representation of features with a distance between them that is less than the marginal value for positive sample pairs and greater than the marginal value for negative sample pairs. A typical ranking loss function pipeline consists of three steps: (1) the features from two (or three) inputs are extracted and then the embedding representations for each feature are computed. (2) A distance metric, such as the Euclidean distance, is defined to measure

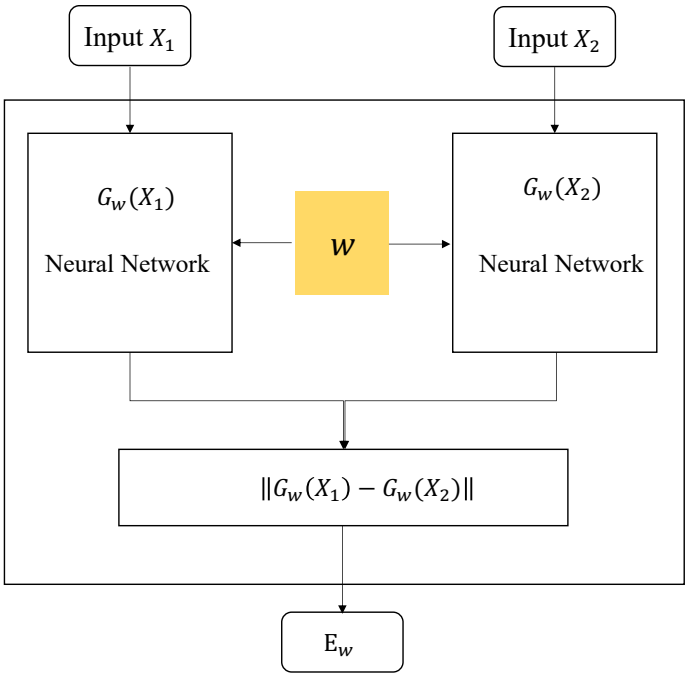


Figure 2.3: The network architecture of a Siamese network.

the similarity of these representations. (3) Finally, if two inputs are similar, the feature extractor generates similar representations for them; otherwise, it will generate different representations for them.

The ranking loss can be divided into two types: Pairwise Ranking loss and Triplet Ranking loss, based on the usage of pairs of training data samples or triplets of training data samples, respectively. Fig 2.4 illustrates an example of the network architecture based on a pairwise ranking loss.

In Fig 2.4, the positive pairs and negative pairs are constructed, respectively. A positive pair is composed of an anchor sample and a positive sample. The network is hoped to close the feature embeddings between the anchor and the positive in the distance metric. In addition, a negative pair consists of an anchor sample and a negative sample. The network aims to push away the feature embeddings between the anchor and the negative. Pairwise ranking loss hopes that the distance of positive pairs is 0 and the distance of negative pairs is greater than the marginal value m. The loss function can be formulated as:

$$L = \begin{cases} dist(f_a, f_p), & \text{if positive pair} \\ max(0, m - dist(f_a, f_n)), & \text{if negative pair} \end{cases} \tag{2.11}$$

where f_a, f_p, f_n are the feature embeddings of anchor, positives, and negatives, respectively.

For positive pairs, the pairwise ranking loss is 0 only when the network generates features for both elements in the pair with no distance between them. For negative pairs, the loss is 0 when the distance between the representations of the two pair elements is greater than the margin m. The aim of m is to focus on more difficult pairs during the network training stage because no effort is wasted on enlarging that distance when the representations of a negative pair are sufficiently distant.

The triplet ranking loss explores the triplets of training data instead of pairs. Fig 2.5 shows an example of a network pipeline based on a triplet ranking loss. A triplet consists of an anchor,

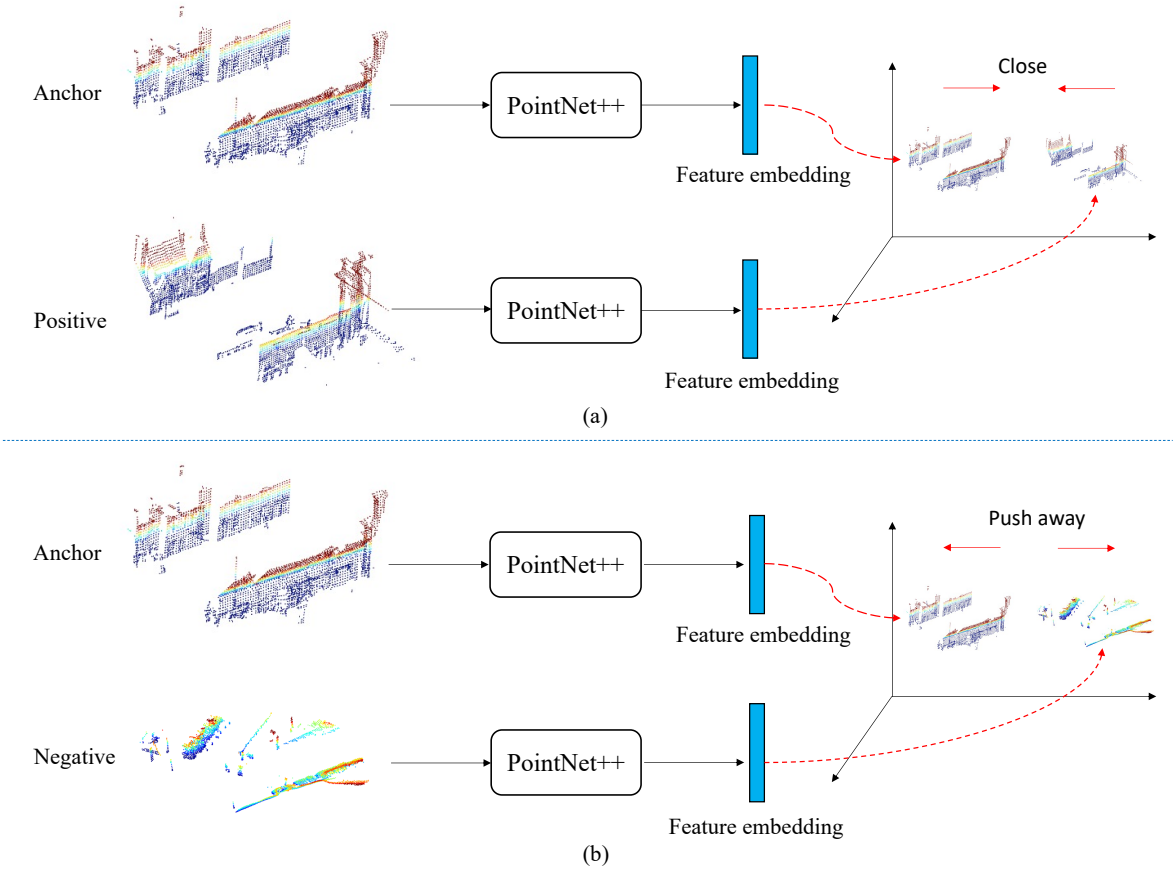


Figure 2.4: An example of a pairwise ranking loss set up to train a net for point cloud based place recognition. a) shows the network should close the feature embeddings between the anchor and the positive, b) shows the network should push away the feature embeddings between the anchor and the negative. A pairwise ranking loss can be used in other tasks or other networks.

a positive, and a negative sample. The objective is the distance $dist(f_a, f_p$ between the anchor and the negative is greater (bigger than the margin m) than the distance $dist(f_a, f_n$ between the anchor and the positive. The triplet ranking loss can be formulated as:

$$L = max(0, m + dist(f_a, f_p) - dist(f_a, f_n)) \tag{2.12}$$

According to Eq. 2.12, the triplets can be divided into three types: Easy Triplets, Hard Triplets, and Semi-Hard Triplets. The visualization of them is vividly illustrated in Fig .

Easy Triplets: $dist(f_a, f_n) > dist(f_a, f_p + m)$**.** In the embedding space, an Easy Triplet is defined as the negative sample being sufficiently distant from the anchor in comparison to the positive sample. The loss will be zero, so the network will regard them as easy training data and there will be no effort to update the network parameters.

Hard Triplets: $dist(f_a, f_n) < dist(f_a, f_p)$**.** In the embedding space, a Hard Triplet is defined as the negative sample being closer to the anchor than the positive. Because the loss is positive, it is a difficult case for network training, and the network should focus on these cases.

Semi-Hard Triplets: $dist(f_a, f_p) < dist(f_a, f_n) < dist(f_a, f_n) + m$**.** In the embedding space, a Hard Triplet is defined as the negative sample being more distant to the anchor than the positive, but the distance is not greater than the margin. The loss is also positive.

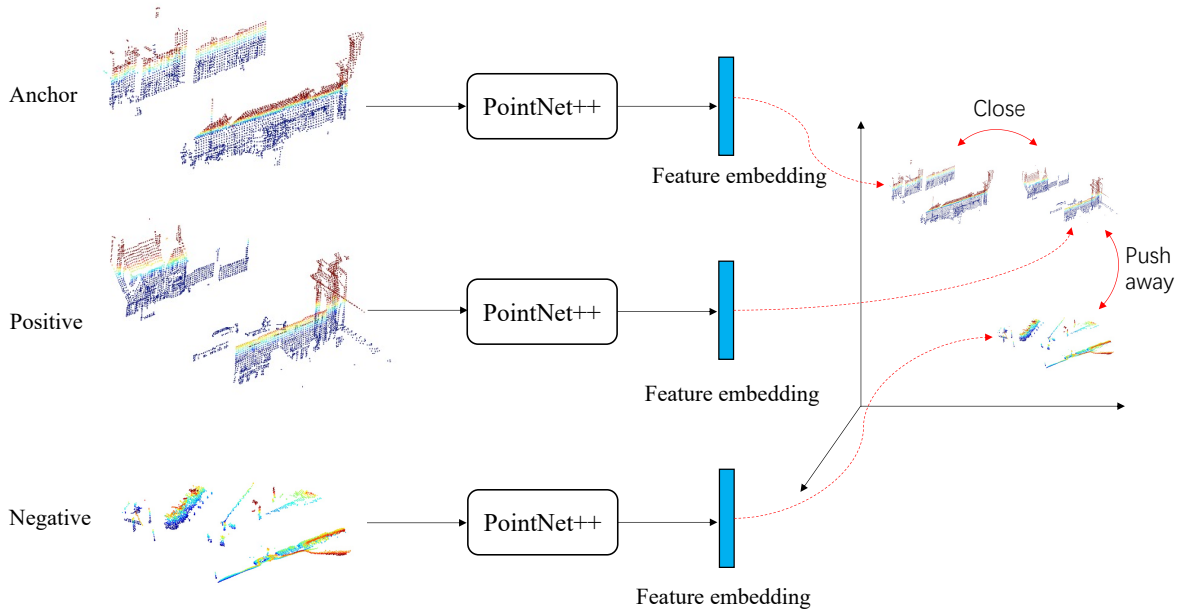


Figure 2.5: An example of a triplet ranking loss set up to train a net for point cloud based place recognition. A triplet ranking loss can be used in other tasks or other networks.

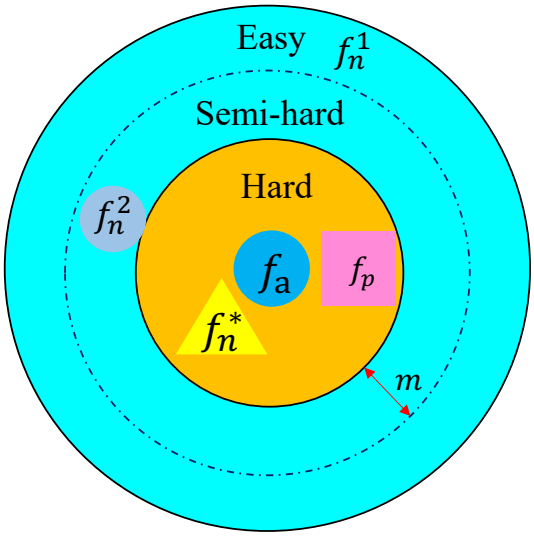


Figure 2.6: Three types of triplets: Easy Triplets, Hard Triplets, Semi-Hard Triplets. f_a is an anchor example, and f_p is a positive example. f_n^1, f_n^2, and f_n^* are the easy, semi-hard, and hard negative examples, respectively.

2.3 Transformer

Transformer [Vaswani et al., 2017] is originally proposed as a sequence-to-sequence model architecture for machine translation, which relies entirely on an attention strategy to obtain global context dependencies between input and output. Owing to its high performance, Transformer has been an off-the-shelf architecture in natural language processing, computer vision [Dosovitskiy et al., 2020], and even other disciplines, like chemistry [Schwaller et al., 2019].

A Transformer is made up of an encoder and a decoder, which are both stacks of identical blocks. Each encoder block consists primarily of a multi-head self-attention module and a position-wise feed-forward network (FFN). A residual connection is used around each module to build a deeper model, followed by layer normalization. In decoder blocks, an additional multi-head attention module over the output of the encoder stack can be inserted between the multi-head self-attention modules and the FFNs. In addition, the self-attention sub-layer in the decoder stack is modified to prevent positions from attending to subsequent positions. Fig. 2.7 shows the overall architecture of the Transformer. The main modules in a Transformer are multi-head attention, position-wise FFN, residual connection and normalization, and positional encoding.

A key challenge of applying a Transformer is its inefficiency in processing long data mainly due to the computation and memory complexity of the self-attention module. Thus, adapting the Transformer to 3D point clouds is a challenging task.

2.3.1 Attention

The input of an attention module includes three vectors: Query, Key, and Value. The output is computed as a weighted sum of the values, with the weight assigned to each value determined by the query's compatibility function with the corresponding key.

The attention mechanism explored in a Transformer is called 'Scaled Dot-Product Attention'. Given the queries $\mathbf{Q}$ and keys $\mathbf{K}$ of dimension d_k, and values $\mathbf{V}$ of dimension d_v, the scaled dot-product attention can be calculated by:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \tag{2.13}$$

$\frac{1}{\sqrt{d_k}}$ is used to alleviate the gradient vanishing problem of the Softmax function because the magnitude of the dot products increases when d_k is a large value. $\text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)$ is often called 'Attention Map'.

Instead of utilizing a single attention function with d_m-dimensional queries, keys, and values, Transformer explores a multi-head attention mechanism, where the queries, keys, and values are linearly projected h times with d_k, d_k, d_v dimensions, respectively. Then, the attention function is performed in parallel on each of these projected versions of queries, keys, and values according to Eq. 2.13, yielding d_v -dimensional output values. The final values are obtained by concatenating all the outputs and projecting them back to the original d_m-dimensional representation. Formally, multi-head attention can be formulated as:

$$\begin{aligned} \text{MultiHeadAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}\left(\text{head}_1, \cdots, \text{ head }_H\right)\mathbf{W}^O, \\ \text{head }_i &= \text{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right), \end{aligned} \tag{2.14}$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_m \times d_v}$ and $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_m}$.

In Transformer, the self-attention strategy is used for the multi-head attention module. In a self-attention layer, all of the queries, keys, and values come from the same place.

2.3.2 Position-wise FFN

In addition to attention modules, Transformer has a fully connected feed-forward network that is applied to each position independently and identically. FFN consists of two linear transformations separated by a ReLU activation, which is formulated as follows:

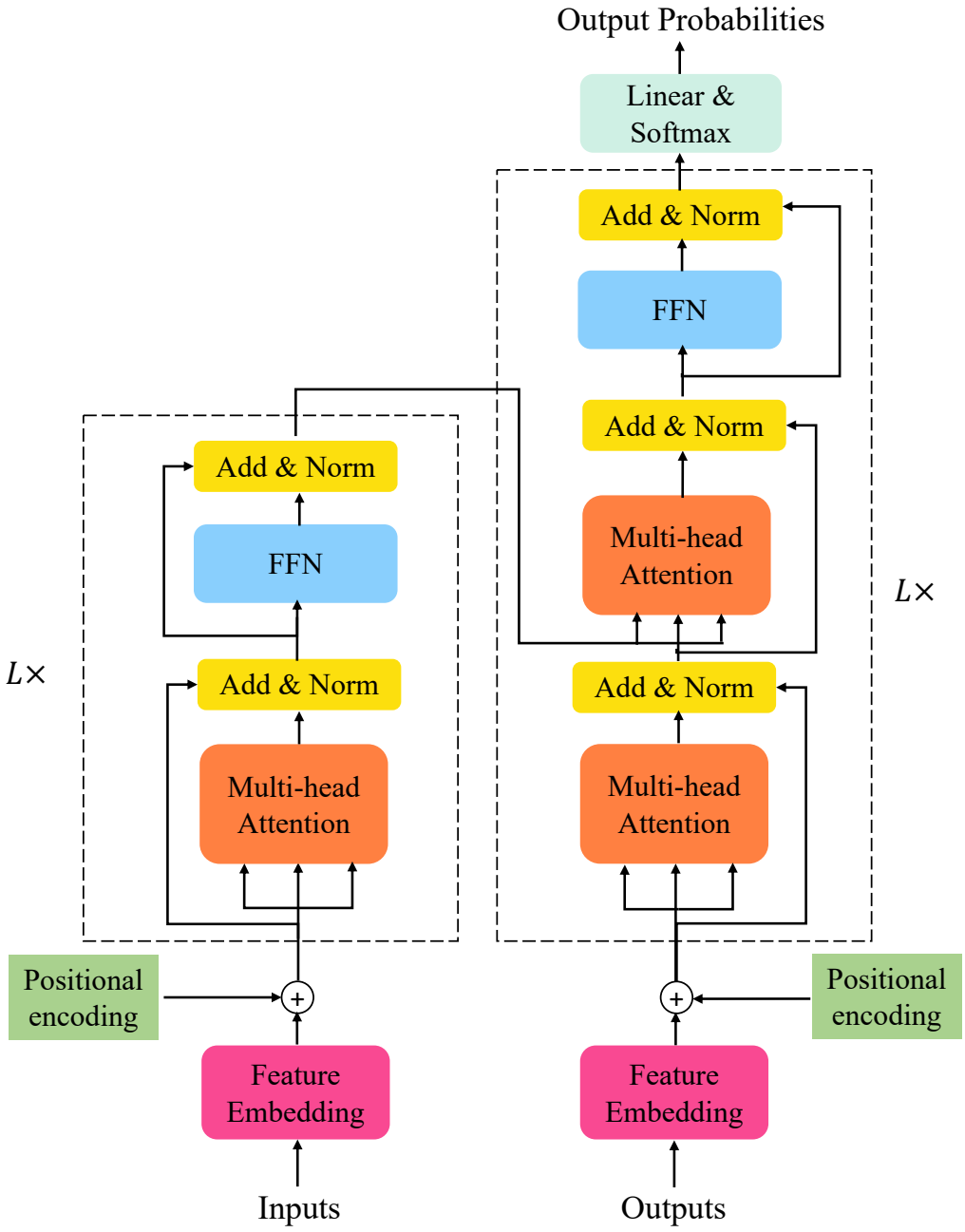


Figure 2.7: Overview of the Transformer architecture.

$$\mathrm{FFN}(\mathbf{H}) = \mathrm{ReLU}(\mathbf{H}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{2.15}$$

where $\mathbf{H}$ is the output vector of the previous layer, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$, and $\mathbf{b}_2$ are trainable parameters in linear layers.

2.3.3 Residual connection and normalization

Inspired by the residual connection design [He et al., 2016a] for building a deeper network, Transformer also employs a residual connection around each module and adds the layer normalization

layers. Thus, each encoder in Transformer, including a multi-head self-attention (MHSA) module and position-wise FFN, can be formulated as follows:

$$\begin{aligned} \mathbf{H} &= \text{LayerNorm}(\text{MHSA}(\mathbf{X}) + \mathbf{X}), \\ \mathbf{H}' &= \text{LayerNorm}\left(\text{FFN}\left(\mathbf{H}\right) + \mathbf{H}\right), \end{aligned} \tag{2.16}$$

where LayerNorm is the layer normalization layer.

2.3.4 Positional encoding

When modeling text sequences, word ordering is important, and it is thus critical to properly encode word positions. Thus, the information about the position of the tokens in the sequence should be injected to fully explore the ordering of the sequence. However, there is no recurrence or convolution (especially for the encoder) in Transformer. To address this problem, Transformer adopts an absolute sinusoidal positional encoding. For each position *pos*, the positional encoding is a sin/cos function of the *pos* with a pre-defined frequency.

$$\text{PE}(pos)_i = \begin{cases} \sin\left(\omega_i pos\right), & \text{if } i \text{ is even} \\ \cos\left(\omega_i pos\right), & \text{if } i \text{ is odd} \end{cases} \tag{2.17}$$

where ω_i is the pre-defined frequency for each dimension. The positional encoding of each position in the sequence is then added to the feature embeddings and fed to Transformer, as shown in Fig 2.7.

3 Point Cloud based Place Recognition

Place recognition and scene localization in large-scale and complex environments is a fundamental challenge with applications ranging from autonomous driving [Häne et al., 2017; Hee Lee et al., 2013] and robot navigation [Fu et al., 2018; Mur-Artal et al., 2015] to augmented reality [Liu et al., 2016a]. In this chapter, two point cloud based place recognition methods are reported. To be specific, the first method is a novel learning-based approach for identifying sub-maps from aggregated point clouds, which includes a point orientation encoding (PointOE) module, a self-attention unit, and a simple yet efficient loss function. The second one is designed for recognizing single raw LiDAR scans.

3.1 Problem statement

Let M_{ref} be a pre-built reference map of 3D point clouds defined with respect to a fixed reference frame, which is divided into a set of submaps/LiDAR scans such that $M_{ref} = \{m_i : i = 1, ..., M\}$. Notably, the submaps pre-built by aggregating multiple LiDAR scans are always covering the same length and include a fixed number of points (e.g. 4096), while the single scan has a larger area and a greater number of points. Each submap/LiDAR scan is tagged with a Universal Transverse Mercator (UTM) coordinates at its centroid/scanning position using GPS/INS. Let Q be a query point cloud with the same coverage with respect to a submap/LiDAR scan in M_{ref}. The point cloud based place recognition problem is defined as retrieving a submap/LiDAR scans m^* from M_{ref} that is structurally closest to Q. Note that under this formulation, Q is not a subset of M_{ref}, since they are independently scanned at different times.

To tackle this problem, a neural network is designed to learn a function $f(\cdot)$ that embeds a local point cloud to a 3D global descriptor of pre-defined size. The goal is to find a submap/LiDAR scan $m^* \in M_{ref}$ such that the distance between global descriptors $f(m^*)$ and $f(Q)$ is minimized:

$$m^* = \underset{m_i \in M_{ref}}{\operatorname{argmin}}\ d(f(Q), f(m_i)), \tag{3.1}$$

where $d(\cdot)$ is a distance metric (e.g., Euclidean distance). In practical implementation, a global descriptor dictionary is built offline for all 3D submaps/LiDAR scans. When a query submap/scan appears, the nearest submap/scan is obtained efficiently by comparing the global descriptor extracted online from the query scan with stored global descriptors.

3.2 Self-Attention and Orientation Encoding Network (SOE-Net)

The SOE-Net consists of three essential steps: the extraction of 3D local descriptors via a PointOE module, the aggregation of local descriptors via a self-attention unit and a NetVlad layer, and

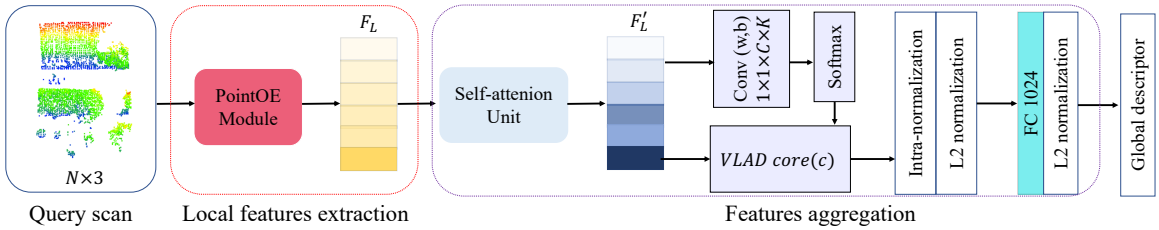


Figure 3.1: Overview of SOE-Net architecture. The network takes a query scan with N points as input and employs the PointOE module to extract point-wise local descriptors F_L. During descriptor aggregation, a self-attention unit is applied to the local descriptors and followed by the NetVLAD layer. Finally, a fully connected (FC) layer is adapted to compress the output descriptor vector, follow by the L2 normalization to produce a global descriptor.

the loss functions for training. Fig. 3.1 shows the overall network architecture of the proposed SOE-Net, where the local descriptor extraction part produces local descriptors from the 3D query scan, and the descriptor aggregation part aims to generate a distinct global descriptor. A detailed explanation of each step will be given in the following.

Given the input as a query point cloud with coordinates denoted as $Q = \{p_1, \cdots, p_N\} \in \mathbb{R}^{N\times 3}$, the designed PointOE module is first used to extract point-wise local descriptors. Unlike previous studies, it extracts relevant local information from eight directions to enhance point-wise feature representation, with details described in Section 3.2.1. Then, a self-attention unit in the descriptor aggregation part is proposed to encode the spatial relationship among point-wise local descriptors, which is explained in Section 3.2.2. Afterward, the NetVLAD layer is adopted to fuse enhanced local descriptors in Section 3.2.2. The training strategy with the proposed "Hard Positive Hard Negative quadruplet" (HPHN quadruplet) loss is presented in Section 4.3.4.

3.2.1 Local descriptor extraction

PointOE Module

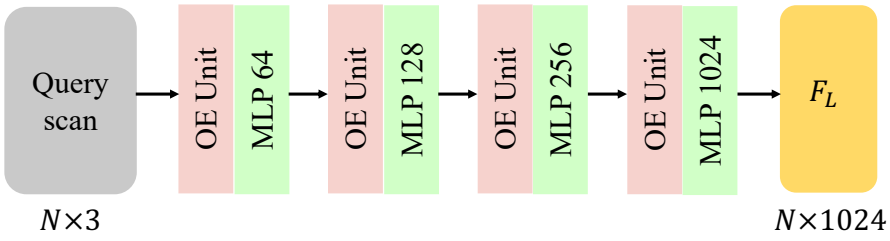


Figure 3.2: The architecture of PointOE module. The input point cloud passes through a series of OE units and MLPs, and local descriptors F_L are generated as output.

The successes of many non-learning-based image retrieval methods are owing to the design of great local image descriptors (e.g., SIFT [Lowe, 2004]). Orientation Encoding (OE) is one of SIFT's most shining highlights, which is also considered to benefit 3D feature description. Inspired by PointSIFT [Jiang et al., 2018], the OE unit is introduced to the proposed SOE-Net. Specifically, it is integrated into PointNet to improve the point-wise feature representation ability. Fig. 3.2 shows the detailed architecture of the PointOE module. To the best of our knowledge, no prior work has explored it for large-scale place recognition and its effectiveness for retrieval has not been verified.

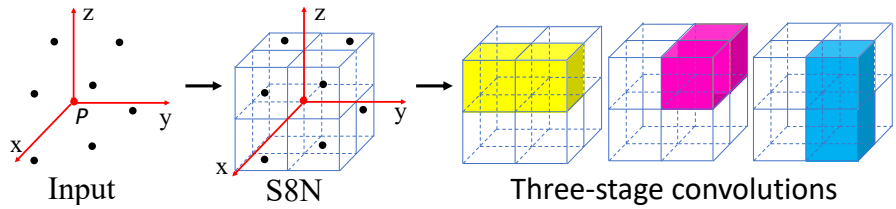


Figure 3.3: The workflow of OE unit.

The inputs to the PointOE module are the 3D coordinates of N points. Following [Qi et al., 2017a], multilayer perceptrons (MLP) are adapted to encode the input 3D coordinates into features of $[64, 128, 256, 1024]$ dimensions. The OE unit is inserted in front of each MLP to improve the representation ability. Local descriptors F_L are generated from this module.

Orientation-encoding Unit. Consider a $N \times C$ matrix as an input that describes a point cloud of size N with a C-dimensional feature for each point, the OE unit will output a feature map with the same dimension $N \times C$. Every point is assigned to a new C-dimensional feature, which integrates the local information from eight orientations. As shown in Fig. 3.3, the OE unit first adopts the Stacked 8-neighborhood Search (S8N) to find the nearest neighbors for a point P in each of the eight octants [Jiang et al., 2018]. Furthermore, the features are extracted using three-stage convolutions from those neighbors, which lie in a $2 \times 2 \times 2$ cube along the $x-, y-, z-$ axis. Formally, these three-stage convolutions are defined as:

$$\begin{aligned} OE_x &= \text{ReLU}(\text{Conv}(w_x, V, b_x)), \\ OE_{xy} &= \text{ReLU}(\text{Conv}(w_y, OE_x, b_y)), \\ OE_{xyz} &= \text{ReLU}(\text{Conv}(w_z, OE_{xy}, b_z)), \end{aligned} \tag{3.2}$$

where $V \in \mathbb{R}^{2\times2\times2\times C}$ are the feature vectors of neighboring points. $w_x \in \mathbb{R}^{2\times1\times1\times C}, w_y \in \mathbb{R}^{1\times2\times1\times C}$ and $w_z \in \mathbb{R}^{1\times1\times2\times C}$ are weights of the three-stage convolutions, b_x, b_y, b_z are the biases of convolution operators. In this way, the OE unit captures the local geometric structure from eight spatial orientations.

3.2.2 Feature aggregation

Self-attention Unit

To introduce long-range context dependencies after extracting local descriptors, a self-attention unit [Zhang et al., 2019a] is designed before fusing them into the NetVLAD layer. The self-attention unit can encode meaningful spatial relationships between local descriptors. Fig. 3.4 presents its architecture. Given local descriptors $F_L \in \mathbb{R}^{N\times C}$, where N is the number of points and C is the number of channels, F_L is fed into two MLPs respectively and generate the new feature maps $X \in \mathbb{R}^{N\times C}$, $Y \in \mathbb{R}^{N\times C}$. Then, the attention map W is calculated, defined as follows:

$$W_{j,i} = \frac{\exp(Y_j \cdot X_i^T)}{\sum_{i,j=1}^{N} \exp(Y_j \cdot X_i^T)}, \tag{3.3}$$

where $W_{j,i}$ indicates that the i^{th} local descriptor impacts on j^{th} local descriptor, with the shape of $N \times N$. Here, it deems as the component that learns the long-range dependency relationship among point-wise local descriptors. More important local descriptors will contribute more to the representation of the target global descriptor. On the other hand, F_L is fed into another MLP to output a new feature map $Z \in \mathbb{R}^{N\times C}$. Afterward, it is multiplied with the transpose of W to

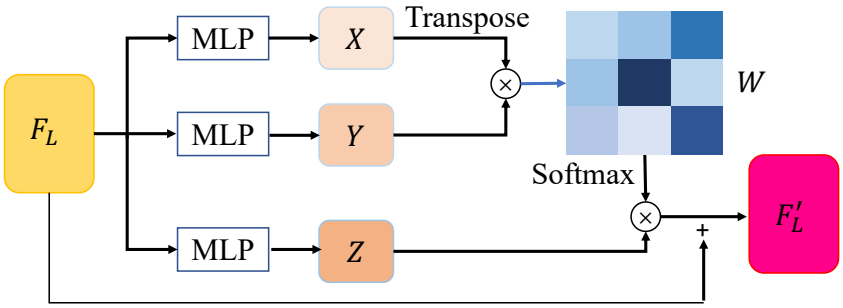


Figure 3.4: The workflow of the self-attention unit.

generate the result $A^P \in \mathbb{R}^{N\times C}$. Finally, a scale parameter α is added on it and added back F_L, which can be defined as follows:

$$F'_L = \mu A^p + F_L = \mu W^T Z + F_L, \tag{3.4}$$

where μ is initialized as zero and gradually assigned more weights with the progress of learning. The final output has a global context view compared with the original local descriptors. This enhances feature integration by combining geometrical and contextual information.

NetVLAD Layer

In this module, the aim is to aggregate the local descriptors into a discriminative and compact global one. Following the configuration in [Angelina Uy & Hee Lee, 2018], a NetVLAD layer is adopted to fuse features. The NetVLAD layer learns K visual words, denoted as $\{v_1, \cdots, v_K | v_k \in \mathbb{R}^C\}$, and generates a $(C \times K)$-dimensional VLAD descriptor. However, the VLAD descriptor is time-consuming for nearest neighbor search, thus a fully connected layer is applied to generate a more compact global descriptor with an L_2 normalization.

3.2.3 Loss function

Before going into the details of the proposed HPHN-quadruplet loss, a short review of the quadruplet loss [Chen et al., 2017] and its improvement are given. To compute the quadruplet loss, each batch of the training data includes T quadruplets. Each quadruplet is denoted as $\Gamma_q = (\delta_a, \delta_p, \delta_n, \delta_n^*)$, where δ_a is an anchor point cloud, δ_p a positive point cloud (structurally similar to the query), δ_n a negative point cloud (structurally dissimilar to the query), δ_n^* a randomly sampled point cloud that is different with $\delta_a, \delta_p, \delta_n$. The quadruplet loss is formulated as:

$$\begin{aligned} L_q = &\frac{1}{T}\sum^{T}\left[||f(\delta_a)-f(\delta_p)||_2^2-||f(\delta_a)-f(\delta_n)||_2^2+\alpha\right]_+ \\ &+\frac{1}{T}\sum^{T}\left[||f(\delta_a)-f(\delta_p)||_2^2-||f(\delta_n^*)-f(\delta_n)||_2^2+\beta\right]_+, \end{aligned} \tag{3.5}$$

where $[\cdots]_+$ denotes the hinge loss, α and β are the constant margins. The first term is a triplet loss which focuses on maximizing the feature distance between the anchor point cloud and the negative point cloud. The second term focuses on maximizing the feature distance between the negative point cloud and the additional point cloud δ_n^*.

To make the positive and negative samples in the quadruplet more effective, the quadruplet loss is extended to the lazy quadruplet loss [Angelina Uy & Hee Lee, 2018] by introducing hard sample mining. The quadruplets now become $\Gamma_{lq} = (\delta_a, \{\delta_p\}, \{\delta_n\}, \delta_n^*)$, where $\{\delta_p\}$ is a collection of ϕ positive point clouds and $\{\delta_n\}$ is a collection of φ negative point clouds. The loss is modified accordingly to

$$\begin{aligned} L_{lq} = & \max_{\substack{i=1\ldots\phi \\ j=1\ldots\varphi}} \left(\left[||f(\delta_a) - f(\delta_p^i)||_2^2 - ||f(\delta_a) - f(\delta_n^j)||_2^2 + \alpha \right]_+ \right) \\ & + \max_{\substack{i=1\ldots\phi \\ j=1\ldots\varphi}} \left(\left[||f(\delta_a) - f(\delta_p^i)||_2^2 - ||f(\delta_n^*) - f(\delta_n^j)||_2^2 + \beta \right]_+ \right). \end{aligned} \tag{3.6}$$

In practice, a common strategy is to set β to be smaller than α (e.g., $\alpha = 0.5, \beta = 0.2$) to make the second term in Eq. 3.6 a relatively weaker constraint. However, this practice is less justified, especially in the scenario of metric learning for large-scale place recognition. In this work, a Hard Positive Hard Negative quadruplet loss (HPHN quadruplet) is proposed, which unifies the margin selection for δ_a and δ_n^*, and meanwhile relies on the hardest positive and the hardest negative samples in the batch to compute the learning signal. In this case, the hardest positive point cloud δ_{hp} is the least structurally similar to the anchor point cloud, which is defined as:

$$\delta_{hp} = \underset{\delta_p^i \in \{\delta_p\}}{\operatorname{argmax}} \; ||f(\delta_a) - f(\delta_p^i)||_2^2, \tag{3.7}$$

The hardest negative point cloud is the most structurally dissimilar to the anchor point cloud. Here, the first finding is the hard negative point cloud δ_{hn} in $\{\delta_n\}$, which is defined as:

$$\delta_{hn} = \underset{\delta_n^j \in \{\delta_n\}}{\operatorname{argmin}} \; ||f(\delta_a) - f(\delta_n^j)||_2^2. \tag{3.8}$$

Additionally, the feature distance is considered from δ_n^* to δ_n:

$$\delta_{hn}^{'} = \underset{\delta_n^j \in \{\delta_n\}}{\operatorname{argmin}} \; ||f(\delta_n^*) - f(\delta_n^j)||_2^2. \tag{3.9}$$

Finally, one of them is selected as the hardest negative training data, which has the minimum distance d_{hn}:

$$d_{hn} = \min(||f(\delta_a) - f(\delta_{hn})||_2^2, ||f(\delta_n^*) - f(\delta_{hn}^{'})||_2^2). \tag{3.10}$$

In conclusion, the HPHN quadruplet loss can be formulated as:

$$L_{HPHN} = \left[||f(\delta_a) - f(\delta_{hp})||_2^2 - d_{hn} + \gamma \right]_+, \tag{3.11}$$

where γ is the unified margin. The first term in Eq. 3.11 is the upper bound of the feature distance of all the positive point cloud pairs, and the second term is the lower bound of the feature distance of all the negative point cloud pairs in a batch.

Although having a form similar to the triplet loss, the loss is still a quadruplet loss that is computed from the sampled quadruplet. Compared with the lazy quadruplet loss, the proposed HPHN quadruplet loss picks the harder term between Eq. 3.8 and Eq. 3.9, instead of using both in the loss computation. Moreover, the same margin is used when either of both is selected. Despite this simple modification, the experimental results in Chapter 7 demonstrate that the proposed HPHN quadruplet loss significantly outperforms the lazy quadruplet loss.

3.2.4 Implementation details

The proposed SOE-Net is implemented in the TensorFlow framework and trained on a single Nvidia Titan Xp GPU with 12G memory. The size of the input points is 4096. The margins γ for the HPHN quadruplet loss are set to 0.5. Similar to all previous methods, the number of clusters K in the NetVLAD layer is set to 64. In the training stage, the batch size is set to 1 in each training iteration. Adam optimizer is used in the models for epoch 20. Same as PCAN [Zhang & Xiao, 2019], two positive point clouds and nine negative point clouds (including one other negative point cloud) are chosen in calculating loss functions. The initial learning rate is set to 0.0005. It decays by 0.7 after every 200K steps.

3.3 Cross Attention Single Scan Place Recognition (CASSPR)

Although previous methods [Angelina Uy & Hee Lee, 2018; Zhang & Xiao, 2019; Xia et al., 2021a] achieve excellent performance on large-scale point cloud based place recognition task, they are trained and evaluated on the 3D reference submaps, which was pre-built by aggregating multiple LiDAR scans. The submaps are always covering 20m in length and include a fixed number of points (e.g. 4096). In practice, the real stored data is actually a combination of single LiDAR scans that cover a 160-200 meter size area and have up to 260k points for Ouster OS1-128. The larger area and greater number of points bring a bigger challenge for achieving point cloud based place recognition. Recently, MinkLoc-SI is a pioneering network for place recognition tasks based on a single LiDAR scan. However, it is only an extension of MinkLoc3D [Komorowski, 2021], a 3D voxel-based learning model using Minkowski Engine [Choy et al., 2019a]. Thus, it still can not solve the two problems: *1) The global descriptors generated by 3D voxelization and 3D convolution networks must suffer the geometric loss due to quantization. 2) The long-range contextual dependencies are ignored by 3D full convolution networks.*

Fig. 3.5 shows the overall network architecture of the CASSPR. It is a two-branch network including a Hierarchical Cross-Attention Transformer(HCAT) module and lightweight self-attention (LSA) units. The point-based branch (Top in Fig. 3.5) preserves the fine-grained geometrical features via running simple per-point MLPs. The voxel-based branch is based on Minkloc3D-SI [Żywanowski et al., 2021], including a local feature extraction network and generalized-mean (GeM) pooling layer. The local feature extraction network is first built with a spherical representation, as explained in Section 3.3.1. Then, a Feature Pyramid Network (FPN) is designed as a pattern to extract high-resolution local descriptors with a large receptive field, including six convolutional layers and one transposed convolution layer (TConv). Point-voxel feature fusion happens due to the HCAT module, with details described in Section 3.3.2. In addition, a lightweight self-attention unit in the local feature extraction part is introduced to encode the spatial relationship among local descriptors, as explained in Section 3.3.3. The training strategy with the loss function is presented in Section 3.3.4.

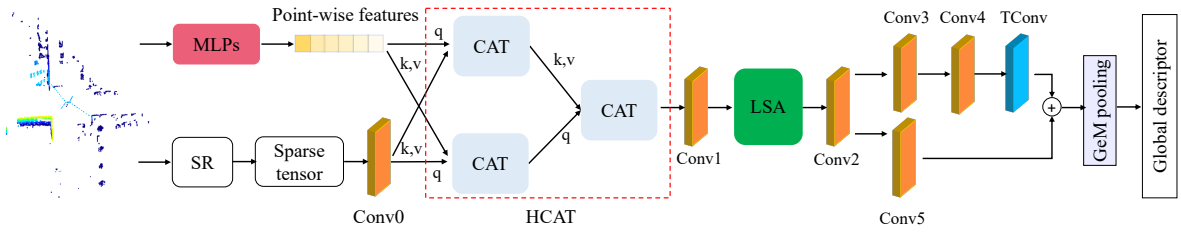


Figure 3.5: The overall network architecture of CASSPR.

3.3.1 Spherical representation and point-voxel fusion

A natural alternative to extracting geometric features from point clouds is to utilize the 3D voxelization on Cartesian coordinate systems and 3D convolution networks. This solution is suitable for indoor scenes with dense and uniform-density points. However, the density of 3D point clouds in urban scenes collected by LiDAR sensors is uneven, where the area closer to the scanner location has much greater density than a far-away region. Consequently, previous 3D voxelization methods based on Cartesian coordinates have had difficulty extracting fruitful features by treating the point cloud as a uniform one and splitting it via the uniform voxel [Zhu et al., 2021].

Motivated by this, a spherical-voxelization method introduced in Minkloc3D [Żywanowski et al., 2021] is used in this work. Each 3D point (x, y, z) on Cartesian coordinate system is transformed to a spherical representation $(\gamma, \theta, \varphi)$ on the spherical coordinate system:

$$\begin{aligned} \gamma &= \sqrt{x^2 + y^2 + z^2}, \\ \theta &= atan2(y, x), \\ \varphi &= atan2(z, \sqrt{x^2 + y^2}), \end{aligned} \tag{3.12}$$

where γ is the distance between the point and the sensor location, θ is the horizontal scanning angle and φ is the vertical scanning angle. With the spherical representation, the grid size would be increased to cover the farther-away area, thus more evenly distributing the points across different regions and giving a more balanced picture against the varying density. Next, the point cloud with coordinates $(\gamma_i, \theta_i, \varphi_i)$ is quantized into a finite number of voxels. The quantization loss is introduced since each cuboid only has one point. A single sparse tensor is then created by the quantized points and an associated feature via Minkowski Engine [Choy et al., 2019a]. The values of the associated feature are initialized to one for non-empty voxels.

Although the above spherical-voxelization operation can effectively maintain the geometric structure from LiDAR scans, the quantization loss cannot be ignored since many points are lost. In order to quantify the information loss, an experiment is conducted on how the number of points changes in the process. As shown in Fig. 3.6, the difference between the original point cloud and its quantization version is visualized. About 2000 out of 4096 points are removed on average for being too close to other points with a quantization size of $(\gamma_i, \theta_i, \varphi_i) = (2.5, 2.0, 1.875)$ proposed in [Żywanowski et al., 2021].

Observed this, an efficient primitive is proposed to extract more informative high-level features from raw single scans, which compensates for the information loss in the voxel-based branch. As illustrated in Fig. 3.5, the upper point-based branch extracts the point-wise features for each individual point via MLPs. The lower voxel-based branch first generates a sparse tensor using the above spherical-voxelization method. Then one 3D convolutional block is used to aggregate the neighboring points, producing sparse feature maps with increasing receptive fields. However, how to fuse the features from different branches is still a challenge.

3.3.2 Hierarchical cross-attention transformer

To efficiently explore the inter-view relationship, the relationship between volumetric view and point view, a Hierarchical Cross-Attention Transformer (HCAT) is proposed to fuse the two features from different views in a unified model. As shown in Fig. 3.7, the HCAT consists of three Cross Attention Transformers (CAT). Each CAT has different roles. The first CAT takes the sparse-voxel feature maps as the query and the point-wise features as the key and value. It extracts point-wise features with reference to sparse feature maps and outputs new enhanced

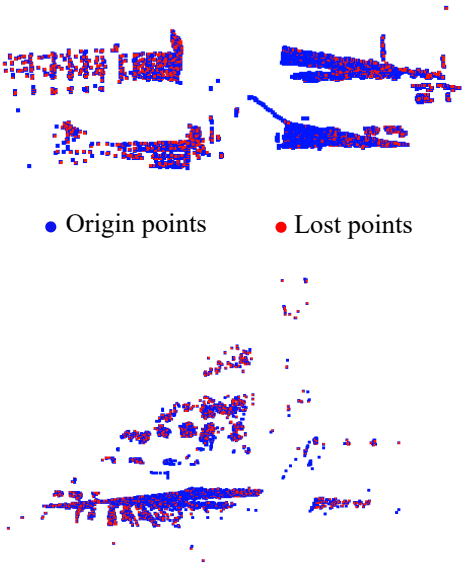


Figure 3.6: Visualization of the points difference between the original and after quantization.

sparse tensors. Oppositely, another CAT produces new enhanced point-wise features by taking the point-wise features as the query and the sparse-voxel features as the key and value. Finally, the aim of the third CAT is to fuse the enhanced sparse tensors and the enhanced point-wise features. The CAT is a residual module, consisting of two sub-layers: Multi-Head Cross-Attention (MHCA) and Feed-Forward Network (FFN). The feed-forward network includes two linear transformation layers with a ReLU activation function in between. Formally, each CAT can be formulated as:

$$\begin{aligned} \mathbf{F}_{CA} &= \mathrm{CAT}(\mathbf{Q}, \mathbf{K}, \mathbf{V}), \\ \mathbf{F}_{CA} &= \widetilde{\mathbf{F}}_{CA} + \mathrm{FFN}\left(\widetilde{\mathbf{F}}_{CA}\right), \\ \widetilde{\mathbf{F}}_{CA} &= \mathbf{Q} + \mathrm{MHCA}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right), \end{aligned} \tag{3.13}$$

where $\mathbf{Q} = F_s \in \mathbb{R}^{N_s \times d}$ is the query, $\mathbf{K} = \mathbf{V} = F_p \in \mathbb{R}^{N_p \times d}$ are the key and value.

In the MHCA layer, the cross-attention is achieved by projecting the $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$with h times. Specially, the weight matrix with 'Scaled Dot-Product Attention' [Vaswani et al., 2017] is calculated for the query, key, and value, which can be formulated as

$$\mathrm{Attention}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = \mathrm{softmax}\left(\frac{\mathbf{Q} + \mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{3.14}$$

Next, the values for h heads are calculated and concatenated together:

$$\text{Multi-Head}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\ \mathrm{head}_1, \ldots, \mathrm{head}_h]\, \mathbf{W}^O, \tag{3.15}$$

$$\mathrm{head}_i = \mathrm{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right), \tag{3.16}$$

where $\mathbf{W}$ is the weight of the attention. $\mathbf{W}_i^Q \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d \times d_v}$, and $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_k}$.

With the above HCAT, the point-wise features from the point-based branch can attend to the sparse tensors from the voxel-based branch. Hence, the quantization loss can be compensated and more accurate features will be got via the HCAT module.

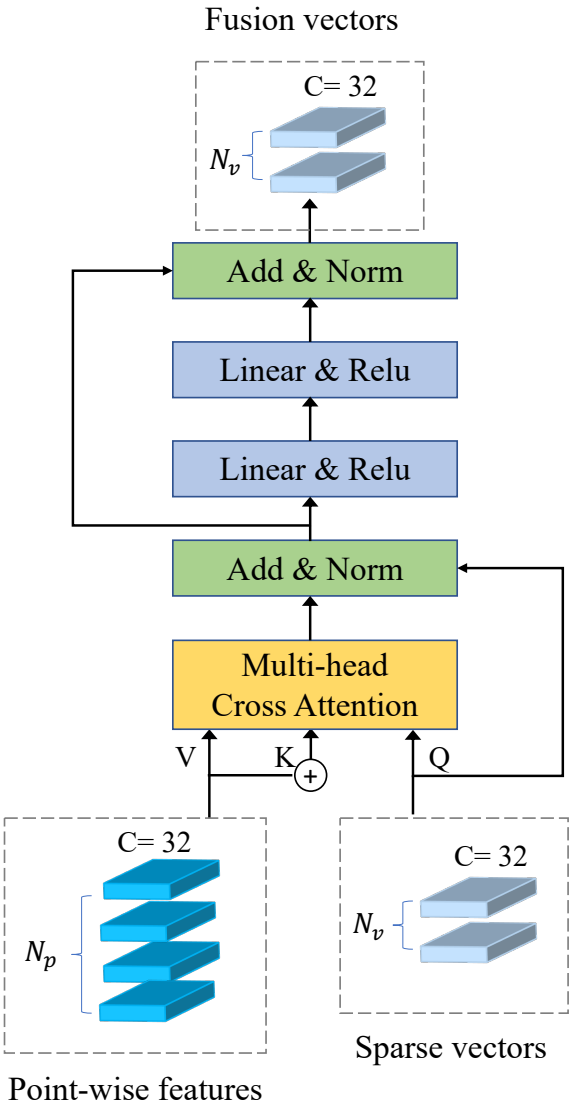


Figure 3.7: The architecture of one Cross-Attention Transformer (CAT). It takes the sparse-voxel vectors as a query and the point-wise features as key and value.

3.3.3 Lightweight self-attention unit

Before going to the details of the proposed lightweight self-attention unit, a short review of the previous self-attention used is given in 3D place recognition or point Transformers [Zhao et al., 2021; Guo et al., 2021]. Given a point-wise local descriptor $D_{local} = \{(p_i, f_i)\}_{i=1}^{N}$, where p_i is the i-th point coordinate and f_i is the feature of p_i. N is the number of points. A dot-product self-attention (DPSA) unit can be formulated as follows:

$$\begin{aligned} \text{DPSA}\left(D_{local}\right) = & \sum_{f_j \in D_{local}} \text{Softmax}[\delta(p_i - p_j)) \\ & + f_i W_i^Q (f_j W_j^K)^T] f_j W_j^V, \end{aligned} \tag{3.17}$$

where W_i^Q, W_j^K, W_j^V are learned matrices, and δ is the positional encoding for point coordinates.

However, computing DPSA directly by multiplying two $N \times C$ matrices is expensive, with $O(N^2)$ space complexity and $O(N^{2.34})$ time complexity [Alman & Williams, 2021]. This quadratic dependency on the number of points has brought a challenge for large-scale 3D place recognition based on a single scan since N is much greater in this work. Additionally, in order to compute the point-based positional encoding one needs to search for K-nearest neighbors, which can be achieved with $O(N \log N)$ space complexity [Park et al., 2022]. This has also become a bottleneck for processing large-scale point clouds.

To make self-attention more effective, a lightweight self-attention (LSA) unit is introduced to reduce both the training and inference time/memory consumption inspired by [Park et al., 2022]. In specially, a memory-efficient positional encoding is introduced to reduce space complexity.

Positional encoding. The input point cloud is voxelized to a set of M triplets $\mathbb{V} = \{(v_i, g_i, c_i)\}_{i=1}^{M}$, including the i-th voxel coordinate v_i, the corresponding voxel feature g_i, and the centroid coordinate c_i of this voxel. Note that $M < N$ since the voxelization step will lose some points. This fact would also imply lower space and time complexity. The lightweight self-attention on c_i can be formulated based on Eq. 3.20:

$$\text{LSA}\,(c_i) = \sum_{g_j \in D_{local}} \text{softmax}\left[g_i W_i^Q (g_j W_j^K)^T + \delta(c_i - c_j))\right] g_j W_j^V, \tag{3.18}$$

where $\delta(c_i - c_j) \in \mathbb{R}^{M \times D}$. D is the dimensions of the positional encoding layers.

Finding neighbor K voxels via voxel hashing will be quick since it only has $O(M)$ time complexity. However, implementing $\delta(c_i, c_j)$ directly using MLP in requires $O(MKD)$ space complexity. To alleviate this problem, a coordinate decomposition approach is inspired by [Park et al., 2022]. Given a query voxel (v_i, g_i, c_i) and a nearest neighbor voxel (v_j, g_j, c_j), the relative position encoding between c_i and c_j can be decomposed as following:

$$\delta(c_i - c_j) = \delta\left[(c_i - v_i) + (v_i - v_j) - (c_j - v_j)\right]. \tag{3.19}$$

Seen in Eq. 3.19, the memory-consuming $\delta(c_i - c_j)$ is decomposed to three parts $\delta(c_i - v_j)$, $\delta(v_i - v_j)$, and $\delta(c_j - c_j)$. It can analyze that: (1) The space complexity of $\delta(c_i - v_j)$ will be $O(MD)$ because of the continuity of c; (2) The positional encoding $\delta(v_i - v_j)$ is more memory-efficient, whose space complexity is $O(KD)$. Note that $K << M$, there can be only K different discretized relative positions of $(v_i - v_j)$. (3) the additional space complexity for $\delta(c_j - c_j)$ does not need to be added since the $\delta(c_i - v_j)$ has been computed before. In conclude, the space complexity of $\delta(c_i, c_j)$ is reduced from $O(MKD)$ to $O(MD + KD)$. The detailed decomposition is visualized vividly in Fig. 3.8.

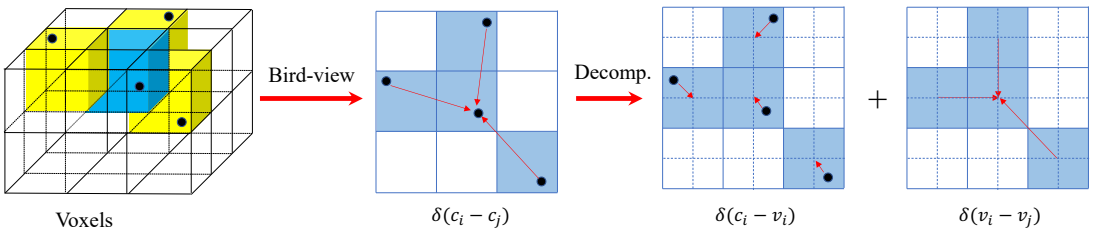


Figure 3.8: Decomposition of positional encoding. The continuous positional encoding $\delta(c_i - c_j)$ is decomposed to $\delta(c_i - v_i)$ and $\delta(v_i - v_j)$. This decomposition can reduce the space complexity from $O(MKD)$ to $O(MD + KD)$.

Same as the local self-attention layer in [Park et al., 2022], the cosine similarity is used instead of the Softmax function in Eq. 3.18. Thus, the Eq. 3.18 can be rewritten as:

$$\mathrm{LSA}\left(c_i\right) = \sum_{g_j \in D_{local}} \mathrm{cosine}\left[g_i W_i^Q (g_j W_j^K)^T + \delta(c_i - c_j))\right] g_j W_j^V . \tag{3.20}$$

3.3.4 Loss function

Same as in Minkloc3D [Żywanowski et al., 2021], a triplet margin loss based on the batch hard negative mining approach is introduced. Hash-based indexing is first utilized to quickly determine whether two scans are structurally similar, or dissimilar, or if the similarity is indefinite. A batch of scans with size n is built by sampling $n/2$ structurally similar scans. Then, the batch is fed into CASSPR to compute global descriptors. Notably, two $n \times n$ boolean masks are computed, one indicating structurally similar pairs and the other structurally dissimilar pairs. The informative training triplets are constructed based on the boolean masks. Besides, only the hardest positive and the hardest negative scans are mined in one batch. The hard mining triples loss can be formulated as:

$$L = MAX\left(m + d\left(\psi_a, \psi_p\right) - d\left(\psi_a, \psi_n\right), 0\right), \tag{3.21}$$

where ψ_a is an anchor point cloud, ψ_p a hardest positive point cloud (structurally similar to the anchor), ψ_n a hardest negative point cloud (structurally dissimilar to the anchor). m is a pre-defined margin parameter.

4 3D Vehicle Detection and Tracking

In this chapter, the methods for vehicle detection and tracking of moving objects (DTMO) using MLS point clouds are presented. The aim is to detect vehicles and estimate the location of moving objects by identifying the trajectory across all frames. The DTMO framework follows a two-step strategy. In the first step, the locations of vehicles are detected using a PointPillars [Lang et al., 2019] method. In the second step, given an initial bounding box of a template object in the first frame from LiDAR scans, a novel lightweight and detector-free 3D single object tracking method named DMT (Detector-free Motion prediction based 3D Tracking) is proposed.

4.1 Problem statement

In this thesis, DTMO can be split into two sub-tasks: object detection and single object tracking. Given a LiDAR scan $P = \{P_1, P_2, ..., P_N | P_i = (x, y, z)\}$, where N is the number of points, the object detection task is defined as locating and classifying the object (e.g. vehicle) in this scan, which aims to predict the 3D bounding box of the object $B = \{x_c, y_c, z_c, h, w, l, \theta\}$. (x_c, y_c, z_c) is the center coordinates of the 3D bounding box, (h, w, l) is the height, width, and length respectively, and θ is the orientation of the bounding box.

Then, the predicted B can be regarded as an initial 3D bounding box B_{init} of the object in the first frame, and let $Q = \{Q_i\}_{i=1}^{M}$ be a query point cloud created by cropping and centering the object in the first frame with B_{init}. The single object tracking task is defined as locating the same object in the search point cloud $P = \{P_i\}_{i=1}^{N^*}$ given the B_{init} frame by frame. M and N^* are the numbers of points in the query point cloud and search point cloud, respectively. Formally, previous state-of-the-art 3D single object trackers [Qi et al., 2020; Zheng et al., 2021a] can be formulated as:

$$Tracker\,(Q, P, B_{init}) \rightarrow (\hat{x}, \hat{y}, \hat{z}, \hat{\theta}), \tag{4.1}$$

where $Q \in \mathbb{R}^{M\times 3}$, $P \in \mathbb{R}^{N\times 3}$ and $B_{init} \in \mathbb{R}^7$. Notably, only the center coordinates and orientation $(\hat{x},\ haty, \hat{z}, \hat{\theta})$ of the target are predicted since the height, width, and length of the object is assumed to be the same in other frames.

The previous trackers employ off-the-shelf detectors on scanned point clouds for target detection. They may easily drift when the point clouds are relatively sparse or incomplete. In this thesis, the potential target center is proposed to predict in a free point-cloud way, which fully explicitly leverages motion cues from previous target states $S_{prev} = \{S_1, S_2, \cdots, S_{t-1}\}$, where the state S_t is the predicted center coordinates in the t-th frame. The whole process is formulated as:

$$Tracker\,(Q, P, B_{init}, \mathcal{M}(S_{prev})) \rightarrow (\hat{x}, \hat{y}, \hat{z}, \hat{\theta}), \tag{4.2}$$

where $\mathcal{M}(\cdot)$ is a motion prediction function that estimates a potential target center in the current frame based on previous target states.

4.2 3D vehicle detection

Fig. 4.1 shows the pipeline of the vehicle detector, which includes data pre-processing, detection network, and post-processing modules. The input point clouds are first filtered by removing points that are not in the front view in the pre-processing module. The filtering strategy reduces many points and improves the computing efficiency of the detection network. Then, a data augmentation method is used to add many training samples in case of over-fitting. Next, an efficient one-stage 3D object detector PointPillars [Lang et al., 2019] to predict the bounding boxes of vehicles. PointPillars consists of a Pillar feature network, a 2D CNN backbone, and a 3D detection head. Finally, the post-processing operations, including score filtering and Non-Maximum Suppression (NMS), are utilized to improve the detection performance in the inference stage.

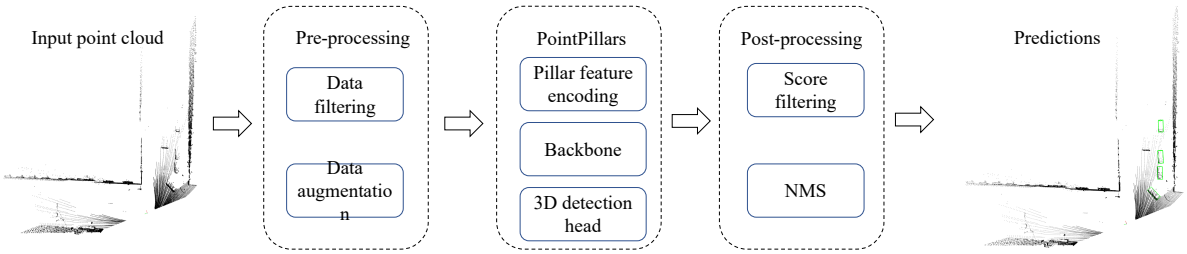


Figure 4.1: Pipeline of the vehicle detector. The pre-processing filters the points outside the front view and augments the training data with additional samples. Next, the PointPillars network is used to estimate 3D bounding boxes for the vehicles. Finally, the score-filtering and NMS methods are introduced to improve detection results.

4.2.1 Pre-processing

The data pre-processing step consists of data filtering and data augmentation operations. The data is first filtered using viewing-frustum culling, which involves constructing the front viewing frustum and removing LiDAR points that are outside of it. Specifically, six surfaces and their corresponding normal vectors are first created based on the calculated eight corners of the field of view. The relative position (inside or outside) between a point and the surface of the apparent cone can be determined by the angle between the vector formed by the point and the surface and the surface normal vector. The point is inside if the angle is obtuse; otherwise, it is outside.

Following SECOND [Yan et al., 2018], the data augmentation includes three strategies: sampling ground truths from the database, global rotation and scaling, and object noise. Because there are too few ground truths during training, the goal of sampling ground truths from the database is to increase the number of ground truths and simulate objects that exist in various environments. A database containing labeled point clouds within the ground truth 3D bounding box is first generated from the training dataset. Then, several labeled point clouds randomly selected from this database are introduced into the current training scans via concatenation. Notably, a collision test is conducted to avoid physically impossible cases. In addition, global rotation and scaling are applied to all ground-truth 3D bounding boxes and scans. The scaling rate is randomly drawn from a uniform distribution [0.95, 1.05], and the global rotation angle is from $[-\pi/4, \pi/4]$. Furthermore, the noise is also introduced into each ground-truth box and its corresponding point clouds independently and randomly, instead of adding noise on all point clouds. The rotations are randomly sampled from a uniform distribution $[-\pi/2, \pi/2]$. The linear translations are drawn from a Gaussian distribution (0, 0.25).

4.2.2 PointPillars

Pillar feature encoding

The input point cloud is first discretized into a set of pillars N_p with the grid size $H \times W$, and each pillar has a fixed number of points, denoted as P_n. One raw point in pillars is augmented with $x_p, y_p, z_p, x_{off}, y_{off}$, where x_p, y_p, z_p is the 3D coordinates of the pillar center, and x_{off}, y_{off} is the offset from the pillar x, y center. Notably, due to the sparsity of the point cloud, the set of pillars will be mostly empty, and the non-empty pillars will have few points in general.

The augmented points are first passed through a simplified version of PointNet [Qi et al., 2017a] to create a tensor with the size of (N_p, P_n, C), where C is the feature channels. Then a max-pooling layer is used to generate an output feature of size (N_p, C). Finally, the feature is scattered back to the original pillar locations to create a pseudo-image of size (H, W, C). Fig. 4.2 shows the architecture of the pillar feature encoding network.

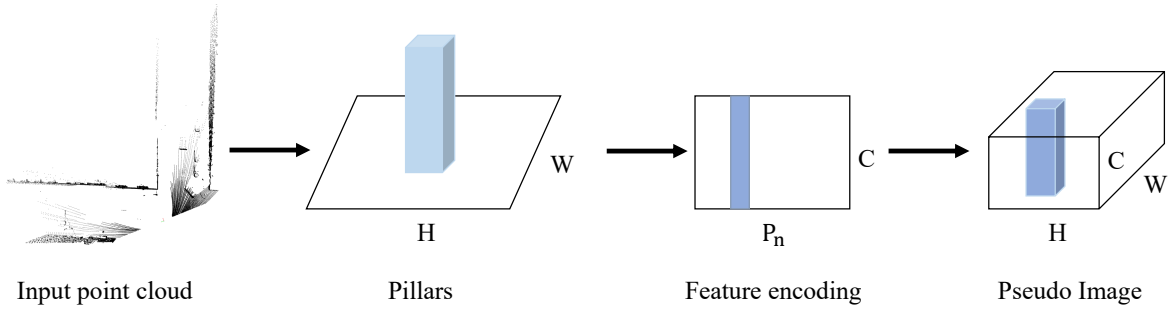


Figure 4.2: Network architecture of the pillar feature encoding. The raw point cloud is first discretized into evenly stacked pillars. Then, the pillars are encoded and scattered back to a 2D pseudo-image.

2D backbone

The 2D CNN backbone has a similar architecture to VoxelNet [Zhou & Tuzel, 2018b], including a top-down sub-network and a down-top sub-network. Fig 4.3 shows the detailed network architecture. The top-down network consists of a series of downsampling blocks. Each block has 2D convolutional layers with the kernel size of 3×3, each followed by Batch Normalization and a ReLu layer. The size of the feature map after each downsampling block is cut in half, but the number of output channels is doubled.

In the down-top network, the transposed 2D convolutional layers are used to upsample the feature maps from the top-down sub-network. Batch Normalization and ReLu layers are also applied to the upsampled features. The final output features are a concatenation of all multi-stage features derived from various strides.

3D detection head

Same as Single Shot Detector (SSD) [Liu et al., 2016b], the pre-defined anchors are used to match the ground-truth boxes using 2D Intersection over Union (IoU). The height and elevation of bounding boxes are not used for matching; instead, in a two-dimensional match, the height and elevation become additional regression targets.

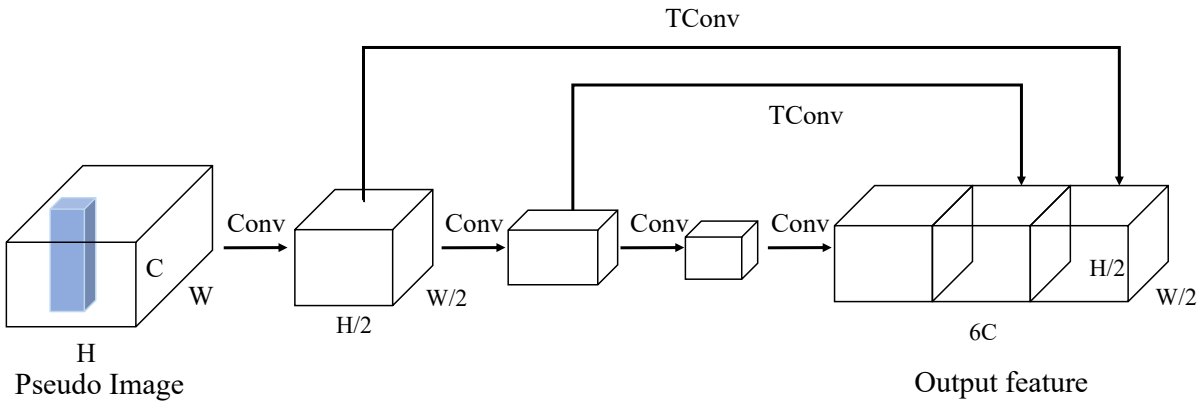


Figure 4.3: Network architecture of the 2D backbone. 'TConv' is a transposed 2D convolutional layer.

Loss function

The training loss includes three components: localization loss, angle loss, and classification loss. The aim of localization loss is to supervise the predicted 3D bounding box. However, the localization loss cannot solve the issue of flipped boxes: Radians 0 and π correspond to the same box but result in a large loss when one is misidentified as the other. An angle loss [Yan et al., 2018] is introduced to address this problem. The object classification loss is to distinguish the labels of objects.

Localization loss. The Huber (smooth-L1 loss) is used to supervise the regression residuals δb between ground-truth 3D bounding boxes $(x_{gt}, y_{gt}, z_{gt}, w_{gt}, h_{gt}, l_{gt}, \theta_{gt})$ and predictions $(x_{pt}, y_{pt}, z_{pt}, w_{pt}), h_{pt}, l_{pt}, \theta_{pt})$.

$$\begin{aligned}
\mathcal{L}_{loc} &= \sum \text{smoothL1}(\Delta b), \\
\Delta b &= \{\Delta x, \Delta y, \Delta z, \Delta w, \Delta h, \Delta l, \Delta\theta\}, \\
\Delta x &= \frac{x_{gt} - x_{pt}}{d_{pt}}, \Delta y = \frac{y_{gt} - y_{pt}}{d_{pt}}, \Delta z = \frac{z_{gt} - z_{pt}}{h_{pt}}, \\
\Delta w &= \log \frac{w_{gt}}{w_{pt}}, \Delta l = \log \frac{l_{gt}}{l_{pt}}, \Delta h = \log \frac{h_{gt}}{h_{pt}}, \\
\Delta\theta &= \theta_{gt} - \theta_{pt},
\end{aligned} \tag{4.3}$$

where $d_{pt} = \sqrt{(w_{pt})^2 + (l_{pt})^2}$ is the diagonal of the base of the predicted bounding box.

Angel loss. The sin-error loss is used to supervise the angle regression, instead of directly predicting the angle offset $\Delta\theta$.

$$\mathcal{L}_\theta = \text{smoothL1}\left(\sin\left(\theta_g t - \theta_p t\right)\right). \tag{4.4}$$

Softmax classification loss is finally utilized to distinguish the flipped boxes.

Classification loss. Due to the extreme imbalance between a few ground truths and the generated anchors, the focal loss [Lin et al., 2017b] is introduced to guide the object classification.

$$\mathcal{L}_{cls} = -\alpha\left(1 - P_{pt}\right)^\gamma \log P_{pt}, \tag{4.5}$$

where P_{pt} is the class probability of a predicted box. In this thesis, α and γ are set to 0.25 and 2, respectively.

Above all, the combined loss is used to train the detection network:

$$\mathcal{L}_{dec} = \frac{1}{N_{pos}} \left(\beta_1 \mathcal{L}_{loc} + \beta_2 \mathcal{L}_{cls} + \beta_3 \mathcal{L}_{\theta} \right), \tag{4.6}$$

where N_{pos} is the number of positive anchors and $\beta_1, \beta_2, \beta_3$ are set to 2, 1, and 0.2, respectively.

4.2.3 Post-processing

In this study, the score-filtering and NMS methods are used to decrease the number of false positives after the inference of detection. The score-filtering strategy simply excludes predicted bounding boxes with extremely low confidence scores via a pre-defined threshold. The NMS method suppresses predictions by removing overlapped bounding boxes since there are no two vehicles with an overlap region in the real world. Notably, the NMS method is performed followed by the score-filtering.

Let $\mathcal{B} = \{b_1, ..., b_{Nb}\}$ is a list of 3D detection boxes with scores $S = \{s_1, .., s_{Nb}\}$, NMS removes the detection box with the maximum score $\mathcal{M}$ from the list $\mathcal{B}$ and adds it to the list final detections $\mathcal{D}$. In addition, NMS removes any box which has an overlap greater than a certain threshold $\mathbb{T}$ with the maximum score $\mathcal{M}$ in $\mathcal{B}$. This operation is repeated for the remaining boxes $\mathcal{B}$. Algorithm 1 shows the detailed flow of NMS.

Algorithm 1 The workflow of NMS.

Input: 3D detection boxes $\mathcal{B}$, corresponding detection scores $S = \{s_1, .., s_{Nb}\}$, pre-defined threshold $\mathbb{T}$.

Initialization. $\mathcal{D} \leftarrow \{\}$
repeat
  $m \leftarrow \text{argmax}\, \mathcal{S}$
  $\mathcal{M} \leftarrow b_m$
  $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
  while $b_i \in \mathcal{B}$ **do**
    if $IOU\,(\mathcal{M}, b_i) \geq N_t$ **then**
      $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$
    end if
  end while
until $\mathcal{B} =$ empty

Output: $\mathcal{D}, \mathcal{S}$.

4.3 Detector-free Motion prediction based 3D Tracking network (DMT)

The overall network architecture of the proposed DMT is shown in Fig. 5.3. Given the query and search point cloud with coordinates denoted as Q and P, and an initial bounding box B_{init}, the backbone is first used to extract target-specific features following [Zheng et al., 2021a], as introduced in Section 4.3.1. Unlike previous studies, a motion prediction module is proposed to estimate a potential target center in the current frame based on the previous target states S_{prev}, with details described in Section 4.3.2. Afterward, an explicit voting module is adapted to modify the coordinates of the coarse predicted center and predict the orientation in Section 4.3.3.

Algorithm 2 The workflow of DMT.

Input: Points Q in query, points P in search area, an initial bounding box B_{init}, previous target states S_{prev} and target-specific search feature f.

1: **Potential target center generation.** Given S_{prev}, predict a coarse target center C_{coarse} in the current frame using a motion prediction module.
2: **Explicit voting.** Feed f and C_{coarse} into an explicit voting module to estimate a target-specific point feature $\hat{f}$ of the target center.
3: **Final box regression.** Regress the 3D bounding box of the target based on $\hat{f}$ using a prediction head.

Output: The 3D bounding box of the target.

The loss function is presented in Section 4.3.4. The training strategy and implementation details are explained in Section 4.3.5. For highlighting the simplicity of DMT, the detailed flow is also sketched in Algorithm 2.

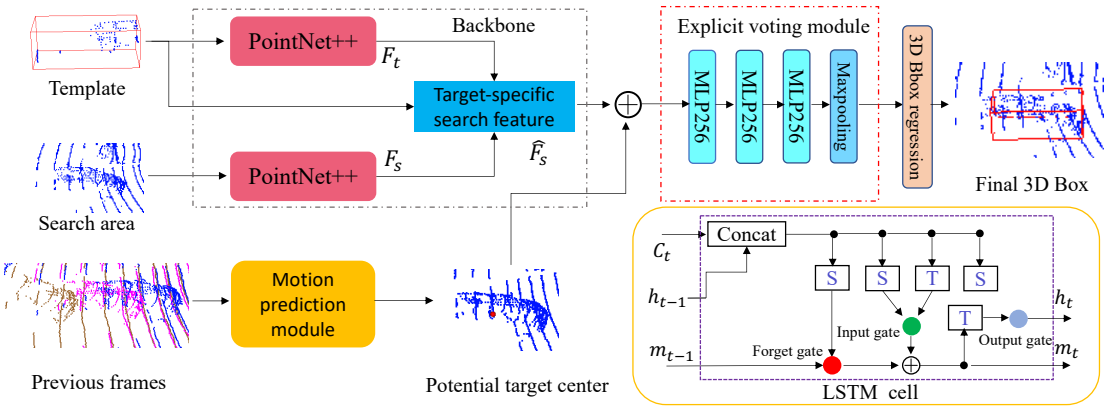


Figure 4.4: Overview of DMT. The backbone network first extracts the target-specific features from the template and search area points following [Zheng et al., 2021a]. Then, the motion prediction module (MPM) estimates the 3D coordinates of a potential target center. Next, the explicit voting module refines the target-specific search feature extracted by the backbone at the coarse predicted center. Finally, a 3D bounding box prediction head regresses the target location. One example of the MPM is an LSTM (lower right corner).

4.3.1 Backbone

The aim of the backbone network is to generate an enhanced target-specific search feature by fusing the template's target information into the search area points. The box-aware feature fusion (BAFF) module in [Zheng et al., 2021a] is adopted as the backbone*, as shown in Fig. 4.5. The template and search area are first feed respectively into PointNet++ [Qi et al., 2017b] for obtaining their features. Then, the BAFF module help augment the search area with target-specific features, which includes BoxCloud [Zheng et al., 2021a] comparison and feature aggregation sub-modules.

BoxCloud comparison. Given the feature of a search area $F_s = \{f_i^s\}_{i=1}^{M_1}$ obtained by PointNet++, the 9D BoxCloud coordinates $C_{bc}^s = \{c_i^s\}_{i=1}^{M_1}$ are predicted for each point feature f_i^s via MLP, where M_1 is the number of the points in C_{bc}^s. The prediction is supervised by a BoxCloud loss, presented in Sec. 4.3.4. Then, the pairwise distance between the predicted C_{bc}^s and the BoxCloud $C_{bc}^t = \{c_i^t\}_{i=1}^{M_2}$ of the template is compared, as shown in Fig. 4.5, where M_2 is the number of the points in C_{bc}^t. Following [Zheng et al., 2021a], the simple l_2 distance is adopted as the

*The framework is not restricted to BAFF, and any suitable backbone could be used.

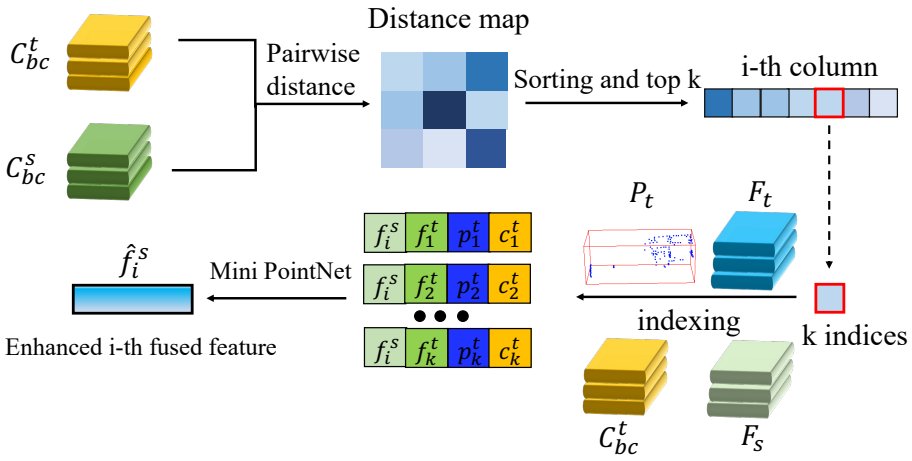


Figure 4.5: The workflow of box-aware feature fusion (BAFF) module. C^s_{bc} is predicted from each search point feature f^s_i via MLP. C^t_{bc}, F_t, P_t are the 9D BoxCloud coordinate, the feature and the spatial 3D coordinates of a template. The module first generates the distance map between BoxCloud C^s_{bc} and C^t_{bc} for retrieving the top-k nearest neighbors with respect to each point in the search area. Then, a mini PointNet is adopted to generate $\hat{f}^s_i$ by aggregating the neighbors' features.

distance metric. After obtaining the distance map, the top k most similar template points for each point in the search area are sorted and selected. The i-th column of the distance map in Fig. 4.5 represents the indices of the k nearest neighbors of the i-th search point.

Feature aggregation. After getting the top k template features, it is hopeful to fuse them into the search area. As illustrated in Fig. 4.5, the k nearest template points for the i-th point p^s_i in the search area are selected according to the i-th column in the distance map. Besides, considering the feature of a template F_t extracted from PointNet++, the corresponding spatial 3D coordinates P_t and 9D BoxCloud coordinate C^t_{bc} of the template points, more informative k tuples $\left\{\left[f^t_j, p^t_j, c^t_j, f^s_i\right], \forall j = 1, \cdots, k\right\}$ are constructed. Finally, a mini-PointNet is used to obtain the aggregated feature of the search point from these pairs, which can be formulated as follows:

$$\hat{f}^s_i = G \odot \left\{MLP(\left[f^t_j, p^t_j, c^t_j, f^s_i\right)\right\}^k_{j=1}), \tag{4.7}$$

where $G\odot$ is a max-pooling operation. Finally, the effective target-specific search feature $\hat{F}_s = \{\hat{f}^s_i\}^{M_2}_{i=1}$ is obtained.

4.3.2 Motion prediction module

The previous end-to-end 3D SOT methods [Qi et al., 2020; Zheng et al., 2021a; Shan et al., 2021] heavily rely on point cloud features for target object detection. However, erroneous detection may occur when the point cloud of the target is incomplete [Giancola et al., 2019]. To alleviate this, explicitly leveraging spatial-temporal information is proposed for 3D SOT. Specifically, a motion prediction module (MPM) $\mathcal{M}$ is introduced based on previous target states (i.e., predicted 3D target center coordinates in the previous frames) to predict a coarse target center in the current frame. Suppose that a tracklet $\{(x_i, y_i, z_i)\}^t_{i=1}$ is known in the previous t frames, the prediction of the target center location in the next $(t+1)$-th frame is formulated as:

$$(\hat{x}_{t+1}, \hat{y}_{t+1}, \hat{z}_{t+1}) = \mathcal{M}(\{(x_i, y_i, z_i)\}^t_{i=1}). \tag{4.8}$$

In the general design, common regression or prediction models can be employed as the MPMs for effective target center prediction. Here, several simple yet effective MPMs are introduced.

Constant velocity model. The constant velocity model assumes that the target acceleration in the current frame is 0 and that the velocity of the target in the current frame should be equal to the velocity in the last frame. Given the target locations in the $(t-1)$ and t-th frames $\{(x_i, y_i, z_i)\}_{i=t-1}^{t}$, the predicted target center coordinate in the $(t+1)$-th frame is calculated as $(2x_t - x_{t-1}, 2y_t - y_{t-1}, 2z_t - z_{t-1})$. Despite the simplicity of this model, this simple model can also work very well in the proposed DMT.

Sequence-to-sequence prediction model. The goal of the proposed MPM is to predict a 3D coordinate based on previous estimated t target coordinates, which is actually a sequence-to-sequence prediction task. Long Short-Term Memory (LSTM) network [Hochreiter & Schmidhuber, 1997] is a typical sequence-to-sequence prediction model that has been widely used in various sequence prediction tasks. In this work, a multi-layer LSTM is chosen since this naive LSTM model can better validate the effectiveness of the proposed tracking method. The conventional LSTM cell is shown in Fig. 4.4 (bottom right). More details can be found in [Hochreiter & Schmidhuber, 1997]. In the implementation, the center coordinates of the 10 consecutive frames from the times $t-10$ to t are selected to predict potential target center coordinates in the $(t+1)$-th frame. In the training stage, multiple training tracklets generated from the KITTI and NuScenes datasets are prepared to train the LSTM respectively. In online tracking, the offline learned LSTM network is directly used for motion prediction without further updating.

Regression model. Traditional learning-based regression models can also be employed as MPMs. In this work, several basic regression models are tried including linear regression, ridge regression, Gaussian processor regression, and Ransac regression. The training for the above models is similar to the LSTM-based MPM, i.e., using the generated tracklet training data for training in an offline manner.

The above basic MPMs can roughly predict the potential target center coordinate based on the previous states. The prediction is not always reliable since the previous target states may be noisy (i.e., the predicted target center does not match the ground truth), or the target changes position in an unexpected way. To alleviate this problem, a lightweight explicit voting module is proposed to further refine the MPM prediction.

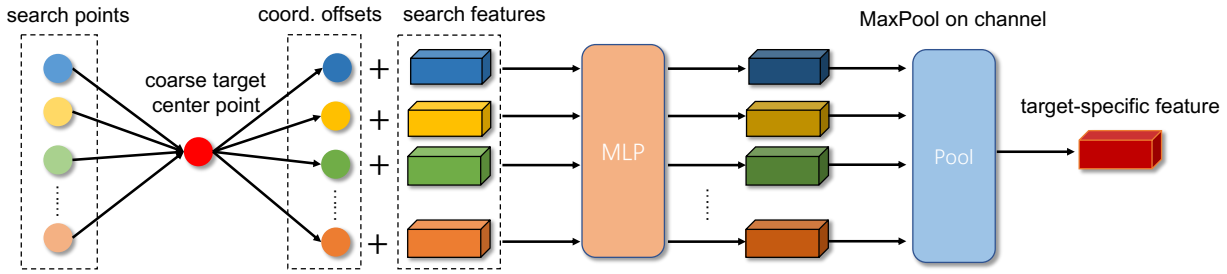


Figure 4.6: The overall pipeline of an explicit voting module (EVM). The proposed EVM first calculates the coordinate offsets between each search point and the coarse predicted target center. Then, the offsets are jointly concatenated with the search features for feature modeling via an MLP. Finally, a permutation-invariant max pooling layer is applied to obtain the target-specific feature of the predicted target center point for the final 3D box prediction.

4.3.3 Explicit voting module

Before going to the details of the proposed explicit voting module (EVM), a short review on the RPN module (VoteNet) used by previous trackers [Qi et al., 2020; Zheng et al., 2021a; Wang et al., 2021; Shan et al., 2021] is given. The architecture of VoteNet includes two aspects: 1) Hough

voting to convert the search area seeds into possible target centers; and 2) clustering neighboring potential target centers to obtain the final target center. For generating the potential target centers, VoteNet estimates the coordinate offsets between each search seed and ground-truth target center, which aims to push the predicted possible target centers and ground-truth target center to be as close as possible. In the proposed DMT, the above two steps can be removed since the coarse target center location in the current frame is provided by the proposed MPM, which makes DMT simpler and lighter.

The overall pipeline of the proposed explicit voting module is shown in Fig. 4.6. As can be seen, after obtaining the coarse target center coordinate $(\hat{x}_{t+1}, \hat{y}_{t+1}, \hat{z}_{t+1})$ estimated by MPM and the target-specific search feature, the goal of the proposed EVM is to estimate effective features on $(\hat{x}_{t+1}, \hat{y}_{t+1}, \hat{z}_{t+1})$. In the design of EVM, coordinate offsets are used as explicit voting signals for estimating the target center feature. Specifically, the coordinate offset between the estimated target center and each search point is first calculated. Then, the coordinate offset with the search point feature is concatenated to obtain a candidate voting feature $f \in \mathbb{R}^{C+3}$, where C denotes the feature dimension. Suppose there are N search points with N corresponding candidate voting features $\{f_i\}_{i=1}^{N}$. The explicit target coordinate voting is formulated as:

$$\bar{f}_i = \text{MLP}(f_i), \quad \hat{f} = \text{MaxPool}(\{\bar{f}_i\}_{i=1}^{N}), \tag{4.9}$$

where $\bar{f}_i \in \mathbb{R}^C$, and $\hat{f} \in \mathbb{R}^C$ is the final estimated target-specific feature at the estimated target center, which is obtained by applying the max pooling operation on the channel dimension of each feature vector in $\{\bar{f}_i\}_{i=1}^{N}$. The estimated feature $\hat{f}$ is finally fed into a prediction head (i.e., MLP) for regressing the bounding box of the target.

In the training stage, given a ground-truth target center location in a frame, the diverse points around the ground-truth center are randomly sampled. For stable training, the maximum distance between the sampled points and the ground-truth center should not be too large, and here it is set to 0.75 meters. During training, the proposed EVM learns to estimate target-specific features of the sampled points that are effective for predicting the final bounding box. Note that the diverse sampled points can effectively mimic the noisy predictions of MPM, which makes the proposed DMT more robust to noise in the predicted target track.

4.3.4 Loss function

Following [Zheng et al., 2021a], the training loss includes three components: classification loss, box-cloud loss, and regression box loss. The former two losses enhance the target-specific feature extracted by the backbone, while the latter supervises the estimated 3D bounding box. In addition, a velocity loss is added for training the MPM (except for the constant velocity model).

Point-wise classification loss. Following [Qi et al., 2020], only search points located on the surface of a ground-truth target are useful in the EVM, and thus labeled as positives, while all others are negatives. Therefore, a standard binary cross entropy loss L_{cla} is adopted to classify the search points.

BoxCloud loss. The BoxCloud features [Zheng et al., 2021a] in the search area are unknown in the inference stage, so it is needed to predict the 9D BoxCloud coordinate C_{bc} in the search area, which is supervised by a smooth-L1 regression loss.

$$\mathcal{L}_{bc} = \frac{1}{\sum_i E_i} \sum_{i=1}^{N} \left\| C_{bc}^i - \hat{C}_{bc}^i \right\| \cdot E_i, \tag{4.10}$$

where $\hat{C}_{bc}$ are ground-truth BoxCloud coordinates pre-calculated before training. E_i is a binary mask, which indicates whether the i-th point is inside an object BBox or not.

3D box regression loss. The final result of DMT is to predict the 3D box parameters $C_{bbox} = \{\hat{x}, \hat{y}, \hat{z}, \hat{\theta}\}$. Following previous work, the Huber (smooth-L1 loss) is adopted to supervise the regression.

$$\mathcal{L}_{bbox} = \left\| C_{bbox} - \hat{C}_{bbox} \right\|, \tag{4.11}$$

where $\hat{C}_{bbox}$ is ground-truth bounding box of the target.

Velocity loss. For training a MPM, the distance between the predicted center coordinates of the target and the ground truth is hoped as small as possible. In this work, the mean squared error loss $\mathcal{L}_v$ is used for supervision:

$$\mathcal{L}_v = \left\| C_{cen}^{t+1} - \hat{C}_{cen}^{t+1} \right\|_2, \tag{4.12}$$

where $C_{cen}^{t+1} = (\hat{x}_{t+1}, \hat{y}_{t+1}, \hat{z}_{t+1})$ (see Eq. (4.8)) is the predicted target center coordinate at the $(t+1)$-th frame and $\hat{C}_{cen}^{t+1}$ is the corresponding ground-truth coordinate. Note that the MPM with $\mathcal{L}_v$ is first trained, and then the following combined loss is used to train the backbone network, EVM, and the prediction head:

$$L = \alpha L_{cla} + \beta L_{bc} + \gamma L_{bbox}, \tag{4.13}$$

where α, β, and γ are hyperparameters to balance their relationship. Here, $\alpha = 0.2$, $\beta = 1.0$, $\gamma = 0.2$.

4.3.5 Implementation details

Following previous 3D trackers [Qi et al., 2020; Zheng et al., 2021a], the template and search point clouds are generated in both the training and testing stages. To fairly compare with recent trackers equipped with online detectors, the same target-specific search feature generation method in BAT [Zheng et al., 2021a] is used, which makes the predictions of BAT and the proposed DMT both based on the same augmented search features.

Search area generation. In practice, the object movement between two consecutive frames is relatively small, so searching the entire frame for the target is unnecessary. Following [Zheng et al., 2021a], the target near the previous object location is looked for to generate search areas for training and testing. During both training and testing, templates and their BBoxes are transformed into the object coordinate system before being sent to the model.

Network architecture. In the proposed MPM, one LSTM layer with 50 hidden units is used as the motion predictor. The input tracklet length is set to 10, meaning that the target states in the previous 10 frames are used for prediction. The model size of this LSTM model is about 50K, which is extremely light. The EVM is implemented as a three-layer MLP with 256 hidden units, where the first two layers are followed by a 1-D batch normalization layer and a ReLU activation layer. The same backbone and the box prediction head as P2B [Qi et al., 2020] and BAT [Zheng et al., 2021a] are used.

Training. In the training stage, the tracklet training data (i.e., each tracklet contains the target center coordinates in every 10 frames and the corresponding ground-truth target center coordinates in the next frame) is first generated to train the LSTM network. The batch size is set to the overall dataset size, and the learning rate and training epochs are respectively set to 1e-3 and 8000. The whole training takes only 28 seconds on the car category of the KITTI dataset, which is efficient. After training the LSTM network in an offline manner, it is used for online testing without further modifications. The proposed DMT is trained for 60 epochs using the Adam optimizer with a batch size of 100. The learning rate is initialized as 1e-3 and decayed with 0.5 in every 5 epochs.

Testing. During testing, the trained DMT is applied to infer 3D bounding boxes of a given target within tracklets frame by frame. For the current frame, the template is updated by fusing the point clouds in the first given BBox and in the previous estimated BBox. To obtain the search area, the previously estimated BBox is enlarged by 2 meters in the current frame and collected the points within the enlarged BBox.

5 3D Shape Completion

The scanned data from LiDAR is often incomplete and noisy owing to the occlusion. Point cloud completion is an essential and challenging task in the fields of photogrammetry and computer vision, which infers the complete structure from partial point clouds. In this chapter, two point cloud completion methods are introduced. To be specific, the first method is an end-to-end network for completing point clouds of 3D vehicle shapes, operating on the partial and sparse point clouds directly. While, more categories (e.g. Car, Airplane, Chair) are extended to complete in the second method, which is a point cloud completion network based on a feature-matching strategy.

5.1 Problem statement

The point cloud completion task can be regarded as a set problem. Let $X = \{P_i : i = 1, ..., N\}$ be a set of the partial points, which are lied on the observed surfaces of a single object as a result of a single observation or a series of observations from a 3D sensor. Further, let $Y = \{P_j : i = 1, ..., M\}$ be a complete set of 3D points uniformly sampled from the object's observed and unobserved surfaces. The 3D shape completion task is defined as predicting Y given X. M and N are the numbers of points in the ground-truth point cloud and partial point cloud, respectively. Formally, the 3D point cloud completion methods can be formulated as:

$$Y = F(X) = F(P_i : i = 1, ..., N) \tag{5.1}$$

where $X \in \mathbb{R}^{N\times 3}$, $Y \in \mathbb{R}^{M\times 3}$ and F is a prediction function. Notably, X is not necessarily a subset of Y and there is no explicit correspondence between points in X and points in Y since they can be obtained from the object surface independently.

In this chapter, supervised learning is used to solve this problem. A deep neural network is trained to predict Y directly from X. A 3D large-scale synthetic CAD model dataset is used to easily obtain training samples of X and Y for supervised learning. The network can be generic across multiple object categories and it makes no assumptions about the underlying object's structure, such as symmetry or planarity.

5.2 Vehicle Point Completion Network (VPC-Net)

Vehicles are the most concerned investigation targets in this research because they are a dynamic and essential component in the 3D urban road environment. An accurate and instant measurement of the vehicles is critical for monitoring their behaviors and extracting their geometric characteristics. MLS systems have been chosen as a key sensor by many autonomous driving companies and research institutes, particularly for vehicle extraction, because they can provide highly accurate geometric information (e.g., 3D coordinates of vehicle points) and reliable ratiometric attributes (e.g., reflectivities of various surface materials) of multiple instances simultaneously.

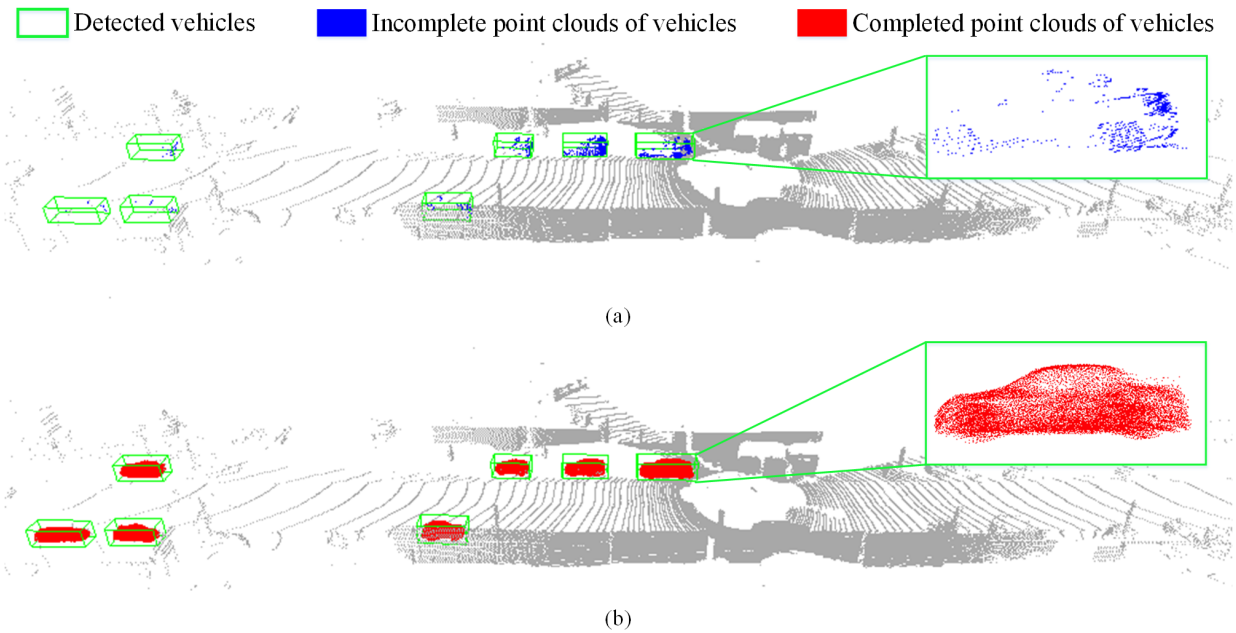


Figure 5.1: Visualization of the incomplete and completed point clouds of vehicles in raw scans. a) A single-frame raw real-scan data from KITTI [Geiger et al., 2013]. b) Completed scan generated by the proposed Vehicle Points Completion-Net (VPC-Net).

However, acquired 3D point clouds of vehicles from MLS systems are inevitably incomplete due to object occlusion or self-occlusion. For instance, in Fig. 5.1a, a few typical point clouds of vehicles on urban roads from the KITTI dataset [Geiger et al., 2013] are illustrated. The missing parts in the scanned point clouds of the vehicles are clearly visible in this figure. Because it has changed the dimension of shapes, biased the volume of objects, and destroyed the topology of the surfaces. This incompleteness significantly limits the potential uses of vehicle point clouds. The complete geometric shapes of vehicles provide solid foundations for 3D perceptual tasks such as instance extraction, type classification, and track estimation in generic applications such as 3D traffic monitoring [Wen et al., 2019]. Recently, Zhang et al. [2020a] proposed an alternative strategy that estimates the vehicle poses first, and then retrieves a similar CAD model of this vehicle from large-scale CAD model datasets to replace the raw point clouds. This method, however, cannot deal with occluded vehicles and cannot preserve the true knowledge of raw point clouds. Furthermore, for specific applications such as measuring vehicle-induced aerodynamic loads in bridge engineering, the entire surface as well as the shape of the measured vehicles is critical to estimating wind pressure caused by vehicles driving close to the sound barrier, which has a significant impact on the design of urban highway viaducts [Pan et al., 2018].

In this section, a new neural network is proposed, named Vehicle Points Completion network (VPC-Net), to synthesize complete, dense, and uniform point clouds for vehicles from MLS data. Given the sparse and partial point clouds of vehicles, the proposed network can generate complete and realistic structures and keep the fine-grained details from the partial inputs, as shown in Fig. 5.1b.

The critical architecture of VPC-Net is shown in Fig. 5.2, which includes an encoder module, a decoder module, and a refiner module. The encoder is to extract the global features from partial and sparse point clouds. Second, the decoder is divided into two parts: (i) it uses the generated global features as input to generate a coarse but complete point cloud, and (ii) it combines the coarse point cloud and global features to produce dense point clouds. Finally, to

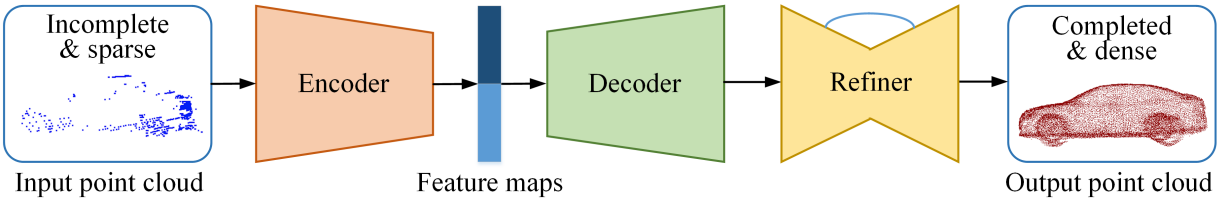


Figure 5.2: Workflow of the proposed VPC-Net.

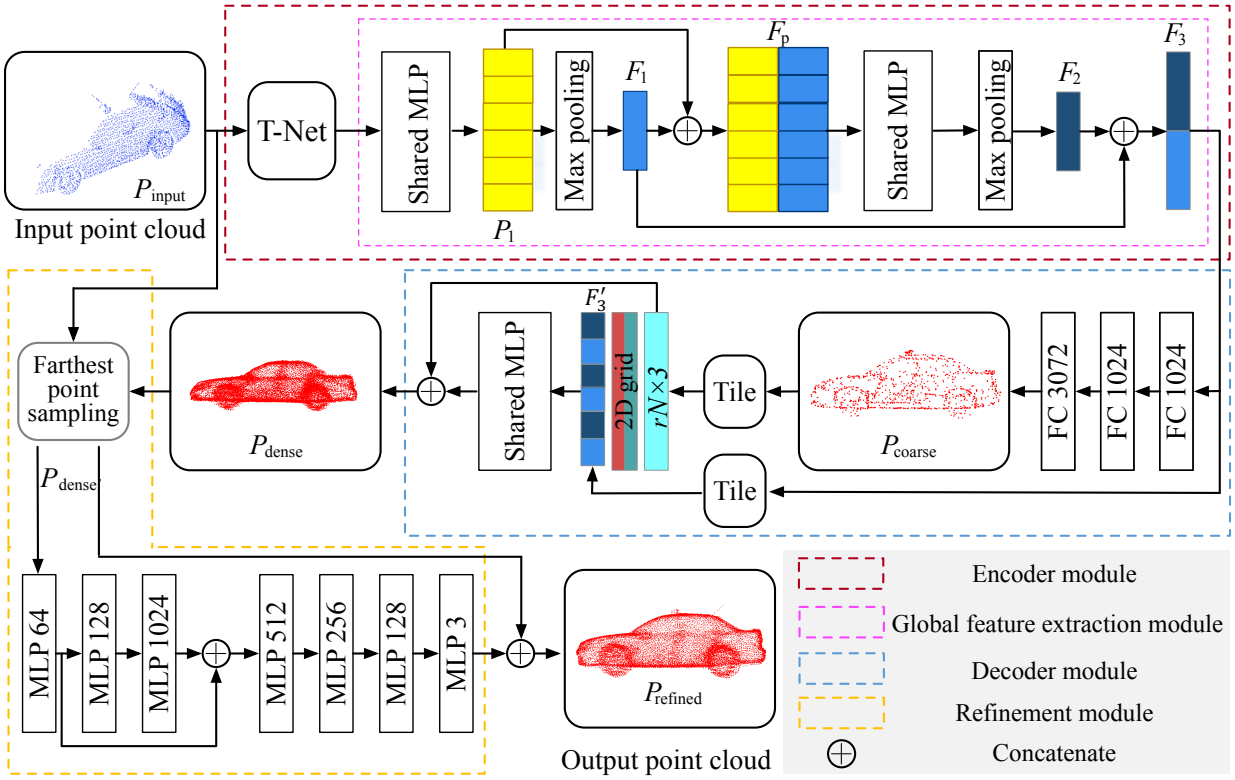


Figure 5.3: The network architecture of VPC-Net.

preserve the original details, the skip connections are used to concatenate the partial inputs with the previous dense point cloud. The refiner refines the fused 3D point clouds further to produce the final completion result. The point clouds generated by the proposed VPC-Net perform three outstanding functions: (i) complete the missing surface with fine-grained structures; (ii) preserve the original details of the inputs; and (iii) produce uniform point clouds.

5.2.1 Encoder

The aim of the encoder is to provide a set of features F for the decoder. As a result, the encoder's feature extraction capability is critical throughout the network. It is significantly beneficial for 3D coordinates regression of the dense point cloud generation if the encoder can effectively combine the local and global features from the partial inputs. The encoder is made up of two modules: a spatial transform network and a global feature extraction module. Formally, it can be formulated by the combination of two functions, defined as follows:

$$F = Q(P_{input}|w_Q), Q = Q_1 \circ Q_2 \tag{5.2}$$

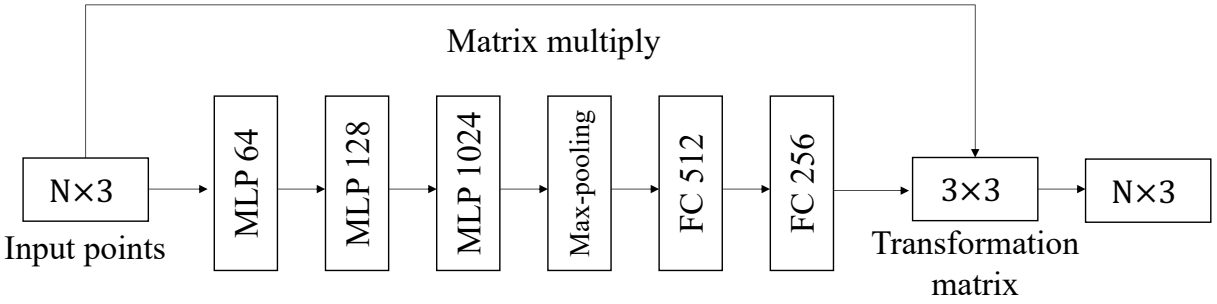


Figure 5.4: The architecture of T-Net in the encoder of VPC-Net.

where Q_1 and Q_2 are the spatial transform network and the global feature extraction module, respectively. w_Q denotes the weight parameters of Q, and P_{input} is the partial point clouds as inputs.

Since the input point clouds of the vehicle are spatially disordered and their poses are diverse, accessing the unified features for neural networks will be difficult. As a result, the input point cloud is hoped to have a neat pose to help with feature extraction. In other words, learned features from input point sets should be geometrically invariant.

A natural solution is to align all point sets to a canonical space. In [Jaderberg et al., 2015], the authors used a spatial transformer for learning feature invariance to translation and rotation in 2D images. Inspired by this, a 3D spatial transform network T-Net [Qi et al., 2017a] is adopted to predict a 3×3 transformation matrix for the original point clouds. Furthermore, this transformation matrix is multiplied by the coordinates of the input points directly. Therefore, the inputs are aligned to a canonical space, allowing the following network to learn a unified and standardized feature attentively.

T-Net is similar to a mini-PointNet [Qi et al., 2017a], with a shared Multiple Layer Perception (MLP) network, a max-pooling layer, and two fully connected layers. The detailed architecture operation is shown in 5.4. It takes raw point clouds as input and produces a 3×3 matrix. In detail, the MLP network first encodes each point to multiple dimensions $[64, 128, 1024]$. A max-pooling layer is used, followed by two fully connected layers with $[512, 256]$ sizes. The regressed matrix is initialized as an identity matrix. All layers, with the exception of the last, are followed by a ReLU activation and a batch normalization layer.

In addition, another important part of the encoder is a global feature extraction module. Generally, it is based on the recently advanced feature extraction network PointNet, which directly operates on point clouds. Inspired by this, the encoder, as illustrated in Fig. 5.3, adopts two stacked PointNet layers to extract the geometric information for the input point cloud. Each PointNet layer comprises one shared MLP and one max-pooling layer as a basic module. In the first PointNet layer, a point-wise feature P_1 is learned from the points of $N_{input} \times 3$ transformed by the STN, where N_{input} is the number of points and 3 is the x,y,z coordinates of each point. Afterwards, a max-pooling layer is employed on P_1 to output a 256-dimensional local feature vector F_1. In the second PointNet layer, the local latent space is concatenated with every independent point feature by feeding F_1 back to the point-wise feature P_1. The global latent vector F_2 is then extracted from the aggregated point features F_p through the second PointNet layer, with the size $F_2 := 1024$.

However, it always loses the fine details of the inputs since the latent space extracted by the last max-pooling layer only represents the rough global shape. Inspired by the skip connection

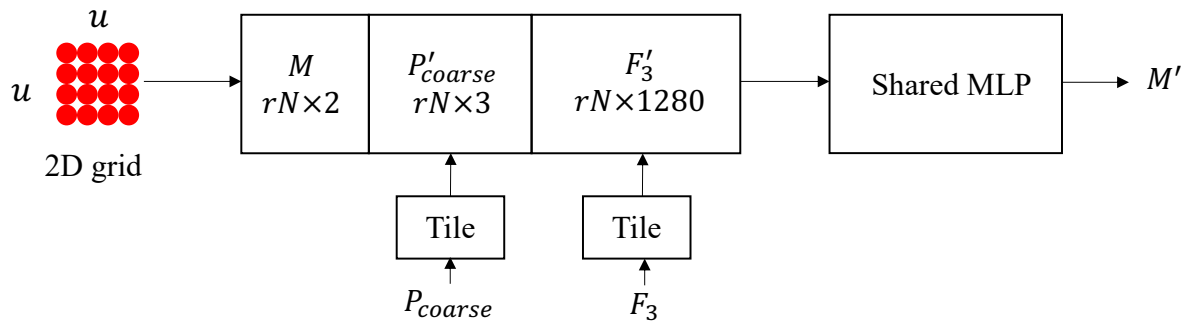


Figure 5.5: The detailed concatenating operation in the decoder of VPC-Net.

from U-Net [Ronneberger et al., 2015], a point feature enhancement (PFE) layer is designed to concatenate the global feature F_2 with the local feature F_1 to synthesize the final feature space F_3. Size $F_3 := 1280$, and it includes both low-level and high-level feature information. Experimental results in Section 8.3.1 show that this design improves the feature extraction ability of the encoder for partial inputs.

5.2.2 Decoder

The decoder is responsible for converting the final global latent vector F_3 into dense, evenly, and complete 3D point clouds. In this stage, a coarse-to-fine completion strategy is applied for generating the 3D coordinates of point clouds. Inspired by the 3D single object reconstruction network RealPoint3D [Xia et al., 2019], three fully connected layers are explored to generate a sparse point cloud with a complete geometric surface. Lastly, it outputs the final vector with $3N$ units, and it is reshaped into an $N \times 3$ coarse point cloud P_{coarse}.

However, the fully connected layer is not suitable for generating dense points. Some points will be over-concentrated when regressing a large number of points since the fully connected layers are not restrictive on the local density. To alleviate the problem, in the second stage, the points in P_{coarse} are first tiled to produce a dense point set $P^{'}_{coarse} := rN \times 3$, where r is the up-sampling rate. Then, a folding-based operation [Yang et al., 2018] is applied to deform a unique 2D grid vector and concatenate with each point of the coarse point cloud to obtain new patches. This operation has the potential to increase the difference between duplicated points. In other words, each coarse point cloud point can be considered a spatial keypoint and used it as the center point to generate a series of surrounding points. To make full use of the features of input point clouds, the points in $P^{'}_{coarse}$, the tiled global feature space $F^{'}_3$, and the deformed 2D grids are concatenated to obtain a new aggregated feature. The detailed concatenating operation is shown in Fig. 5.5. The coordinates of points on a zero-centered $u \times u$ grid ($u^2 = rN \times 3$) are first deformed into a $rN \times 2$ matrix M [Yang et al., 2018]. Then, M is concatenated with the coordinates of the tiled coarse point cloud $P^{'}_{coarse}$ and the duplicated global feature vector $F^{'}_3$. Furthermore, the aggregated feature is passed through a shared MLP with sizes $[512, 512, 3]$ to generate a new $rN \times 3$ matrix $M^{'}$. This shared MLP can be regarded as a non-linear mapping that transforms the 2D grid into a smooth 2D manifold in 3D space [Yuan et al., 2018]. Finally, the dense point cloud $P_{dense} := rN \times 3$ is generated by adding the coordinates of each point in $P^{'}_{coarse}$ to the matrix $M^{'}$.

5.2.3 Refiner

Although the decoder produces impressive results, the fine-grained details of the inputs are always lost, and the points are unevenly distributed. To address these problems, the partial input P_{input} is combined with the decoder outputs P_{dense}. The details of the input point cloud can be fully retained by this operation. The linear combination, however, will result in non-uniform merged points because the two point clouds have different densities and may overlap. Thus, a uniformly distributed subset point cloud $P^{'}_{dense}$ is sampled with a size of $rN \times 3$ using farthest point sampling (FPS). The hyperparameter r is the same in the decoder and the refiner. In this work, $r = 16$.

The refiner can be regarded as a point feature residual network. The refiner is hoped to predict per-wise offsets o_x, o_y, o_z for every point in $P^{'}_{dense}$. Therefore, the points $P^{'}_{dense}$ are passed through a series of MLPs to predict point feature residuals since neural networks are better at residuals [Wang et al., 2018]. Specifically, a bottom-up and top-down strategy is used to refine the point coordinates, inspired by the structure of an encoder-decoder network. The refiner consists of seven MLPs. It first encodes each point into multiple dimensions $[64, 128, 1024]$. It is then decoded to generate the offsets of each point with dimensions of $[512, 256, 128, 3]$. Except for the final layer, which is followed by a batch normalization layer and a Tanh activation, all other MLPs are preceded by a batch normalization layer. Furthermore, the local feature is expected to be preserved in the subsequent layers. As shown in Fig. 5.3, the feature with dimensions of 64 is combined with the bottleneck layer with a size of 1024. Overall, the generated point clouds $P^{'}_{dense}$ in this refiner can be formulated as

$$P_{refined} = R(P^{'}_{dense}) + P^{'}_{dense} \tag{5.3}$$

where $R\{.\}$ predicts point-wise displacements by the refiner.

5.2.4 Loss function

In this work, the topological distance between the completed object by the proposed VPC-Net and the ground truth is defined as the loss function. Inspired by [Fan et al., 2017], the Chamfer Distance (CD) and Earth Mover's Distance (EMD) are used to optimize the network. Distance metric functions are highly efficient and invariant to permutations of the relative ordering of points. The CD between the completed point cloud P_c and the ground truth P_{gt} is defined as

$$d_{chamfer}(P_c, P_{gt}) = \sum_{x \in P_c} \min_{y \in P_{gt}} \parallel x - y \parallel_2^2 + \sum_{x \in P_{gt}} \min_{y \in P_c} \parallel x - y \parallel_2^2 \tag{5.4}$$

where $P_c, P_{gt} \subseteq R^3$. Intuitively, it aims to find the nearest neighbor between the two point sets in two directions. Each point of P_c is mapped to the closest point in P_{gt}, and vice versa. Thus, the sizes of P_c and P_{gt} are not required to be the same. For the nearest neighbor search, it is a computationally light function with $O(nlogn)$ complexity. However, it cannot guarantee the consistency of predicted points [Mandikal & Radhakrishnan, 2019], and it is sensitive to the detailed geometry of outliers [Tatarchenko et al., 2019]. To alleviate these problems, the EMD between P_c and P_{gt} is proposed by

$$d_{EMD}(P_c, P_{gt}) = \min_{\phi: P_c \to P_{gt}} \sum_{p \in P_c} \parallel p - \phi(p) \parallel_2 \tag{5.5}$$

where $P_c, P_{gt} \subseteq R^3$, $\phi : P_c \to P_{gt}$ is a bijection. Unlike CD, the sizes of P_c and P_{gt} must be the same since it is a point-to-point mapping function. However, its computing complexity $O(n^2)$ is too expensive. This makes it not suitable for generating dense point sets in the training.

As a result, a training strategy is proposed to make use of both distance functions. The EMD loss for P_c predicted by the encoder is used to ensure that the generated coarse point cloud is even and has general geometry. The CD loss is used to optimize the predicted dense point clouds P_{dense} and P'_{dense}. In more formal terms, the total loss is defined as

$$L(P_{coarse}, P_{dense}, P'_{dense}, P_{gt}) = d_{EMD}(P_{coarse}, \tilde{P_{gt}}) + \gamma d_{chamfer}(P_{dense}, P_{gt}) + \beta d_{chamfer}(P'_{dense}, P_{gt}) \tag{5.6}$$

where $\tilde{P_{gt}}$ is the subsampled ground truth with the same size as P_{coarse}. γ and β are hyperparameters to balance their relationship.

5.2.5 Implementation details and training process

The proposed VPC-Net was implemented in the TensorFlow framework and trained on a single NVIDIA Titan Xp GPU with 12 GB of memory. In the training stage, the batch size was set to eight. The Adam optimizer was used in the models for 100 K steps. The size of the coarse output generated by the encoder was 1024. The initial learning rate was set to 0.0001. The learning rate was decayed by 0.7 after every 50 K steps and clipped by 10^{-6}. $\gamma$and β were made equal. They gradually increased from 0.01 to 1 in the first 50 K steps. Notably, the resolutions of the inputs were various, from a few hundred points to thousands of points. Additionally, to

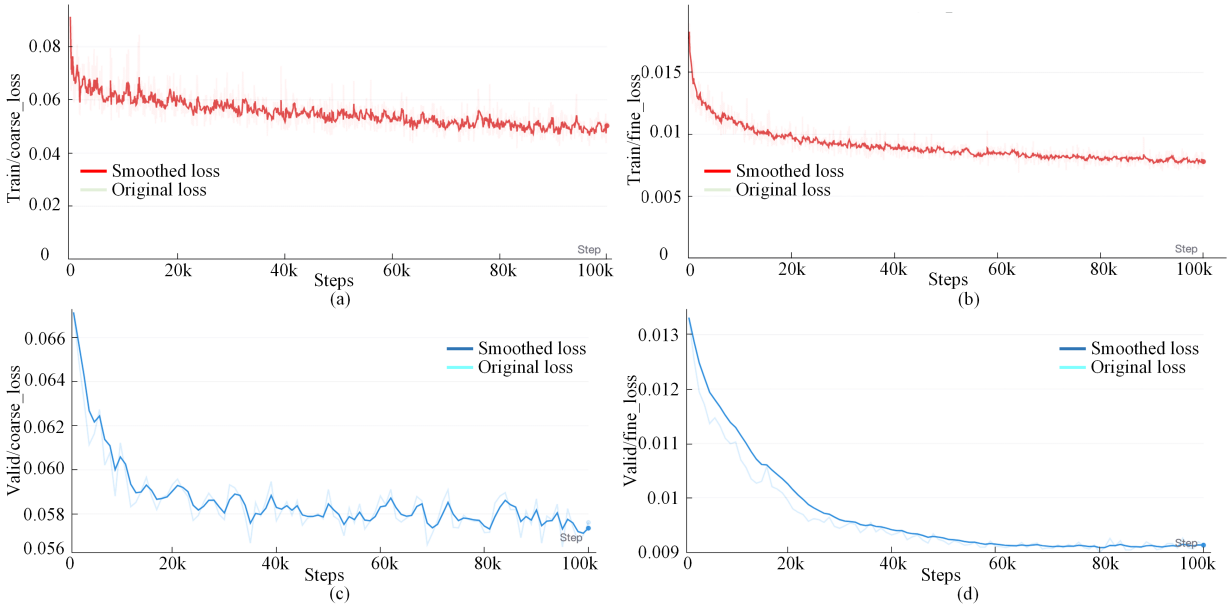


Figure 5.6: Visualization of the VPC-Net training process. EMD errors for coarse point cloud generated by the decoder in a) the training stage and b) the validation stage. CD errors for dense point cloud produced by the refiner in c) the training stage and d) the validation stage.

demonstrate the training process more vividly, the learning curve of the proposed VPC-Net is plotted (see Fig. 5.6 for illustration). The training losses and validation losses both consisted of two different types of losses. One is the CD for generated coarse point clouds, while the other is the EMD for the produced dense point clouds. As shown in Figs. 5.6a and 5.6b, the training losses gradually decreased as the number of training steps increased and converged until 100 K steps. The validation losses are shown in Figs. 5.6c and 5.6d, which also prove the proposed VPC-Net converges at 100 K training steps.

5.3 Asymmetrical Siamese Feature Matching Network (ASFM-Net)

Although VPC-Net performs well in completing 3D vehicles in an urban street environment, a more general 3D completion neural network is expected to be applied to multiple categories, including unseen categories. Thus, the second work is a novel neural network focusing on the completion of multiple object categories, which investigates the importance of shaping prior information via a feature-matching strategy and is termed an Asymmetrical Siamese Feature Matching Network (ASFM-Net).

Before going to the details of the proposed ASFM-Net, a short review of previous multiple-category completion methods [Yuan et al., 2018; Tchapmi et al., 2019; Sarmad et al., 2019; Wang et al., 2020d; Wen et al., 2020; Zhang et al., 2020b] and analysis of their shortcomings are given. PCN [Yuan et al., 2018] firstly proposed a learning-based completion method that operates on the point clouds directly. Afterward, TopNet [Tchapmi et al., 2019] designed a hierarchical-structured decoder based on a rooted tree for point cloud generation. Furthermore, RL-GAN-Net [Sarmad et al., 2019] introduced a reinforcement learning agent to control the generative adversarial network to generate a high-fidelity completed shape. SoftPoolNet [Wang et al., 2020d] analyzed the max-pooling operation causes some information loss when extracting global features, they proposed a soft pooling approach that selects multiple high-scoring activations. To preserve local structures, SA-Net [Wen et al., 2020] explored a skip-attention mechanism to transfer local features to the decoder. However, they all explored an encoder-decoder way to complete point clouds, relying on the global feature extracted by the encoder. Recently, RFA [Zhang et al., 2020b] realized this problem and employed a feature aggregation strategy to enhance the representation of global features. However, it still can not solve the fundamental problem: *The global features only extracted from the partial inputs must be incomplete and lose the geometric details.*

In the proposed ASFM-Net, this problem is transferred to how to make up for the loss of incomplete global features. A natural solution is to reduce the distribution gap between the partial global feature and the complete global feature. Thus, an asymmetrical Siamese auto-encoder network [Pham et al., 2020] is proposed to push the latent spaces extracted from the partial and complete point clouds to be as close as possible, as shown in Fig. 5.7.

In this way, the incomplete global feature is actually enhanced by the shape priors, including the class labels and the complete object geometric information. Compared with previous completion methods, the global feature extracted by encoders will be more fruitful after passing through the asymmetrical Siamese auto-encoder network. Then, the complete point clouds with more fine-grained details can be reconstructed using this global feature. In order to generate the final point clouds with the desired resolution, an iterative refinement unit is introduced in the last stage.

The overall network architecture of the proposed ASFM-Net is shown in Fig. 5.8. Given a partial input point cloud, an asymmetrical Siamese auto-encoder is first adopted to reconstruct the coarse point cloud Y_{coarse} in an unsupervised learning way. It maps the partial and complete point clouds in a pre-built database into a shared latent space, with details described in Section 5.3.1. Then, a refinement unit is proposed to refine Y_{coarse} for producing fine-grained details, which is explained in Section 5.3.2.

5.3.1 Asymmetrical Siamese auto-encoder

To make the global features extracted from partial point clouds have more shape prior information, an asymmetrical Siamese auto-encoder network is explored in an unsupervised learning way, as

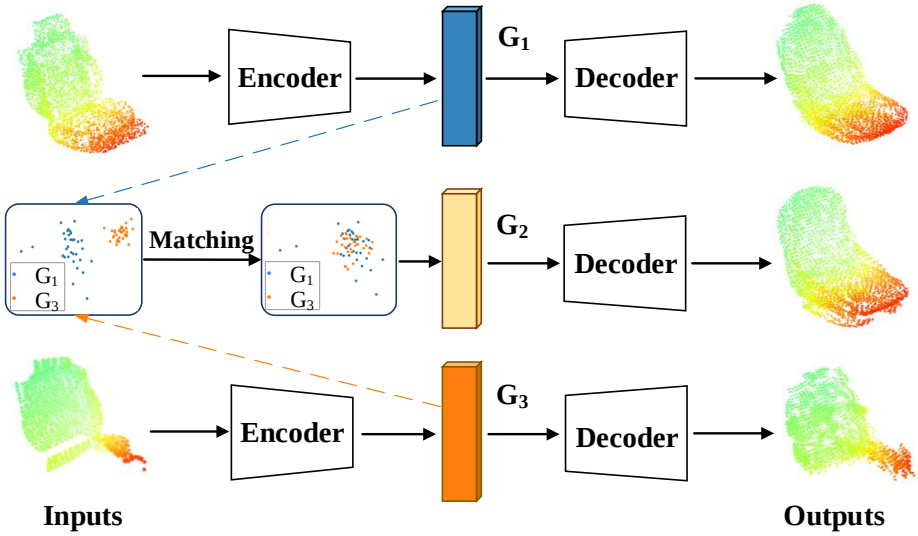


Figure 5.7: Illustration of the feature matching strategy in the proposed ASFM-Net. The first and the third row show the auto-encoders for the complete and partial point clouds, respectively. G_1 and G_3 represent the global features encoded from complete and partial point clouds, respectively. It is clearly seen the spatial distribution between G_1 and G_3 becomes consistent (G_3 evolved into G_2) after feature matching. The matched features G_2 can be used to generate the complete outputs.

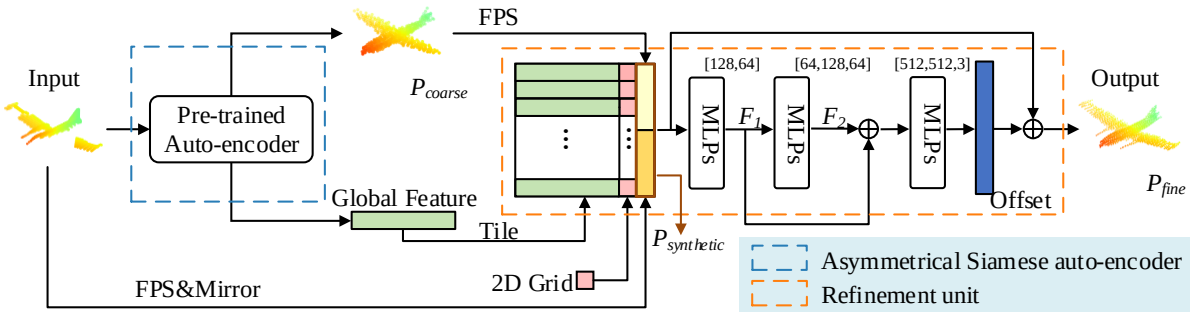


Figure 5.8: The overall architecture of ASFM-Net. ASFM-Net adopts a coarse-to-fine fashion to generate a dense and complete output: the asymmetrical Siamese auto-encoder module (blue) aims to provide a coarse point cloud and a global feature with shape prior; the refinement unit aims to preserve the details in the input and reconstruct the complete output with fine-grained geometry.

shown in Fig. 5.10. It consists of two AutoEncoder modules and a metric learning mechanism. The two AutoEncoder modules have identical architecture, and PCN [Yuan et al., 2018] is chosen as the backbone network. Note that any off-the-shelf point cloud feature extraction networks, e.g., PointNet [Qi et al., 2017a], FoldingNet [Yang et al., 2018], etc., can replace PCN serving as the backbone seamlessly. However, it is experimentally found PCN in the proposed ASFM-Net achieves the best performance.

Backbone network PCN. The encoder in PCN consists of two stacked PointNet (PN) layers. A shared MLP consumes $N \times 3$ input point cloud into a point-wise feature vector F_1. Then, a max-pooling operation is used to obtain a global feature g_1. Secondly, another PN layer takes F_1 and g_1 as inputs. It first concatenates g_1 to each point feature in F_1 and then passes the augmented feature through a shared MLP and a max-pooling layer for getting the final global feature g_2. The decoder in PCN adapts a multistage point generation strategy, which includes a

fully-connected decoder [Achlioptas et al., 2018] and a folding-based decoder [Yang et al., 2018]. In the first stage, the fully-connected decoder generates a coarse point cloud by passing g_2 through three fully-connected layers. In the second stage, the folding-based decoder refines the coarse output to a dense point cloud with fine-grained details.

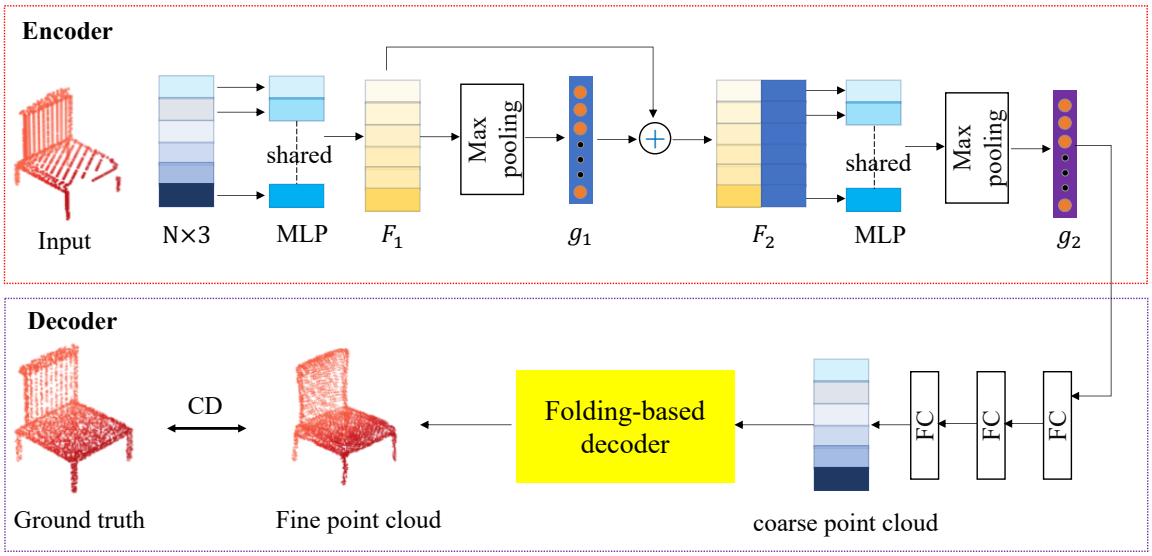


Figure 5.9: The network architecture of PCN [Yuan et al., 2018]. The encoder extracts a global feature vector g_2 from the input point cloud. The decoder adopts g_2 to first produce a coarse point cloud followed by a fine output. The Chamfer Distance (CD) is used to measure the difference between the outputs and the ground truth.

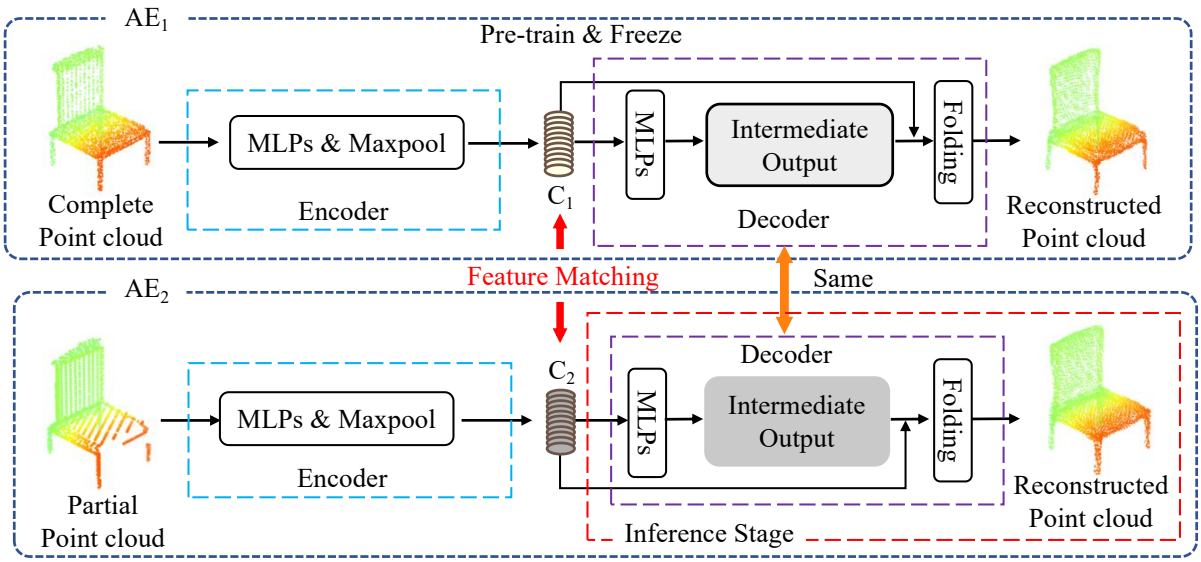


Figure 5.10: The network architecture of asymmetrical Siamese auto-encoder. An AutoEncoder AE_1 (the upper part) is first trained and freezes the weights to produce a codeword C_1 from a complete point cloud. Then, the encoder of another AutoEncoder AE_2 is trained to map the partial point codeword C_2 to be consistent with C_1. In the inference stage, the decoder of AE_1 is applied to transform the C_2 to a new and complete reconstructed point cloud.

Network details. Inspired by FoldingNet [Yang et al., 2018], an AutoEncoder AE_1 (the upper part in Fig. 5.10) is first trained for complete point clouds. The encoder takes each complete point cloud in the pre-built database as input and maps it to a high-dimensional codeword C_1. A decoder reconstructs point clouds to the original shape using this codeword. Note that a point cloud does not match a ground truth label, it thus is in a self-supervised learning way. In the experiments, the codeword length is set as 1024. Once the training process is finished, all weights of the AutoEncoder AE_1 will be frozen. Then, the second AutoEncoder AE_2 (the lower part in Fig. 5.10) is designed for partial point clouds. The encoder also maps the partial point cloud into a 1024-dimensional codeword C_2. The distribution of C_1 and C_2 is expected to be consistent by optimizing the feature matching distance. In the inference stage, the decoder of AE_1 with fixed weights will transform the C_2 to a new and complete reconstructed point cloud. Notably, the weights of the encoder in AE_2 are only updated using a feature-matching loss in the training stage. Experiments in Section 8.3.2 demonstrate the codeword obtained using the proposed feature matching strategy is more effective than the global features directly extracted from the partial inputs.

5.3.2 Refinement unit

Although the asymmetrical Siamese auto-encoder can extract a more effective global feature and generate a coarse point cloud P_{coarse}, the fine details of the input are inevitably lost. To preserve the detailed information of the input point cloud, following [Wang et al., 2020a], the partial inputs with the P_{coarse} are concatenated to form a synthetic point cloud $P_{synthetic}$ using the farthest points sampling algorithm and mirror operations. Various symmetry operations are explored, including plane symmetry, projective symmetry, and affine transformation operations. Experiments confirm that the XY-plane symmetry achieves the best performance. Inspired by FoldingNet [Yang et al., 2018], a 2D grid generator is utilized and concatenates these 2D grids with each point coordinate to increase the variability of each point. In order to narrow the distribution difference between the partial and the complete point cloud, the refinement unit concatenates the global feature with the coordinate of each point in $P_{synthetic}$. Due to the superiority of neural networks in residuals prediction [Wang et al., 2018], the refinement unit predicts the coordinate offset for every point between the point set $P_{synthetic}$ and the ground truth point cloud. Specifically, the $P_{synthetic}$ is passed through a series of bottom-up and top-down structural styles of MLPs. Overall, the final completed point cloud P_{fine} after the refinement unit can be expressed as:

$$P_{fine} = R\left(P_{synthetic}\right) + P_{synthetic}, \tag{5.7}$$

where $R(\cdot)$ denotes the function of predicting the coordinate residuals for the $P_{synthetic}$. Besides, P_{fine} is regarded as the synthetic point cloud $P_{synthetic}$ for a new loop when a higher point resolution is required. The point resolution will be doubled by iterating the refinement operation continuously.

5.3.3 Loss function

The training loss consists of two components, a feature-matching loss, and a reconstruction loss. The former requires a more similar distribution of partial and complete point clouds and the latter expects the topological distance between the completed point clouds and the ground truth as small as possible.

Feature matching loss. Various metrics for feature matching have been experimented with, such as cosine similarity [Nguyen & Bai, 2010] and Euclidean distance [Danielsson, 1980]. Finally,

the Euclidean distance is chosen due to its best performance on this work. The similarity between two high-dimensional feature vectors can be calculated as the following equation:

$$\mathcal{L}_{feat}(X,Y) = \sum_{i=0}^{n} \|F_{p_i} - F_{c_i}\|_2, \tag{5.8}$$

where X and Y represent the partial and complete point clouds, respectively. $F_p = (x_1, x_2, \cdots, x_n)^T$ and $F_c = (y_1, y_2, \cdots, y_n)^T$ denote the features encoded from X and Y.

Reconstruction loss. Following the previous work VPC-Net, Chamfer Distance (CD) is used to evaluate the similarity between two sets of point clouds. There are two forms of CD: CD-T and CD-P. The definitions of CD-T and CD-P between two point clouds P and Q are as follows:

$$\begin{aligned}
\mathcal{L}_{P,Q} &= \frac{1}{N_P} \sum_{p \in P} \min_{q \in Q} \|p - q\|_2^2, \\
\mathcal{L}_{Q,P} &= \frac{1}{N_Q} \sum_{q \in Q} \min_{p \in P} \|p - q\|_2^2, \\
\mathcal{L}_{CD-T}(P,Q) &= \mathcal{L}_{P,Q} + \mathcal{L}_{Q,P}, \\
\mathcal{L}_{CD-P}(P,Q) &= (\sqrt{\mathcal{L}_{P,Q}} + \sqrt{\mathcal{L}_{Q,P}})/2,
\end{aligned} \tag{5.9}$$

where N_P and N_Q are the amounts of points in P and Q, respectively. Notably, CD-P is used in all experiments during the training stage.

Overall loss. The overall loss function is the weighted sum of the feature-matching loss and the reconstruction loss. Both the predicted coarse point clouds P_{coarse} and final results P_{final} are optimized via the CD loss. More formally, the overall loss is defined as:

$$\begin{aligned}
\mathcal{L}_{sum} = \alpha \mathcal{L}_{feat}(X,Y) + \gamma \mathcal{L}_{CD}(P_{final}, P_{gt}) \\
+ \beta(\mathcal{L}_{CD}(P_{coarse}, P_{gt}),
\end{aligned} \tag{5.10}$$

where P_{gt} is the ground truth point cloud. α, β, and γ are all hyperparameters to balance their relationship, which are changed with the training steps synchronously.

6 Experiments

In this chapter, the experimental setup is explained in detail, including experimental datasets, as well as the evaluation metrics for experimental results analysis.

6.1 Experimental design

Extensive experiments on various datasets were conducted to evaluate the performance of the methods proposed in Sections 3-5. These experiments are classified into three major groups.

The first group is the test of point cloud based place recognition methods. Since point cloud based place recognition methods were divided into 3D submap-based place recognition and single-scan-based place recognition, these proposed methods were tested using different benchmark datasets. Specifically, SOE-Net is a 3D submap-based place recognition method, the experiments are conducted on the Oxford RobotCar [Maddern et al., 2017] and three in-house datasets proposed in [Angelina Uy & Hee Lee, 2018]. To showcase the performance of the proposed CASSPR on single scans, the TUM City Campus [Zhu et al., 2020] and USyd Campus [Zhou et al., 2020] datasets are also used for training and evaluation. Both of them were collected from LiDAR sensors mounted on a moving vehicle in multiple dynamic urban environments at different times. Note that the data are collected from different LiDAR sensors (Two Velodyne HDL-64E LiDAR sensors in the TUM City Campus, one Velodyne VLP-16 in USyd Campus), and in different cities and countries (Germany, Australia). The methods were implemented in the framework of Tensorflow and PyTorch and performed on four NVIDIA V100 (Pascal) 32 GB GPUs.

The second group is the test of the performance of the 3D object detection and tracking methods using the TUM City Campus dataset [Zhu et al., 2020] and two public benchmark datasets, including KITTI [Geiger et al., 2013] and NuScenes [Caesar et al., 2020] datasets. The proposed 3D detector was tested on the TUM City Campus dataset, while it was trained on KITTI. The proposed single object tracker DMT was trained and tested on KITTI and NuScenes datasets, respectively. The experiments were implemented using Pytorch and conducted on a computer with four NVIDIA V100 (Pascal) 32 GB GPUs.

The third group is the test of the performance of the proposed 3D shape completion methods using several public benchmark datasets, including the 3D synthetic datasets and raw LiDAR datasets. To test the generalization ability of shape completion methods, the real-scanned LiDAR datasets were used to test while the proposed methods were trained on the 3D synthetic datasets. Specifically, the VPC-Net was tested on ShapeNet [Chang et al., 2015], KITTI [Geiger et al., 2013], and TUM [Zhu et al., 2020] datasets. The ASFM-Net was tested on two synthetic benchmark datasets, including PCN [Yuan et al., 2018] and Completion3D datasets, both created from the ShapeNet dataset. The generalization test is conducted on the KITTI dataset. The experiments were implemented using TensorFlow and conducted on a computer with an NVIDIA TITAN X (Pascal) 12 GB GPU.

The details of experimental datasets and the prepossessing will be provided in the following sections.

6.2 Experimental datasets

Three groups of experimental datasets are designed in this work. The first one is for the test of point cloud based place recognition methods. The second one is the test of the 3D object detection and tracking methods. The third one is for the test of the 3D shape completion methods.

6.2.1 Oxford RobotCar and In-house datasets

The Oxford RobotCar [Maddern et al., 2017] dataset is published by University of Oxford, which consists of data recorded over a year-long and over 1000 km. It is collected by a SICK LMS-151 2D LiDAR scanner mounted on a car that travels around the region of Oxford repeatedly at different times. 44 sets of full and partial runs are used in the place recognition work. For each run, the collected 2D scans are accumulated to build a unique environment map. The map is then used to construct a database of submaps that represent unique local areas of the region for each run. Each submap is built with respect to the UTM coordinate frame using GPS/INS readings [Angelina Uy & Hee Lee, 2018].

The training and testing submaps are split into 70% and 30% for each run, respectively. And the training and testing submaps have fixed regular intervals of 10m and 20m, respectively. 3D points that are within a 20m trajectory are included in each submap. Thus, in total, 21,711 training submaps are used for training and 3030 testing submaps.

Different from Oxford RobotCar dataset, the three in-house datasets of a university sector (U.S.), a residential area (R.A.), and a business district (B.D.) are created from five different runs using a Velodyne-64 LiDAR sensor. For testing the generalization ability of the proposed methods trained only on Oxford RobotCar, all of them are used as testing maps. Henceforth, this is referred to as *Baseline Network*. Furthermore, each run of the U.S. and R.A. is geographically divided into training and testing reference maps, which are added to *Refinement Network*.

To better learn geometric features, the non-informative ground planes of all reference submaps are removed. The size of each submap is downsampled to 4096 points. In training, point clouds are regarded as correct matches if they are at a maximum of 10 m apart and wrong matches if they are at least 50 m apart. In testing, the retrieved point cloud is regarded as a correct match if the distance is within 25m between the retrieved point cloud and the query scan. Fig. 6.1 shows examples of downsampled point clouds from Oxford RobotCar, U.S., R.A., and B.D., respectively. The numbers of training and testing submaps used in the Baseline and Refinement networks are presented in Table. 6.1.

Table 6.1: Number of training and testing submaps for Baseline and Refinement networks on Oxford RobotCar dataset and in-house datasets.

Datasets	Training		Test	
	Baseline Network	Refinement Network	Baseline Network	Refinement Network
Oxford RobotCar	21711	21711	3030	3030
In-house datasets		6671	4542	1766

6.2.2 TUM City Campus dataset

TUM City Campus dataset [Zhu et al., 2020] contains two recording runs at the city campus of Technical University of Munich, which has been acquired by Fraunhofer IOSB with their

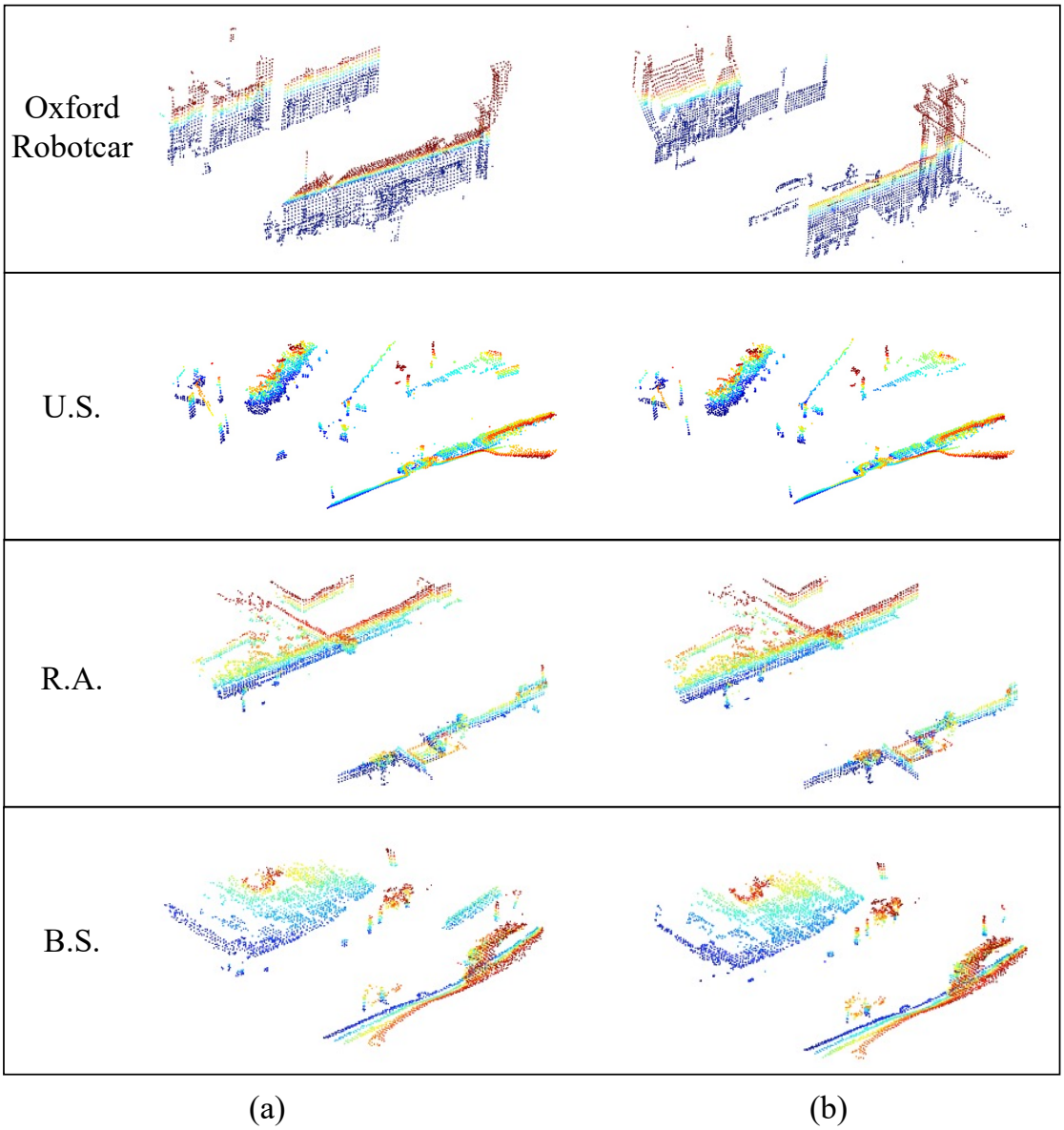


Figure 6.1: The downsampled point clouds from Oxford RobotCar, U.S., R.A., and B.D., respectively. a) shows the query submap, and b) shows the corresponding point cloud with the same location in another run.

MODISSA (Mobile Distributed Situation Awareness) sensor platform in April 2016 and December 2018 respectively. Two Velodyne HDL-64E LiDAR sensors are mounted on MODISSA, where each one has a 10 Hz rotational frequency. The entire point cloud covers an urban area of approximately 0.2 km^2, with around 1 km along roadways. The acquisition resulted in more than 10500 scans for each run. A single scan includes 130K points per rotation and covers a large area of 120 meters in diameter. In addition, the scans are recorded synchronously with position and orientation using an Applanix POS LV 520 inertial navigation system (INS), which was augmented by real-time kinematic (RTK) correction data of the German SAPOS network. Thus, each reference scan can be built with respect to the UTM coordinate frame using accurate GPS/INS readings. The data acquisition and the examples of single reference scan are shown in 6.3.

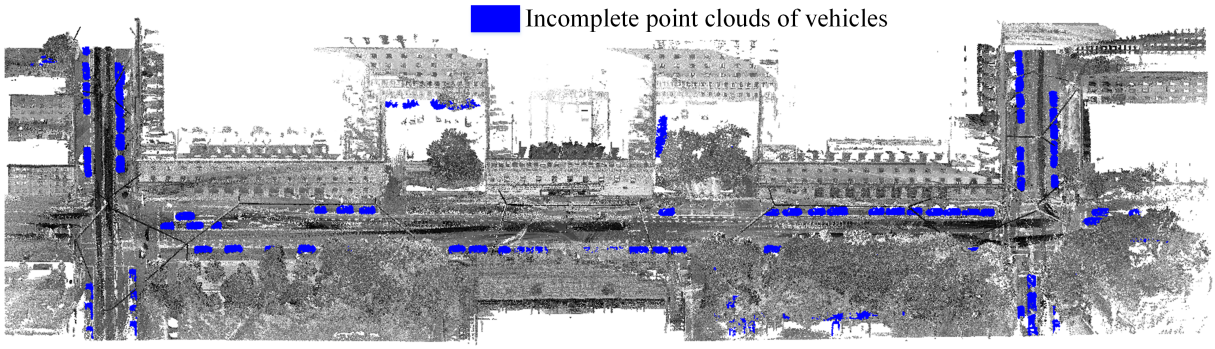


Figure 6.2: Point clouds of Arcisstrasse from the TUM City Campus dataset. The vehicle points and background points are shown in blue and gray colors, respectively.

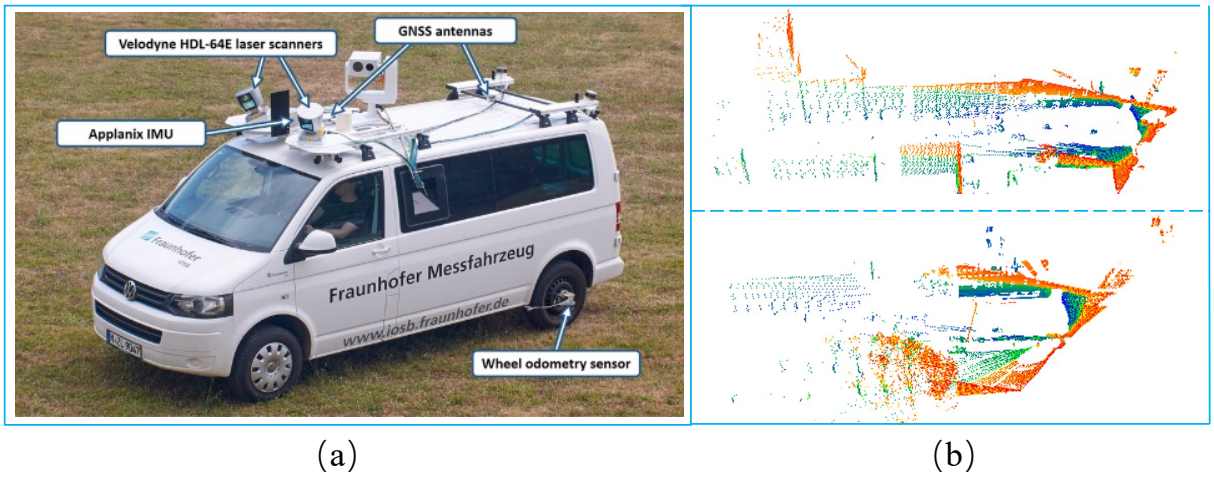
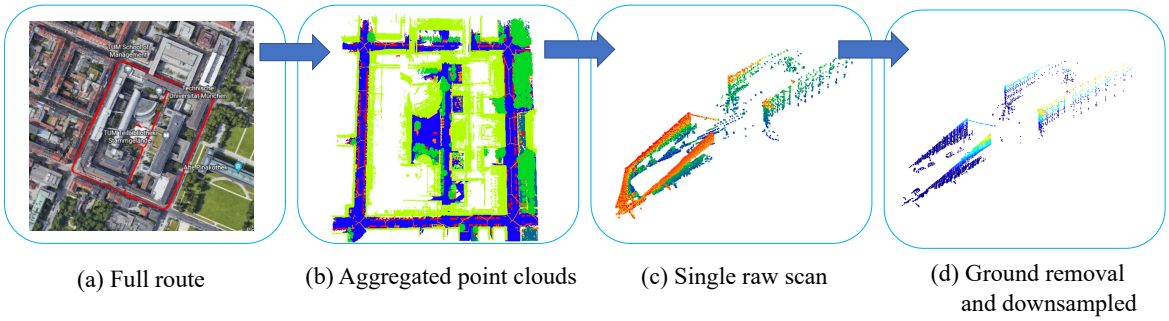


Figure 6.3: TUM City Campus dataset. a) Collected by MODISSA platform from Fraunhofer IOSB .[Borgmann et al., 2018] b) Examples of data acquired by the two obliquely mounted laser scanners.

In addition, the TUM City Campus dataset provides an aggregated point cloud of the whole obtained sequence. It includes more than 40 million annotated points with labels for eight classes of objects. In Fig. 6.2, an illustration of scanned vehicles on the Arcisstrasse of this dataset is given. As shown in Fig 6.2, the point clouds of vehicles in the TUM dataset are denser than those in the KITTI dataset. They are also incomplete, although the missing content is less severe.

Reference scan preprocessing. The pre-processing of the reference scan includes three main steps: coordinates transformation of the LiDAR sensor location, ground removal, and point cloud downsampling. The coordinates of one LiDAR location are first transformed from a local Euclidean coordinate system East-North-Up to the geocentric Earth-centered, Earth-fixed (ECEF) coordinate system using an affine transformation matrix. The transformed ECEF coordinates are then converted to Latitude, Longitude and Altitude coordinate system (LLA) for generating a visualization of the vehicle trajectory (The red line in Fig. 6.5). Finally, they are converted to UTM coordinates as the ground-truth locations. The LiDAR frames are split at fixed regular intervals of 5 meters without overlapping based on the sensor position of each point cloud frame. The ground planes are removed in the collected point cloud frames. Besides, the point clouds are downsampled to 4096 points using a voxel grid filter. Each frame is shifted to



(a) Full route (b) Aggregated point clouds (c) Single raw scan (d) Ground removal and downsampled

Figure 6.4: TUM City Campus dataset preprocessing: a) A full trajectory from the TUM dataset. b) An aggregated point cloud of the whole obtained sequence in the red line shown in a). c) An example of a single raw LiDAR scan. d) A downsampled scan that is removed the ground plane and all points within [-1,1] m.

zero mean and normalized within the range of [-1, 1]. Fig. 6.4 illustrates the pipeline of a single scan pre-processing, including a reference map, a raw LiDAR scan, and the downsampled scan.

Data splitting and evaluation The LiDAR scans collected at each run of the TUM datasets are split into two disjoint reference maps used for training and testing. Fig. 6.5 shows the training area and test area. Each run is geographically split into 80% and 20% for training and testing without intersection. This resulted in 675 scans to train and 162 scans to test. During training, the point clouds are defined at most 5m apart as structurally similar and at least 12.5m apart as structurally dissimilar. In the inference stage, a single scan from one testing reference map is used as a query to find matches from the other testing reference map in a different year that is within the range of 5m. Fig. 6.6 shows examples of downsampled point clouds from the TUM dataset.

6.2.3 USyd Campus dataset

The USyd Campus (USyd) Dataset [Zhou et al., 2020] contains LiDAR scans collected from a buggy-like car when driving the same route around University of Sydney over 50 weeks in varying weather conditions. An array of sensors include a Velodyne VLP-16 LiDAR, six cameras, and GPS/IMU. The locations recorded from GPS are served as ground truth in the place recognition task. Fig 6.7 shows the trajectory of the car and training/test areas. Following [Żywanowski et al., 2021], the consecutive LiDAR scans are split with the 5 meters distance, resulting in about 735 scans per every 40 runs. And each scan includes up to 25,000 3D points and covers a 100 meters size area. The distributions are kept as same as the raw scans and the point clouds are downsampled to 4096 points using a voxel grid filter. Notably, the ground planes of every single scan are not removed.

The vehicle trajectory and training/test areas are shown in Fig. 6.7. The four test sections include $100 \times 100m^2$ areas. In total, 19,138 training and 8797 test 3D LiDAR scans are obtained. Fig. 6.8 shows examples of downsampled point clouds from the USyd dataset.

6.2.4 3D vehicle dataset

In this work, the synthetic CAD models on the category of cars from ShapeNet[Chang et al., 2015] are used to create a 3D vehicle dataset containing pairs of partial and complete point clouds, in order to train the proposed VPC-Net. Specifically, the 3D vehicle dataset includes a

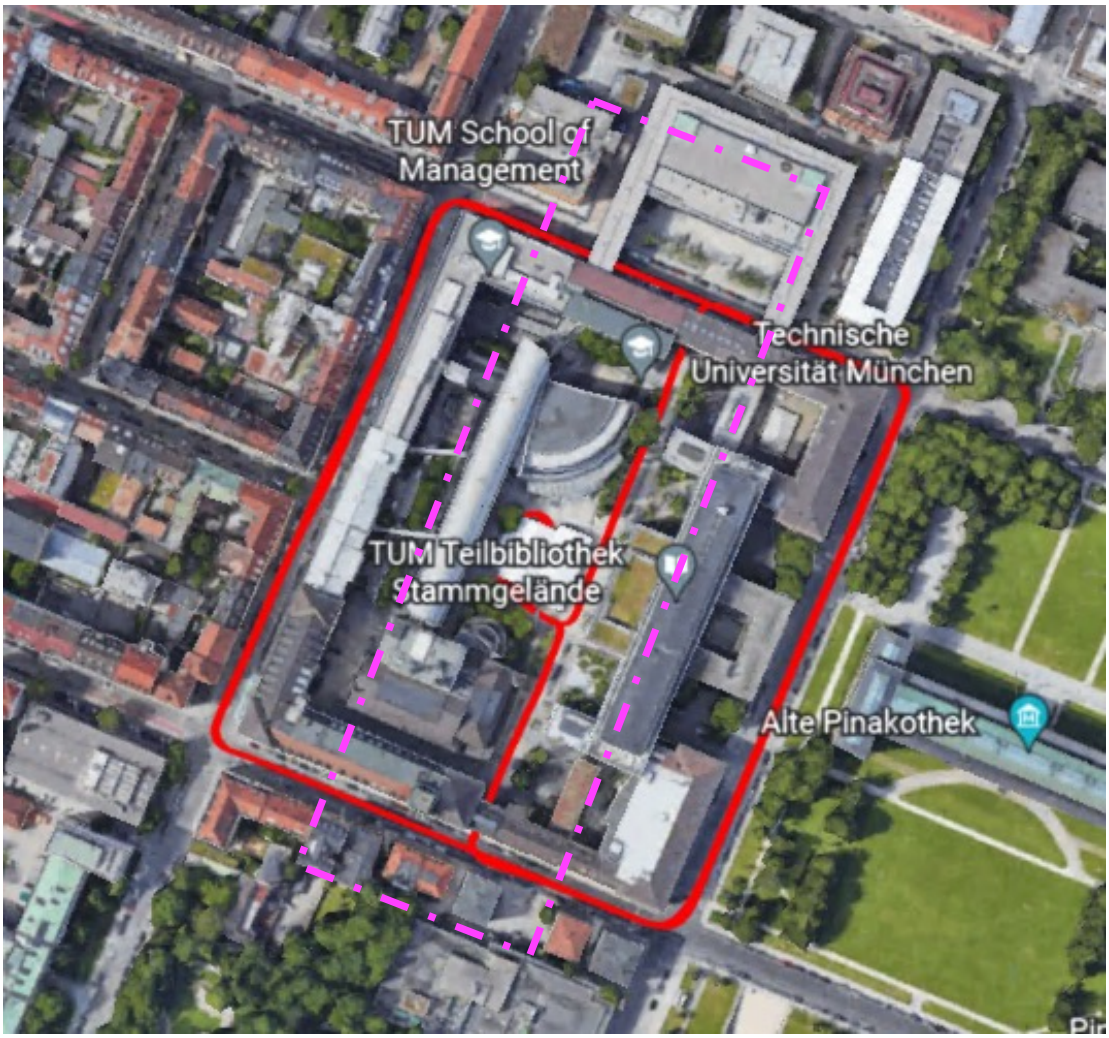


Figure 6.5: Visualization of the trajectory of the vehicle that repeatedly drives through TUM city campus in 2016 and 2018. The red lines in the magenta box are the test area, and the others are the training area.

total of 5677 different instances of vehicles, which are split into training, validation, and test data. Among them, 100 instances are used for validation, and 150 instances are utilized for testing. The remaining instances are reserved for training. For creating complete point clouds as ground truth, for each CAD model of a vehicle instance, 16,384 points are sampled uniformly on the surface of each CAD model of a vehicle as the synthetic point cloud. Fig. 6.9 shows examples of complete point clouds of vehicle instances from CAD models in ShapeNet. Instead of using subsets of complete point clouds as partial inputs, the CAD models of vehicle instances are rendered to a set of depth images from a variety of view angles and then back-projected to different view planes to generate partial point clouds. This operation can make the incomplete distribution of partial point clouds closer to real-scan data.

Following the data generation in PCN [Yuan et al., 2018], the pipeline of generating partial inputs from the ShapeNet dataset is illustrated in Fig. 6.10. The depth images are generated by placing a virtual RGB-D camera at different view angles. The camera is designed to be oriented toward the center of the 3D model. A series of viewpoints are randomly selected to generate

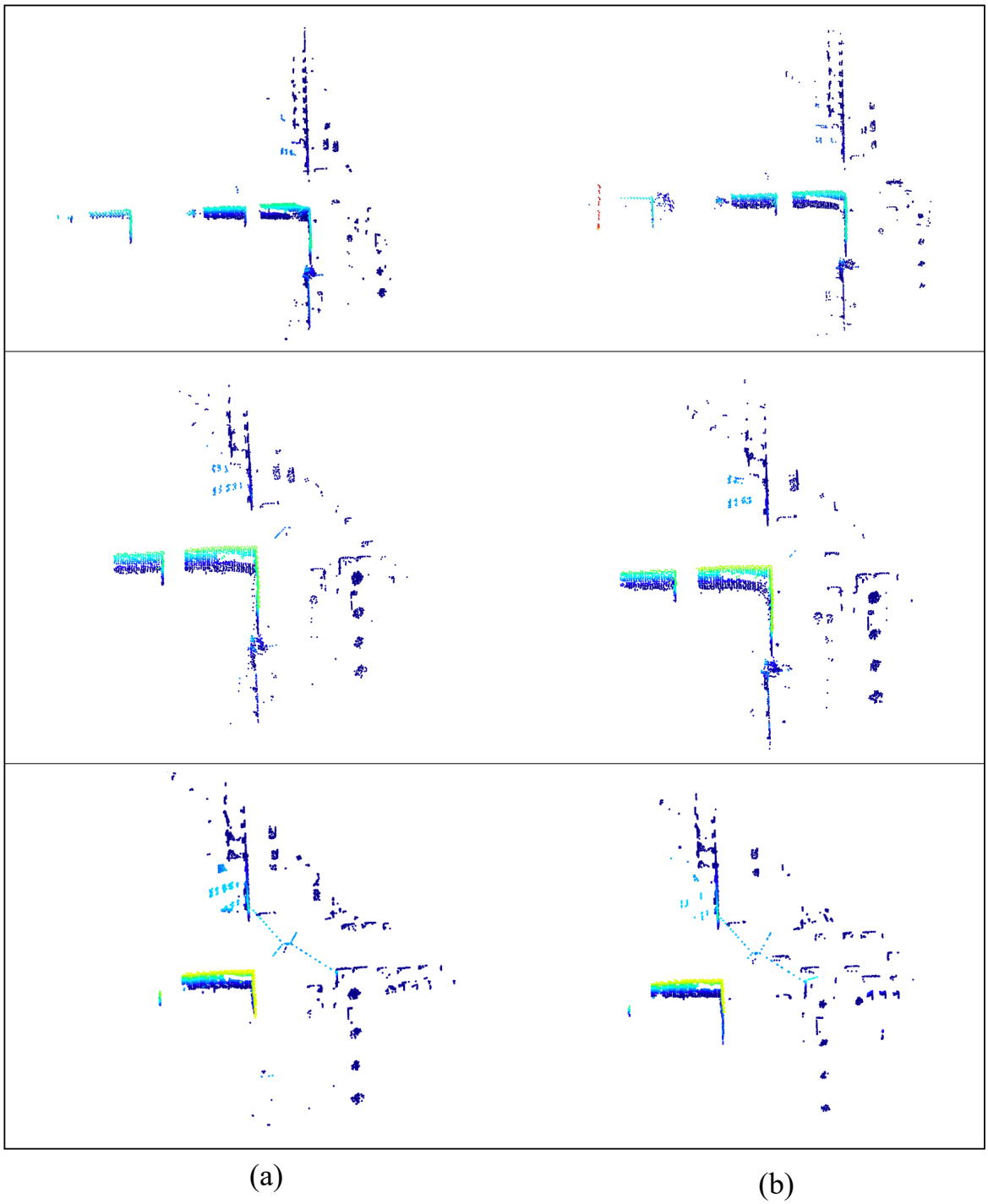


Figure 6.6: The downsampled point clouds from the TUM dataset. a) shows the query single scan, and b) shows the corresponding point cloud with the same location in another run.

incomplete shape scans through limited view access. Lastly, the resulting depth maps are back-projected to form partial point clouds. In this work, eight randomly distributed viewpoints are chosen to generate eight partial point clouds for each training 3D CAD model of a vehicle. Notably, the resolution of these partial scans can be different. The reason for generating training point clouds from a synthetic 3D dataset is that it consists of a wide variety of complete and detailed 3D

Figure 6.7: Visualization of the trajectory repeatedly driving through the University of Sydney campus. The red lines in the magenta box are the test area, and the others are the training area.

vehicle models, while they are not available in real-scanned LiDAR datasets. Moreover, scanning thousands of vehicles using LiDAR systems for acquiring complete point clouds as the ground truth is quite time-consuming and labor-intensive, which is not a practical solution. Recently, some high-quality 3D reconstruction datasets have emerged such as ScanNet [Dai et al., 2017a] and S3DIS [Armeni et al., 2017], which can also provide training data with high quality. However, they are mainly focused on indoor scenes, not including any objects in outdoor scenarios.

6.2.5 PCN dataset and Completion3D benchmark

PCN dataset [Yuan et al., 2018] is created from the ShapeNet dataset [Chang et al., 2015], containing pairs of complete and partial point clouds. Notably, each complete model includes 16,384 points and is corresponding to eight partial point clouds. The dataset covers 30974 CAD models from 8 categories: airplane, cabinet, car, chair, lamp, sofa, table, and watercraft. Following [Yuan et al., 2018], the number of models for validation and testing are 100 and 150, respectively. The remaining models are used for training. In the experiments, the complete shapes are uniformly downsampled from 16,384 points to 4096. The performance is evaluated on the resolution containing 4096 points.

Completion3D benchmark* is released by TopNet [Tchapmi et al., 2019], which is a subset of the ShapeNet dataset derived from the PCN dataset. Different from the PCN dataset, the resolution of both partial and complete point clouds is 2048 points. Moreover, each complete model is only corresponding to one partial point cloud. The train/test split is the same as the PCN dataset.

*https://completion3d.stanford.edu/.

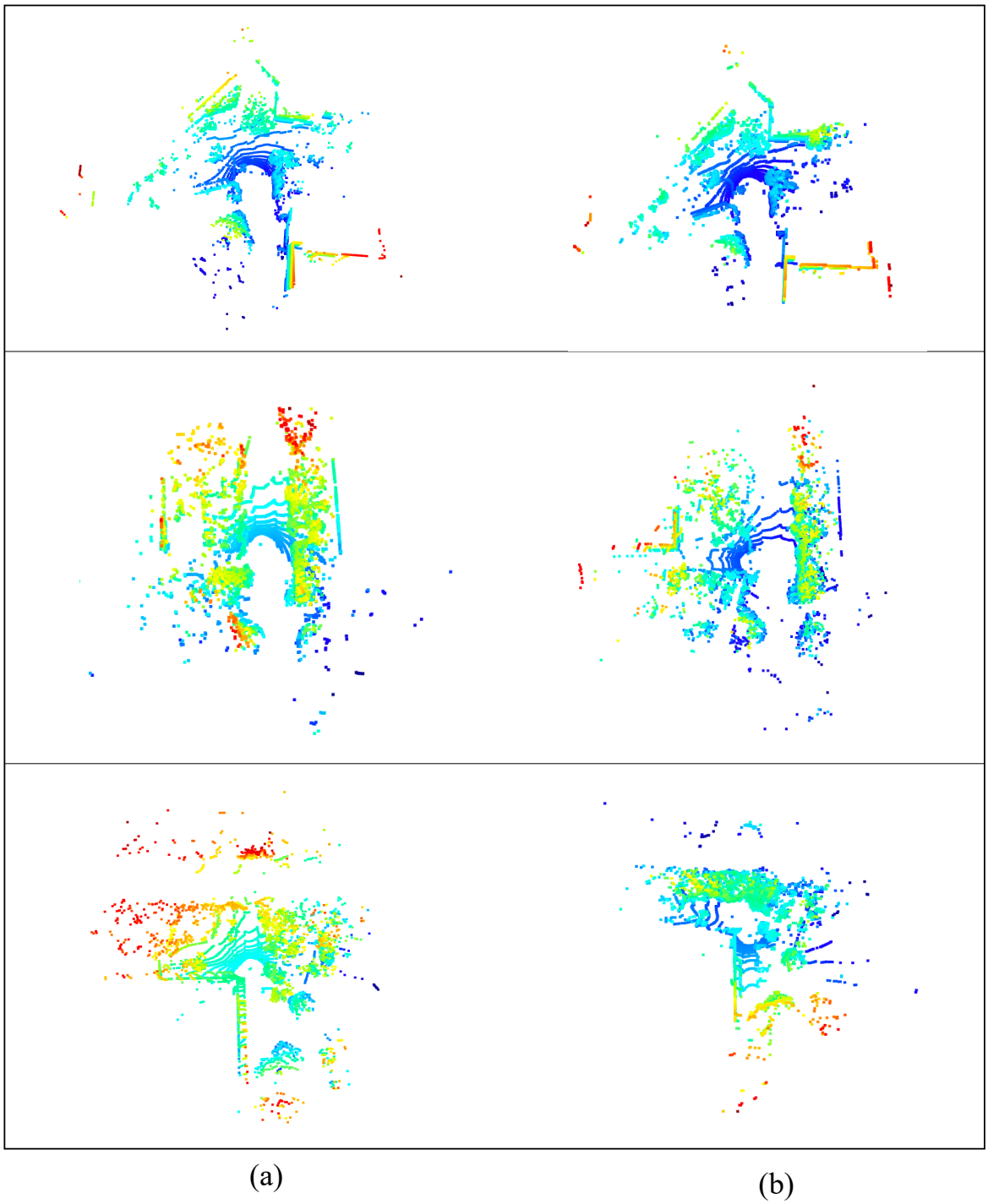


Figure 6.8: The downsampled point clouds from the USyd dataset. a) shows the query single scan, and b) shows the corresponding point cloud with the same location in another run.

6.2.6 KITTI 3D object detection dataset

The KITTI 3D object detection dataset [Geiger et al., 2013] provides raw point clouds collected by the Velodyne HDL-64E rotating 3D laser scanner and annotations for vehicle instances in the form of 3D bounding boxes. It records six hours of traffic scenarios, which are diverse and capture real-world traffic situations with many static and dynamic vehicles. The raw dataset includes five

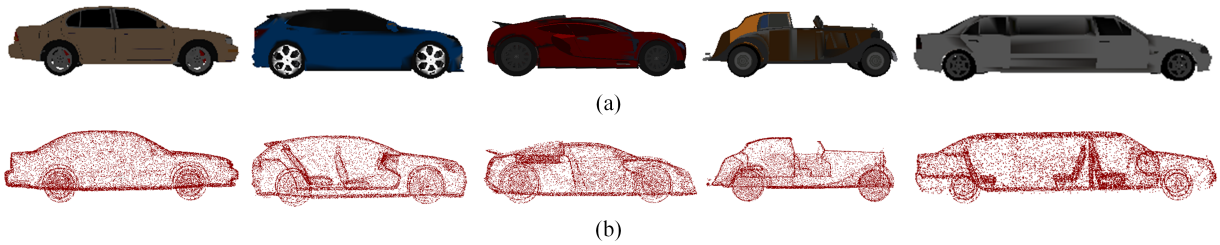


Figure 6.9: Examples of CAD models and sampled point clouds of vehicle instances from the ShapeNet dataset. a) CAD models of vehicle instances stored in ShapeNet. b) Generated complete point clouds sampled uniformly from these CAD models.

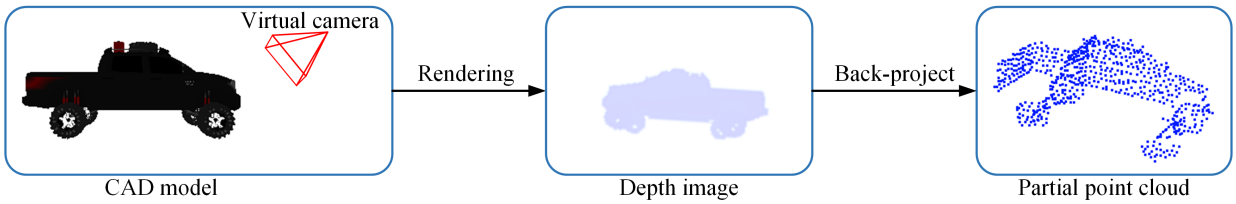


Figure 6.10: The pipeline of partial input generation.

categories, namely 'Road', 'City', 'Residential', 'Campus', and 'Person'. The data category 'City' is composed of about 28 sequences (i.e., 8477 frames). In each sequence of the raw data, apart from objects annotated with 3D bounding boxes, tracklets, and calibration are also provided. Three example frames in 'City' are shown in Fig. 6.11. The point clouds of vehicles are very sparse and exhibit a significant loss of content, while another is that target vehicles appear in an arbitrary location with variable sizes.

6.2.7 NuScenes dataset

The NuScenes dataset [Caesar et al., 2020] consists of 1000 scenes collected in Boston (Seaport and South Boston) and Singapore (One North, Holland Village, and Queenstown) from a 32 beams LiDAR scan, six cameras, and five radars, with a total length of 242 km. NuScenes annotated 23 object categories with accurate 3D bounding boxes across the entire dataset to facilitate object detection and tracking tasks, including various vehicles, types of pedestrians, mobility devices, and other objects. In total, NuScenes contains 1.4 million camera images, 400,000 Lidar sweeps, 1.3 million RADAR sweeps, and 1.1 million object bounding boxes in 40,000 keyframes. Specifically, the NuScenes dataset contains 32,302 frames in the car category, which is five times larger than the KITTI dataset. Following [Zheng et al., 2021a], the training set of NuScenes is used for training, and the validation set is used for testing. Tracklets with no points in the first bounding boxes are ignored during the evaluation.

6.3 Evaluation metrics

6.3.1 Evaluation metric of point cloud based place recognition

In order to evaluate the performance of place recognition, the location of each submap/scan is tagged with a UTM coordinate based on GPS/INS units. The evaluation is performed by selecting a single query submap/scan from the testing map while matching the nearest submaps/scans

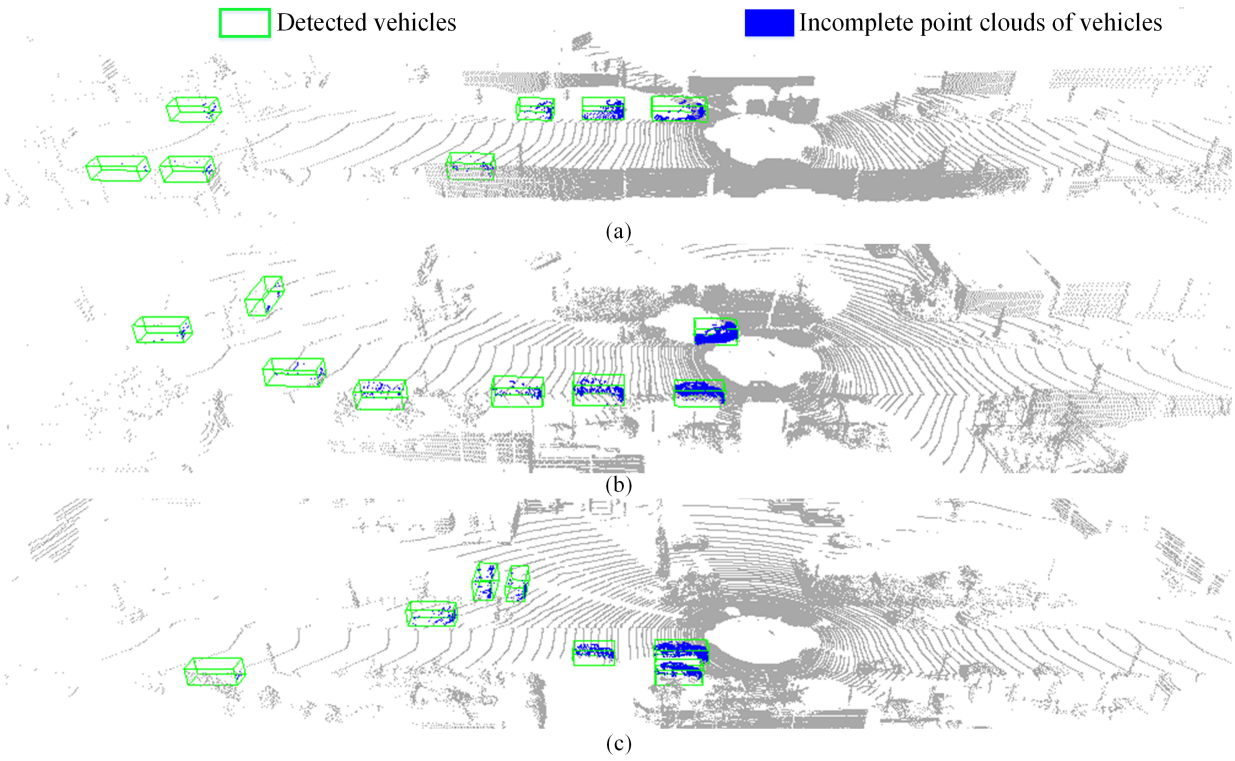


Figure 6.11: Example frames of the 'City' category from the KITTI dataset. The vehicle points, background points, and bounding boxes are shown in blue, gray, and green colors, respectively.

within d meters from another collection of reference submaps/scans at a different time. Note that the testing map is in a never-before-seen area. The Euclidean distance of the global descriptors is calculated and then nearest neighbors are performed for matching. Average Recall at Top N is used as an evaluation metric, which means the location is correctly recognized if the N most similar scans matched from the database contain at least one location within the distance d from the query. Top 1 counts the number of times the first match from the database matches the query location, which is actually important for robotic applications. The Top 1% results are also presented for comparing the state-of-the-art solutions following [Angelina Uy & Hee Lee, 2018].

In the TUM and USyd datasets, the retrieved scan is regarded as a correct match if the distance is within $d = 5m$ between the retrieved point cloud and the query scan. For Oxford RobotCar and in-house datasets, the threshold is set $d = 25m$.

6.3.2 Evaluation metric of 3D object detection and tracking

In this thesis, the performance of 3D object detection and tracking is evaluated by three widely used metrics: 'Average Precision (AP)', 'Success', and 'Precision'. AP is defined as the area under the curve (AUC) of precision-recall for object detection. The precision measures the percentage of correct predictions among all predictions and the recall measures the percentage of correct predictions among all ground truths. When the Intersection-over-Union (IoU) is greater than a predefined threshold, the prediction is correct. The 3D IoU is formulated as:

$$\phi_t = \frac{B_t^G \cap B_t^D}{B_t^G \cup B_t^D}, \tag{6.1}$$

where B_t^G and B_t^D are the ground-truth bounding box and the bounding box predicted by the detector, respectively.

For single object tracking, One Pass Evaluation (OPE) [Wu et al., 2013] is applied to measure the Success and Precision of different trackers. For a predicted bounding box and a ground-truth bounding box, 'Success' is defined as the IoU between them. 'Precision' is defined as the AUC for the distance error curve from 0 to 2m, which is measured between the centers of the two boxes. The success and precision metrics respectively measure the box overlap and center distance error between the predicted bounding box and the ground-truth bounding box.

6.3.3 Evaluation metric of 3D shape completion

The performance of the 3D shape completion method is evaluated by two commonly applied metrics: CD (see Eq. 5.4) and EMD (see Eq. 5.5), between the completed point cloud and the ground truth. The definitions of CD and EMD have been given in Section 5.2.4, where CD-T and CD-P are also defined by Eq. 5.9. For computing the metrics with a lower computational cost, the dimensions of both the ground truth and completed point clouds are normalized by regarding the length of the bounding box of length as one unit. For the Completion3D benchmark, the online leaderboard adopts CD-T. Thus, CD-T is adopted for experiments in Section 7.3.2 and CD-P for the other experiments.

7 Results and Analysis

In this chapter, the qualitative and quantitative experimental results of the proposed methods introduced in Chapters 3-5 are provided.

7.1 Point cloud based place recognition results

7.1.1 SOE-Net

Baseline network

The baseline network is compared with PointNetVLAD (PN_VLAD) [Angelina Uy & Hee Lee, 2018] as a baseline and the state-of-the-art methods PCAN [Zhang & Xiao, 2019], LPD-Net [Liu et al., 2019], DH3D [Du et al., 2020], and DAGC [Sun et al., 2020]. For a fair comparison, the same evaluation metrics are used, including the Average Recall at Top N and Average Recall at Top 1%. The final global descriptors of all networks are 256-dim. Table 7.1 shows the top 1% recall of each network on the four datasets. The recall values of DH3D for U.S., R.A. and B.D. are not reported in [Du et al., 2020].

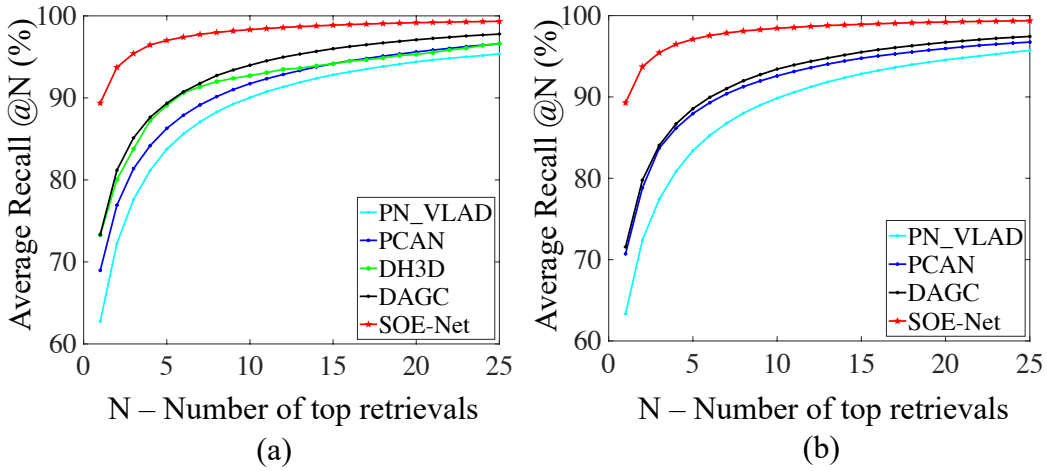


Figure 7.1: Average recall of SOE-Net tested on Oxford RobotCar. a) Baseline: shows the average recall when SOE-Net is only trained on Oxford RobotCar. b) Refinement: shows the average recall when SOE-Net is trained on Oxford RobotCar, U.S. and R.A. DH3D is not trained on this dataset in [Du et al., 2020].

The results show that the proposed baseline network outperforms others significantly on the Oxford RobotCar dataset. The best performance on Oxford RobotCar reaches the recall of 96.40 % at top 1%, exceeding the recall of the current state-of-the-art method LPD-Net by 1.52 %. Furthermore, SOE-Net achieves the recall of 93.17%, 91.47%, 88.45% on the unseen datasets

Table 7.1: The average recall (%) at top 1% for each network.

	SOE-Net (Proposed)	DAGC	DH3D	LPD	PCAN	PN_VLAD
Oxford	**96.40**	87.49	84.26	94.92	83.81	80.31
U.S.	93.17	83.49	-	**96.00**	79.05	72.63
R.A.	**91.47**	75.68	-	90.46	71.18	69.75
B.D.	88.45	71.21	-	**89.14**	66.82	65.30

Table 7.2: Average recall (%) at top 1% (@1%) and top 1 (@1) for each of the models trained on Oxford RobotCar, U.S. and R.A..

	Ave recall @1%			Ave recall @1		
	SOE-Net(Proposed)	DAGC	PCAN	SOE-Net(Proposed)	DAGC	PCAN
Oxford	**96.43**	87.78	86.40	**89.28**	71.39	70.72
U.S.	**97.67**	94.29	94.07	**91.75**	86.34	83.69
R.A.	**95.90**	93.36	92.27	**90.19**	82.78	82.26
B.D.	**92.59**	88.51	87.00	**88.96**	81.29	80.11

respectively, which is similar or slightly weaker than LPD-Net. However, both of them improve the performance by a large margin compared with other methods. Notably, LPD-Net relies on ten handcrafted features, which has complex network architecture and high computational cost. Fig. 7.1 (a) shows the recall curves of PointNetVLAD, PCAN, DAGC, and SOE-Net for the top 25 retrieval results. Notably, the recall at top 1 of SOE-Net reaches a recall of 89.37%, indicating the proposed network effectively captures the task-relevant local information and generate more discriminative global descriptors.

Refinement network

To improve the generalizability of the network on the unseen scenarios, [Zhang & Xiao, 2019; Angelina Uy & Hee Lee, 2018; Sun et al., 2020] further add U.S. and R.A. to the training data. The proposed refinement work is trained following the same training sets. As illustrated in Table 7.2, SOE-Net still significantly outperforms the state-of-the-art method DAGC on all datasets. By comparing Table 7.1 and Table 7.2, it becomes clear that adding more data from different scenarios improves the performance of SOE-Net on the unseen dataset B.D.. In other words, given more publicly accessible datasets of real scans, SOE-Net has huge potential for LiDAR based localization. In Fig. 7.1 (b) the recall curves of the refinement network of PointNetVLAD, PCAN, DAGC, and SOE-Net are plotted for the top 25 retrieval results. It demonstrates that the global descriptors generated by SOE-Net are more discriminative and generalizable than all previously tested state-of-the-art methods.

Results visualization

In addition to quantitative results, the selected qualitative results of some correctly retrieved matches are shown in Fig. 7.2. A full traversal is chosen randomly as the reference map on four benchmark datasets, respectively. Four query point clouds are chosen from other randomly selected traversals on their respective datasets, with each representing one sample submap from individual testing areas. For each instance, the query point cloud and the top 3 retrieved matches are shown on the left. It becomes clear that the best match has a very similar scene as the query point cloud. Besides, the location of each point cloud is displayed in the reference map on the right. For each query, the location of the top 1 result (indicated by the blue circle) is correctly overlapped with the query location (represented by the red cross). It shows that the proposed SOE-Net indeed has the ability to recognize places.

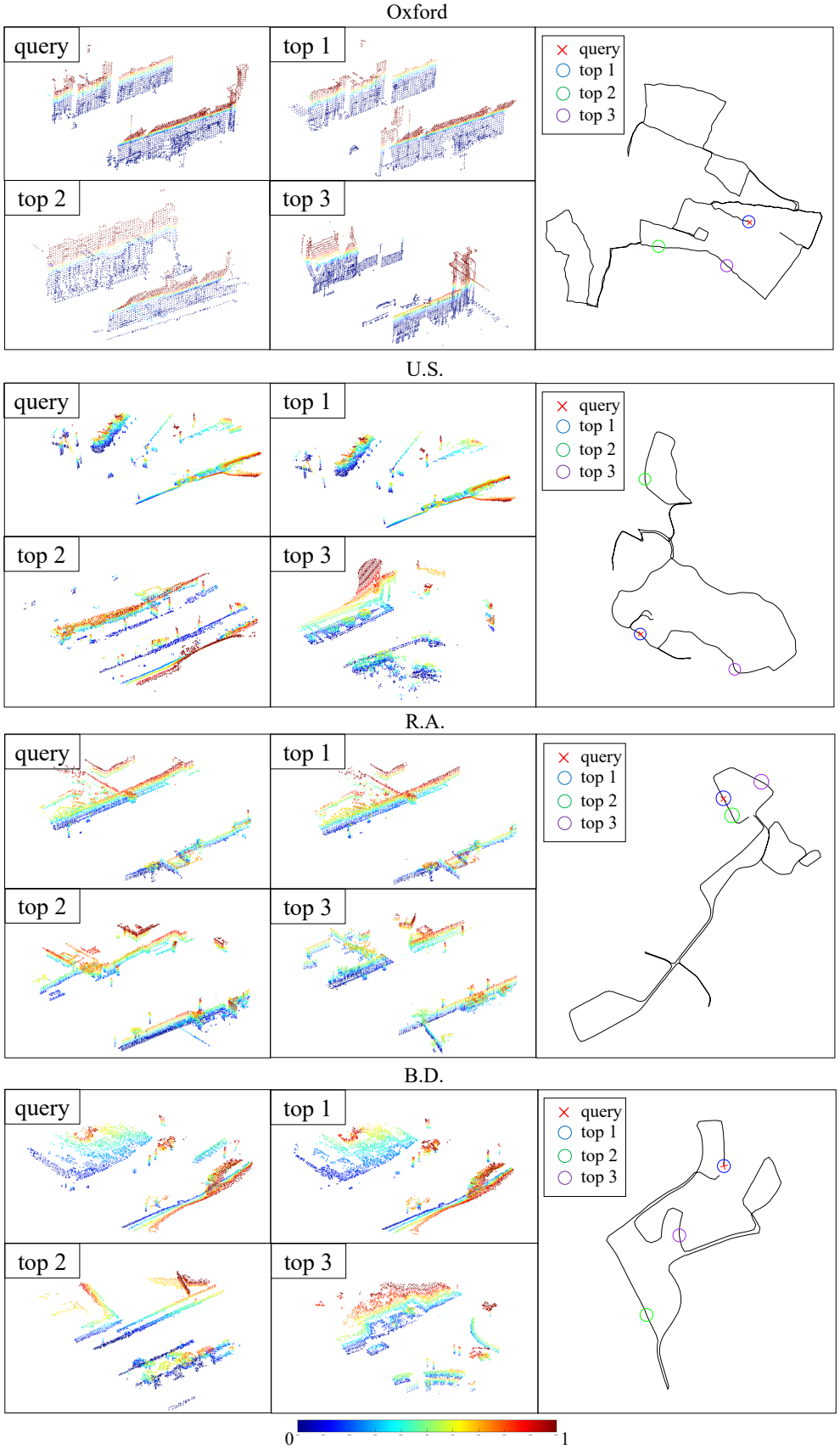


Figure 7.2: Visualizations of example retrieval results of SOE-Net on four benchmark datasets. For each retrieval, the query point cloud and the top 3 retrieved results are displayed. Locations of these point cloud are also indicated in the associated reference map. Colors in these point clouds represent heights above the ground.

Table 7.3: Average recall at top 1% (@1% and top 1 (@1) for each of the models trained on the TUM dataset.

TUM dataset	Ave recall @1%	Ave recall @1
PointNetVlad [Angelina Uy & Hee Lee, 2018]	76.3	61.9
PCAN [Zhang & Xiao, 2019]	87.8	71.2
SOE-Net [Xia et al., 2021a]	83.5	66.9
MinkLoc3D [Komorowski, 2021]	82.7	66.9
MinkLoc3D-S [Żywanowski et al., 2021]	85.7	69.1
CASSPR (Proposed)	**97.1**	**85.6**

7.1.2 CASSPR

Comparisons on TUM City Campus Dataset

The proposed CASSPR is compared with the state-of-the-art: PointNetVlad [Angelina Uy & Hee Lee, 2018], PCAN [Zhang & Xiao, 2019], SOE-Net [Xia et al., 2021a], MinkLoc3D [Komorowski, 2021], and MinkLoc3D-S [Żywanowski et al., 2021]. Besides, the same evaluation metrics are used and the dimensions of all global descriptors are set to 256-dim. Table 7.3 shows the top 1% and top 1 recall of each network on the TUM City Campus dataset.

The results show that the proposed CASSPR outperforms others significantly on the TUM dataset. The best performance on the TUM dataset reaches the recall of 97.1 % at top 1%, exceeding the recall of the current state-of-the-art method by 9.3 %. Furthermore, CASSPR achieves the recall of 85.6% at top 1, which has a significant advantage (16.6%) over MinkLoc3D-S. Fig. 7.3 shows the recall curves of PointNetVLAD, PCAN, SOE-Net, Minkloc3D, MinkLoc3D-S, and CASSPR for the top 25 retrieval results. Notably, the recall at top 1 reaches a recall of 86.4%, indicating the proposed CASSPR effectively captures the task-relevant local information and generates more discriminative global descriptors.

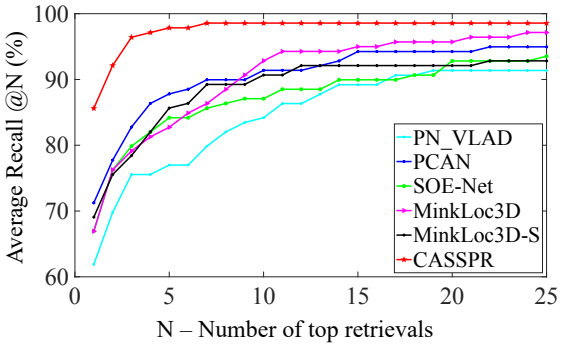


Figure 7.3: Average recall of CASSPR tested on the TUM dataset.

Results visualization on the TUM dataset

In addition to quantitative results, the selected qualitative results of some correctly retrieved matches and one failed case are shown in Fig. 7.4. A full traversal is made as the reference map on the TUM city campus dataset. Three query point clouds are chosen from the 2016 traversal, with each representing one single scan from the testing areas. For each scan, the query point cloud and the top 3 retrieved matches are shown on the left. It becomes clear that the best match has a very similar scene to the query point cloud. Besides, the location of each point cloud in the reference map is displayed on the right. For each query, the location of the top 1 result

USyd dataset	AR @1%	AR @1
PointNetVlad [Angelina Uy & Hee Lee, 2018]	81.7	60.7
PCAN [Zhang & Xiao, 2019]	86.4	68.7
SOE-Net [Xia et al., 2021a]	78.9	52.8
MinkLoc3D [Komorowski, 2021]	98.1	91.7
MinkLoc3D-S [Żywanowski et al., 2021]	98.8	93.9
CASSPR (Proposed)	**98.9**	**97.6**

Table 7.4: Average recall (%) at top 1% (@1%) and top 1 (@1) for each of the models trained on the USyd dataset.

(indicated by the blue circle) is correctly overlapped with the query location (represented by the red cross). Fig 7.4(a) illustrates the successful cases, which demonstrate CASSPR indeed has excellent recognition ability.

Fig 7.4(b) shows a failed retrieval case, in which the top 1 retrieved match has a similar scene to the query scan. It is seen that the highly similar scenes with different locations still confuse CASSPR. This is reasonable since CASSPR achieves recognition via fully mining the geometric information.

Comparisons on the USyd dataset.

Due to the small size of the TUM dataset, further experiments are conducted on a larger USyd Campus (USyd) Dataset in order to better demonstrate the superiority of CASSPR. As the references, the previous SOTA are trained and tested based on their published codes, including PointNetVlad [Angelina Uy & Hee Lee, 2018], PCAN [Zhang & Xiao, 2019], SOE-Net [Xia et al., 2021a], MinkLoc3D [Komorowski, 2021], and MinkLoc3D-S [Żywanowski et al., 2021]. The evaluation results are shown in Table. 7.4.

CASSPR achieves the best performance of 98.9% / 97.6% at AR@1% / AR@1, exceeding the performance of the previous SOTA MinkLoc3D-S by 3.9% on AR@1. It demonstrates that CASSPR can generate more discriminative global descriptors compared with point-based or voxel-based baselines.

Comparisons on the Oxford RobotCar and in-house datasets.

To further demonstrate the capability of CASSPR, the experiments are conducted on benchmark datasets introduced in [Angelina Uy & Hee Lee, 2018], which include the Oxford RobotCar dataset and three in-house datasets: University Sector (U.S.), Residential Area (R.A.), Business District (B.D.). Following the baseline networks proposed in [Angelina Uy & Hee Lee, 2018; Zhang & Xiao, 2019; Liu et al., 2019; Hui et al., 2022; Xia et al., 2021a; Komorowski, 2021; Zhou et al., 2021; Hui et al., 2021,?; Fan et al., 2022; Żywanowski et al., 2021], CASSPR is trained and tested on Oxford RobotCar dataset. U.S., R.A., and B.D. datasets are used to verify the generalization ability of models on unseen scenarios. Notably, the submaps in the Oxford RobotCar dataset are created by concatenating consecutive 2D scans from SICK LMS151 2D LiDAR during the 20m. The spherical representation is thus not suitable for this type of data, as demonstrated in [Żywanowski et al., 2021]. The same settings are set in Minkloc3D, except that the LSA unit is added, and followed by each 3D convolution layer. The HCAT is removed because the

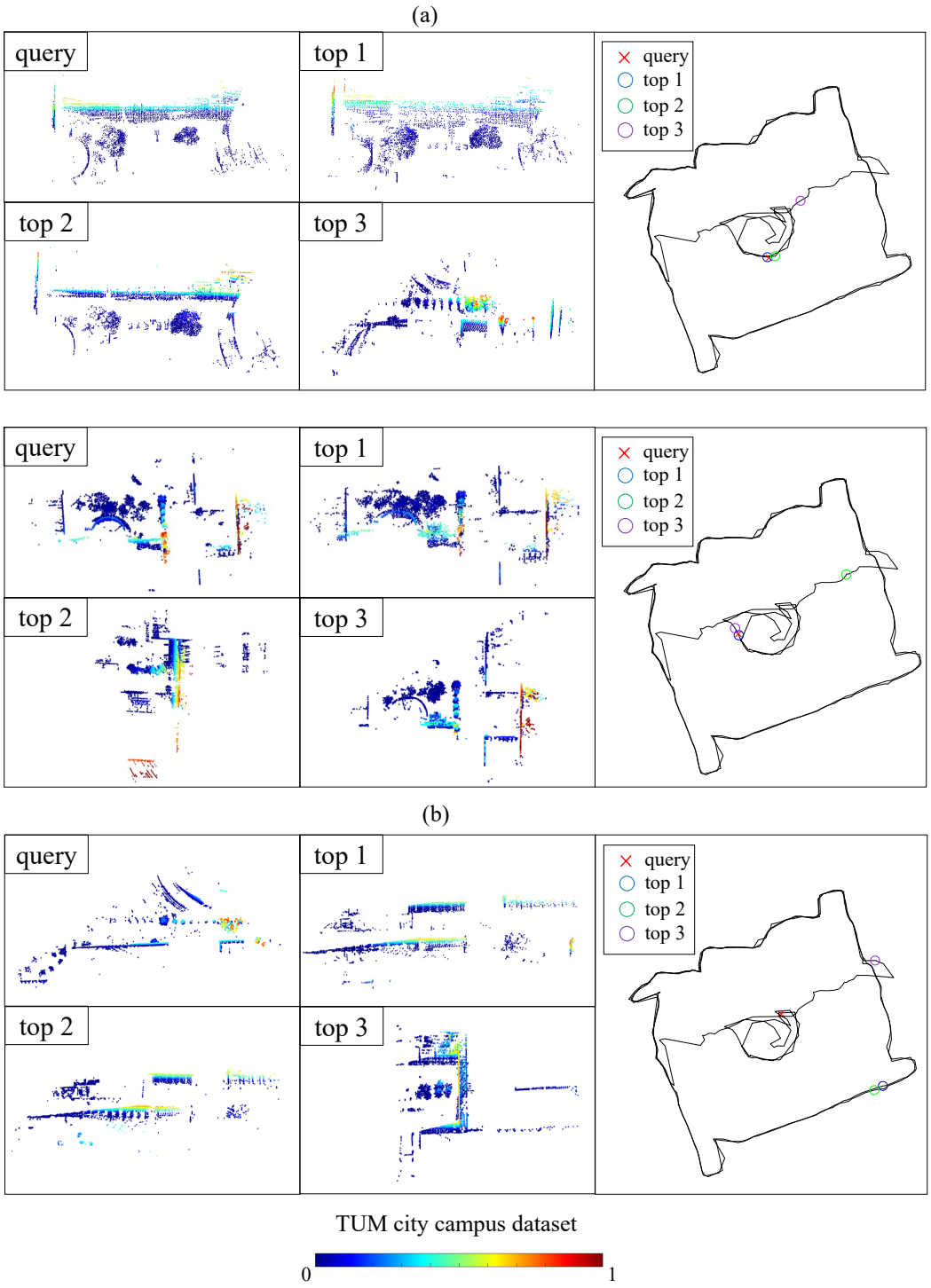


Figure 7.4: Visualizations of example retrieval results of CASSPR on the TUM dataset. a) shows the successful retrieval cases. b) show one failed case.

Table 7.5: Average recall (%) at top 1% (@1%) and top 1 (@1) for each of the models trained on the Oxford RobotCar, U.S. and R.A..

	Oxford		U.S.		R.A.		B.D.	
	AR @1	AR @1%	AR @1	AR @1%	AR @1	AR @1%	AR @1	AR @1%
PointNetVLAD [Angelina Uy & Hee Lee, 2018]	62.8	80.3	63.2	72.6	56.1	60.3	57.2	65.3
PCAN [Zhang & Xiao, 2019]	69.1	83.8	62.4	79.1	56.9	71.2	58.1	66.8
LPD-Net [Liu et al., 2019]	86.3	94.9	87.0	96.0	83.1	90.5	82.5	89.1
EPC-Net [Hui et al., 2022]	86.2	94.7	-	96.5	-	88.6	-	84.9
SOE-Net [Xia et al., 2021a]	89.4	96.4	82.5	93.2	82.9	91.5	83.3	88.5
MinkLoc3D [Komorowski, 2021]	93.0	97.9	86.7	95.0	80.4	91.2	81.5	88.5
NDT-Transformer [Zhou et al., 2021]	93.8	97.7	-	-	-	-	-	-
PPT-Net [Hui et al., 2021]	93.5	98.1	90.1	97.5	84.1	93.3	84.6	90.0
SVT-Net [Fan et al., 2022]	93.7	97.8	90.1	96.5	84.3	92.7	85.5	90.7
MinkLoc3D-S [Żywanowski et al., 2021]	92.8	81.7	83.1	67.7	72.6	57.1	70.4	62.2
CASSPR	**95.6**	**98.5**	**92.9**	**97.9**	**89.5**	**94.8**	**87.9**	**92.1**

quantization loss can be ignored when using extremely small voxel sizes, as demonstrated in [Xu et al., 2021].

CASSPR is compared with the state-of-the-art methods, including PointNetVLAD [Angelina Uy & Hee Lee, 2018], PCAN [Zhang & Xiao, 2019], LPD-Net [Liu et al., 2019], EPC-Net [Hui et al., 2022], SOE-Net [Xia et al., 2021a], Minkloc3D [Komorowski, 2021], NDT-Transformer [Zhou et al., 2021], PPT-Net [Hui et al., 2021], SVT-Net [Fan et al., 2022], and MinkLoc3D-S [Żywanowski et al., 2021]. The dimensions of all global descriptors are set to 256-dim. The evaluation results are shown in Table. 7.5. The results of previous methods are reported in their papers following the same evaluation protocol. It is clearly seen that CASSPR achieves state-of-the-art results on the Oxford RobotCar, with 1.7% improvements at recall @1 than the baseline method, MinkLoc3D. Compared with NDT-Transformer which is also based on a Transformer architecture, CASSPR introduces a fast and lightweight self-attention unit and achieves a remarkable improvement. Compared with PCAN and SOE-Net, the proposed CASSPR exceeds the recall of PCAN and SOE-Net by 25.6% and 5.5% at recall @1, respectively. This indicates the proposed LSA is more effective than the attention strategy used in PCAN and SOE-Net, effectively capturing the long-range spatial relationships. In addition, CASSPR still surpasses other methods significantly on all in-house datasets, although there exist huge differences in data distribution between Oxford Robotcar and in-house datasets. It demonstrates that the global descriptors generated by CASSPR have better generalization ability than all previously state-of-the-art methods.

7.2 Detection and tracking results

7.2.1 Vehicle detection

In this section, the proposed 3D vehicle detection method is trained on the KITTI 3D object detection benchmark and then tested on the TUM city campus dataset. TUM dataset is extremely small and only has point cloud data without image data, the 3D ground-truth bounding boxes of the vehicles are manually annotated in each LiDAR scan. Gabelsbergerstrasse is chosen as the test area due to the high complexity of the street environment and the large number of vehicles passing by.

Detection results

The annotated vehicles in the KITTI benchmark are classified into Simple, Moderate, and Hard difficulties based on their bounding box height in the image plane, occlusion level, and truncation level. Table 7.6 shows the definition of difficulties in the KITTI benchmark. As the TUM data has no image data, the difficulties cannot be distinguished based on the definition. All objects

are thus considered in the evaluation since they cannot be filtered based on their bounding box height in the image plane. The Average Precision (AP) is calculated in both 3D and BEV under the IoU thresholds of 0.7 and 0.5. 3D AP computes the 3D IoU of the predicted and ground-truth 3D bounding boxes. In BEV AP, the predicted and ground truth 3D bounding boxes are first projected to 2D bounding boxes, and then the 2D IoU is calculated.

Table 7.7 presents the detection results on the KITTI and TUM datasets. Two conclusions are easily obtained from the table. (1) For the KITTI dataset, the proposed detector achieves high performance under the IoU thresholds of 0.7 and 0.5, while it performs worse testing on the TUM dataset, only 25.94% and 54.15% on 3D AP and BEV AP respectively. This indicates the generalization ability of the introduced detector is not well on the TUM dataset. Besides, some visualization results are given in Fig. 7.5 and Fig. 7.6. (2) When the IoU threshold is set as 0.5, the performance on the TUM dataset is increased significantly compared with the IoU thresholds of 0.7, reaching 60.32% and 68.27% on 3D AP and BEV AP respectively. This demonstrates that the proposed detector can determine which point clouds are vehicles in the input scans, but the predicted locations and sizes of bounding boxes are not precise.

Table 7.6: The definition of difficulties in the KITTI benchmark.

Difficulty	Min. height	Max. Occlusion	Max. Truncation
Easy	40 Pixel	Fully visible	15%
Moderate	25 Pixel	Partly occluded	30%
Hard	25 Pixel	Difficult to see	50%

Table 7.7: Average precision for the proposed detector tested on the KITTI and TUM datasets. Notably, the results on the KITTI dataset are for Easy level.

IoU	KITTI		TUM	
	3D AP	BEV AP	3D AP	BEV AP
0.7	75.54	83.53	25.94	54.15
0.5	89.29	89.48	60.32	68.27

Results visualization

Fig. 7.5 and Fig. 7.6 show some selected cases of the proposed detector tested on the KITTI 3D detection benchmark and TUM city campus dataset, respectively. The green one is the predicted bounding box and the blue one is the ground truth. As seen from the figures, the LiDAR scan in the TUM dataset differs significantly from the scans in the KITTI dataset. The LiDAR point cloud in the TUM dataset is actually aggregated from two Velodyne scans at the same time, covering a much wider range compared with point clouds in the KITTI. Fig. 7.6(a) and Fig. 7.6(c) show the proposed detector can detect the targets accurately and tightly, indicating it has good generalization ability in some cases. In Fig. 7.6(b), the vehicle in the top right corner is not detected by the proposed detector. This scenario is never seen in the KITTI dataset, which covers two streets with a single scan. In addition, some points floating on the ground are incorrectly detected as vehicles, indicating the proposed detector is not very robust to noise.

Computational time analysis

In this section, the required computational time of the proposed detector is analyzed. Here, the proposed method is tested on all frames of the annotated TUM city campus dataset with a single Tesla T4 GPU (15G) and an Intel(R) Xeon(R) CPU @ 2.00GHz. The proposed detection framework achieves real-time efficiency with 38.38ms per scan. During the inference stage, the

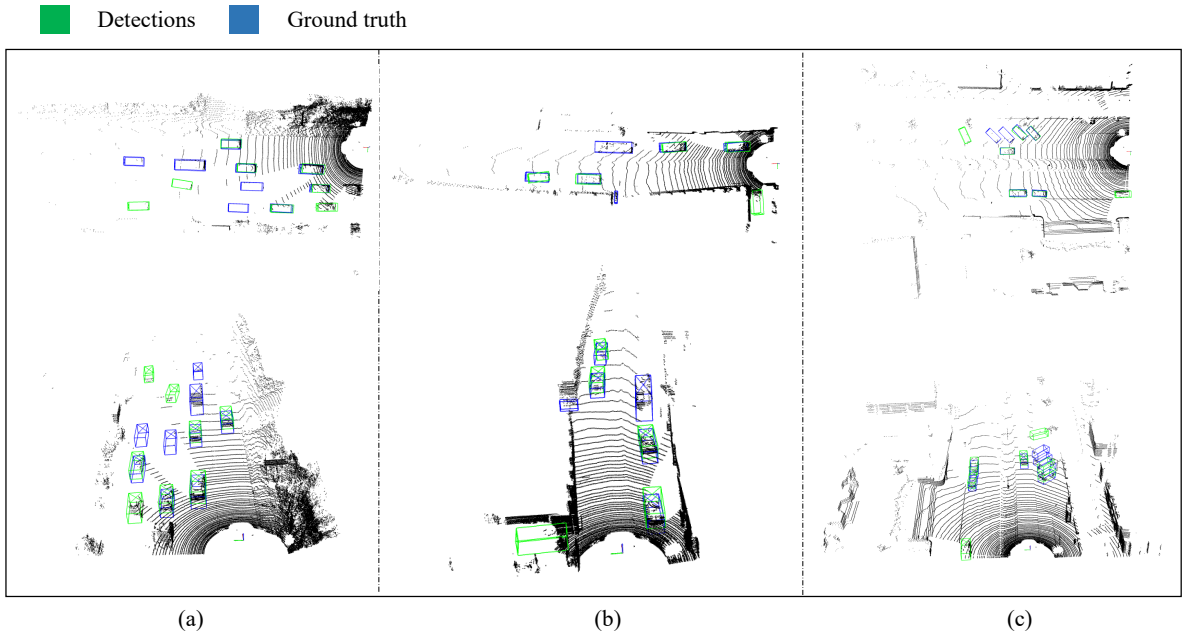


Figure 7.5: Visualizations of vehicle detection results on the KITTI 3D detection benchmark. The predicted bounding boxes and ground truth are shown in green and blue, respectively. (a)-(c) show results for test instances from different locations, with different views.

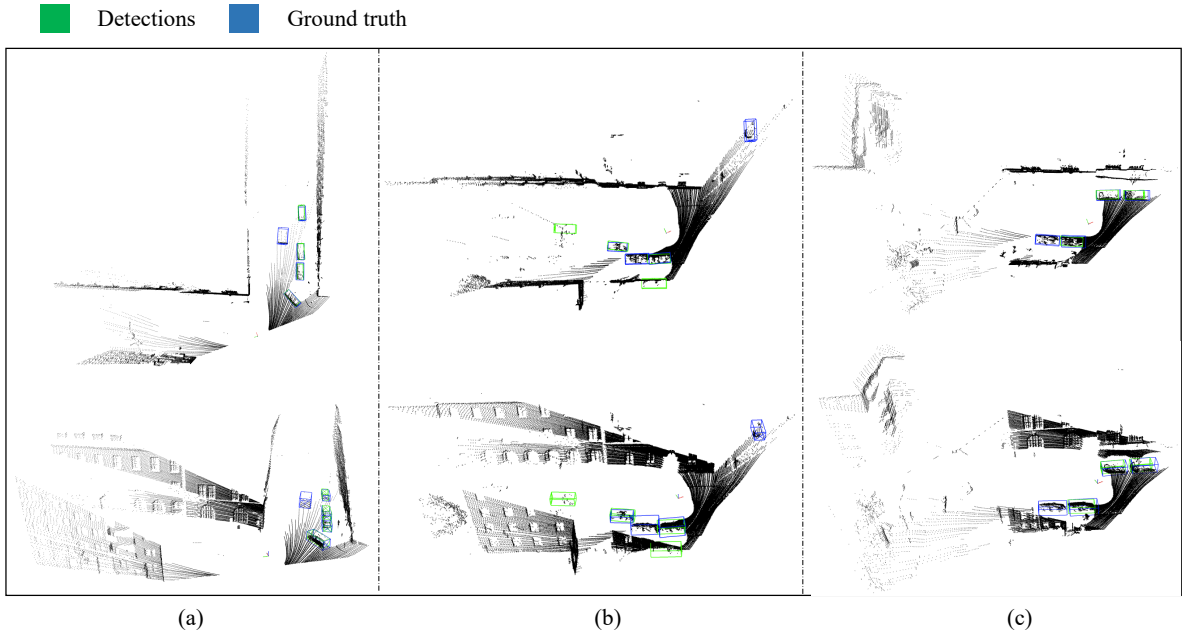


Figure 7.6: Visualizations of vehicle detection results on the TUM dataset. The predicted bounding boxes and ground truth are shown in green and blue, respectively. (a)-(c) show results for test instances from different locations, with different views.

single LiDAR scan is loaded and filtered based on the range first, which takes 12.17ms. Then, the points are organized into pillars, and the pillars are decorated, encoded, and scattered to the pseudo-image, costing 5.19ms in this step. Afterward, the pseudo image is passed through the 2D backbone, which takes 1.97ms. The following detection heads spend 0.065ms on predicting the 3D bounding boxes for vehicles. Finally, the post-processing NMS takes 20.77 ms to filter out the redundant detection on the CPU. A conclusion can be drawn that the data pre-processing and post-processing consume a large amount of time in the proposed detection framework.

7.2.2 DMT

Comparison with State-of-the-arts

The proposed DMT is compared with the state-of-the-art methods: SC3D [Achlioptas et al., 2018], its follow-up SC3D-RPN [Zarzar et al., 2019], FSiamese [Zou et al., 2020], 3DSiamRPN [Fang et al., 2020], P2B [Qi et al., 2020], MLVSNet [Wang et al., 2021], PTT [Shan et al., 2021], and BAT [Zheng et al., 2021a]. For a fair comparison, the same evaluation metrics are used. In this section, the default setting of the MPM is an LSTM prediction model. Fig. 7.7 and Table 7.8 show the success and precision of each network on the KITTI and NuScenes datasets. The success and precision values for other methods are those reported in their published papers [Achlioptas et al., 2018; Zarzar et al., 2019; Zou et al., 2020; Fang et al., 2020; Qi et al., 2020; Wang et al., 2021; Shan et al., 2021; Zheng et al., 2021a]. DMT is first quantitatively evaluated on KITTI, and then extend the comparisons to NuScenes.

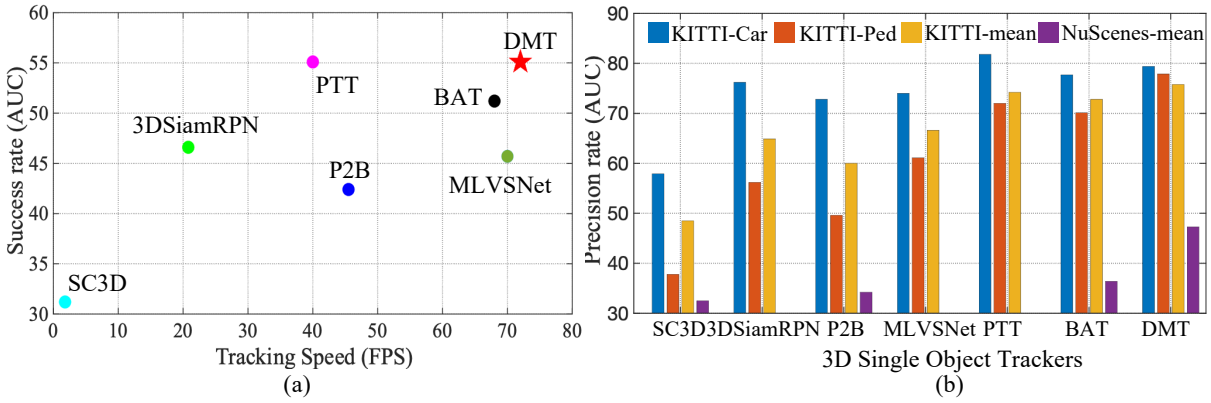


Figure 7.7: a) Tracking accuracy v.s. speed on the car category of KITTI benchmark. DMT outperforms state-of-the-art 3D single-object trackers in terms of both tracking accuracy and speed. b) Precision comparison on KITTI-Car, KITTI-Pedestrian, KITTI-mean, and NuScenes-mean.

Comparisons on KITTI. Following [Qi et al., 2020; Zheng et al., 2021a], the search area centered on the previous result in the inference stage is generated to meet the requirement of real scenarios. The results in Table 7.8 show that the proposed DMT outperforms other 3D trackers significantly. Specifically, when mixing all categories together to test the average performance following previous trackers, the average performance of DMT reaches 55.1, outperforming BAT by ∼4% on Success, indicating the effectiveness of the proposed DMT. When comparing PTT on rigid object (e.g., Van) tracking, DMT has a significant advantage (∼10%) over PTT on the less-frequent van category in terms of the success metric. However, DMT does not achieve the highest performance in the more-frequent car category. The transformer-based tracker PTT can learn better features of rigid objects since they have complex network architectures and more parameters but rely on more data to train the networks. Qualitative results are given in Section 7.2.2.

Table 7.8: Results of the Success and Precision of different 3D trackers with different categories on the KITTI and NuScenes dataset. 'Ped' represents 'Pedestrian'.

	Dataset	KITTI					NuScenes				
	Category	Car	Ped	Van	Cyclist	Mean	Car	Trunk	Trailer	Bus	Mean
	Frame Number	6424	6088	1248	308	14068	32302	8646	2297	2215	45460
Success (%)	SC3D [Giancola et al., 2019]	41.3	18.2	40.4	41.5	31.2	30.6	23.5	27.4	23.6	28.7
	SC3D-RPN [Zarzar et al., 2019]	36.3	17.9	-	43.2	-	-	-	-	-	-
	FSiamese [Zou et al., 2020]	37.1	16.2	-	47.0	-	-	-	-	-	-
	3DSiamRPN [Fang et al., 2020]	58.2	35.2	45.6	36.1	46.6	-	-	-	-	-
	P2B [Qi et al., 2020]	56.2	28.7	40.8	32.1	42.4	34.6	25.2	30.0	28.4	32.3
	MLVSNet [Wang et al., 2021]	56.0	34.1	52.0	34.3	45.7	-	-	-	-	-
	PTT [Shan et al., 2021]	**67.8**	44.9	43.6	37.2	**55.1**	-	-	-	-	-
	BAT [Zheng et al., 2021a]	60.5	42.1	52.4	33.7	51.2	36.8	28.6	31.8	30.2	34.7
	DMT (Proposed)	66.4	**48.1**	**53.3**	**70.4**	**55.1**	**43.8**	**51.3**	**46.8**	**38.2**	**44.0**
Precision (%)	SC3D [Giancola et al., 2019]	57.9	37.8	47.0	70.4	48.5	35.9	24.8	24.8	21.8	32.5
	SC3D-RPN [Zarzar et al., 2019]	51.0	47.8	-	81.2	-	-	-	-	-	-
	FSiamese [Zou et al., 2020]	50.6	32.2	-	77.2	-	-	-	-	-	-
	3DSiamRPN [Fang et al., 2020]	76.2	56.2	52.8	49.0	64.9	-	-	-	-	
	P2B [Qi et al., 2020]	72.8	49.6	48.4	44.7	60.0	37.6	25.2	26.7	27.6	34.2
	MLVSNet [Wang et al., 2021]	74.0	61.1	61.4	44.5	66.6	-	-	-	-	-
	PTT [Shan et al., 2021]	**81.8**	72.0	52.5	47.3	74.2	-	-	-	-	-
	BAT [Zheng et al., 2021a]	77.7	70.1	**67.0**	45.4	72.8	39.5	28.4	30.5	29.5	36.4
	DMT (Proposed)	79.4	**77.9**	65.6	**92.6**	**75.8**	**48.3**	**51.1**	**40.3**	**31.9**	**47.3**

To demonstrate the generalizability for non-rigid object tracking, DMT is compared with other trackers on Pedestrian and Cyclist. For Pedestrian, DMT outperforms BAT and PTT by ∼8% and ∼6% on Precision respectively, indicating the effectiveness of the proposed tracking pipeline. Amazingly, DMT outperforms BAT and PTT by a large margin for the cyclist category, achieving about ∼47%/∼45% improvement on Precision. This phenomenon can be explained as follows: 1) the amount of training and testing samples are extremely small; 2) DMT is more robust to interference with non-rigid objects in the search area; 3) DMT is simple yet effective, thus relying on fewer data to learn better networks. The visualized results are shown in Fig. 7.8. This also demonstrates that DMT can achieve better performance, especially when having fewer data compared with BAT.

Comparisons on NuScenes. For the car category, DMT reaches the best performance of 43.8/48.3 on Success/Precision, exceeding the performance of current state-of-the-art method BAT [Zheng et al., 2021a] by ∼7%/∼9% respectively. Notably, for the truck and trailer categories, DMT achieves about ∼23% and 20% improvements over BAT on Precision, which demonstrates that the motion-guided pipeline is more effective, especially on the more challenging dataset. Moreover, for the bus category, which has the fewest training samples, DMT still outperforms BAT by a large margin of 8% in terms of the Success metric. Compared with the baseline method BAT, the performance of DMT shows significant improvements (∼10% on average) in terms of all categories. Note that PTT/MLVSNet did not present results on NuScenes in their papers.

Computational cost analysis

In this section, the required computational resources of different 3D trackers are analyzed in terms of the number of parameters, floating point operations (FLOPs), and running speed. For a fair comparison, here, DMT is tested on all frames of the KITTI-Car with a single NVIDIA RTX3090 GPU. As shown in Table 8.7 and Fig. 7.7 (Left), DMT uses less time per frame with fewer FLOPs compared with other trackers. Notably, despite the fact that DMT includes an LSTM model, the number of parameters of DMT are the same as P2B, while DMT is significantly faster (57% improvement in frame per second) and simpler (36% improvement in FLOPs) by using the same RTX3090 GPU. In addition, the running speed of MLVSNet is close to DMT. However, DMT

is lighter (i.e., with fewer model parameters) and can achieve much better performance on the KITTI dataset (see Table 7.8), demonstrating that DMT is simple yet effective.

Table 7.9: Computational cost requirements of different 3D single object trackers on KITTI-Car. * indicates the frame per second is reported from the corresponding paper.

Method	Modality	Params	FLOPs	Frame Per Second	Platform
SC3D [Giancola et al., 2019]	LiDAR	-	-	1.8*	1080Ti
FSiamese [Zou et al., 2020]	LiDAR+RGB	-	-	4.9*	1080Ti
3DSiamRPN [Fang et al., 2020]	LiDAR	-	-	20.8*	1080Ti
P2B [Qi et al., 2020]	LiDAR	**5.4M**	4.65G	45.5	3090
MLVSNet [Wang et al., 2021]	LiDAR	7.6M	-	70.0*	1080Ti
PTT [Shan et al., 2021]	LiDAR	-	-	45	3090
BAT [Zheng et al., 2021a]	LiDAR	5.9M	3.05G	68.0	3090
DMT (Proposed)	LiDAR	**5.4M**	**2.98G**	**71.5**	3090

Results visualization on the KITTI dataset

According to the different categories and difficulties of the targets, some advantageous cases of DMT on the KITTI dataset are selected and visualized in Fig. 7.8. Four frames sorted by time from a full sequence are selected from the cyclist and car categories, respectively. For the cyclist target, the point clouds of the target and the tracked results are shown at the top of Fig. 7.8. In this example, BAT tracks the cyclist wrongly when there are two similar cyclists in the surrounding area. DMT can track the target accurately and tightly, indicating DMT is more robust to complex scenarios. Furthermore, the tracked results on the car category are displayed, which is shown at the bottom of Fig. 7.8. Here BAT fails in the extremely sparse scenes (fewer than 10 points), but DMT works well, which shows that DMT is indeed robust to point sparsity.

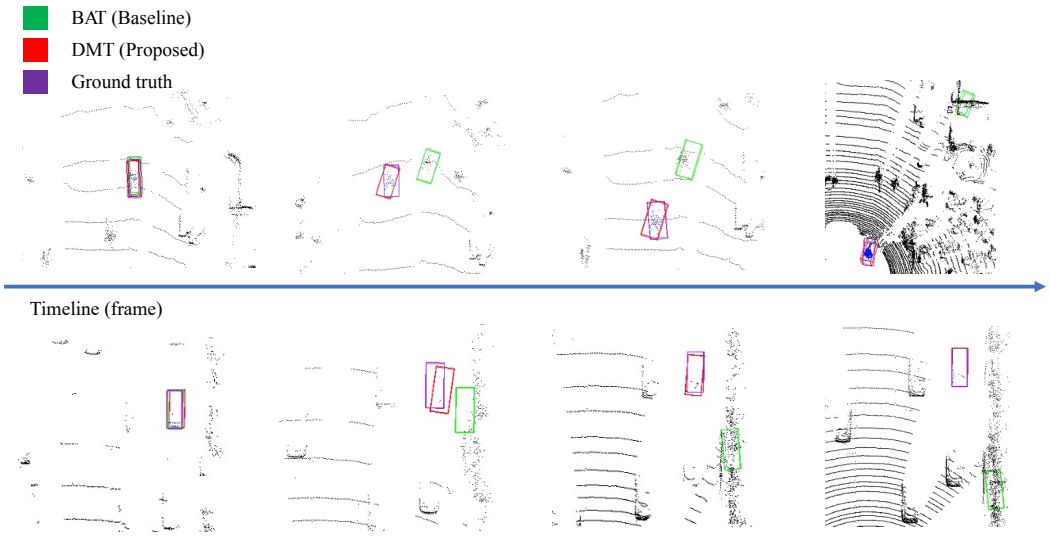


Figure 7.8: Visualizations of example results of DMT compared with BAT on the KITTI dataset. The point clouds of tracked objects are shown in blue. (Top) shows results for test instances from the cyclist category. There are two cyclists in a neighboring area, and DMT can maintain the correct track while BAT drifts to the wrong object. (Bottom) shows results for test instances from the car category. Although the point clouds are extremely sparse (< 10 points), DMT still tracks the object.

Table 7.10: Vehicle tracking results compared with BAT on the TUM dataset.

	Precision	Success
BAT [Zheng et al., 2021a]	58.6	48.3
DMT (Proposed)	**64.6**	**55.8**

Application on the TUM dataset

In this section, the proposed DMT is applied to track the vehicles on the TUM dataset. Notably, DMT is only trained on the KITTI dataset. Same with the experiments of the proposed detector, the Gabelsbergerstraße is selected as the test area. Table 7.10 shows the vehicle tracking results of DMT compared with BAT on the TUM dataset. From Table 7.10, DMT achieves 64.6% and 55.8% on Precision and Success, outperforming BAT by 8% and 7.5% respectively. This demonstrates DMT has a better generalization ability when applying an unseen dataset. To vividly present the superiority of DMT, the tracking examples are visualized in Fig 7.9. Three frames sorted by time from each full sequence are selected. In these examples, DMT can track the vehicle accurately from the beginning frame to the end while BAT tracks the vehicle more incorrectly with time going on. The examples demonstrate DMT is more robust to unknown scenarios, even though the test dataset is extremely different from the training dataset.

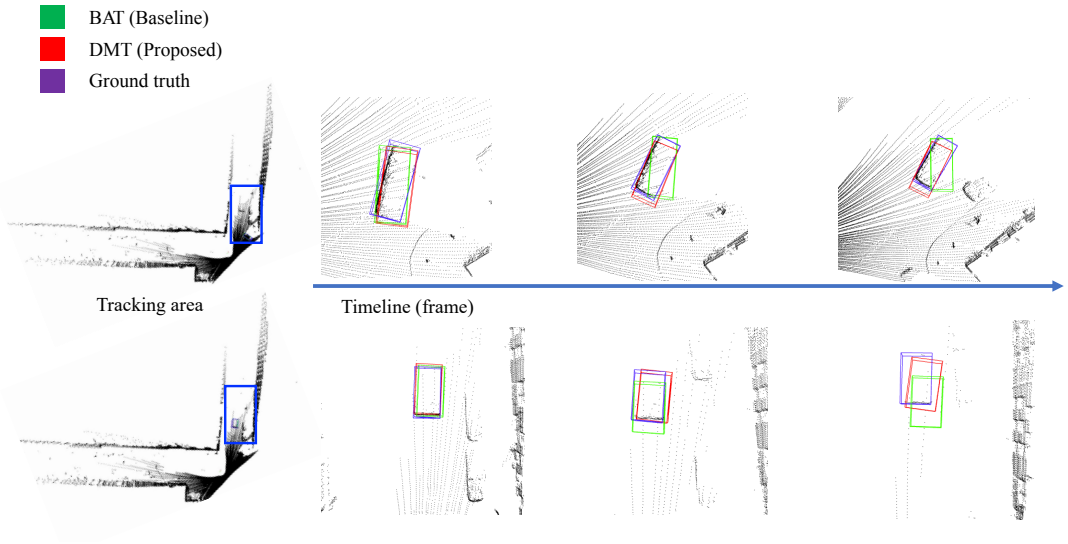


Figure 7.9: Visualizations of vehicle tracking results of DMT compared with BAT on the TUM dataset. The BAT and DMT are trained on the KITTI dataset and then tested on the TUM dataset. Although the point clouds are different from the KITTI dataset, DMT still tracks the vehicle well.

7.3 Shape completion results

7.3.1 VPC-Net

Point completion on the ShapeNet dataset

For evaluating the performance of VPC-Net in completing point clouds of synthetic models, VPC-Net is compared against the following state-of-the-art methods on the ShapeNet testing data, including 3D-EPN [Dai et al., 2017b], PCN [Yuan et al., 2018], and TopNet [Tchapmi et al., 2019]. 3D-EPN [Dai et al., 2017b] is a typical volumetric completion method, which was trained on the large-scale synthetic dataset as well. PCN [Yuan et al., 2018] is a pioneering method that completes partial inputs using point clouds directly, which conducted end-to-end training through

an auto-encoder. TopNet [Tchapmi et al., 2019] is the newest end-to-end point cloud completion method. For a fair comparison, all methods were trained and tested on the same data for all experiments. The size of the output point cloud and the ground truth was fixed to 16,384 points. Quantitative results are shown in Table 7.11, respectively.

Table 7.11: Quantitative comparison (smaller value represents better performance) of VPC-Net against the state-of-the-art methods on ShapeNet.

Methods	Mean Chamfer Distance per point (10^{-3})	Mean Earth Mover's Distance per point (10^{-2})
3D-EPN [Dai et al., 2017b]	22.308	10.7080
PCN [Yuan et al., 2018]	11.668	6.0480
TopNet [Tchapmi et al., 2019]	13.765	9.6840
VPC-Net (Proposed)	**8.662**	**5.1677**

Table 7.11 shows that VPC-Net outperforms other methods significantly. In this table, the value of CD and EMD metrics are scaled by 1000 and 100, respectively. A relative improvement is obtained on the average CD value by 25.7% and the average EMD value by 14.6% over the second-best approach PCN. Note that the values of EMD are much higher than those of CD. The reason is that EMD is a one-to-one distance matching metric, whereas CD can have one-to-many correspondences between points.

Point completion on the KITTI dataset

For evaluating the performance of VPC-Net on real scan LiDAR data, VPC-Net is tested for point cloud completion on the KITTI dataset. 2483 partial point clouds of cars are extracted from every frame based on their bounding boxes. Each extracted point cloud is transformed into the bounding box's coordinate system and then completed by VPC-Net trained on the ShapeNet dataset. Lastly, they are turned back to the world coordinates. Considering the extra noisy points from the ground or nearby objects within the bounding box of the car, the FPS operation is removed in the refiner since it would bring this noise into the final completed results. Note that there are no ground truth point clouds in this dataset.

The qualitative results are shown in Fig. 7.10. The single frame raw data and five detected vehicles from the testing data are visualized in Fig. 7.10a. Fig. 7.10b shows five sparse and partial input point clouds, while Figs. 7.10c and 7.10d display the completed point clouds by PCN and VPC-Net, respectively. From Fig. 7.10, VPC-Net has a better generalization capability and has complete shapes that show that the point sets are evenly distributed on the vehicle surface. Note that both networks were trained on the same ShapeNet training set and tested on KITTI. For example, for Example 2 in Fig. 7.10, the result generated by VPC-Net includes the details of missing parts, and all points are more evenly distributed on the geometric surface, while point sets completed by PCN are messy and lose detailed structures of the rear of the car. Many points from PCN escaped the car surface, which can be observed in Example 3, Example 4, and Example 5.

Based on the obtained outputs and comparisons, it can be concluded from Fig. 7.10 that VPC-Net is robust to different resolutions of input point clouds, which is an essential characteristic for handling real scan data. For example, the point clouds of Examples 1 and 3 have 12 and 100 points, respectively, while 903 points are included in the case of Example 5. In spite of this, VPC-Net is able to produce uniformly, dense, and complete point clouds with finely detailed structures.

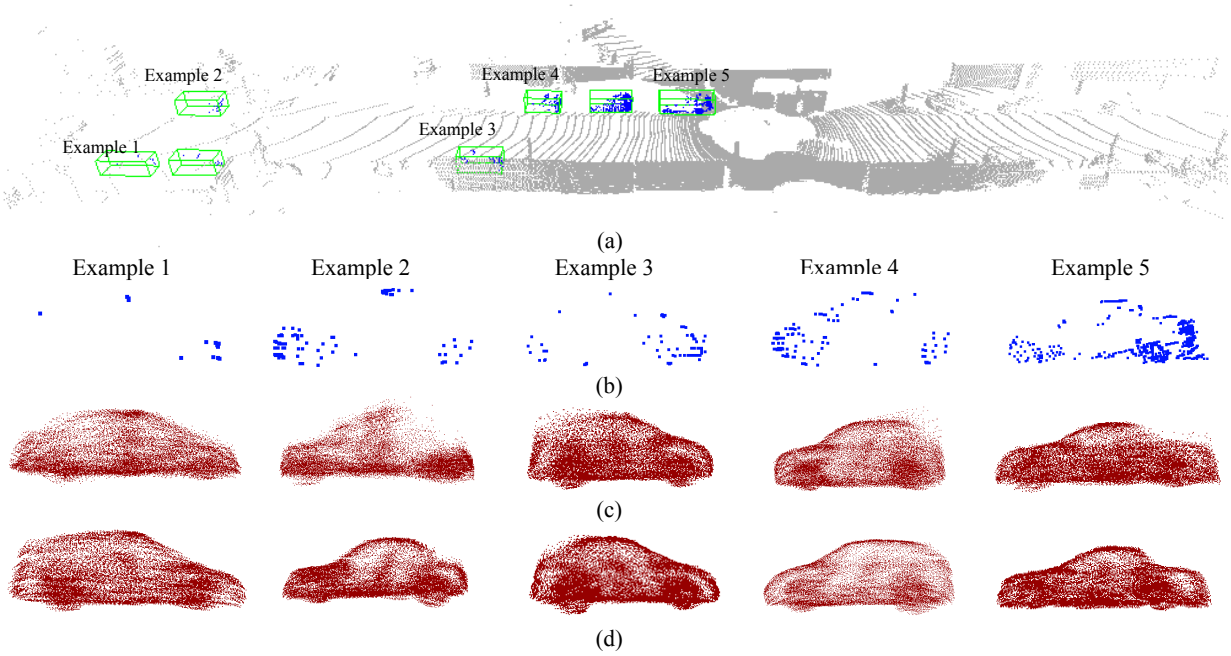


Figure 7.10: Completed 3D point clouds using real-scan data from the KITTI dataset. a) shows five detected vehicle examples in a single frame. b) shows the input partial point clouds. c) shows the completed point clouds by PCN and d) shows the completed point clouds by VPC-Net.

Point completion on the TUM dataset

To further illustrate VPC-Net's effectiveness and generalization ability on real scan data, the TUM dataset is selected as a test set. There are no complete point clouds as ground truth for the TUM dataset either. The qualitative results of some vehicle instances are selected and shown in Fig. 7.11.

Unlike point clouds from the KITTI dataset, point clouds from the TUM dataset are very dense. These partial point clouds contain 4200 points on average here. In spite of this, VPC-Net can still generate detailed information not only in partial inputs but also for the missing structures. For example, in the fourth row of Fig. 7.11, the point cloud completed by VPC-Net preserves the shape of the input and reconstructs the wheels and other missing parts. This verifies that VPC-Net can transfer easily between the different distributions without any fine-tuning operations, whether partial point clouds are from the KITTI dataset, the TUM dataset, or the ShapeNet dataset.

7.3.2 ASFM-Net

Evaluation on the PCN dataset

PCN dataset [Yuan et al., 2018] is created from the ShapeNet dataset [Chang et al., 2015], containing pairs of complete and partial point clouds. Notably, each complete model includes 16384 points and is corresponding to eight partial point clouds. The dataset covers 30974 CAD models from 8 categories: airplane, cabinet, car, chair, lamp, sofa, table, and watercraft. Following [Yuan et al., 2018], the number of models for validation and testing are 100 and 150, respectively. The remaining models are used for training. The complete shapes are uniformly downsampled from 16384 points to 4096. The performance is evaluated on the resolution containing 4096 points.

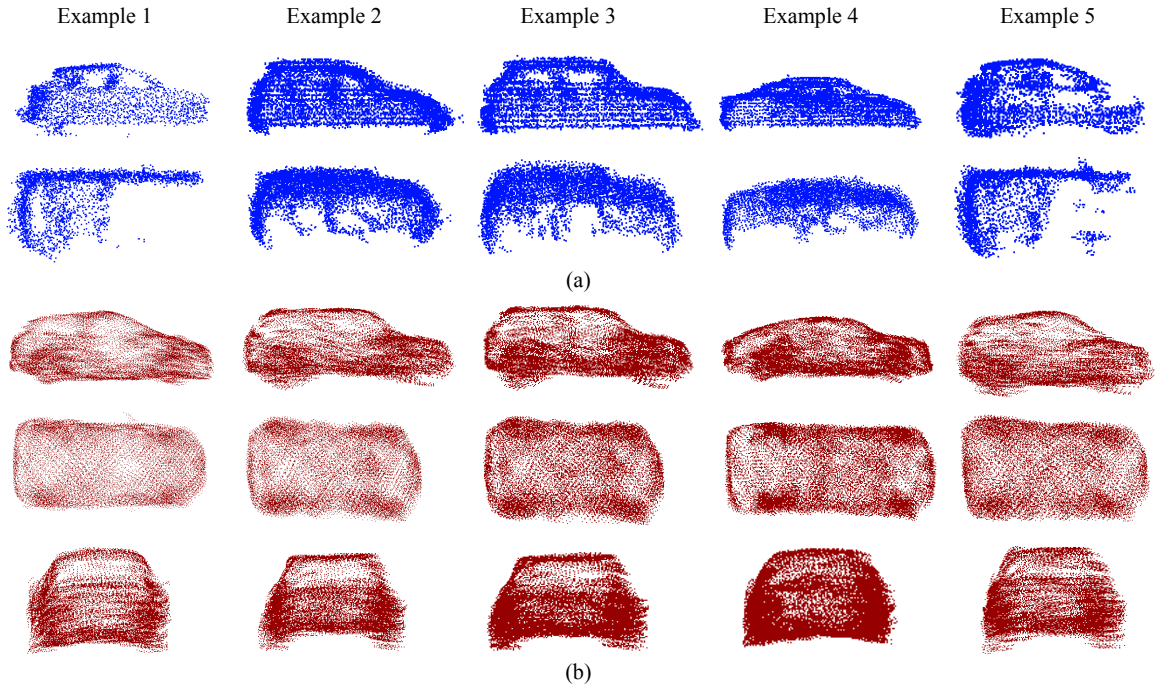


Figure 7.11: Completed 3D point clouds using real-scan data from the TUM dataset. a) shows five vehicle examples of partial point clouds seen from side view and top view. b) shows the completed point clouds displayed from different viewpoints: side view, top view, and rear view.

Table 7.12: Quantitative comparison on known categories on the PCN dataset. Point resolutions for the output and ground truth are 4096. For CD-P, lower is better.

Methods	Chamfer Distance(10^{-3})								
	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft	Average
TopNet [Tchapmi et al., 2019]	8.21	15.99	13.28	15.80	15.29	17.49	13.27	13.81	14.14
PCN [Yuan et al., 2018]	7.95	15.59	13.10	15.47	15.31	16.78	13.22	13.37	13.85
RFA [Zhang et al., 2020b]	7.49	15.68	13.52	14.00	12.33	16.50	11.99	11.40	12.87
ASFM-Net (Proposed)	**6.75**	**14.85**	**12.51**	**13.17**	**11.66**	**15.38**	**11.49**	**10.96**	**12.09**

ASFM-Net is compared against several state-of-the-art point cloud completion methods such as PCN [Yuan et al., 2018], TopNet [Tchapmi et al., 2019] and RFA [Zhang et al., 2020b] qualitatively and quantitatively. Table 7.12 shows that ASFM-Net achieves the lowest CD values in all eight categories, which demonstrates the superior performance in this dataset. Especially, ASFM-Net improves the average CD values by 6.1% compared to the second-best method RFA. Besides, the visualization results are shown in the column labeled 4096 in Fig. 7.12. ASFM-Net can not only predict the missing part of the object but also preserve the details of the input point cloud. For example, in the fifth row of Fig. 7.12, TopNet and PCN totally failed. They cannot complete the missing areas, and even destroy the original shape of the lamp. RFA attempts to repair the lamp but fails to output a satisfactory result. In contrast, the completed point cloud by the proposed ASFM-Net preserves the detailed structure from the input and reconstructs the missing lamp cover successfully.

Evaluation on the Completion3D benchmark

Completion3D benchmark* is released by TopNet [Tchapmi et al., 2019], which is a subset of the ShapeNet dataset derived from the PCN dataset. Different from the PCN dataset, the resolution

Table 7.13: Quantitative comparison on known categories on the Completion3D benchmark. Point resolutions for the output and ground truth are 2048. For CD-T, lower is better.

Methods	Chamfer Distance(10^{-4})								
	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft	Average
FoldingNet [Yang et al., 2018]	12.83	23.01	14.88	25.69	21.79	21.31	20.71	11.51	19.07
PCN [Yuan et al., 2018]	9.79	22.70	12.43	25.14	22.72	20.26	20.27	11.73	18.22
PointSetVoting [Zhang et al., 2021]	6.88	21.18	15.78	22.54	18.78	28.39	19.96	11.16	18.18
AtlasNet [Groueix et al., 2018]	10.36	23.40	13.40	24.16	20.24	20.82	17.52	11.62	17.77
RFA [Zhang et al., 2020b]	6.52	26.60	10.83	27.86	23.21	23.58	11.66	7.41	17.34
TopNet [Tchapmi et al., 2019]	7.32	18.77	12.88	19.82	14.60	16.29	14.89	8.82	14.25
SoftPoolNet [Wang et al., 2020d]	4.89	18.86	10.17	15.22	12.34	14.87	11.84	6.48	11.90
SA-Net [Wen et al., 2020]	5.27	14.45	7.78	13.67	13.53	14.22	11.75	8.84	11.22
GR-Net [Xie et al., 2020b]	6.13	16.90	8.27	12.23	10.22	14.93	10.08	5.86	10.64
CRN [Wang et al., 2020a]	3.38	13.17	8.31	10.62	10.00	12.86	9.16	5.80	9.21
SCRN [Wang et al., 2020b]	3.35	12.81	7.78	9.88	10.12	12.95	9.77	6.10	9.13
VRC-Net [Pan et al., 2021]	3.94	10.93	6.44	9.32	8.32	11.35	8.60	5.78	8.12
ASFM-Net (Proposed)	**2.38**	**9.68**	**5.84**	**7.47**	**7.11**	**9.65**	**6.25**	**4.84**	**6.68**

of both partial and complete point clouds is 2048 points. Moreover, each complete model is only corresponding to one partial point cloud. The train/test split is the same as the PCN dataset.

In the column labeled 2048 in Fig. 7.12, the results of the visual comparison between ASFM-Net and other approaches are presented, from which the more reasonable ability to infer the missing parts and the more effective fidelity of ASFM-Net. The quantitive comparison results of ASFM-Net with the other state-of-the-art point cloud completion approaches are shown in Table 7.13. It is apparent from this table that the proposed ASFM-Net achieves the best performance concerning chamfer distance averaged across all categories. Compared with the second-best method VRC-Net [Pan et al., 2021], ASFM-Net improves the performance of averaged chamfer distance with a margin of 17.7%.

Car completion on the KITTI dataset

Following PCN, the proposed ASFM-Net is evaluated for car completion on the KITTI [Geiger et al., 2013] dataset. The KITTI dataset includes many real-scanned partial cars collected by a Velodyne 3D laser scanner. In this experiment, one sequence of raw scans is taken. It contains 2483 partial car point clouds, which are collected from 98 real cars under 425 different frames. Same as PCN, every point cloud is completed with a model trained on cars from ShapeNet, and then transformed back to the world frame. Notably, there are no ground truth point clouds in the KITTI dataset.

Table 7.14: Fidelity distance (FD) and consistency comparison on the KITTI dataset.

Methods	Input	TopNet	PCN	RFA	ASFM-Net
FD	-	0.041	0.041	0.072	**0.025**
Consistency	0.052	**0.014**	0.016	0.033	0.020

Therefore, fidelity distance (FD), consistency, and minimal matching distance (MMD) are proposed by PCN as evaluation metrics. MMD is to measure how much the completion output resembles a typical car. However, this metric is not meaningful due to only being trained in the car category. In other words, the prior information is already known that this object must be a car. Thus, MMD is ignored as an efficient evaluation metric in this paper. The fidelity is to measure the similarity between the input and completed point clouds, which calculates the average distance

*https://completion3d.stanford.edu/.

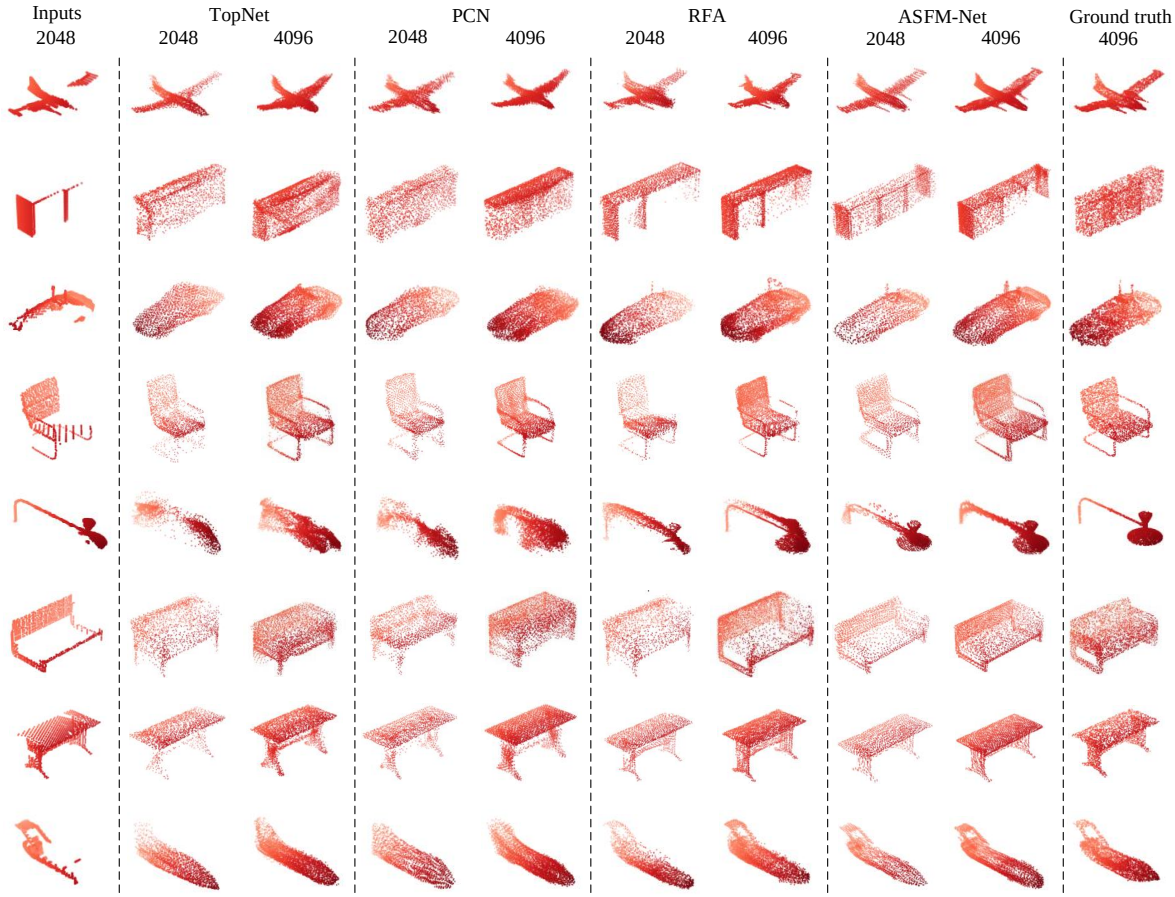


Figure 7.12: Qualitative comparison on known categories on the Completion3D benchmark and the PCN dataset. The 2048 and 4096 resolutions of the outputs are completed from the Completion3D benchmark and the PCN dataset, respectively.

between each point in the input and the nearest point in the completed point cloud. Consistency is to measure how consistent the outputs reconstructed by networks are against variations in the inputs, which calculates the average Charmfer distance among the completion results of the same instance in consecutive frames. The quantitative results are shown in Table 7.14. From Table 7.14, it can be seen that ASFM-Net achieves excellent fidelity performance. The value is far below the other three methods, which illustrates ASFM-Net can preserve the original shape of the input effectively. The consistency performance of ASFM-Net is slightly weaker than TopNet and PCN. However, ASFM-Net improves the performance by a large margin compared with raw inputs. To explore the consistency performance more clearly, the results completed by different methods for one specific vehicle in different frames are visualized in Fig. 7.13. The raw point clouds of the same vehicle have significantly different shape appearances since they are collected at different moments from a moving vehicle. This is the reason that the value of consistency for inputs is high. From Fig. 7.13, PCN and TopNet are good at generating a general shape of the car while ASFM-Net reconstructs plausible shapes and keeps fine-grained details from the input. It demonstrates that ASFM-Net is more flexible when the appearance of the input point cloud changes greatly.

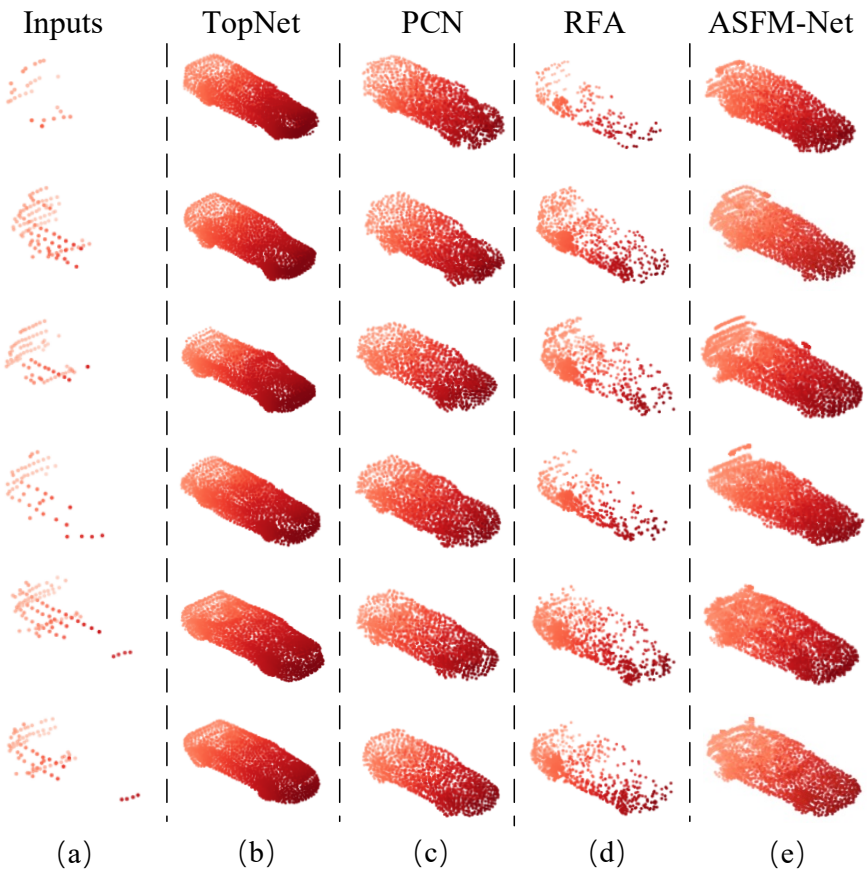


Figure 7.13: Qualitative comparison on the KITTI dataset. From a) to e): Raw point clouds of the same vehicle scanned in consecutive frames, shape completion using TopNet, PCN, RFA, and ASFM-Net, respectively.

8 Discussion

This chapter covers the discussions of the various algorithms and methods presented in Chapters 3-5. Based on extensive experimental results, the ablation studies, the advantages, limitations, and applications of those algorithms and methods are thoroughly discussed.

8.1 Discussion on place recognition

8.1.1 SOE-Net

Ablation study

Ablation studies evaluate the effectiveness of different proposed components in SOE-Net, including both the PointOE module and the self-attention unit. The performance of the proposed HPHN quadruplet loss is also analyzed. All experiments are conducted on Oxford RobotCar.

PointOE module and self-attention unit. The effectiveness of the proposed PointOE module and the self-attention unit is tested, using PointNetVLAD and PCAN as baselines, PN_VLAD, PCAN. Either the PointOE module or self-attention unit is integrated into PointNetVLAD, referred as PN_VLAD-OE and PN_VLAD-S. Both two components are then combined into PointNetVLAD, denoted as PN_VLAD-SOE. Besides, PointNet is replaced by PointNet++ [Qi et al., 2017b] in the local descriptor extraction stage, referred to as PN++_VLAD. All networks are trained with lazy quadruplet loss, with results shown in Table 8.1.

Table 8.1: Ablation studies of self-attetnion unit and PointOE module on Oxford RobotCar. The results show the average recall (%) at top 1% (@1%) and at top 1 (@1) for each model.

	Ave recall @1%	Ave recall @1
PN_VLAD	81.01	62.76
PN++_VLAD	89.10	76.23
PCAN	83.81	69.05
PN_VALD-S	86.71	73.03
PN_VALD-OE	92.20	82.21
PN_VALD-SOE	93.41	84.20

Comparing with PointNetVLAD, PN_VLAD-S sees an improvement of 5.7% and 10.27% on the average recall at top 1% and top 1, respectively. The performance of PN_VLAD-S also exceeds the recall of PCAN by 2.9% and 3.98%, respectively, indicating the proposed self-attention unit is more effective than the attention strategy used in PCAN. This is due to the context information has a significant effect on aggregating local descriptors into a global one, and the proposed self-attention unit can learn long-range spatial relationships between local descriptors. With the proposed PointOE module, SOE-Net brings significant improvements on the average recall by 11.19% and 19.45%, respectively, when compared with PointNetVLAD. Besides, PointNet++ enhances PointNet features with a hierarchical encoding pipeline, but still does not explicitly encode orientation. The comparison with PN++_VLAD demonstrates the superiority of OE for

3D descriptor learning for place recognition. Combining both modules can improve the performance by 12.40% and 21.44% on average recall, respectively. The ablation studies demonstrate the significant role of each module in SOE-Net.

HPHN quadruplet loss. To evaluate the proposed HPHN quadruplet loss, the performance of the proposed SOE-Net trained with different losses is compared. As shown in Table 8.2, the network performance is better when trained on the proposed HPHN quadruplet loss. The performance on Oxford RobotCar reaches 96.40% recall at top 1% and 89.47% recall at top 1, exceeding the same model trained with the lazy quadruplet loss by 2.99% and 5.17%, respectively, demonstrating the superiority of the proposed HPHN quadruplet loss.

Table 8.2: Results of the average recall (%) at top 1% and at top 1 of SOE-Net trained with different losses on Oxford RobotCar.

	Ave recall @1%	Ave recall @1
Lazy quadruplet	93.41	84.20
HPHN quadruplet	**96.40**	**89.37**

Output dimension analysis

In this section, the performance of the global descriptor with different output dimensions is analyzed. The results of average recall at top 1% for the global descriptor produced by SOE-Net and DAGC are shown in Table 8.3. Two conclusions can be drawn from this table: (1) SOE-Net outperforms DAGC, even if the generated global descriptor has a smaller dimension; (2) when the output dimension decreases from 256 to 128, the performance of SOE-Net only declines by around 1%-3% on each benchmark. When the dimension expands to 512, the performance only changes by about 0.3%-1%.

Table 8.3: Results of the average recall (%) at top 1% of different global descriptor dimensions on Oxford RobotCar. D is the output dimension of global descriptors.

	SOE-Net			DAGC		
	D=128	D=256	D=512	D=128	D=256	D=512
Oxford	95.30	96.40	**96.70**	84.43	**87.49**	85.72
U.S.	91.24	93.17	**94.47**	81.17	**83.49**	83.02
R.A.	90.53	**91.47**	91.00	72.39	**75.68**	74.46
B.D.	85.88	88.45	**89.29**	69.57	**71.21**	68.74

Values of margin analysis

In this section, the network performance with different margins in the HPHN quadruplet loss is explored using Oxford RobotCar. Table 8.4 shows results of average recall at top 1% and top 1 with different margins for the SOE-Net architecture. Seen from the table, SOE-Net achieves the best performance with a margin value of 0.5. When the values expand to 0.7, the performance steadily degrades. This implies the distance between positive and negative pairs is sufficient with lower values of margin. On the other hand, when the value is set to 0.4, the performance decreases. The fixed value of margin is thus set as 0.5 in SOE-Net.

Table 8.4: Margin analysis in the HPHN quadruplet loss. SOE-Net is chosen as a baseline and evaluate it on Oxford RobotCar.

Margin	Ave recall @1%	Ave recall @1
0.4	95.87	88.84
0.5	**96.40**	**89.37**
0.6	96.23	89.30
0.7	95.63	88.46

8.1.2 CASSPR

Ablation study

To assess the relative contribution of each module, the LSA and the HCAT (including the point branch) are removed from CASSPR one by one, denoted as CASSPR_HCAT and CASSPR_LSA, respectively. The key/value and query for the fusion unit of HCAT is also swtiched, namely the point branch acting as a query, and the voxel branch as a key and value, denoted as CASSPR_Switch. All networks are trained on the TUM dataset, with results shown in Table 8.5. CASSPR_HCAT outperforms MinkLoc3D-S in recall by 8.6%, indicating the proposed HCAT is a crucial part of a successful fusion strategy. The LSA unit brings significant improvements on the average recall metric (7.9%), when compared against MinkLoc3D-S performance. Combining both modules achieves the best performance, improving the performance by 11.4% and 16.5%. It is also observed inferior performance of CASSPR_Switch to the proposed method when changing the sequence of key/value and query in the second stage.

Table 8.5: Ablations of the hierarchical cross-attention transformer (HCAT) and lightweight self-attention (LSA) on the TUM dataset.

Method	MHCAT	LSA	Ave recall @1%	Ave recall @1
MinkLoc3D [Komorowski, 2021]			82.7	66.9
MinkLoc3D-S [Żywanowski et al., 2021]			85.7	69.1
CASSPR_MHCAT	✓		94.3	77.7
CASSPR_LSA		✓	93.6	84.9
CASSPR_Switch	✓	✓	89.2	77.7
CASSPR	✓	✓	**97.1**	**85.6**

Number of lightweight self-attention units

In this section, the network performance with different numbers of lightweight self-attention (LSA) units is also explored on the Oxford RobotCar and in-house datasets. Specifically, the HCAT module is removed and the LSA units is inserted one by one after the each convolutional layer. The network is denoted as CASSPR_LSA. '0' means that no LSA unit is added, thus the network structure is exactly the same as MinkLoc3d.

Table 8.6 shows results of average recall at top 1% and top 1 with different numbers of LSA units for the CASSPR_LSA architecture. Seen from the table, three conclusions can be drawn: (1) For Oxford RobotCar, it is observed that CASSPR_LSA outperforms MinkLoc3d by 0.1% and 0.5% on the average recall at top 1% and top 1 respectively when only adding one unit, indicating the effectiveness of the LSA unit. (2) CASSPR_LSA achieves the best performance with 6 LSA units. This implies the network has the best global awareness when all attention units are used. (3) Comparing the performance when the number of units is 5 and 6, the network achieves the same performance on Oxford RobotCar, while it has the better performance tested on in-house datasets when the number is 6. This demonstrates the LSA unit benefits the generalization ability.

Table 8.6: Average recall (%) at top 1% (@1%) and top 1 (@1) for CASSPR with different number of LSA units trained on Oxford RobotCar, U.S. and R.A..

Number of LSA	Oxford RobotCar		U.S.		R.A.		B.D.	
	AR @1	AR @1%	AR @1	AR @1%	AR @1	AR @1%	AR @1	AR @1%
0	93.0	97.9	86.7	95.0	80.4	91.2	81.5	88.5
1	93.5	98.0	86.9	95.0	80.0	88.8	80.9	87.3
2	93.5	98.0	87.2	96.0	80.2	89.3	80.8	87.4
3	92.9	97.6	90.1	96.5	81.0	91.1	82.6	89.2
4	92.9	97.6	87.2	95.6	82.7	92.5	82.5	89.1
5	**94.7**	**98.4**	88.0	94.4	86.2	93.4	82.7	89.0
6	**94.7**	**98.4**	**91.4**	**97.1**	**86.8**	**93.5**	**86.3**	**91.0**

The maximum range of 3D LiDAR

In this section, the performance of the global descriptor with different maximum ranges of points from LiDAR is analyzed. CASSPR is used as a baseline and the experiments are conducted on the USyd Campus dataset. The maximum range of points is set from 20m to 100m at 20m intervals since the Velodyne VLP-16 sensor utilized in the USyd dataset has a range of about 100m. Fig. 8.1 presents the CASSPR results obtained for varying maximum ranges of points.

As seen from the figure, the best results are obtained when the maximum measurement range is set to at least 40m. Three conclusions can be drawn:(1) Compared the performance from 20m to 100m, the average recall at top 1% only ranges from 93.9% to 98.4%, demonstrating CASSPR is robust to the different maximum ranges of points. (2) The location of the 3D points is sparse at larger maximum ranges, making it difficult to identify meaningful location identification features. (3) When the smallest maximum range is 20m, the performance at top 1 drops to 77.7%. The result implies the limited range contains less information and is more likely to contain highly similar scenarios.

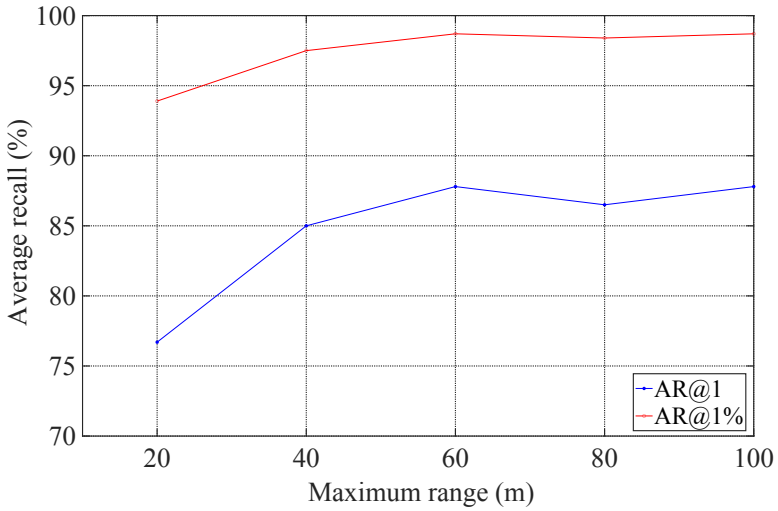


Figure 8.1: Average recall of CASSPR tested on USyd with different maximum distance of points from the 3D LiDAR position. Maximum ranges less than 40m show a drop in performance.

Computational cost analysis

In this section, the required computational resources of different global descriptors in terms of the number of parameters and time efficiency are analyzed. For a fair comparison, here, all

Table 8.7: Computational cost requirements of different 3D global descriptors on the TUM dataset.

Methods	Parameters (M)	Time Usage (ms)			
		total	Point branch	HCAT	LSA
PointNetVLAD [Angelina Uy & Hee Lee, 2018]	19.8	12.6	-	-	-
PCAN [Zhang & Xiao, 2019]	20.4	66.8	-	-	-
SOE-Net [Xia et al., 2021a]	19.4	66.9	-	-	-
MinkLoc3D-S [Żywanowski et al., 2021]	1.1	6.0	-	-	-
CASSPR (Proposed)	3.8	29.7	5.0	12.7	1.4

scans of the TUM dataset are tested with a single NVIDIA V100 GPU. As shown in Table 8.7, CASSPR takes 29.7ms to encode the one scan into a global descriptor with only 3.8M parameters. Although CASSPR is a point-voxel fusion architecture, CASSPR has lower trainable parameters compared with the point-based methods, including PointNetVLAD, PCAN and SOE-Net. For inference time, CASSPR is faster than the attention-based methods PCAN and SOE-Net in terms of running time per frame (29.7ms vs. 66.8ms). However, compared with the voxel-based method MinkLoc3D-S, CASSPR costs more parameters and running times since CASSPR additionally includes a point branch, a HCAT module, and a LSA unit. In addition, the running time of each module in CASSPR is presented, including the point branch, HCAT, and LSA. The LSA is light and fast, only spends 1.4ms. The most time-consuming module is HCAT since it still uses a conventional dot-production attention layer.

8.2 Discussion on detection and tracking

8.2.1 Ablation studies

In this section, the effectiveness of important modules in DMT is analyzed, including both the motion prediction module (MPM) and the explicit voting module (EVM).

Table 8.8: Ablation studies of motion prediction module (MPM) and explicit voting module (EVM) on KITTI-Car.

Method	MPM	EVM	Success	Precision
BAT[Zheng et al., 2021a]			60.5	77.7
DMT_MP	✓		-	37.0
DMT_EV		✓	54.0	64.1
DMT	✓	✓	**66.4**	**79.4**

An ablation study is first conducted on the necessity of the EVM and MPM. All studies are conducted on KITTI-Car. The EVM and the MPM is removed one by one, denoted as DMT_MP and DMT_EV. Both variations have the same structure as DMT except for the removed module. The baseline model is BAT.

The results are shown in Table 8.8. Four conclusions are obtained from these results. (1) The potential target center estimated by the MPM is extremely inaccurate, only reaching 37% on Precision. Note that the MPM in DMT cannot regress the orientation of the target, and thus the Success value cannot be computed. (2) The precision without EVM is 37% (DMT_MP), and with EVM is 79.4% (DMT), which demonstrates that EVM can estimate an effective target-specific point feature to further refine the prediction of MPM. (3) Comparing DMT_EV with BAT, the performance of DMT_EV degrades about 6%/13% in terms of Success/Precision. This is consistent with the expectation that a simpler explicit voting module is used, removing the complicated RPN module. (4) The full pipeline achieves the best performance, which demonstrates the two

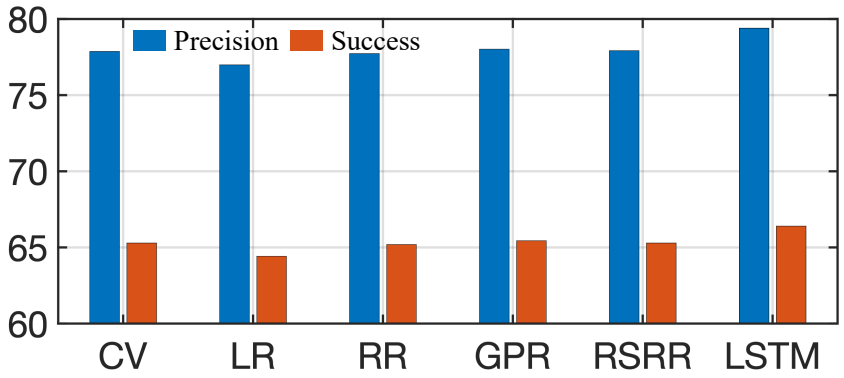


Figure 8.2: Comparison of using various regression or prediction models as the motion prediction module on KITTI-Car.

modules are mutually beneficial and necessary. In addition, even if the MPM provides inaccurate results, DMT achieves satisfactory performance due to the explicit voting module.

8.2.2 The choice of motion prediction module

In Fig. 8.2, various types of motion prediction models are compared on KITTI-Car. The compared models include Constant Velocity (CV), Linear Regression (LR), Ridge Regression (RR), Gaussian Process Regression (GPR), RANSAC with Ridge Regression (RSRR), and LSTM models. The LR, RR, GPR, and RSRR models are trained in the same way as the LSTM model, i.e., using the same sampled tracklets from the training data in KITTI-Car for offline training. These models are then applied for motion prediction during online testing without further updating. In Fig. 8.2, the differences among various models are not significant, which implies that DMT is not sensitive to the MPM selection. This is because EVM is trained to predict GT bounding boxes from diverse sampled locations in the training stage, which makes it more robust to noisy predicted target center locations. The sequence-to-sequence prediction LSTM model achieves the best Precision (79.4%) and Success (66.4%) due to its better ability of sequence modelling.

8.2.3 Template generation strategy

The performance of DMT with four template generation strategies is explored following [Zheng et al., 2021a], including "the first ground-truth", "the previous result", "the first ground-truth and previous result", and "all previous results". Table 8.9 shows results of Success/Precision with different settings for different trackers on KITTI-Car. Note that P2B, BAT, and DMT use the same PointNet++ backbone. The specific design in DMT enables it to achieve better tracking performance than the other trackers under different template generation settings. Specifically, DMT achieves the best performance when using the "all previous" strategy, outperforming BAT and P2B by large margins (∼8%, ∼12% respectively). Another finding is P2B, BAT, and DMT all report degraded results under the "all previous" setting since these trackers did not train the networks using all previous results for efficiency, while SC3D did. Despite this, the superior overall performance of DMT in Table 8.9 suggests that DMT better utilizes motion cues from all previous predictions compared with BAT.

8.2.4 Sampling distance for training EVM

In this section, the network performance with different sampling distances (i.e., the distances between the sampled points and the ground-truth center) is explored in the training of EVM.

Table 8.9: Different strategies for template generation. 3D trackers are evaluated on KITTI-Car.

	Method	The First GT	Previous result	First & Previous	All Previous
Success	SC3D[Giancola et al., 2019]	31.6	25.7	34.9	41.3
	P2B [Qi et al., 2020]	46.7	53.1	56.2	51.4
	BAT [Zheng et al., 2021a]	51.8	59.2	60.5	55.8
	DMT (Proposed)	**54.3**	**63.8**	**66.4**	**63.5**
Precision	SC3D[Giancola et al., 2019]	44.4	35.1	49.8	57.9
	P2B [Qi et al., 2020]	59.7	68.9	72.8	66.8
	BAT [Zheng et al., 2021a]	65.5	75.6	77.7	71.4
	DMT (Proposed)	**67.2**	**76.7**	**79.4**	**75.9**

As mentioned in Section 4.3.3, the distance should not be too large to maintain stable training. An ablation experiment on KITTI-Car is conducted choosing the distance values from 0.65 to 0.95. As shown in Table 8.10, the performance of DMT reaches its peak with a distance value of 0.75. When the distance expands to 0.95, the performance steadily degrades. This implies the distances between sampled points and the ground-truth center are still a little large so that some outliers are picked. On the other hand, the network performance drops when the distance is set to 0.65. Thus, the value is fixed at 0.75 for the best performance.

Table 8.10: Sampling distance analysis for DMT. DMT is evaluated on KITTI-Car.

Distance(m)	Success(%)	Precision(%)
0.65	64.0	77.0
0.75	**66.4**	**79.4**
0.85	63.0	77.5
0.95	63.0	76.8

8.2.5 Number of sampled training points

In the practical implementation, various points around the ground-truth target center are sampled to mimic motion predictions during the online tracking process. In this section, how the number of sampled points affects the final tracking performance is studied. Specifically, the number of sampled points is varied and reported the corresponding performance in Table 8.11 on KITTI-Car. Sampling dense points (i.e., 64) leads to better performance because dense sampling provides more comprehensive cases for learning a more robust EVM. It is also noticed that the performance is not saturated, implying that better performance can be obtained by sampling a larger number of points. However, in the current experiments, DMT is limited by the GPU memory size.

Table 8.11: Sampling point number analysis for DMT. DMT is evaluated on KITTI-Car.

Number	Success(%)	Precision(%)
8	61.1	75.0
16	62.2	75.7
32	64.5	78.0
64	**66.4**	**79.4**

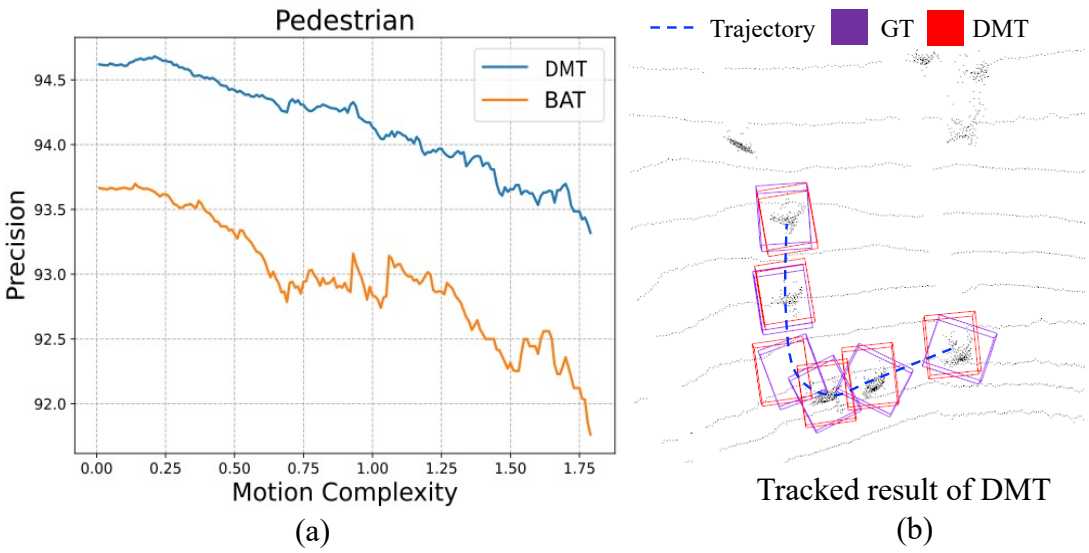


Figure 8.3: (a) Comparison of BAT and DMT under various motion complexity on KITTI-Pedestrian. (b) Example results of DMT for complex motion patterns.

8.2.6 Robustness test for object motion patterns

To better demonstrate the effectiveness of DMT on complex motion patterns, Fig. 8.3(a) shows the comparison of DMT and BAT on tracklets with different motion complexity. Here, motion complexity is defined as the average error of a simple constant velocity model. DMT still performs better than the RPN-based 3D tracker BAT when the motion complexity increases, which demonstrates the robustness of DMT to complicated motion patterns. The reason is that the diverse points are randomly sampled when training the EVM, which makes DMT more effectively handle various motion patterns. To further demonstrate the superiority clearly, one tracklet of a pedestrian having a complex trajectory is visualized in Fig. 8.3(b). DMT can track the target accurately despite the complicated motion pattern.

8.3 Discussion on 3D shape completion

8.3.1 VPC-Net

Visualization of completion details

To better gain further insights into the details of completion performance, the residual distance between corresponding points from the outputs of VPC-Net to the ground truth is visualized in Fig. 8.4. The 10 different vehicles are from ShapeNet test data. This figure provides detailed information about which vehicle parts were completed correctly. Different colors encode the normalized distance between the corresponding shapes. Fig. 8.4 clearly shows that the output point clouds completed by the proposed VPC-Net recovered most of the vehicle parts correctly. In addition, by observing the red area in Examples 1, 3, 5, and 6, it can be seen that VPC-Net cannot capture the fine-grained details in terms of the roof of vehicles. However, from the perspective of human perception, it can be tolerated since humans tend to judge an object's quality by its global features and will tolerate small inaccuracies in shape or location [Tatarchenko et al., 2019].

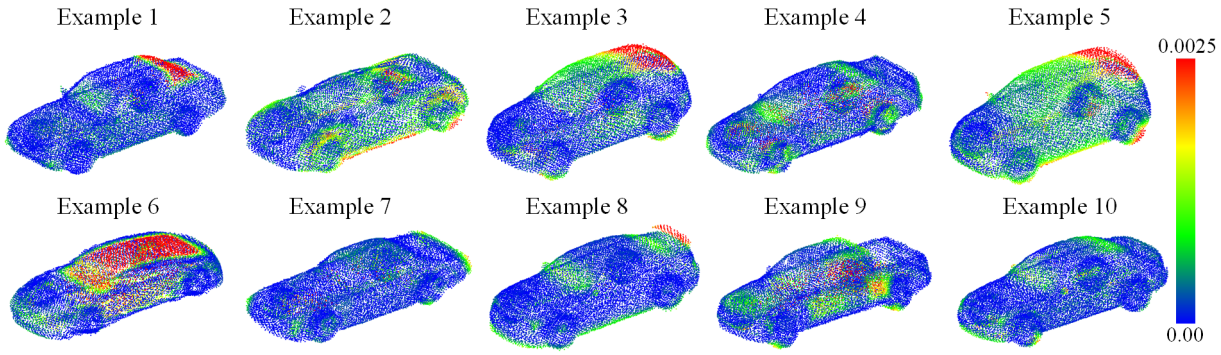


Figure 8.4: Visualizing point distances between the completed point clouds with ground truth point clouds.

Table 8.12: Performance comparison of the proposed VPC-Net with different components. The mean Chamfer Distance (CD) and Earth's Mover Distance (EMD) per point are reported, multiplied by 10^3 and 10^2, respectively.

STN	PFE	Refiner	CD	EMD
			11.668	6.0480
✓			8.922	5.1947
✓	✓		8.916	5.1777
✓	✓	✓	**8.662**	**5.1677**

Ablation study

The ablation studies evaluated the effectiveness of the different proposed components in VPC-Net, including the spatial transform network (STN), the point feature enhancement operation (PFE), and the refiner. Four models are developed: (1) a model without STN, PFE, or the refiner, (2) a model with STN only, (3) a model with both STN and PFE, and (4) a model with STN, PFE, and the refiner. CD and EMD are used as the evaluation metric, and the quantitative results of these models are shown in Table 8.12. All experiments were conducted on the ShapeNet dataset, and the resolution of the points was 16,384. It is clear that the full pipeline has the best performance. As shown in Table 8.12, with the proposed STN module, the proposed model achieves an improvement of 23.5% and 14.1% on CD and EMD, respectively. This is because the rigid geometric transformation has a significant effect on extracting features from partial inputs, while STN can learn invariance to translation and rotation. With the proposed PFE module, VPC-Net improves (0.1 %, 0.3 %) the CD and EMD. This confirms that enhancing the global feature is essential to generate a more accurate coarse point cloud. The proposed refiner module can further improve the performance by 3 % and 0.2 % in terms of the CD and EMD. The improvement in the CD is especially significant. This is because the refiner actually improves the fine-grained details of the completed point clouds, and the CD is better for measuring the fine-detailed structure of objects than the EMD. As pointed out in [Fan et al., 2017], the CD will produce points outside the main body at the correct locations. The EMD roughly captures the mean shape and is considerably distorted, which means it will ignore some flying but correct points. The ablation studies demonstrate that each proposed module plays a significant role in VPC-Net for performance improvements. Removing any modules will decline the performance, which proves that each proposed module contributes.

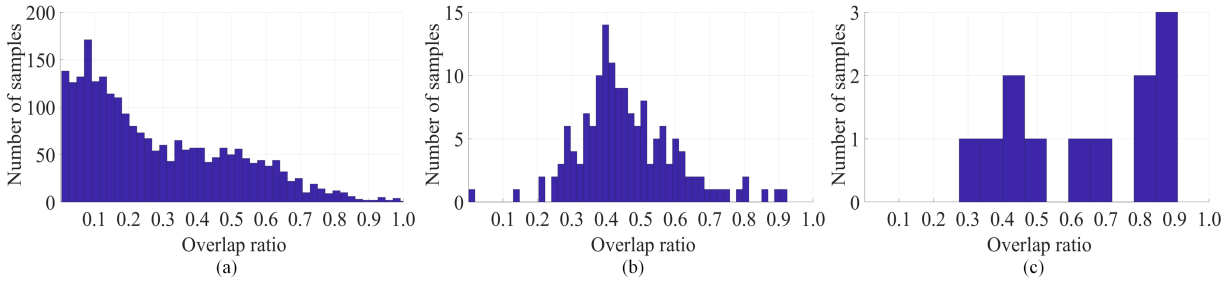


Figure 8.5: Completeness on the tested datasets. Overlap ratio between input point clouds and completed point clouds in a) KITTI dataset, b) ShapeNet dataset, and c) TUM dataset.

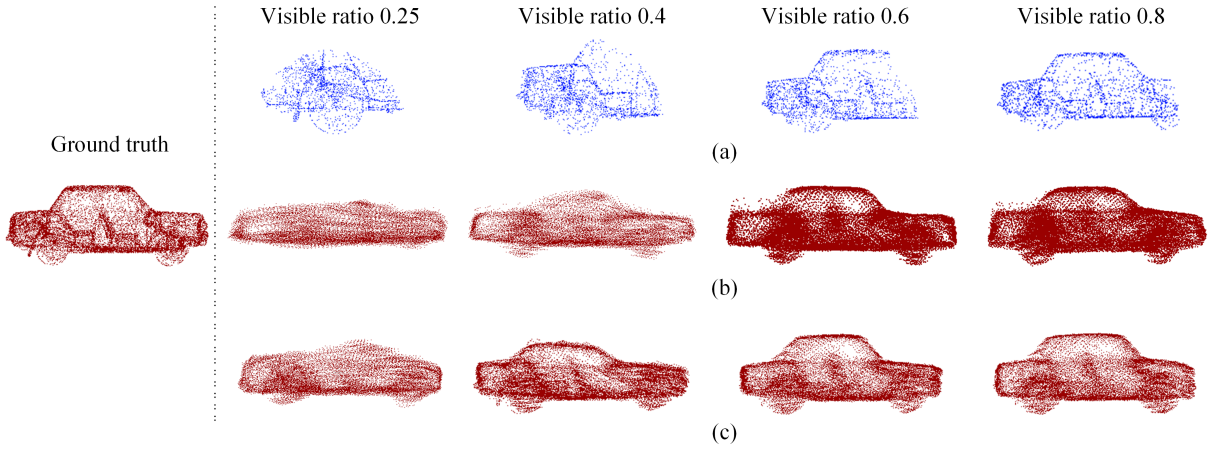


Figure 8.6: Qualitative results on the inputs with different amounts of missing content. a) shows partial point clouds with different levels of visibility. b) and c) show the completed point clouds by PCN and VPC-Net, respectively.

Robustness test

The experiments are carried out to evaluate the robustness of VPC-Net for input point clouds with various degrees of missing information. First, the completeness statistics of the test data from the ShapeNet dataset, the KITTI dataset, and the TUM-MLS-2016 dataset were collected, as shown in Fig. 8.5. The overlap ratio R_o between the input partial point clouds and the completed point clouds is referred to as the completeness metric, which is defined by

$$R_o = S_p / S_c \tag{8.1}$$

where S_p and S_c are surface areas of input partial point clouds and completed point clouds, respectively.

As can be seen in Fig. 8.5, most input instances from the KITTI dataset are very sparse, and completeness is less than 50%. In contrast, the examples from the TUM dataset have enough completeness since that dataset provides the aggregated point clouds, not the original scan data. The completeness of the test data from the ShapeNet dataset is a normal distribution. Based on the experimental results, VPC-Net can handle these inputs with different completeness.

To better illustrate the robustness, the robustness test experiment was performed on the ShapeNet test data since there are ground truth point clouds. The incompleteness degree d of

Table 8.13: Quantitative results on inputs with different amounts of missing content. The CD is reported by PCN and VPC-Net, multiplied by 10^3.

Visible Ratio	25%	40%	60%	80%
PCN [Yuan et al., 2018]	21.555	13.979	12.002	11. 884
VPC-Net	**14.786**	**12.377**	**7.926**	**7.612**

input point clouds is changed, where d ranges from 20% to 75%. The qualitative and quantitative results are shown in Fig. 8.6 and Table 8.13, respectively. The visible ratios 0.25, 0.4, 0.6, and 0.8 mean that four incomplete inputs lack 75%, 60%, 40%, and 20% of the ground truth data, respectively. As illustrated in Fig. 8.6 and Table 8.13, three conclusions can be drawn: (1) VPC-Net is more robust than PCN when dealing with a high degree of incompleteness. For example, when the visible ratio is 0.25, VPC-Net is able to generate the general shape of the car, but PCN fails. (2) When more regions are missing, CD and EMD errors slowly increase. This implies VPC-Net is still robust when meeting inputs with different incompleteness degrees. (3) The outputs completed by both methods are plausible when dealing with incomplete inputs with a large percentage of missing information. For example, the car generated by the proposed VPC-Net is a cabriolet, while the ground truth is a non-convertible car. However, this ambiguity is a common issue [Fan et al., 2017], because even for humans, it is difficult to know what this car is like based on just one wheel.

Registration Test

An even density and completeness are key factors for a successful registration between two point clouds [Xu et al., 2019]. Correspondingly, the registration result can also reflect the quality (e.g., the evenness of point density or the completeness of points) of the input point clouds [Yuan et al., 2018]. Here, similar to the test conducted in the work of the baseline method PCN [Yuan et al., 2018], the registration experiments between pairs of vehicle point clouds are conducted. Comparing the registration accuracy using incomplete and complete point clouds demonstrates the feasibility of the proposed vehicle point cloud completion method. The vehicle point clouds of adjacent frames in the same Velodyne sequence from the KITTI dataset were chosen as test data. Two types of inputs in the registration method are adopted: one represents the partial point clouds from the real-scan data, while the other represents the completed point clouds by the proposed VPC-Net. Moreover, a simple point-to-point Iterative Closest Point (ICP) [Besl & McKay, 1992] was applied as a registration algorithm, which minimizes distances iteratively between points from two point clouds. Notably, the ICP algorithm is not the only choice for registration tasks. Any registration algorithm that can be applied to illustrate the completed results has a good and consistent shape for the same vehicle instances in different frames. The average rotational and translational error in the registration results with partial and complete input point clouds were compared. The rotational error E_R and translational error E_T are defined as follows, respectively:

$$E_R = 2\cos^{-1}(2 < R_1, R_2 >^2 -1) \tag{8.2}$$

$$E_T = ||T_1 - T_2||_2 \tag{8.3}$$

where R_1 and T_1 are the rotation and translation of the ground truth in the KITTI dataset, respectively. R_2 and T_2 are the rotation and translation measured by the ICP method, respectively.

As shown in Table 8.14, the quantitative results demonstrate that the complete point clouds generated by VPC-Net provide a more accurate estimation of translation and rotation than the in-

Table 8.14: Averaged rotation and translation errors of point cloud registration using different inputs.

Inputs	Average error	
	rotation (°)	translation (m)
Partial inputs	13.9422	7.0653
Complete inputs	**7.9599**	**4.2059**

Table 8.15: Quantitative comparison of point cloud registration task with different inputs.

	Partial inputs		Complete outputs	
Example	Rotation error	Translation error	Rotation error	Translation error
1	4.5159	1.4715	**1.8219**	**0.5904**
2	11.4627	2.1093	**0.5678**	**0.1060**
3	4.5159	1.4715	**1.8219**	**0.5904**
4	143.9396	58.0907	**1.5606**	**0.7201**
5	178.6335	54.5161	**3.1471**	**1.5235**
6	14.8757	7.8894	**2.4544**	**1.2499**
7	3.1952	1.8321	**2.4083**	**1.3489**
8	1.7482	0.6973	**0.9957**	**0.2084**
9	**0.0270**	**1.3128**	5.5954	3.0927
10	**0.6646**	**1.3941**	4.1969	3.8149

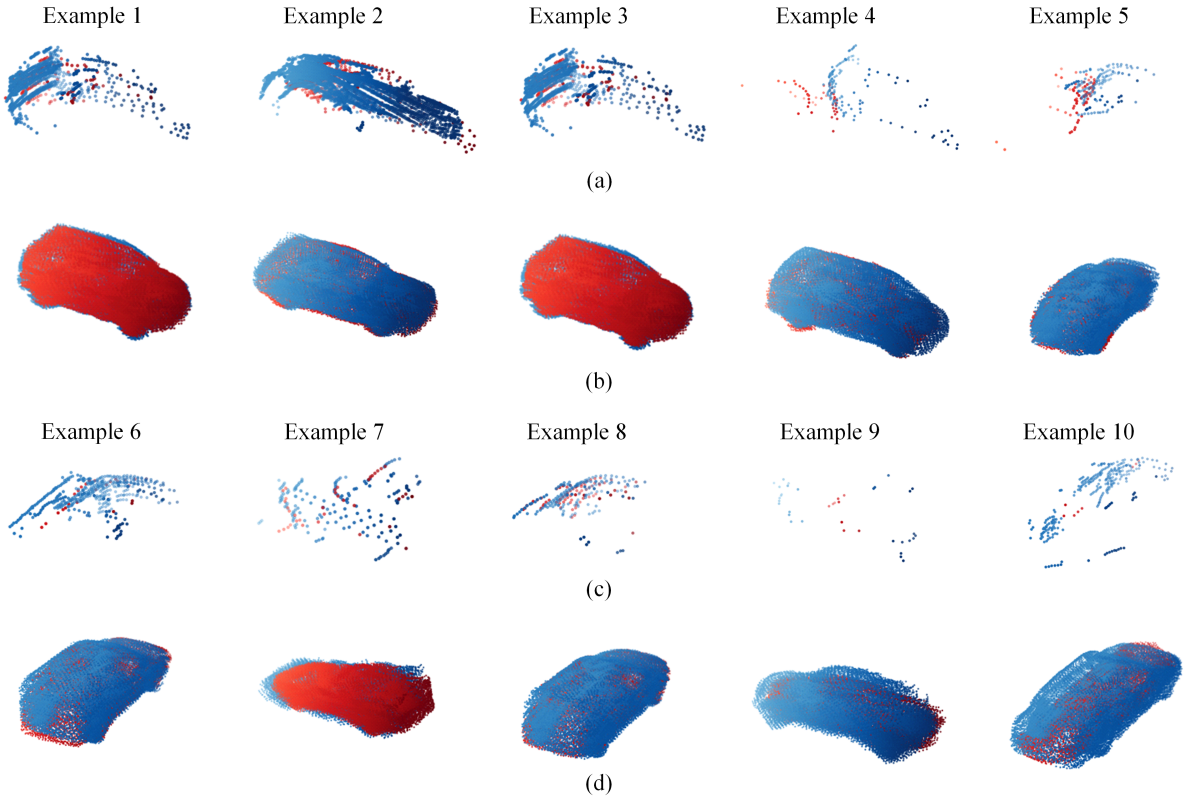


Figure 8.7: Qualitative comparison of point cloud registration task with different inputs. a) and c) Registered results with partial point clouds. b) and d) Registered completed results of the same examples.

complete point clouds when conducting the registration test. Specifically, rotation and translation accuracy improves by 42.9% and 40.5%, respectively.

In Fig. 8.7, 10 qualitative examples are displayed. The completed point clouds have large overlapping regions recovered by VPC-Net, which demonstrates that VPC-Net can generate consistent shapes with high quality for the same vehicle in different frames. The corresponding rotation and translation errors for these examples are listed in Table 8.15. As can be seen from Example 1 to Example 8, the registration using complete point clouds shows an improvement in both rotation and translation accuracy. The improvement is most significant when the error with partial inputs is relatively large. Examples 9 and 10 are failure cases where the registered partial inputs have better performance than registered complete inputs. However, this is explained by the qualitative results in Fig. 8.7: the registered partial inputs have too few points, only about 10, so the ICP method is not able to compute the errors accurately.

8.3.2 ASFM-Net

Ablation Study

The ablation study evaluates the effectiveness of different proposed modules in ASFM-Net, including both the pre-trained asymmetrical Siamese auto-encoder and refinement unit. All experiments are conducted on the Completion3D benchmark. The CD-P is selected as the evaluation metric.

Table 8.16: Ablation studies of asymmetrical Siamese auto-encoder and refinement unit on the Completion3D benchmark.

Methods	Chamfer Distance(10^{-3})								
	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft	Average
PCN [Yuan et al., 2018]	14.10	27.03	20.28	27.02	24.99	27.16	22.30	17.03	22.49
SA-PCN Decoder	13.32	25.69	19.82	23.67	22.98	24.12	22.46	16.84	21.11
PCN-Refine	10.57	23.48	18.87	21.45	18.37	22.59	18.27	13.98	18.45
TopNet [Tchapmi et al., 2019]	13.88	28.07	19.69	24.74	23.36	26.12	22.48	16.66	21.88
SA-TopNet Decoder	13.09	25.55	20.22	24.37	23.03	24.78	21.55	17.42	21.25
TopNet-Refine	10.96	23.86	18.78	21.41	17.85	22.05	18.31	14.08	18.41
w/o Refine	16.02	27.32	20.58	28.38	26.08	27.74	24.78	17.56	23.56
ASFM-Net	**10.39**	**22.42**	**18.24**	**19.32**	**17.31**	**21.66**	**17.82**	**13.74**	**17.61**

Pre-trained asymmetrical Siamese auto-encoder. In this section, the effectiveness of the asymmetrical Siamese auto-encoder which learns shape prior was evaluated. The pre-trained asymmetrical Siamese auto-encoder was used to replace the encoder modules of PCN and TopNet but keep their decoder modules respectively, referred to as SA-PCN Decoder and SA-TopNet Decoder. The quantitative results are illustrated in Table 8.16. With the proposed asymmetrical Siamese auto-encoder module, ASFM-Net brings significant improvements on the average chamfer distance by 6.1% and 2.9%, respectively, when compared with PCN and TopNet. This is due to the global features extracted by the asymmetrical Siamese auto-encoder include shape priors, while the global features directly encoded from the partial inputs are less informative. The comparison results demonstrate the superiority of the Siamese auto-encoder for 3D global feature learning for object completion.

The refinement unit. TopNet and PCN were chosen as the baseline. The refinement unit is first just integrated into TopNet and PCN, referred as TopNet-Refine and PCN-Refine. The quantitative results are shown in Table 8.16. Compared with TopNet and PCN, both TopNet-Refine and PCN-Refine improve the performance across all categories significantly. The performance of TopNet-Refine and PCN-Refine also exceeds the average chamfer distance of TopNet and PCN by 15.86% and 17.96%, respectively. In addition, the refinement unit is removed from the proposed ASFM-Net and only asymmetrical Siamese auto-encoder is used, referred to as w/o Refine. These results show that the refinement unit contributes to learning more perfect shape information with fine-grained details and is helpful for other point cloud completion networks.

Robustness Test

To further evaluate the robustness of the models, the experiments were conducted on the input point clouds with various visible ratios. The visible ratio R_v between the partial and the complete point clouds is defined as:

$$R_v = N_p / N_c \tag{8.4}$$

where N_p is the resolution of the points of a partial point cloud under the currently visible radius in the spherical space and N_c is the total points of complete point clouds. R_v ranges from 20% to 80% with a step of 20%.

The quantitative and qualitative results are shown in Table 8.17 and Fig. 8.8, respectively. From them, the following two conclusions can be concluded: (1) ASFM-Net can deal with high missing degrees more robustly. Even though the visibility is only 0.2, ASFM-Net still generates the overall shape of the airplane. Both TopNet and RFA miss a good aircraft shape and generate uneven points. The result of PCN is completely failed. (2) ASFM-Net reaches the best performance no matter in any visible ratios, which demonstrates ASFM-Net is more robust to occlusion data.

Table 8.17: Quantitative comparison on known categories under different visible ratios. The CD-P is reported by TopNet, PCN, RFA, and ASFM-Net, multiplied by 10^3.

Methods	Visible Ratio						
	20%	30%	40%	50%	60%	70%	80%
TopNet [Tchapmi et al., 2019]	41.92	34.72	30.16	27.27	25.44	24.27	23.47
PCN [Yuan et al., 2018]	45.65	37.82	33.08	30.06	27.64	25.77	24.01
RFA [Zhang et al., 2020b]	41.58	34.61	29.64	25.83	22.74	20.74	19.06
ASFM-Net	**39.94**	**31.71**	**25.97**	**22.03**	**19.29**	**17.47**	**16.27**

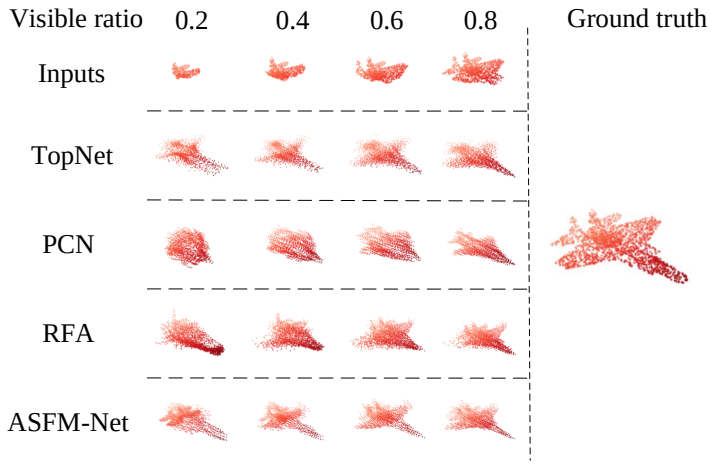


Figure 8.8: Qualitative comparison on inputs with different visible ratios. From top to down: Partial point clouds with different levels of visibility, completed point clouds by TopNet, PCN, RFA, and ASFM-Net.

Since the asymmetrical Siamese auto-encoder is trained using the models with known categories in the pre-built database, it is essential to explore its impact on the entire network ASFM-Net when facing unknown objects. In this section, eight novel categories were selected for evaluation from the ShapeNet dataset, which was divided into two groups: one is the bed, bench, bookshelf, and bus (visually similar to the training categories), another is guitar, motor-

bike, pistol, and skateboard (visually dissimilar to the training categories). All experiments are conducted on the Completion3D benchmark.

Completion on novel categories

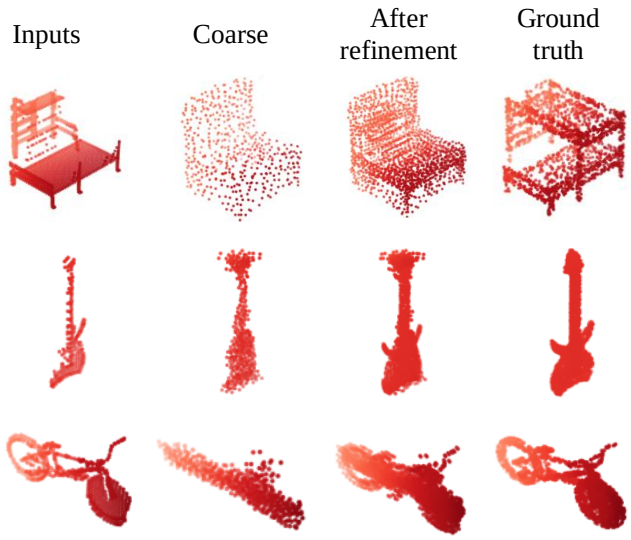


Figure 8.9: Qualitative point cloud completion result on the novel categories. 'Coarse' means the outputs are only completed by the asymmetrical Siamese auto-encoder. 'After refinement' means the final point cloud completed by ASFM-Net including a refinement unit.

The qualitative and quantitative results are shown in Table 8.18 and Fig. 8.9, respectively. From Fig. 8.9, the coarse outputs generated by the asymmetrical Siamese auto-encoder are wrong. It mistakenly completes the bed as the chair, the guitar as the lamp, and the motorbike as the watercraft. This is consistent with the expectation that the prior category information is learned from the known training categories. However, even if the asymmetrical Siamese auto-encoder provides wrong results, ASFM-Net can possibly reconstruct satisfactory point clouds (Row 2,3) thanks to the refinement unit. Besides, as shown in Table 8.18, ASFM-Net outperforms other state-of-the-art methods on all novel categories. Notably, ASFM-Net can improve performance by a large margin on visually dissimilar categories (e.g. the pistol and skateboard). This demonstrates that ASFM-Net has better generalizability than all previously tested state-of-the-art methods.

Table 8.18: Quantitative comparison on novel categories on the Completion3D benchmark. Point resolutions for the output and ground truth are 2048. For CD-P, lower is better.

Methods	Chamfer Distance(10^{-3})								
	bed	bench	bookshelf	bus	guitar	motor	pistol	skateboard	Average
TopNet [Tchapmi et al., 2019]	39.76	20.64	28.82	17.77	15.62	22.52	22.13	18.26	23.19
PCN [Yuan et al., 2018]	38.73	21.28	29.26	18.47	17.19	23.10	20.34	17.70	23.26
RFA [Zhang et al., 2020b]	34.67	19.27	23.38	18.05	17.21	21.33	19.93	18.95	21.98
ASFM-Net	**31.94**	**17.31**	**23.19**	**17.02**	**11.97**	**16.81**	**15.83**	**14.50**	**18.57**

8.3.3 Application

Apart from evaluating the effectiveness of the proposed methods, more complete and denser point clouds can be helpful for many common tasks [Yuan et al., 2018]. The completed results were applied to a 3D vehicle monitoring task. The proposed VPC-Net can provide complete shape information about vehicles, which can be regarded as an assistant for this task. It also demonstrates that the proposed method is suitable for real-time applications. The goal is to

provide the shape of the vehicles for the monitoring task only based on the existing raw LiDAR data.

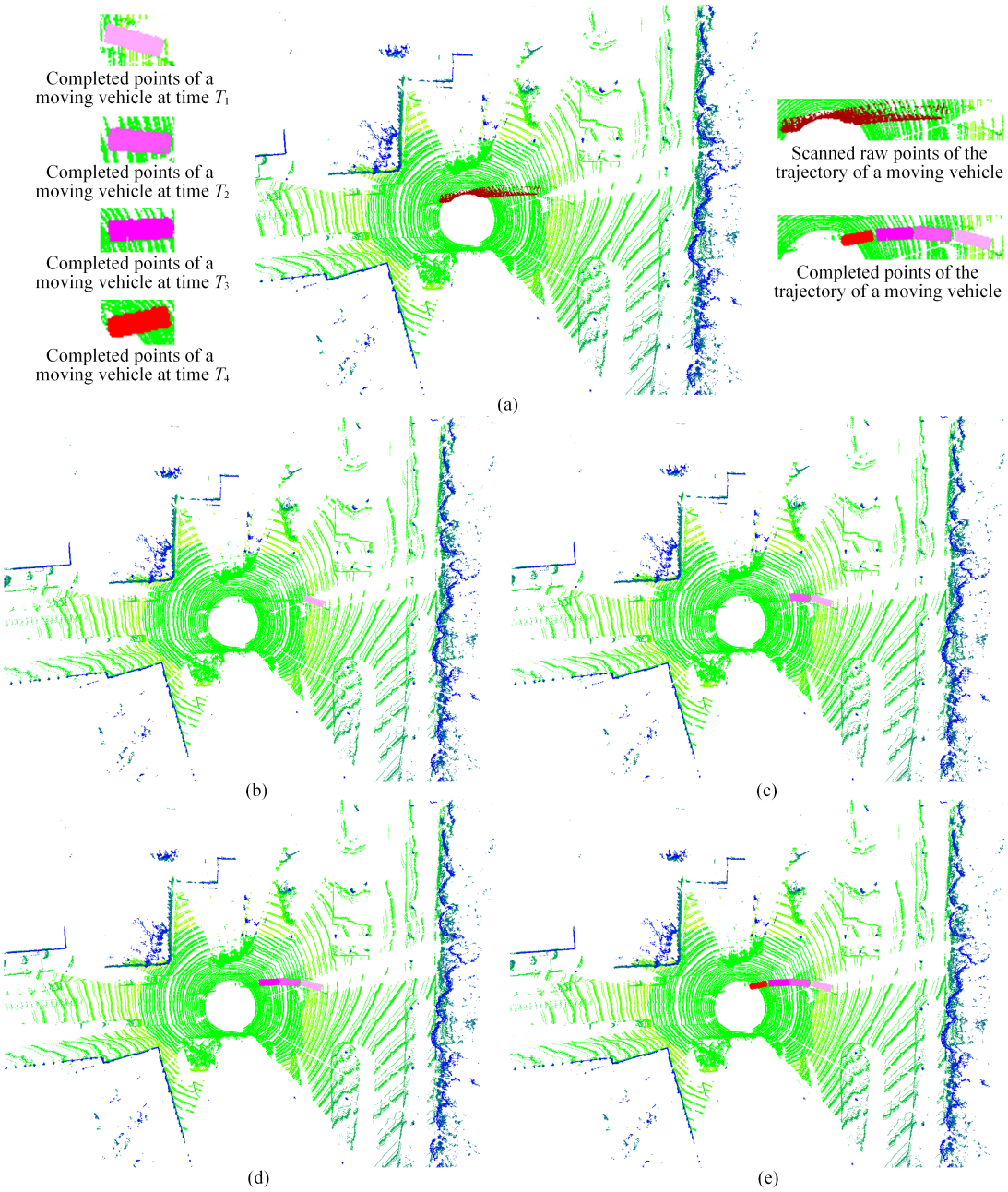


Figure 8.10: Application to 3D traffic monitoring. a) shows a 3D traffic scene at the crossroad visualized via the SLAM technique. b), c), d) and e) show the vehicle point clouds completed by VPC-Net. Different colors represent the vehicle appearing at different times.

Therefore, one Velodyne HDL-64E rotating 3D laser scanner is placed at the center of the crossroads to collect spatially dense and accurate 3D information. The round hole in Fig. 8.10a is the location of the LiDAR system. The typical monitoring technique Simultaneous Localization and Mapping (SLAM) [Cadena et al., 2016] is leveraged to estimate the vehicles in a 3D map while simultaneously localizing the object within it. The velocity, orientation, and trajectory of vehicles can be obtained using the SLAM method. However, it cannot reconstruct the complete

shape of moving vehicles, as shown in Fig. 8.10a. In Fig. 8.10a, the brown point clouds represent a moving car passed in this LiDAR-based system, and form a band shape. Dynamic vehicles were detected from each frame's raw data and completed by VPC-Net trained on the ShapeNet dataset. Figs. 8.10b-d show the completed vehicle appeared on these crossroads at continuous time T_1, T_2, T_3, and T_4, respectively. As can be seen, the proposed VPC-Net can be applied to the real-time 3D vehicle monitoring task. Furthermore, the completed point clouds have full-content information on vehicle models. As pointed out by [Pan et al., 2018], the complete shape of the measured vehicles plays an important role in designing the structure of urban highway viaducts, since it is key to estimating wind pressure caused by vehicles driving close to the sound barrier. Thus, the complete shape of vehicles will help traffic managers make the right decisions when designing highway viaducts. In addition, the 3D shape acquisition of vehicles is critical in the dynamic 3D reconstruction of traffic on road tasks [Zhang et al., 2020a]. However, they used the 3D CAD vehicle models from ShapeNet instead of real vehicles to simulate real traffic scenes. This strategy cannot deal with occluded vehicles, nor can it preserve their real shape knowledge. From this point of view, the shape of moving vehicles completed by the proposed method can support dynamic 3D traffic scene reconstruction tasks.

9 Conclusion and Outlook

In this chapter, the most important conclusions are drawn based on the work carried out within this thesis, and the new possible directions for further research work are outlined to address the limitations of the proposed methods. The conclusions are organized according to their relevance to the specific goals of this thesis as well as the research question in Section 1.3.

9.1 Conclusion

Research question I: Which accuracy of place recognition can be achieved by MLS point clouds in an urban area collected at different times?

First, the task of point cloud based place recognition is addressed. In this work, two robust and discriminative 3D global descriptors have been designed for recognizing the places of the MLS point clouds in a large-scale urban area, one focuses on identifying submaps from the aggregated point clouds, and another is for recognizing single LiDAR scans. The representation ability of the global descriptors is developed via improving receptive fields of points, the context dependencies among local descriptors, the improvement of the loss function, and the involvement of the Transformer. The improvement in place recognition results when applying the PointOE module reveals the importance of improving receptive fields. From the result of SOE-Net, a conclusion is drawn that a novel HPHN quadruplet loss can achieve more discriminative and generalizable global descriptors. On the other hand, the long-range context dependencies among local descriptors are investigated by utilizing the self-attention unit in SOE-Net and the LSA units in CASSPR. In addition, the novel hierarchical cross-attention Transformer in CASSPR is also involved in improving the discriminate features. The experiments on several benchmark datasets presenting various urban scenarios have validated the effectiveness of the aforementioned aspects in the task of point cloud based place recognition. The qualitative and quantitative results reveal the superiority of the proposed methods over other state-of-the-art methods on the benchmark datasets. Notably, the performance on the Oxford RobotCar and TUM datasets reaches a recall of 94.7% and 85.6% at top 1 retrieval, respectively. Despite the fact that the proposed methods produce promising results in place recognition, there are still notable limitations. For example, the margin in the HPHN quadruplet loss needs to be set beforehand. Exploring adaptive margins that can better distinguish positive and negative pairs should be considered in the future. The high GPU-memory requirement has also limited the application of the proposed hierarchical cross-attention Transformer to large batch sizes for training.

Research question II: Which success and precision rate of object detection and tracking (e.g. vehicles) in an urban street environment can be achieved using features learned from 3D sparse point clouds?

The second task addressed is object detection and tracking during driving on the urban road. Here, a real-time and effective pipeline is proposed to detect and track vehicles using MLS scans in two steps. The first one is vehicle detection, which locates and recognizes the vehicles in 3D

space. It can be achieved by predicting the 3D bounding boxes of the vehicles. The second one is the tracking of vehicles. It tracks the specific vehicle in successive scans given an initial detection result in the first frame. Combining the two steps, self-driving roles can fully know the locations of the vehicles that lie in the current urban scene and keep tracking them to avoid collisions. For vehicle detection, an overall precision of about 68% can be finally achieved on the TUM City Campus dataset. As for tracking, a novel lightweight 3D tracker is proposed. A motion prediction module is designed for predicting a potential target center, explicitly leveraging spatial-temporal correlations from previous frames to explore prior knowledge. In addition, a simplified voting module is proposed to regress an accurate 3D box with the guidance of the potential target center. Experiments on the KITTI Geiger et al. [2012] and NuScenes Caesar et al. [2020] benchmark datasets demonstrate the superiority of DMT over other state-of-the-art 3D SOT methods. Notably, the performance on the NuScenes dataset reaches ∼10% improvement on average, while running faster and lighter than the previous state-of-the-art methods. The current pipeline of vehicle detection and tracking in the urban area, including the procedures: data preprocessing, detection of vehicles, post-processing, and tracking of vehicles, however, has some obvious limitations. The detection results heavily influence the final tracking results. The success of tracking is dependent on optimal parameterization for each previous step. In the future, combining these steps into an end-to-end network for efficiency is a research direction.

Research question III: Which completeness and robustness can be achieved for completing point clouds of objects which are only partially scanned?

The final task addressed is 3D shape completion after the detection and tracking progress. Here, two shape completion methods are proposed to synthesize complete, dense, and uniform point clouds given a partial object from MLS data. The completion quality is improved by improving the feature extraction ability of the encoder, refining the results with fine-grained details, and learning prior information. The experimental results of VPC-Net demonstrate the importance of improving global features extracted from the encoder by including an STN and PFE layer. In addition, the improvement in shape completion results when adding a refiner module illustrates the refiner can preserve the fine details of input point clouds. From the result of ASFM-Net, a conclusion can be drawn that shapes prior information benefits generating a more informative global feature. The experiments on the various urban datasets, including the KITTI and TUM City Campus datasets, have demonstrated the effectiveness of the proposed modules and strategies. In addition, the fine-grained and highly accurate completion results when applying to the input point clouds with different point densities or intense noise and outliers reveal that the proposed methods have good robustness. Besides, the **1st place** in the leaderboard of Completion3D is achieved, exceeding the previous state-of-the-art over 12%. However, there are some obvious limitations of the proposed methods. For example, the designed refiner increases the number of training parameters. Considering the ambiguity of the completion at test time, in the future, generating multiple plausible shapes and then assessing the plausibility of several various completions will be explored. Besides, completing other objects in urban scenes, such as buildings, traffic signs, road lanes, and so on, should be investigated.

9.2 Outlook

Given the drawbacks of the proposed methods and their shortcomings in real-world applications, the following aspects can be investigated in the future:

- The global descriptors for point cloud based place recognition should be optimized. First, the the generalization ability of models should be further improved. Although the proposed

methods performed well on benchmark datasets, their performance on unseen datasets was not entirely satisfactory, limiting their applications in the real world. Currently, there are some transfer learning methods working on improving the generalization ability, so that further work could be conducted on the investigation of transfer learning methods. Second, the current methods focus solely on learning geometric features from the coordinates of 3D points. However, other information, like color and reflectance, is not thoroughly investigated. Further research could be conducted into attribute information provided by point clouds or other sensorÂ sources. Multi-sensor or multi-model fusion is a good exploration direction to improve efficiency. Third, current benchmark datasets for point cloud based place recognition tasks are still small. The 3D localization community will benefit from the creation of a larger dataset that includes different countries, lighting, driving directions, and precipitation.

- The process of 3D object detection and tracking should be further optimized. First, the detection performance could be improved. The 3D detector could be optimized according to the characteristics of point clouds. For example, the pillars can be designed following a dynamic step, depending on the number of points in each pillar. Second, the current 3D single object tracker focuses more on dynamic objects, like vehicles, pedestrians, and cyclists. In a real-world application, more categories, even unseen objects should be also tracked. Thus, it is urgent to present a more strong and more efficient 3D single object tracker. Third, multi-object tracking is a future research direction. Achieving simultaneous tracking of multiple classes of objects in a single LiDAR scan is a challenging task, but has great significance to avoid obstacles for autonomous vehicles.

- The 3D shape completion could be further improved. First, the completed objects can be expanded from a single object to a scene. The completion of a single object is based on the geometric information of the object itself, whereas the completion of a scene needs to take into account the spatial connections between objects and the contextual semantic information. Scene completion significantly benefits the scene understanding. Second, combining completion and some downstream tasks (e.g. object detection and tracking) into one is a worthwhile direction to pursue. The incompleteness of the data is one of the major challenges encountered in object detection and tracking tasks. Thus, it is natural to believe that the two tasks are closely linked and complementary. Third, almost all 3D shape completion methods are trained on synthetic 3D model datasets since it is hard to collect the complete real-scanned datasets. However, there is a domain gap between synthetic data and real-world data. Improving the generalization ability on real-scanned data should be explored by reducing the domain gap.

Bibliography

Achlioptas P, Diamanti O, Mitliagkas I, Guibas L (2018) Learning representations and generative models for 3d point clouds. In: International Conference on Machine Learning: 40–49.

Alman J, Williams VV (2021) A refined laser method and faster matrix multiplication. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA): 522–539.

Angelina Uy M, Hee Lee G (2018) Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 4470–4479.

Anguelov D, Srinivasan P, Koller D, Thrun S, Rodgers J, Davis J (2005) SCAPE: shape completion and animation of people. In: ACM SIGGRAPH 2005 Papers: 408–416.

Ao S, Guo Y, Hu Q, Yang B, Markham A, Chen Z (2022) You Only Train Once: Learning General and Distinctive 3d Local Descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Arandjelovic R, Gronat P, Torii A, Pajdla T, Sivic J (2016) NetVLAD: CNN architecture for weakly supervised place recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 5297–5307.

Armeni I, Sax S, Zamir AR, Savarese S (2017) Joint 2d-3d-semantic data for indoor scene understanding. arXiv preprint arXiv:1702.01105.

Arroyo Contera R et al. (2017) Topological place recognition for life-long visual localization.

Asvadi A, Girao P, Peixoto P, Nunes U (2016) 3D object tracking using RGB and LIDAR data. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC): 1255–1260.

Bai X, Luo Z, Zhou L, Fu H, Quan L, Tai CL (2020) D3feat: Joint learning of dense detection and description of 3d local features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 6359–6367.

Bailey T, Durrant-Whyte H (2006) Simultaneous localization and mapping (SLAM): Part II. IEEE Robotics & Automation Magazine, 13 (3): 108–117.

Besl PJ, McKay ND (1992) Method for registration of 3-D shapes. In: Sensor Fusion IV: Control Paradigms and Data Structures, 1611: 586–606.

Bibi A, Zhang T, Ghanem B (2016) 3d part-based sparse tracker with automatic synchronization and registration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 1439–1448.

Borgmann B, Schatz V, Kieritz H, Scherer-Klöckling C, Hebel M, Arens M (2018) Data processing and recording using a versatile multi-sensor vehicle. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, 4 (1): 21–28.

Boric S, Schiebel E, Schlogl C, Hildebrandt M, Hofer C, Macht DM et al. (2021) Research in autonomous driving–A historic bibliometric view of the research development in autonomous driving. International Journal of Innovation and Economic Development, 7 (5): 27–44.

Broggi A, Bertozzi M, Fascioli A, Bianco CGL, Piazzi A (1999) The ARGO autonomous vehicleâs vision and control systems. International Journal of Intelligent Control and Systems, 3 (4): 409–441.

Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R (1993) Signature verification using a" siamese" time delay neural network. Advances in Neural Information Processing Systems, 6: 737–744.

Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, Reid I, Leonard JJ (2016) Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on Robotics, 32 (6): 1309–1332.

Caesar H, Bankiti V, Lang AH, Vora S, Liong VE, Xu Q, Krishnan A, Pan Y, Baldan G, Beijbom O (2020) nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 11621–11631.

Cai Q, Pan Y, Yao T, Yan C, Mei T (2018) Memory matching networks for one-shot image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 4080–4088.

Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H et al. (2015) Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012.

Chauve AL, Labatut P, Pons JP (2010) Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition: 1261–1268.

Chen S, Liu B, Feng C, Vallespi-Gonzalez C, Wellington C (2020) 3D point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. IEEE Signal Processing Magazine, 38 (1): 68–86.

Chen W, Chen X, Zhang J, Huang K (2017) Beyond triplet loss: a deep quadruplet network for person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 403–412.

Choy C, Gwak J, Savarese S (2019a) 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 3075–3084.

Choy C, Park J, Koltun V (2019b) Fully convolutional geometric features. In: Proceedings of the IEEE International Conference on Computer Vision: 8958–8966.

Cohen TS, Geiger M, Köhler J, Welling M (2018) Spherical cnns. arXiv preprint arXiv:1801.10130.

Cui Y, Fang Z, Zhou S (2020) Point Siamese network for person tracking using 3D point clouds. Sensors, 20 (1): 143.

Dai A, Chang AX, Savva M, Halber M, Funkhouser T, Nießner M (2017a) Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 5828–5839.

Dai A, Ruizhongtai Qi C, Nießner M (2017b) Shape completion using 3d-encoder-predictor cnns and shape synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 5868–5877.

Danielsson PE (1980) Euclidean distance mapping. Computer Graphics and Image Processing, 14 (3): 227–248.

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

Deng H, Birdal T, Ilic S (2018b) Ppfnet: Global context aware local features for robust 3d point matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 195–205.

Dickmanns ED, Behringer R, Dickmanns D, Hildebrandt T, Maurer M, Thomanek F, Schiehlen J (1994) The seeing passenger car'VaMoRs-P'. In: Proceedings of the Intelligent Vehicles' 94 Symposium: 68–73.

Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S et al. (2020) An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Du J, Wang R, Cremers D (2020) Dh3d: Deep hierarchical 3d descriptors for robust large-scale 6dof relocalization. In: Proceedings of the European Conference on Computer Vision: 744–762.

Dubé R, Dugas D, Stumm E, Nieto J, Siegwart R, Cadena C (2017) Segmatch: Segment based place recognition in 3d point clouds. In: 2017 IEEE International Conference on Robotics and Automation (ICRA): 5266–5272.

Durrant-Whyte H, Bailey T (2006) Simultaneous localization and mapping: part I. IEEE Robotics & Automation Magazine, 13 (2): 99–110.

Elbaz G, Avraham T, Fischer A (2017) 3D point cloud registration for localization using a deep neural network auto-encoder. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 4631–4640.

Fan H, Su H, Guibas LJ (2017) A point set generation network for 3d object reconstruction from a single image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 605–613.

Fan Z, Song Z, Liu H, Lu Z, He J, Du X (2022) :

Fang Z, Zhou S, Cui Y, Scherer S (2020) 3D-SiamRPN: An End-to-End Learning Method for Real-Time 3D Single Object Tracking Using Raw Point Cloud. IEEE Sensors Journal, 21 (4): 4995–5011.

Fei-Fei L, Fergus R, Perona P (2006) One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (4): 594–611.

Fernández-Moral E, Mayol-Cuevas W, Arevalo V, Gonzalez-Jimenez J (2013) Fast place recognition with plane-based maps. In: 2013 IEEE International Conference on Robotics and Automation: 2719–2724.

Fernández-Moral E, Rives P, Arévalo V, González-Jiménez J (2016) Scene structure registration for localization and mapping. Robotics and Autonomous Systems, 75: 649–660.

Finman R, Paull L, Leonard JJ (2015) Toward object-based place recognition in dense rgb-d maps. In: ICRA Workshop Visual Place Recognition in Changing Environments, 76: 480.

Frome A, Huber D, Kolluri R, Bülow T, Malik J (2004) Recognizing objects in range data using regional point descriptors. In: Proceedings of the European Conference on Computer Vision: 224–237.

Fu Y, Yan Q, Yang L, Liao J, Xiao C (2018) Texture mapping for 3d reconstruction with rgb-d sensor. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 4645–4653.

Geiger A, Lenz P, Stiller C, Urtasun R (2013) Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research, 32 (11): 1231–1237.

Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition: 3354–3361.

Giancola S, Zarzar J, Ghanem B (2019) Leveraging shape completion for 3d siamese tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 1359–1368.

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

Granström K, Schön TB, Nieto JI, Ramos FT (2011) Learning to close loops from range data. The International Journal of Robotics Research, 30 (14): 1728–1754.

Groueix T, Fisher M, Kim VG, Russell BC, Aubry M (2018) A papier-mâché approach to learning 3d surface generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 216–224.

Guo MH, Cai JX, Liu ZN, Mu TJ, Martin RR, Hu SM (2021) Pct: Point cloud transformer. Computational Visual Media, 7 (2): 187–199.

Gurumurthy S, Agrawal S (2019) High fidelity semantic shape completion for point clouds using latent optimization. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV): 1099–1108.

Han X, Leung T, Jia Y, Sukthankar R, Berg AC (2015) Matchnet: Unifying feature and metric learning for patch-based matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 3279–3286.

Han X, Li Z, Huang H, Kalogerakis E, Yu Y (2017) High-resolution shape completion using deep neural networks for global structure and local geometry inference. In: Proceedings of the IEEE International Conference on Computer Vision: 85–93.

Häne C, Heng L, Lee GH, Fraundorfer F, Furgale P, Sattler T, Pollefeys M (2017) 3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. Image and Vision Computing, 68: 14–27.

Harary G, Tal A, Grinspun E (2014) Context-based coherent surface completion. ACM Transactions on Graphics (TOG), 33 (1): 1–12.

He C, Zeng H, Huang J, Hua XS, Zhang L (2020) Structure aware single-stage 3d object detection from point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 11873–11882.

He K, Zhang X, Ren S, Sun J (2016a) Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 770–778.

He L, Wang X, Zhang H (2016b) M2DP: A novel 3d point cloud descriptor and its application in loop closure detection. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS): 231–237.

Hee Lee G, Faundorfer F, Pollefeys M (2013) Motion estimation for self-driving cars with a generalized camera. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2746–2753.

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation, 9 (8): 1735–1780.

Hu J, Lu J, Tan YP (2014) Discriminative deep metric learning for face verification in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 1875–1882.

Huang J, Feris RS, Chen Q, Yan S (2015) Cross-domain image retrieval with a dual attribute-aware ranking network. In: Proceedings of the IEEE International Conference on Computer Vision: 1062–1070.

Huang Z, Yu Y, Xu J, Ni F, Le X (2020) Pf-net: point fractal network for 3d point cloud completion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 7662–7670.

Hui L, Cheng M, Xie J, Yang J, Cheng MM (2022) Efficient 3D point cloud feature learning for large-scale place recognition. IEEE Transactions on Image Processing, 31: 1258–1270.

Hui L, Yang H, Cheng M, Xie J, Yang J (2021) Pyramid point cloud transformer for large-scale place recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision: 6098–6107.

Jaderberg M, Simonyan K, Zisserman A et al. (2015) Spatial transformer networks. In: Advances in Neural Information Processing Systems: 2017–2025.

Jiang M, Wu Y, Zhao T, Zhao Z, Lu C (2018) Pointsift: A sift-like network module for 3d point cloud semantic segmentation. arXiv preprint arXiv:1807.00652.

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

Kart U, Kamarainen JK, Matas J (2018) How to make an rgbd tracker? In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops: 0–0.

Kart U, Lukezic A, Kristan M, Kamarainen JK, Matas J (2019) Object tracking by reconstruction with view-specific discriminative correlation filters. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 1339–1348.

Kazhdan M, Bolitho M, Hoppe H (2006) Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, 7.

Khoury M, Zhou QY, Koltun V (2017) Learning compact geometric features. In: Proceedings of the IEEE International Conference on Computer Vision: 153–161.

Kim G, Kim A (2018) Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS): 4802–4809.

Koch G, Zemel R, Salakhutdinov R et al. (2015) Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop, 2: 0.

Kokkinos I, Bronstein MM, Litman R, Bronstein AM (2012) Intrinsic shape context descriptors for deformable shapes. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition: 159–166.

Komorowski J (2021) Minkloc3d: Point cloud based large-scale place recognition. In: Proceedings of the IEEE Winter Conference on Applications of Computer Vision: 1790–1799.

Kulis B et al. (2013) Metric learning: A survey. Foundations and Trends in Machine Learning, 5 (4): 287–364.

Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O (2019) Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 12697–12705.

Li B, Wu W, Wang Q, Zhang F, Xing J, Yan J (2019) Siamrpn++: Evolution of siamese visual tracking with very deep networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 4282–4291.

Li B, Yan J, Wu W, Zhu Z, Hu X (2018) High performance visual tracking with siamese region proposal network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 8971–8980.

Li G, Liu L, Zheng H, Mitra NJ (2010) Analysis, reconstruction and manipulation using arterial snakes. ACM Transactions on Graphics (TOG), 29 (6): 152.

Li W, Zhao R, Xiao T, Wang X (2014) Deepreid: Deep filter pairing neural network for person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 152–159.

Li Y, Wu X, Chrysathou Y, Sharf A, Cohen-Or D, Mitra NJ (2011) Globfit: Consistently fitting primitives by discovering global relations. In: ACM SIGGRAPH 2011 papers: 1–12.

Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017a) Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2117–2125.

Lin TY, Goyal P, Girshick R, He K, Dollár P (2017b) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision: 2980–2988.

Litany O, Bronstein A, Bronstein M, Makadia A (2018) Deformable shape completion with graph convolutional autoencoders. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 1886–1895.

Liu H, Zhang G, Bao H (2016a) Robust keyframe-based monocular SLAM for augmented reality. In: 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR): 1–10.

Liu M, Sheng L, Yang S, Shao J, Hu SM (2020) Morphing and sampling network for dense point cloud completion. In: Proceedings of the AAAI Conference on Artificial Intelligence, 34 (07): 11596–11603.

Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016b) Ssd: Single shot multibox detector. In: European Conference on Computer Vision: 21–37.

Liu Z, Zhou S, Suo C, Yin P, Chen W, Wang H, Li H, Liu YH (2019) Lpd-net: 3d point cloud learning for large-scale place recognition and environment analysis. In: Proceedings of the IEEE International Conference on Computer Vision: 2831–2840.

Lowe DG (2004) Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60 (2): 91–110.

Lu F, Chen G, Liu Y, Qu Z, Knoll A (2020) Rskdd-net: Random sample-based keypoint detector and descriptor. arXiv preprint arXiv:2010.12394.

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

Luo Z, Zhou L, Bai X, Chen H, Zhang J, Yao Y, Li S, Fang T, Quan L (2020) Aslfeat: Learning local features of accurate shape and localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 6589–6598.

Maddern W, Pascoe G, Linegar C, Newman P (2017) 1 Year, 1000km: The Oxford RobotCar Dataset. The International Journal of Robotics Research (IJRR), 36 (1): 3–15.

Magnusson M, Andreasson H, Nüchter A, Lilienthal AJ (2009) Automatic appearance-based loop detection from three-dimensional laser data using the normal distributions transform. Journal of Field Robotics, 26 (11-12): 892–914.

Malassiotis S, Strintzis MG (2007) Snapshots: A novel local surface descriptor and matching algorithm for robust 3d surface alignment. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (7): 1285–1290.

Mandikal P, Radhakrishnan VB (2019) Dense 3d point cloud reconstruction using a deep pyramid network. In: 2019 IEEE Winter Conference on Applications of Computer Vision: 1052–1060.

Mao J, Shi S, Wang X, Li H (2022) 3D Object Detection for Autonomous Driving: A Review and New Outlooks. arXiv preprint arXiv:2206.09474.

Mao J, Xue Y, Niu M, Bai H, Feng J, Liang X, Xu H, Xu C (2021) Voxel transformer for 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision: 3164–3173.

Mian AS, Bennamoun M, Owens R (2006) Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (10): 1584–1601.

Moosmann F, Stiller C (2013) Joint self-localization and tracking of generic objects in 3D range data. In: 2013 IEEE International Conference on Robotics and Automation: 1146–1152.

Mur-Artal R, Montiel JMM, Tardos JD (2015) ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Transactions on Robotics, 31 (5): 1147–1163.

Nan L, Sharf A, Zhang H, Cohen-Or D, Chen B (2010) Smartboxes for interactive urban reconstruction. In: ACM SIGGRAPH 2010 papers: 1–10.

Nebel D, Kaden M, Villmann A, Villmann T (2017) Types of (dis-) similarities and adaptive mixtures thereof for improved classification learning. Neurocomputing, 268: 42–54.

Nguyen HV, Bai L (2010) Cosine similarity metric learning for face verification. In: Asian Conference on Computer Vision: 709–720.

Oh Song H, Xiang Y, Jegelka S, Savarese S (2016) Deep metric learning via lifted structured feature embedding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 4004–4012.

Osep A, Mehner W, Mathias M, Leibe B (2017) Combined image-and world-space tracking in traffic scenes. In: 2017 IEEE International Conference on Robotics and Automation (ICRA): 1988–1995.

Pan L, Chen X, Cai Z, Zhang J, Zhao H, Yi S, Liu Z (2021) Variational relational point completion network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 8524–8533.

Pan Y, Wang D, Shen X, Xu Y, Pan Z (2018) A novel computer vision-based monitoring methodology for vehicle-induced aerodynamic load on noise barrier. Structural Control and Health Monitoring, 25 (12): 2271.

Panphattarasap P (2019) Urban patterns: using spatial arrangement for vision-based place recognition and localisation. PhD thesis, University of Bristol.

Park C, Jeong Y, Cho M, Park J (2022) Fast Point Transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 16949–16958.

Patil A, Malla S, Gang H, Chen YT (2019) The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. In: 2019 International Conference on Robotics and Automation (ICRA): 9552–9557.

Pauly M, Mitra NJ, Giesen J, Gross MH, Guibas LJ (2005) Example-based 3d scan completion. In: Symposium on Geometry Processing: 23–32.

Pauly M, Mitra NJ, Wallner J, Pottmann H, Guibas LJ (2008) Discovering structural regularity in 3d geometry. In: ACM SIGGRAPH 2008 papers: 1–11.

Petrelli A, Di Stefano L (2011) On the repeatability of the local reference frame for partial shape matching. In: 2011 International Conference on Computer Vision: 2244–2251.

Pham QH, Uy MA, Hua BS, Nguyen DT, Roig G, Yeung SK (2020) LCD: learned cross-domain descriptors for 2D-3D matching. In: Proceedings of the AAAI Conference on Artificial Intelligence, 34 (07): 11856–11864.

Pomerleau D, Jochem T (1996) Rapidly adapting machine vision for automated vehicle steering. IEEE Expert, 11 (2): 19–27.

Qi CR, Su H, Mo K, Guibas LJ (2017a) Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 652–660.

Qi CR, Yi L, Su H, Guibas LJ (2017b) Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems: 5099–5108.

Qi H, Feng C, Cao Z, Zhao F, Xiao Y (2020) P2B: Point-to-box network for 3D object tracking in point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 6329–6338.

Radenović F, Tolias G, Chum O (2018) Fine-tuning CNN image retrieval with no human annotation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41 (7): 1655–1668.

Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems: 91–99.

Rock J, Gupta T, Thorsen J, Gwak J, Shin D, Hoiem D (2015) Completing 3d object shape from one depth image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2484–2493.

Röhling T, Mack J, Schulz D (2015) A fast histogram-based similarity measure for detecting loop closures in 3-d lidar data. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems: 736–741.

Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention: 234–241.

Rusu RB, Blodow N, Beetz M (2009) Fast point feature histograms (FPFH) for 3d registration. In: 2009 IEEE International Conference on Robotics and Automation: 3212–3217.

Rusu RB, Blodow N, Marton ZC, Beetz M (2008) Aligning point cloud views using persistent feature histograms. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems: 3384–3391.

Sarmad M, Lee HJ, Kim YM (2019) Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 5898–5907.

Scheidegger S, Benjaminsson J, Rosenberg E, Krishnan A, Granström K (2018) Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering. In: 2018 IEEE Intelligent Vehicles Symposium (IV): 433–440.

Schlichting A, Brenner C (2014) Localization using automotive laser scanners and local pattern matching. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings: 414–419.

Schnabel R, Degener P, Klein R (2009) Completion and reconstruction with primitive shapes. In: Computer Graphics Forum, 28 (2): 503–512.

Schroff F, Kalenichenko D, Philbin J (2015) Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 815–823.

Schwaller P, Laino T, Gaudin T, Bolgar P, Hunter CA, Bekas C, Lee AA (2019) Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. ACS central science, 5 (9): 1572–1583.

Shan J, Zhou S, Fang Z, Cui Y (2021) PTT: Point-Track-Transformer Module for 3D Single Object Tracking in Point Clouds. arXiv preprint arXiv:2108.06455.

Shen CH, Fu H, Chen K, Hu SM (2012) Structure recovery by part assembly. ACM Transactions on Graphics (TOG), 31 (6): 1–11.

Shi S, Guo C, Jiang L, Wang Z, Shi J, Wang X, Li H (2020a) Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 10529–10538.

Shi S, Wang Z, Shi J, Wang X, Li H (2020b) From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43 (8): 2647–2664.

Söderlund H (2019) Real-time detection and tracking of moving objects using deep learning and multi-threaded kalman filtering: A joint solution of 3d object detection and tracking for autonomous driving.

Spezialetti R, Salti S, Stefano LD (2019) Learning an effective equivariant 3d descriptor without supervision. In: Proceedings of the IEEE International Conference on Computer Vision: 6401–6410.

Stavens DM (2011) Learning to drive: Perception for autonomous cars. Stanford University.

Stutz D, Geiger A (2018) Learning 3d shape completion from laser scan data with weak supervision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 1955–1964.

Sun Q, Liu H, He J, Fan Z, Du X (2020) DAGC: Employing dual attention and graph convolution for point cloud based place recognition. In: Proceedings of the 2020 International Conference on Multimedia Retrieval: 224–232.

Sun Y, Abidi MA (2001) Surface matching by 3d point's fingerprint. In: Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, 2: 263–269.

Sun Y, Chen Y, Wang X, Tang X (2014) Deep learning face representation by joint identification-verification. Advances in Neural Information Processing Systems, 27.

Sung M, Kim VG, Angst R, Guibas L (2015) Data-driven structural priors for shape completion. ACM Transactions on Graphics (TOG), 34 (6): 1–11.

Tagliasacchi A, Olson M, Zhang H, Hamarneh G, Cohen-Or D (2011) Vase: Volume-aware surface evolution for surface reconstruction from incomplete point clouds. In: Computer Graphics Forum, 30 (5): 1563–1571.

Tatarchenko M, Richter SR, Ranftl R, Li Z, Koltun V, Brox T (2019) What do single-view 3d reconstruction networks learn? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 3405–3414.

Tchapmi LP, Kosaraju V, Rezatofighi H, Reid I, Savarese S (2019) Topnet: Structural point cloud decoder. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 383–392.

Tevs A, Huang Q, Wand M, Seidel HP, Guibas L (2014) Relating shapes via geometric symmetries and regularities. ACM Transactions on Graphics (TOG), 33 (4): 1–12.

Thomas H, Qi CR, Deschaud JE, Marcotegui B, Goulette F, Guibas LJ (2019) Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE International Conference on Computer Vision: 6411–6420.

Thrun S, Wegbreit B (2005) Shape from symmetry. In: Proceedings of the IEEE International Conference on Computer Vision: 1824–1831.

Tombari F, Salti S, Stefano LD (2010) Unique signatures of histograms for local surface description. In: European Conference on Computer Vision: 356–369.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Advances in neural information processing systems, 30.

Vo AV, Truong-Hong L, Laefer DF, Bertolotto M (2015) Octree-based region growing for point cloud segmentation. ISPRS Journal of Photogrammetry and Remote Sensing, 104: 88–100.

Wang J, Song Y, Leung T, Rosenberg C, Wang J, Philbin J, Chen B, Wu Y (2014) Learning fine-grained image similarity with deep ranking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 1386–1393.

Wang M, Tseng YH (2011) Incremental segmentation of lidar point clouds with an octree-structured voxel space. The Photogrammetric Record, 26 (133): 32–57.

Wang N, Zhang Y, Li Z, Fu Y, Liu W, Jiang YG (2018) Pixel2mesh: Generating 3d mesh models from single rgb images. In: Proceedings of the European Conference on Computer Vision: 52–67.

Wang X, Ang Jr MH, Lee GH (2020a) Cascaded refinement network for point cloud completion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 790–799.

Wang X, Ang Jr MH, Lee GH (2020b) A self-supervised cascaded refinement network for point cloud completion. arXiv preprint arXiv:2010.08719.

Wang Y, Sun Z, Xu CZ, Sarma SE, Yang J, Kong H (2020c) Lidar iris for loop-closure detection. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS): 5769–5775.

Wang Y, Tan DJ, Navab N, Tombari F (2020d) Softpoolnet: Shape descriptor for point cloud completion and classification. In: Proceedings of the European Conference on Computer Vision: 70–85.

Wang Z, Xie Q, Lai YK, Wu J, Long K, Wang J (2021) MLVSNet: Multi-Level Voting Siamese Network for 3D Visual Tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision: 3101–3110.

Wen C, Dai Y, Xia Y, Lian Y, Tan J, Wang C, Li J (2019) Toward efficient 3d colored Mapping in GPS-/GNSS-denied environments. IEEE Geoscience and Remote Sensing Letters, 17 (1): 147–151.

Wen X, Li T, Han Z, Liu YS (2020) Point cloud completion by skip-attention network with hierarchical folding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 1939–1948.

Weng X, Wang J, Held D, Kitani K (2020a) 3d multi-object tracking: A baseline and new evaluation metrics. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS): 10359–10366.

Weng X, Wang Y, Man Y, Kitani KM (2020b) Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 6499–6508.

Wu H, Li Q, Wen C, Li X, Fan X, Wang C (2021) Tracklet Proposal Network for Multi-Object Tracking on Point Clouds. In: Zhou ZH (ed) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21: 1165–1171. Main Track.

Wu P, Hoi SC, Xia H, Zhao P, Wang D, Miao C (2013) Online multimodal deep similarity learning with application to image retrieval. In: Proceedings of the ACM International Conference on Multimedia: 153–162.

Wu S, Huang H, Gong M, Zwicker M, Cohen-Or D (2015) Deep points consolidation. ACM Transactions on Graphics (ToG), 34 (6): 1–13.

Xia Y, Wang C, Xu Y, Zang Y, Liu W, Li J, Stilla U (2019) RealPoint3D: Generating 3d point clouds from a single image of complex scenarios. Remote Sensing, 11 (22): 2644.

Xia Y, Xu Y, Li S, Wang R, Du J, Cremers D, Stilla U (2021a) Soe-net: A self-attention and orientation encoding network for point cloud based place recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 11348–11357.

Xia Y, Xu Y, Wang C, Stilla U (2021b) VPC-Net: Completion of 3d vehicles from MLS point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 174: 166–181.

Xie H, Yao H, Zhou S, Mao J, Zhang S, Sun W (2020a) Grnet: Gridding residual network for dense point cloud completion. In: European Conference on Computer Vision: 365–381.

Xie H, Yao H, Zhou S, Mao J, Zhang S, Sun W (2020b) Grnet: Gridding residual network for dense point cloud completion. In: European Conference on Computer Vision: 365–381.

Xu Q, Zhong Y, Neumann U (2022) Behind the curtain: Learning occluded shapes for 3D object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence, 36 (3): 2893–2901.

Xu TX, Guo YC, Lai YK, Zhang SH (2021) TransLoc3D: Point cloud based large-scale place recognition using adaptive receptive fields. arXiv preprint arXiv:2105.11605.

Xu Y, Boerner R, Yao W, Hoegner L, Stilla U (2019) Pairwise coarse registration of point clouds in urban scenes using voxel-based 4-planes congruent sets. ISPRS Journal of Photogrammetry and Remote Sensing, 151: 106–123.

Yamany SM, Farag AA (2002) Surface signatures: an orientation independent free-form surface representation scheme for the purpose of objects registration and matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (8): 1105–1120.

Yan Y, Mao Y, Li B (2018) Second: Sparsely embedded convolutional detection. Sensors, 18 (10): 3337.

Yang Y, Feng C, Shen Y, Tian D (2018) Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 206–215.

Yang Z, Sun Y, Liu S, Jia J (2020) 3dssd: Point-based 3d single stage object detector. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 11040–11048.

Yang Z, Sun Y, Liu S, Shen X, Jia J (2019) Std: Sparse-to-dense 3d object detector for point cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision: 1951–1960.

Yew ZJ, Lee GH (2018) 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In: Proceedings of the European Conference on Computer Vision: 630–646.

Yi D, Lei Z, Liao S, Li SZ (2014) Deep metric learning for person re-identification. In: 2014 22nd International Conference on Pattern Recognition: 34–39.

Yuan W, Khot T, Held D, Mertz C, Hebert M (2018) Pcn: Point completion network. In: 2018 International Conference on 3D Vision (3DV): 728–737.

Zaharescu A, Boyer E, Varanasi K, Horaud R (2009) Surface feature detection and description with applications to mesh matching. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition: 373–380.

Zarzar J, Giancola S, Ghanem B (2019) Efficient bird eye view proposals for 3D Siamese tracking. arXiv preprint arXiv:1903.10168.

Zeng A, Song S, Nießner M, Fisher M, Xiao J, Funkhouser T (2017) 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 1802–1811.

Zhang H, Goodfellow I, Metaxas D, Odena A (2019a) Self-attention generative adversarial networks. In: International Conference on Machine Learning: 7354–7363.

Zhang J, Chen W, Wang Y, Vasudevan R, Johnson-Roberson M (2021) Point set voting for partial point cloud analysis. IEEE Robotics and Automation Letters, 6 (2): 596–603.

Zhang S, Wang C, He Z, Li Q, Lin X, Li X, Zhang J, Yang C, Li J (2020a) Vehicle global 6-DoF pose estimation under traffic surveillance camera. ISPRS Journal of Photogrammetry and Remote Sensing, 159: 114–128.

Zhang W, Xiao C (2019) PCAN: 3D attention map learning using contextual information for point cloud based retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 12436–12445.

Zhang W, Yan Q, Xiao C (2020b) Detail preserved point cloud completion via separated feature aggregation. In: Proceedings of the European Conference on Computer Vision: 512–528.

Zhang W, Zhou H, Sun S, Wang Z, Shi J, Loy CC (2019b) Robust multi-modality multi-object tracking. In: Proceedings of the IEEE International Conference on Computer Vision: 2365–2374.

Zhang Z, Peng H (2019) Deeper and wider siamese networks for real-time visual tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 4591–4600.

Zhao H, Jiang L, Jia J, Torr PH, Koltun V (2021) Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision: 16259–16268.

Zheng C, Yan X, Gao J, Zhao W, Zhang W, Li Z, Cui S (2021a) Box-Aware Feature Enhancement for Single Object Tracking on Point Clouds. In: Proceedings of the IEEE International Conference on Computer Vision: 13199–13208.

Zheng Q, Sharf A, Wan G, Li Y, Mitra NJ, Cohen-Or D, Chen B (2010) Non-local scan consolidation for 3d urban scenes. ACM Transactions on Graphics (TOG), 29 (4): 94–1.

Zheng W, Tang W, Jiang L, Fu CW (2021b) SE-SSD: Self-ensembling single-stage object detector from point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 14494–14503.

Zhong Y (2009) Intrinsic shape signatures: A shape descriptor for 3d object recognition. In: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops: 689–696.

Zhou L, Zhu S, Luo Z, Shen T, Zhang R, Zhen M, Fang T, Quan L (2018) Learning and matching multi-view descriptors for registration of point clouds. In: Proceedings of the European Conference on Computer Vision (ECCV): 505–522.

Zhou W, Berrio JS, De Alvis C, Shan M, Worrall S, Ward J, Nebot E (2020) Developing and testing robust autonomy: The university of sydney campus data set. IEEE Intelligent Transportation Systems Magazine, 12 (4): 23–40.

Zhou Y, Tuzel O (2018a) Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 4490–4499.

Zhou Y, Tuzel O (2018b) Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 4490–4499.

Zhou Z, Zhao C, Adolfsson D, Su S, Gao Y, Duckett T, Sun L (2021) Ndt-transformer: Large-scale 3d point cloud localisation using the normal distribution transform representation. In: IEEE International Conference on Robotics and Automation: 5654–5660.

Zhu J, Gehrung J, Huang R, Borgmann B, Sun Z, Hoegner L, Hebel M, Xu Y, Stilla U (2020) TUM-MLS-2016: An annotated mobile LiDAR dataset of the TUM city campus for semantic point cloud interpretation in urban areas. Remote Sensing, 12 (11): 1875.

Zhu X, Zhou H, Wang T, Hong F, Ma Y, Li W, Li H, Lin D (2021) Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition: 9939–9948.

Zou H, Cui J, Kong X, Zhang C, Liu Y, Wen F, Li W (2020) F-Siamese Tracker: A Frustum-based Double Siamese Network for 3D Single Object Tracking. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS): 8133–8139.

Żywanowski K, Banaszczyk A, Nowicki MR, Komorowski J (2021) MinkLoc3D-SI: 3d LiDAR place recognition with sparse convolutions, spherical coordinates, and intensity. IEEE Robotics and Automation Letters, 7 (2): 1079–1086.

Acknowledgment

From the beginning of my PhD study to the completion of my doctoral dissertation, I have devoted myself to the development of deep learning methods on point clouds for autonomous driving and robotics applications. Life during doctoral studies is struggling and lonely, especially during the coronavirus pandemic. There are many people without whom this dissertation would not have been completed, or would not have ended in such a successful manner. I would like to express my sincere gratitude to all those who have contributed to my success.

First and foremost, I would like to convey my sincere thanks to Prof. Uwe Stilla, who offered me the opportunity to do this research in his team at Technical University of Munich. I really feel lucky to be one of his students since I benefited greatly from his rigorous research attitude while studying for my PhD under his supervision. Prof. Uwe Stilla provided me with many valuable suggestions in academic studies, including how to write a good conclusion, how to give a good talk, and so on. He also encouraged me to work with other research groups around the world and to visit University of Oxford. His insightful feedback, constructive criticism, and encouragement have helped me develop my ideas, refine my research questions, and improve my writing. I am grateful for his wisdom, expertise, and patience, which have been invaluable in helping me navigate the complexities of conducting research and writing a dissertation.

I would also like to express my deep appreciation to the members of my dissertation committee, Prof. Daniel Cremers and Prof. Yusheng Xu, for their valuable feedback, critical insights, and constructive comments that have significantly improved the quality of my dissertation. Their expertise, experience, and knowledge have been instrumental in helping me develop my research methodology and interpret my findings. I am grateful for their willingness to share their time, expertise, and resources, and for their support and encouragement throughout my doctoral journey. I would also like to express my gratitude to Prof. Marco Körner for his chairmanship of the examination committee.

I would like to thank my colleagues, Prof. Ludwig Hoegner, Philipp Roman Hirt, Michael Greza, Lukas Lucks, Manoj Kumar Biswanath, Olaf Wysocki, Jingwei Zhu, Max Hoedel, Joachim Gehrung, for their friendship, support, and camaraderie. I am grateful for the many memories we have shared over the years. I would also like to convey my special thanks to Dr. Joao Henriques in Visual Geometry Group at University of Oxford for providing valuable guidance and support on my research. Besides, I would like to express my thanks to my Chinese colleagues in Visual Geometry Group, Dr. Tengda Han, Chuhang Zhang, Guanqi Zhan, Minghao Chen, Junyu Xie, Jianyuan Wang, Dr. Shangzhe Wu, for their support and help to make my time at University of Oxford a happy one. I would also convey my thanks to Qianru Zhao, Xin Chen, Lin Zhou, Chenyu Fang, Peng Luo, Qiangqiang Wu, Xinyi Li, Hu Cao, for their support and their willingness to listen.

I am also deeply indebted to my family, for their love, support, and encouragement. Their unwavering belief in me, their sacrifices, and their constant encouragement have been a source of strength and inspiration throughout my PhD journey. I am very grateful for their unwavering support, and their constant encouragement.