University of Cyprus

DEPARTMENT OF COMPUTER SCIENCE

Graph-based Neural Network Models for 3D Shape Segmentation

Marios Loizou

A dissertation submitted to the University of Cyprus

in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

April, 2024

VALIDATION PAGE

Doctoral Candidate: Marios Loizou

Doctoral Dissertation Title: Graph-based Neural Network Models for 3D Shape Segmentation

The present Doctoral Dissertation was submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy at the Department of Computer Science and was approved on ***April 26, 2024*** *by the members of the Examination Committee.*

Examination Committee:

Research Supervisor _____
Associate Professor Evangelos Kalogerakis

Research Supervisor _____
Professor Yiorgos Chrysanthou

Committee Chair _____
Professor Constantinos Pattichis

Committee Member _____
Assistant Professor Andreas Aristidou

Committee Member _____
Associate Professor Amit Haim Bermano

Committee Member _____
Assistant Professor Minhyuk Sung

DECLARATION OF DOCTORAL CANDIDATE

The present Doctoral Dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned through references, notes, or any other statements.

Marios Loizou

.....

Περίληψη

Η κατάτμηση τρισδιάστατων σχημάτων στα συστατικά τους μέρη αποτελεί ένα διαχρονικό πρόβλημα στην Υπολογιστική Όραση και τα Γραφικά Υπολογιστών. Πρόσφατες επιτυχίες στην τρισδιάστατη Βαθιά Μάθηση οδήγησαν σε πληθώρα μεθόδων για την εκμάθηση αποτελεσματικών αναπαραστάσεων, χρήσιμων για υψηλού επιπέδου προβλήματα επεξεργασίας σχημάτων, συμπεριλαμβανομένης της κατάτμησης σχημάτων. Παρά αυτή τη σημαντική πρόοδο, οι περισσότερες μέθοδοι βασίζονται στην επεξεργασία τοπικών γεωμετρικών γειτονιών και συχνά παραβλέπουν το ευρύτερο πλαίσιο, όπως η δομή, οι συμμετρίες και οι αντιστοιχίες με άλλα σχήματα που συχνά είναι χρήσιμα για τον εντοπισμό και την εξαγωγή τμημάτων από γεωμετρικά χαρακτηριστικά των σχημάτων. Επιπλέον, οι τρισδιάστατες συλλογές δεδομένων που χρησιμοποιούνται ευρέως, περιλαμβάνουν κυρίως τεχνητά αντικείμενα με απλή δομή και στερούνται μοντέλων μεγάλης κλίμακας με υψηλή δομική πολυπλοκότητα.

Αυτή η διατριβή παρουσιάζει νευρωνικές μεθόδους βασισμένες σε γράφους, οι οποίες μοντελοποιούν πολύπλοκες δομικές και χωρικές σχέσεις εντός του ίδιου τρισδιάστατου σχήματος καθώς και μεταξύ των σχημάτων, τα οποία αναπαρίστανται ως γράφοι, με αποτέλεσμα την κατάτμηση σχημάτων με μεγαλύτερη συνέπεια και ακρίβεια. Επιπλέον, η διατριβή παρουσιάζει τη πρώτη συλλογή τρισδιάστατων δεδομένων μεγάλης κλίμακας που αποτελείται από επισημασμένα μοντέλα τρισδιάστατων κτηρίων, και η οποία είναι διαθέσιμη στο ευρύ κοινό. Τα κτήρια έχουν μεγαλύτερη δομική πολυπλοκότητα σε σύγκριση με αντικείμενα που προσφέρονται από κοινές συλλογές δεδομένων. Αυτό έχει ως αποτέλεσμα, η συγκεκριμένη συλλογή δεδομένων να αποτελεί ένα χρήσιμο σημείο αναφοράς για τη συγκριτική αξιολόγηση αλγορίθμων κατάτμησης γεωμετρικών δεδομένων μεγάλης

κλίμακας, που παρουσιάζουν πολύπλοκη δομή.

Πιο συγκεκριμένα, η διατριβή προτείνει τις ακόλουθες τρεις αρχιτεκτονικές νευρωνικών δικτύων βασισμένες σε γράφους, για τη κατάτμηση σχημάτων. Αρχικά, η διατριβή παρουσιάζει τη μέθοδο PB-DGCNN, η οποία μπορεί να ανιχνεύσει τα σύνορα τμημάτων σε 3Δ σχήματα. Αυτή η μέθοδος αξιοποιεί ένα συνελικτικό νευρωνικό δίκτυο για γράφους, που συλλαμβάνει ιεραρχικά τοπικές και μη-τοπικές αλληλεπιδράσεις ενός ζεύγους σημείων, για να μάθει αναπαραστάσεις κατάλληλες για την εξαγωγή υποψηφίων συνόρων μεταξύ τμημάτων. Ο συγκεκριμένος ανιχνευτής συνόρων είναι ανεξαρτήτου κατηγορίας. Μπορεί να εκπαιδευτεί για τον εντοπισμό συνόρων είτε μεταξύ σημασιολογικών τμημάτων, είτε μεταξύ γεωμετρικών πρωτοτύπων που χρησιμοποιούνται συνήθως στη 3Δ μοντελοποίηση. Το PB-DGCNN εξάγει υποψήφια σύνορα σε νέφη σημείων (point clouds) που αναπαρίστανται ως πιθανότητες ανά σημείο. Αυτές οι πιθανότητες συνόρων μπορούν εύκολα να χρησιμοποιηθούν ως διμερής όρος σε μια προσέγγιση κοπής γράφων για την εξαγωγή συνόρων που οριοθετούν τμήματα. Η μέθοδος αποδεικνύεται ότι βελτιώνει τη κατάτμηση του σχήματος παράγοντας πιο καθαρά σύνορα τμημάτων.

Στη συνέχεια, η διατριβή παρουσιάζει το CrossShapeNet, ένα νευρωνικό δίκτυο βασισμένο σε γράφους που μεταδίδει χαρακτηριστικά σημείων σε σχήματα εντός μιας συλλογής, με σκοπό το συγχρονισμό τους και τη βελτίωση της συνοχή τους για τη κατάτμηση 3Δ σχημάτων. Ειδικότερα, η μέθοδος αυτή εισάγει ένα cross-shape attention μηχανισμό για να επιτρέψει τις αλληλεπιδράσεις μεταξύ των χαρακτηριστικών σημείων ενός σχήματος και εκείνων άλλων σχημάτων. Ο μηχανισμός αξιολογεί το βαθμό αλληλεπίδρασης μεταξύ των σημείων και παράλληλα διευθετεί τη διάδοση των χαρακτηριστικών σε διάφορα σχήματα, βελτιώνοντας την ακρίβεια και τη συνοχή των αποτελεσμάτων σε επίπεδο χαρακτηριστικών σημείων, με σκοπό την κατάτμηση των σχημάτων. Επιπλέον, προτείνεται ένα μέτρο ανάκτησης σχήματος, το οποίο επιλέγει κατάλληλα σχήματα για τη cross-shape

attention λειτουργία, για κάθε δοκιμαστικό σχήμα. Σε σύγκριση με προηγούμενες μεθόδους, το CrossShapeNet επιτυγχάνει την υψηλότερη απόδοση κατάτμησης σε όρους μέσου Part IoU στο PartNet, ένα απαιτητικό σημείο αναφοράς για συγκριτική αξιολόγηση στην κατάτμηση τεχνητών αντικειμένων.

Τέλος, η διατριβή παρουσιάζει το BuildingNet, τη πρώτη συλλογή δεδομένων μεγάλης κλίμακας που είναι διαθέσιμη στο ευρύ κοινό και περιλαμβάνει επισημασμένα 3Δ μοντέλα κτηρίων, των οποίων το εξωτερικό και ο περιβάλλων χώρος έχουν ετικεταριστεί με συνέπεια. Αυτή η συλλογή δεδομένων παρέχει 513,000 ετικεταρισμένα γεωμετρικά πρωτότυπα σε 2,000 μοντέλα κτηρίων και περιλαμβάνει δύο συγκριτικές αξιολογήσεις για την κατάτμηση κτηρίων σε μορφή πλέγματος (mesh) και νέφους σημείων (point cloud). Επιπλέον, παρουσιάζεται ένα νευρωνικό δίκτυο γράφων που ετικετοδοτεί κτήρια σε μορφή πλέγματος, αναλύοντας τις χωρικές και δομικές σχέσεις των γεωμετρικών τους πρωτοτύπων. Αυτό το δίκτυο θεωρεί κάθε υποομάδα του πλέγματος (mesh subgroup) ως έναν κόμβο του γράφου και επωφελείται από σχέσεις, όπως γειτνίαση, συμμετρία και συγκράτηση, μεταξύ ζευγών υποομάδων. Η τελική ετικέτα του πλέγματος προκύπτει μέσω της μετάδοσης νευρωνικών μηνυμάτων (neural message passing) στο γράφο.

Η διατριβή ολοκληρώνεται με μια συζήτηση για μελλοντικές κατευθύνσεις έρευνας στην κατάτμηση σχήματος, όπως η αξιοποίηση αυτοεποπτευόμενων διαδικασιών προεκπαίδευσης, μοντέλων ανοιχτού λεξιλογίου και μάθηση δομής χωρίς επίβλεψη για περαιτέρω βελτίωση προσεγγίσεων, βασιζόμενες σε γράφους, για την κατάτμηση καθώς και την κατανόηση 3Δ σχημάτων και 3Δ σκηνών γενικότερα.

Abstract

The segmentation of 3D shapes into their constituent parts has been a long-standing problem in Computer Vision and Graphics. Recent breakthroughs in 3D Deep Learning led to numerous methods for learning effective representations useful for high-level shape processing tasks, including shape segmentation. Despite these significant advances, most methods rely on processing local geometric neighborhoods and often disregard broader context, such as structure, symmetries, and correspondences with other shapes that are often useful for discovering and extracting parts in geometric shape representations. Moreover, commonly used 3D datasets comprise mainly man-made objects with simple structure and lack large-scale models with high structural complexity.

This thesis presents graph-based neural methods that model complex structural and spatial relations within the same shape as well as across shapes in their graph representation to produce more consistent and accurate shape segmentations. Additionally, the thesis introduces the first publicly available large-scale dataset of annotated 3D building models. Buildings have more challenging structural complexity compared to objects in common benchmarks, thus, the dataset serves as a useful benchmark to evaluate segmentation algorithms on large-scale, structurally complex geometric data.

More specifically, the thesis proposes the following three graph-based architectures for shape segmentation. First, the thesis presents PB-DGCNN, a method capable of detecting part boundaries in 3D shapes. This method leverages a graph convolutional network that hierarchically captures local and non-local pairwise

point interactions to learn representations suitable for extracting candidate part boundaries. This boundary detector is class-agnostic. It can be trained to localize boundaries of either semantic parts or geometric primitives commonly used in 3D modeling. PB-DGCNN outputs candidate boundaries on point clouds represented as per-point probabilities; these boundary probabilities can be easily adopted as a pairwise term in a graph cuts formulation to extract boundaries demarcating parts. The method is demonstrated to improve shape segmentation by producing cleaner part boundaries.

The thesis next presents CrossShapeNet, a graph-based network that propagates point-wise feature representations across shapes within a collection to better synchronize them and improve their consistency for 3D shape segmentation. Specifically, the model introduces a cross-shape attention mechanism to enable interactions between a shape's point-wise features and those of other shapes. The mechanism assesses both the degree of interaction between points and also mediates feature propagation across shapes, improving the accuracy and consistency of the resulting point-wise feature representations for shape segmentation. Moreover, a shape retrieval measure is proposed, which selects suitable shapes for cross-shape attention operations for each test shape. Compared with previous methods, CrossShapeNet achieves the highest segmentation performance in terms of mean Part IoU in PartNet, a challenging benchmark for fine-grained part segmentation on man-made objects.

Finally, the thesis introduces BuildingNet, the first publicly available large-scale dataset of annotated 3D building models whose exteriors and surroundings are consistently labeled. The dataset provides 513K annotated mesh primitives across 2K building models and includes two evaluation benchmarks for mesh and point

cloud segmentation. Moreover, a graph neural network is presented that labels building meshes by analyzing spatial and structural relations of their geometric primitives. This network treats each mesh subgroup as a graph node, and takes advantage of relations, such as adjacency, symmetry, and containment, between pairs of subgroups. The final mesh labelling is achieved through neural message passing in the graph.

The thesis concludes with a discussion for future research directions in shape segmentation, such as leveraging self-supervised pre-training procedures, open-vocabulary models, and unsupervised structure learning for further improving graph-based approaches for segmentation as well as 3D shape and scene understanding in general.

Acknowledgements

Upon embarking on this PhD journey six and a half years ago, no one could have prepared me for all the challenges that lay ahead. Starting as a novice researcher, I was fortunate to have two astounding academics by my side to guide me through this process, encourage me during the lows, and provide me with constant support. Their vast knowledge of the field, keen eye for detail, and unwavering determination for excellence are qualities every young researcher should aspire to. My deepest appreciation goes to my advisors, Evangelos Kalogerakis and Melinos Averkiou, for their support, guidance, and inspiration throughout this doctoral pursuit. I hope that I have lived up to their standards; that would be my greatest accomplishment.

To my fellow lab members, Pratheba Selvaraju, Dmitry Petrov, Siddhant Garg, and Gopal Sharma, your collaboration and shared passion for research have enriched my experience immeasurably.

To my beloved wife, Theodora, your patience, understanding, and encouragement sustained me during the most demanding periods of this journey. I am truly fortunate to have you by my side.

Lastly, to my parents, your boundless love, encouragement, and sacrifices have always propelled me forward. This achievement is as much yours as it is mine.

This thesis represents the culmination of years of hard work, dedication, and support from all of you. I am immensely grateful for the invaluable contributions each of you has made to my life and academic journey.

Dedication

To my loving wife Theodora for her enduring support,

and to our precious son Sotiris who fills our lives

with endless joy and wonder.

Thesis Contributions

This thesis is founded on the knowledge acquired by the author's involvement in the authorship of the following journal articles and conference papers:

Journal Articles

[1] Marios Loizou, Melinos Averkiou, and Evangelos Kalogerakis, "Learning Part Boundaries from 3D Point Clouds", *Computer Graphics Forum (also in the Proceedings of the Symposium on Geometry Processing)*, vol. 39, no. 5, 2020.

[2] Marios Loizou, Siddhant Garg, Dmitry Petrov, Melinos Averkiou, and Evangelos Kalogerakis, "Cross-Shape Attention for Part Segmentation of 3D Point Clouds", *Computer Graphics Forum (also in the Proceedings of the Symposium on Geometry Processing)*, vol. 42, no. 5, 2023.

Conference Proceedings

[3] Pratheba Selvaraju, Mohamed Nabail, Marios Loizou, Maria Maslioukova, Melinos Averkiou, Andreas Andreou, Siddhartha Chaudhuri, and Evangelos Kalogerakis, "BuildingNet: Learning to Label 3D Buildings", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The importance of 3D shape segmentation lies in its core role in understanding and interpreting complex 3D environments, crucial for advancements in computer vision and graphics, robotics, and interactive technologies. Shape segmentation deals with the decomposition of 3D models into labeled semantic parts or geometric primitives. Accurate segmentation enables the analysis and manipulation of 3D shapes, facilitating applications in autonomous navigation [131, 260], medical diagnosis [135, 253] augmented reality [233, 68], and architectural design [45, 147].

The need for understanding parts in geometric shape representations is greater than ever. Due to the developments over the past decade of 3D acquisition technologies such as LiDAR sensors, 3D scanners and RGB-D cameras [67, 162, 185], we are witnessing an explosion of real-world 3D point cloud datasets [37, 4, 216, 27, 203, 183, 69, 9, 212, 76]. Accurately segmenting the point clouds into objects and parts is crucial for automatic recognition, shape and scene understanding in general. At the same time, a largely increasing number of deep learning architectures have been proposed to process point clouds for shape understanding [172, 173, 231, 265, 174, 247]. Finally, with respect to other common shape representations, such as polygon meshes, we are similarly witnessing an increasing number of 3D mesh-based datasets, which are in turn useful as synthetic datasets to train computer vision algorithms and for 3D modeling tasks in graphics [28, 250, 249, 158, 103].

A driving force for the development of deep neural networks for processing 3D geometric representations lies on the success of Convolutional Neural Net-

works [57, 120] in computer vision tasks. Classic CNNs [109, 202, 72, 80, 143] exploit several inductive biases that arise from the structure of their input domain [234]. In this case, images are represented as structured multi-channel 2D grids, that exhibit fixed ordering and adjacency between their pixels. Methods that operate on this type of data need to account for the translation invariance, as in the case of object classification where the output prediction should be invariant of any shift transformations of the input image. This is achieved with global symmetric functions such as global pooling which is agnostic of such transformations. In the case of image segmentation, since the output segmentation mask needs to be shifted accordingly with the input, models need to capture geometric priors such as translation equivariance. In CNNs this is achieved by leveraging shared weights in the form of convolutional filters that respect the locality property of the image, where adjacent pixels share similar characteristics. Moreover, after each convolution layer, local pooling is applied to adjacent regions of the image resulting in a coarsened grid. This effectively increases the receptive field of each layer while it enables convolution filters to capture fine and coarse details at different scales of the image, by simultaneously retaining the number of parameters constant at each scale.

Extending these mechanisms to other input domains, such as 3D point clouds, is not trivial. In contrast to 2D images, point clouds are represented as unstructured and unordered sets [19]. Methods that operate on this type of data need to be permutation invariant, i.e, their output prediction should remain unaffected by any permutation of the ordering of the input point set. The first point-based deep architectures [172, 258] showed that is possible to process such unordered sets by first applying a learnable transformation (e.g., in the case of PointNet [172] this trasformation is implemented by a shared Multi Layer Perceptron [182]) to each element of the set independently, and then aggregating the learned representations by global pooling, which imposes permutation equivariance over the input. While these methods have revolutionized the field of geometric deep learning, they overlook spatial and geometric relationships among input elements due to the absence of inherent connectivity within the data. Naturally, the spatial relationships of a point cloud can be modeled by a graph, whose nodes represent individual points and edges are formed between points capturing spatial and geometric relationships between them. This urges the need to devise neural networks

that can operate on graphs, which again features permutation invariance, not only at level of nodes but at level of edges as well. Formally, this type of neural architectures are called Graph Neural Networks (GNNs) [19] which are currently the cornerstone of geometric deep learning. By design these architectures exhibit the same traits as CNNs, such as permutation equivariance and locality by employing shared learnable filters in local neighborhoods modeled by a nearest- neighbor graph (nn-graph). In addition, they can explicitly capture long-range interactions through edges connecting non-local neighborhoods, or alternatively through the use of local pooling and coarsening of the input graph. Permutation invariance can also be achieved through global pooling operations (a more comprehensive design blueprint of Graph Neural Networks and geometric deep learning can be found in [19]).

Despite the advances in the field geometric deep learning and more specifically in the area of shape segmentation, there are several shortcomings that most approaches do not address. The first is that segmentation methods often predict part labels to individual points. This widely adopted approach usually results in erroneous segmentations specifically near the boundaries of parts where the predicted part probabilities are more unstable. For example, consider the back of the chair and the vertical bars of the frame that connect the back with the rest of the chair shape. These two distinct semantic parts lie in close spatial proximity and are usually co-planar. A neural network might be able, with high confidence, to assign the correct label to points that lie further away from the boundaries of these two parts. But this confidence diminishes for points that are near the boundaries, especially in the case where each point in represented only by its 3D Euclidean coordinates (see Figure 1.1 left). Even if the point normals are available, due to the co-planarity between these parts, most methods will not be able to correctly distinguish points that lie in these boundary areas. Thus, segmentation artifacts will appear since the two parts will start to bleed into each other.

Another crucial factor for shape understanding is learning effective point-wise representations in the first place. There has been a lot of research in developing deep neural architectures to learn point-wise representations of shapes through convolution and attention layers, useful for performing high-level tasks, such as shape segmentation. The common denominator of these networks is that they output a representation for each shape point by weighting and aggregating repre-

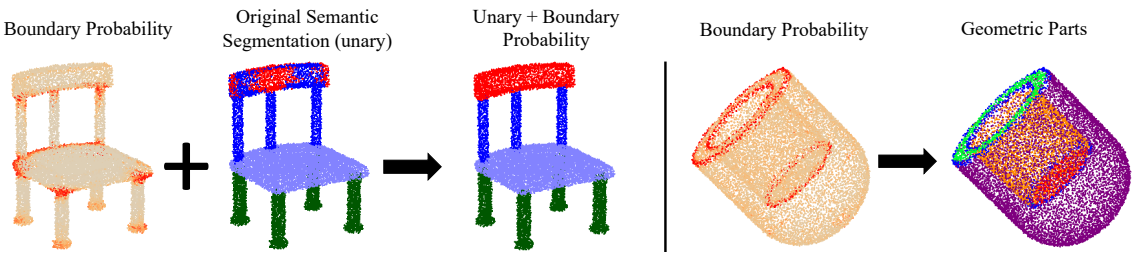

Figure 1.1: The PB-DGCNN method predicts part boundaries in 3D point clouds using a graph convolutional network which outputs a probability per point to lie on a boundary between parts in a 3D shape. *Left:* The output probability per point can be used in pairwise terms to improve graph-based semantic segmentation methods by localizing boundaries between semantic parts. *Right:* It can also be used in the geometric decomposition of point clouds into regions enclosed by sharp boundaries detected by our method.

sentations and relations with other points within the same shape. This limitation prevents them from fully exploiting relations between points and parts across different shapes and producing more synchronized feature representations for more consistent segmentation results.

Lastly, most methods for shape segmentation are primarily designed for simple man-made objects with relatively simple structure. While they showcase good segmentation performance on this type of data, they are less successful in more complex, large-scale shapes such as 3D buildings. These methods struggle to explicitly analyze and leverage the structural relationships inherited in architectural data, thus presenting a significant challenge for shape segmentation in this context.

Addressing these challenges necessitates a diverse approach rather than a singular methodology. The thesis develops various strategies to address these challenges in the context of graph-based networks. Each strategy targets a specific limitation within the segmentation process, by providing a targeted solution to each challenge.

1.2 Overview and Contribution

Each method introduced in this thesis represents a standalone approach to segmentation with a distinct flavour, yet all of the methods share a key characteristic: they all rely on a graph-based architecture either at the shape level, as in the

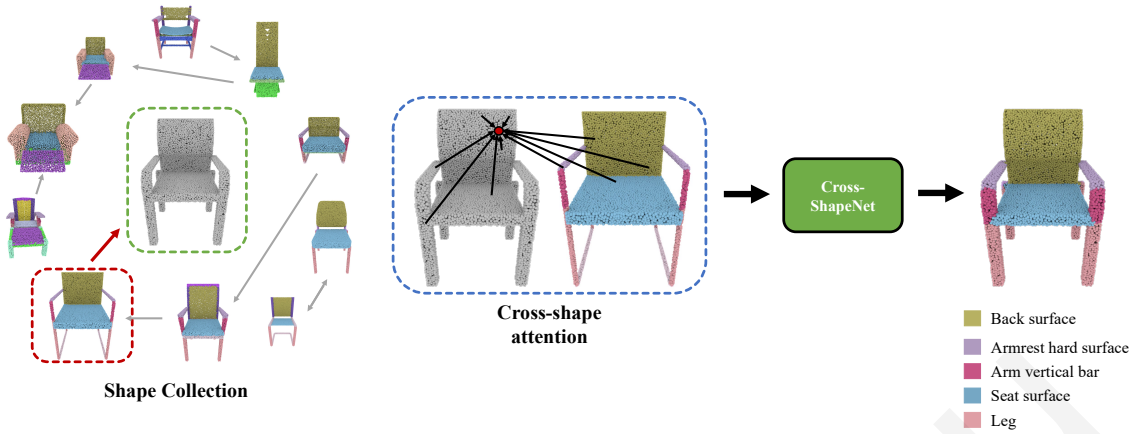

Figure 1.2: *Left:* Given an input shape collection, our method constructs a graph where each shape is represented as a node and edges indicate shape pairs that are deemed compatible for cross-shape feature propagation. *Middle:* Our network is designed to compute point-wise feature representations for a given shape (grey shape) by enabling interactions between its own point-wise features and those of other shapes using our cross-shape attention mechanism. *Right:* As a result, the point-wise features of the shape become more synchronized with ones of other relevant shapes leading to more accurate fine-grained segmentation.

case of detecting part boundaries or segmenting 3D buildings, or at the level of an entire collection of shapes, enabling the propagation of point-wise embeddings across different shapes. As already stated in the previous section, the very nature of 3D shapes requires the development of neural architectures that operate on graphs, since the latter can model spatial and structural relationships that are inherited in this type of data. Furthermore, transferring point representations across shapes through a collection involves constructing a graph whose nodes represent individual shapes, while its edges establish connections between shapes deemed compatible for feature propagation.

First, in Chapter 3 a novel neural network is presented, called PB-DGCNN (Probabilistic Boundary-Dynamic Graph Convolutional Neural Network), which is specifically designed for identifying part boundaries within 3D shape point clouds. This method utilizes a graph convolutional network to assign a probability to each point, indicating its likelihood of being located at a boundary that separates two or more parts in a 3D shape. This boundary detector is quite generic, as it can be trained to localize boundaries of semantic parts or geometric primitives commonly

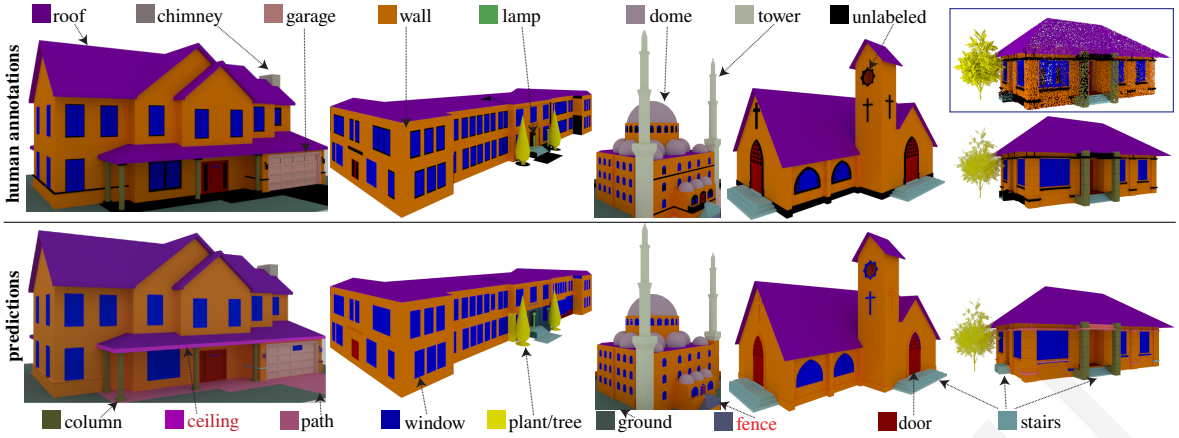

Figure 1.3: *Top:* We introduce a dataset of 3D building meshes with annotated exteriors. *Bottom:* We also present a graph neural network that processes building meshes and labels them by encoding structural and spatial relations between mesh components. *Blue box:* Our dataset also includes a point cloud track. Examples of erroneous network outputs are in red text.

used in 3D modeling (see Figure 1.1). Experiments on geometric and semantic segmentation datasets demonstrate that PB-DGCNN can extract more precise and accurate boundaries that are closer to the ground truth ones, outperforming existing alternatives. Additionally, an application of this method to fine-grained shape segmentation is presented, where it demonstrates significant improvements in terms of part labeling performance.

Next, Chapter 4 introduces a cross-shape attention mechanism that enables interaction and propagation of point-wise feature representations across shapes of an input collection. In this architecture, the representation of a point in a shape is learned by combining representations originating from points in the same shape as well as other shapes, as illustrated in Figure 1.2 (middle). The rationale for such an approach is that if a point on one shape is related to a point on another shape e.g., they lie on geometrically or semantically similar patches or parts, then cross-shape attention can promote consistency in their resulting representations and part label assignments. We leverage neural attention to determine and weigh pairs of points on different shapes. We integrate these weights in our cross-shape attention scheme to learn more consistent point representations for the purpose of semantic shape segmentation.

Finally, Chapter 5 presents BuildingNet, the first publicly available large-scale dataset of annotated 3D building models whose exteriors and surroundings are

consistently labeled. The dataset provides 513,000 annotated mesh primitives across 2,000 building models (examples are depicted in Figure 1.3). We include a benchmark for mesh and point cloud labeling, and evaluate several mesh and point cloud labeling networks. These methods were developed primarily for smaller single objects or interior scenes and are less successful on architectural data. In addition, we introduce a graph neural network that labels building meshes by analyzing spatial and structural relations of their geometric primitives. Our GNN treats each subgroup as a node, and takes advantage of relations, such as adjacency and containment, between pairs of nodes. Neural message passing in the graph yields the final mesh labeling. Our experiments show that this approach yields significantly better results for 3D building data than prior methods.

Furthermore, this thesis in Chapter 2 provides a comprehensive overview of 3D point cloud and mesh processing methods that specifically target the problem of shape segmentation. It also includes an examination of feature curve and edge detection approaches, along with a discussion on relevant 3D datasets for single shapes, indoor scenes, and urban environments. The chapter also provides an analysis of graph-based neural network architectures proposed in this thesis by examining their alignment with the foundational principles of GNNs. Chapter 6 concludes with a discussion on promising future research directions in shape segmentation. It proposes leveraging self-supervised knowledge distillation for automatic extraction of coherent regions from 3D buildings, coupled with a pretext task to learn dependencies between them using a point transformer architecture. Additionally, it outlines a procedure for expanding Cross-ShapeNet to 3D scenes by incorporating recent open-vocabulary approaches.

Chapter 2

Related work

This chapter presents previous approaches related to shape segmentation. While the methods presented in this thesis are predominantly deep learning-based, we begin with an overview of early shape segmentation techniques to ensure completeness (Section 2.1). Subsequently, we present a comprehensive analysis of 3D deep learning methods for processing point clouds (Section 2.2) and highlight key differences with the PB-DGCNN and Cross-ShapeNet methods that are outlined in Chapter 3 and Chapter 4, respectively. Furthermore, we explore previous techniques focused on edge and feature curve detection in point clouds (Section 2.3), which are related to PB-DGCNN.

Moreover, we survey neural networks specifically designed for 3D mesh comprehension and building segmentation tasks (Sections 2.4 and 2.5). Concurrently, we catalog segmentation datasets featuring 3D shapes, indoor scenes, and urban environments (Section 2.6) to underscore the importance of datasets such as BuildingNet, as detailed in Chapter 5. Lastly, we provide an analysis of the graph-based architectures employed in the methods outlined in this thesis, placed side by side with the design blueprint of Graph Neural Networks as introduced in [19] (Section 2.7). This examination aims to highlight the alignment and divergence of our approaches with the foundational principles outlined in the literature.

2.1 Early shape segmentation methods

Early shape segmentation methods rely on classic clustering methods and handcrafted features. First, we begin with a brief overview of classic shape segmentation techniques designed for 3D point clouds, following the taxonomy of Nguyen

and Le [162]. Region-based methods utilize neighborhood information to merge adjacent points that share similar characteristics into isolated regions methods of arbitrary shape. The seminal work by Besl and Jain [14] used variable-order surface fitting and was validated on range images. An initial coarse segmentation is extracted, using the surface curvature sign to assign a surface type label to each pixel. Then, an iterative refinement step is conducted using a region growing strategy with variable-order surface fitting. Koster and Spann [108] introduced a bottom-up hierarchical region-growing approach, which utilizes the mutual inlier ratio (MIR) of adjacent regions as a decisive merging factor. In Rusu *et al.* [187], the authors explored extracting object maps from 3D indoor household environments represented as point clouds. Initially, a geometrical mapping step is conducted to obtain a uniformly sampled point set with extracted geometric features like surface normals and curvature. Following this, a functional mapping procedure generates 3D object maps through a region-based segmentation process, guided by the previously extracted features.

Attribute-based methods, in contrast, employ clustering techniques that leverage point cloud attributes. For instance, methods such as [53, 54] utilized a clustering algorithm to segment airborne laser scanning data by incorporating attributes like point coordinates, tangent planes, and height differences. Unlike previous approaches that focused on specific surface types, attribute-based methods demonstrate the ability to capture more complex shapes.

Other methods utilize geometric primitive fitting for point cloud segmentation. Vosselman *et al.* [219] employ the Hough transform [49] to detect 3D planes and a two-step process for extracting cylinders based on point normals. Schnabel *et al.* [190] adopted the RANSAC algorithm [55] for estimating primitive shapes like planes, spheres, cylinders, cones, and toruses with a localized sampling approach. Building upon this, Li *et al.* [127] introduce a method that learns structural relations among locally fitted primitives, facilitating a global alignment procedure for all primitives. Gelfand and Guibas [58] propose a unique approach focusing on detecting slippable shapes, which possess rigid motions allowing transformed versions to slide along the original shape without gaps.

Lastly, graph-based methods represent point clouds as graphs, with points as nodes and edges connecting neighboring points. Golovinskiy and Funkhouser [61] construct a nearest-neighbors graph and utilize a min-cut approach for foreground-

background extraction. Strom *et al.* [207] adapt the work of Felzenszwalb and Huttenlocher [52] for segmenting colored LIDAR scans. Additionally, several approaches leverage probabilistic graphical models such as Conditional Random Fields [111] and Markov Networks [44], for labeling points [186] or segmenting point clouds from urban environments [191].

Another popular approach in shape segmentation is based on mesh representation. The methods listed here were quantitatively compared in [32]. Initially, Shlafman *et al.* [201] proposed a surface decomposition method utilizing the K-means clustering algorithm. This involved selecting k representative faces, ensuring maximal spatial proximity, and assigning faces to each representative based on geometric criteria to ensure coplanar faces in close proximity belonged to the same segment.

Katz *et al.* [93] introduced a pose-invariant hierarchical mesh segmentation method. They transformed the mesh using multi-dimensional scaling and detected feature points to guide segmentation. The core component was extracted using spherical mirroring, while the rest segments where obtained by subtracting the core component from the mesh. Then, a cut refinement step was employed to smooth out boundaries between segments. This hierarchical process continued until termination conditions were met. Attene *et al.* [5] introduced another hierarchical mesh segmentation algorithm that leverages primitive fitting. This method adopts a bottom-up approach, initially assigning each face to a single cluster. Next, a binary tree of clusters is constructed, with adjacent clusters merged if they are better approximated by a primitive compared to all other pairs. This merging process continues until a user-defined number of segments is reached.

Lai *et al.* [113] proposes a two-phase procedure for mesh segmentation. Initially, faces are over- segmented based on the highest probability of reaching them through a random walk on the mesh dual graph. Then, segments are hierarchically merged using relative lengths of intersections and total perimeters until reaching a user-defined segment count. Golovinskiy and Funkhouser [60] introduced two hierarchical clustering algorithms. In the first, Normalized cuts, segments are merged based on area-normalized cut cost, promoting small boundaries along concavities while maintaining similar areas across segments. The second, Randomized cuts, uses a randomized minimum cuts to guide segmentation boundary placement, terminating when reaching the user-defined segment count.

Shapira *et al.* [194] introduced a segmentation method based on the Shape Diameter Function (SDF), measuring volume diameters around surface points. Initially, they used a Gaussian Mixture Model to assign each face a probability vector for SDF cluster membership. Then, they refined segmentation using the alpha expansion graph-cut algorithm [18], considering probabilistic vectors, boundary smoothness, and concaveness.

A notable exception to previous works was presented by Kalogerakis *et al.* [89], offering an approach for segmenting and labeling parts within 3D meshes. Inspired by the joint image segmentation and recognition paradigm in computer vision, they adopt a data-driven strategy leveraging a graphical probabilistic model [111] to simultaneously segment and recognize semantic parts of meshes. Their method constructs an objective function based on unary terms derived from local surface descriptors, providing cues for face labeling, and pairwise terms that penalize misclassifications of neighboring faces to smooth out segment boundaries. This objective function is learned from a collection of human-annotated meshes, demonstrating that mesh segmentation algorithms can benefit significantly from data-driven learned models.

2.2 3D deep learning for point clouds

Here, we provide an overview of the related work on 3D deep learning for point clouds. Over the past few years, various types of neural networks have emerged for processing point clouds. The pioneering works of Qi *et al.* [172] and Zaheer *et al.* [258] introduced permutation-invariant learned transformations for directly processing unordered point sets. Subsequently, numerous studies investigated more sophisticated point aggregation mechanisms to better model spatial point distributions [173, 123, 205, 118, 196, 140, 43, 262, 146, 264, 174]. Alternatively, point clouds can be processed through various projection-based approaches. One approach is to convert point representations into volumetric grids [239, 154, 37, 177, 200, 141] and processes them utilizing 3D convolutions. Instead of uniform grids, hierarchical space partitioning structures, such as kd-trees, octrees, or lattices, can be utilized to define regular convolutions [179, 101, 224, 225, 208, 226]. Another approach involves projecting point clouds onto local views and treating them as regular grids for processing with image-based convolutional networks. This method was intro-

duced by Su *et al.* [209] and has been followed by subsequent methods for tasks such as object classification [171], which fuse volumetric and multi-view representations, and shape segmentation [88, 81]. Additionally, networks can incorporate point-wise convolution operators to directly process point clouds [126, 79, 241, 136, 64, 6, 73, 227, 245, 237, 107, 213]. On the other hand, shapes can be treated as graphs by connecting each point to other points within neighborhoods in a feature space. Graph convolution and pooling operations can then be performed either in the spatial domain [231, 199, 137, 115, 220, 262, 116, 122, 86, 244, 223, 70], or spectral domain [251, 16, 20, 17, 159]. Attention mechanisms have also been investigated to modulate the importance of graph edges and point-wise convolutions [242, 244, 223, 256].

Most previous approaches primarily address the problem of semantic segmentation. A number of methods have also been proposed to perform geometric decompositions of point clouds based on convexity analysis [87, 206, 42], primitive fitting [124, 127, 268, 197], graph cuts [61, 99], and clustering [15, 263]. In both semantic segmentation and geometric decomposition scenarios, part boundaries often tend to become fuzzy and noisy (see Figure 1.1 - left). To improve the quality of segmentation and align boundaries with underlying surface feature curves, such as creases, some methods employ simple geometric criteria, most commonly normal differences [169, 92, 206], within pairwise terms modeling the probability of boundaries between points. Similar pairwise terms have also been used in mesh segmentation approaches [94, 60, 89, 87]. In contrast, the PB-DGCNN method learns a pairwise term indicating the existence of part boundaries directly on 3D point clouds. As we show in our experiments, our learned boundaries are more accurate and are able to improve the quality of semantic segmentation and geometric decomposition to a larger degree compared to other alternatives.

Finally, several recent works [265, 66, 51, 155, 255, 240, 166, 114, 247, 192, 30, 238] introduced a variety of transformer-inspired models for point cloud processing tasks. Moreover, graph neural network approaches have been shown to model non-local interactions between points within the same shape [231, 122, 244, 70]. None of the above approaches have investigated the possibility of extending attention across shapes, in order to model point-wise interactions within a pair of shapes. A notable exception are the methods by Wang *et al.* [223] and Cao *et al.* [22] that propose cross-attention mechanisms across given pairs of point cloud instances

representing different transformations of the same underlying shape for the specific task of rigid registration. Cross-ShapeNet instead introduces cross-attention across shapes within a large collection without assuming any pre-specified shape pairs. Our method aims to discover useful pairs for cross-shape attention and learns representations by propagating them within the shape collection. Moreover, it shows that the resulting features yield more consistent 3D shape segmentation than several other existing point-based networks.

2.3 Feature curve and edge detection on 3D shapes

The PB-DGCNN approach is related to learning methods for detecting edges, or feature curves on 3D shapes. This is because part boundaries often coincide with surface feature curves, such as creases, ridges and valleys. Detecting such feature curves relies on geometric features, such as curvature extrema or normal discontinuities that can be detected on meshes, RGB-D data or point clouds [165, 10, 33, 23, 106, 90, 210, 82], however, their extraction often depends on hand-tuned procedures. Most similar to PB-DGCNN, EC-Net [254] attempts to learn edges on point clouds using a deep architecture operating on isolated point cloud regions (patches). Our method instead trains a neural network that operates directly on the whole point cloud encoding both local and global structure without any patch extraction or pre-processing. In our experiments, we show that our approach is much more accurate for part boundary detection compared to EC-Net even when the latter is trained on the same dataset as our method.

2.4 Deep nets for 3D mesh understanding

A few recent neural architectures have been proposed for processing meshes. Some network directly operate on the mesh geometric or topological features [153, 71, 112, 192], spectral domain [17, 159, 251, 175], while others transfer representations learned by other networks operating, e.g., on mesh views or voxels [88, 229, 110]. BuildingNet is complementary to these approaches. It is specifically designed to process meshes with pre-existing structure in the form of mesh components (groups of triangles), which are particularly common in 3D building models. CRFs and various grouping strategies with heuristic criteria have been proposed to ag-

gregate such components into labeled parts [229]. Our method instead uses a GNN to label components by encoding spatial and structural relations between them in an end-to-end manner. From this aspect, our method is also related to approaches that place objects in indoor scenes using GNNs operating on bounding box object representations with simple spatial relations, [267, 222], and GNN approaches for indoor scene parsing based on graphs defined over point clusters [117]. Our GNN instead aims to label mesh components represented by rich geometric features, and captures spatial and structural relations specific to building exteriors.

2.5 3D building mesh segmentation and labeling

There has been relatively little work in this area. Early approaches for semantic segmentation of buildings relied on shallow pipelines with hand-engineered point descriptors and rules [214, 152]. A combinatorial algorithm that groups faces into non-labeled components spanning the mesh with high repetition was proposed in [41]. A user-assisted segmentation algorithm was proposed in [40]. Symmetry has been proposed as a useful cue to group architectural components [102, 157]. BuildingNet instead aims to label 3D building meshes with a learning-based approach based on modern deep backbones for extracting point descriptors. It also incorporates repetitions as a cue for consistent labeling, along with several other geometric and structural cues.

2.6 3D segmentation datasets

Here, we explore various 3D segmentation datasets with different levels of granularity, to highlight the necessity for datasets containing annotated 3D buildings, such as BuildingNet.

2.6.1 3D shape semantic segmentation datasets

Existing datasets and benchmarks for 3D shape semantic segmentation are limited to objects with relatively simple structure and small number of parts [32, 89, 77, 249, 158, 252]. The earliest such benchmark [32, 89] had 380 objects with few labeled parts per shape. More recently, Uy *et al.* [216] released a benchmark with 15K

scanned objects but focuses on object classification, with part-level segmentations provided only for chairs. The most recent and largest semantic shape segmentation benchmark of PartNet, introduced by Yu *et al.* [252] contains 27*K* objects in 24 categories, such as furniture, tools, and household items. However, even with PartNet's fine-grained segmentation, its categories still have a few tens of labeled parts on average. BuildingNet introduces a dataset for part labeling of 3D buildings, pushing semantic segmentation to much larger-scale objects with more challenging structure and several tens to hundreds of parts per shape.

2.6.2 3D indoor scene datasets

Another related line of work has introduced datasets with object-level annotations in real-world or synthetic 3D indoor environments [78, 4, 163, 204, 27, 37, 125, 266, 56]. In contrast, BuildingNet focuses on building exteriors, a rather under-investigated domain with its own challenges. While an indoor scene is made of objects, which are often well-separated or have little contact with each other (excluding floors/walls), a building exterior is more like a coherent assembly of parts (windows, doors, roofs) i.e., a single large shape with multiple connected parts, including surroundings (e.g., landscape). Building exteriors share challenges of single-shape segmentation (i.e., segment parts with clean boundaries along contact areas) as well as scene segmentation (i.e., deal with the large-scale nature of 3D data). Buildings also come in a variety of sizes, part geometry and style [145], making this domain challenging for both shape analysis and synthesis.

2.6.3 3D urban datasets

With the explosion of autonomous driving applications, large-scale 3D point cloud datasets capturing urban environments have appeared [161, 69, 183, 9, 212]. These datasets include labels such as roads, vehicles, and sidewalks. Buildings are labeled as a single, whole object. The BuildingNet dataset contains annotations of building parts, which has its own challenges. The RueMonge14 dataset contains 3D building frontal facades captured from a street in Paris with 8 labels related to buildings [180]. Our buildings are instead complete 3D models with significantly more challenging diversity in geometry, style, function, and with more fine-grained part labels.

2.7 Blueprint for graph neural networks

In the work Bronstein *et al.* [19], a unified design blueprint for geometric deep learning is outlined. As discussed in Section 1.1, various geometric priors can be combined to learn stable representations of high-dimensional data, achieved through diverse transformations. One crucial aspect is the *symmetry* of an object, defined as a transformation preserving a certain property. Here, we define an object as a specific instance within an input domain, where a neural network operates. For 2D images, symmetry entails translation invariance, while for point clouds this translates to permutation invariance, due to their unordered nature. Although such transformations are adequate for understanding *global* symmetries, they may overlook *local* symmetries within objects. To capture these, *geometrically stable* transformations are required to preserve properties across local neighborhoods. Thus, transformations that respect translation or permutation equivariance are employed for images or point clouds, respectively. While reinforcing the global symmetry prior, these transformations operate independently across separate regions, lacking exploitation of multiscale structure. Hence, by imposing transformations that leverage *scale separation*, neural networks can capture long-range interactions across various object scales.

These geometric priors establish the necessary conditions for learning high-dimensional representations and guide the design choices for neural network architectures. Firstly, a *local equivariant map* is crucial for preserving local symmetries, while a *global invariant map* consolidates local features into a unified global representation. Additionally, a *coarsening operator* facilitates long-range correlations across varying scales. These three fundamental building blocks offer essential inductive biases for various geometric deep learning methods, tailored to their input domains.

In our context, where we focus on neural networks operating on graphs, we employ local permutation-equivariant transformations, implemented through shared MLPs across different regions of the input graph. We utilize local pooling layers, such as sum, mean, or maximum operations, to aggregate representations of local neighborhoods and downsample the graph, while also retaining permutation invariance within the local neighborhood. Finally, a global permutation-invariant

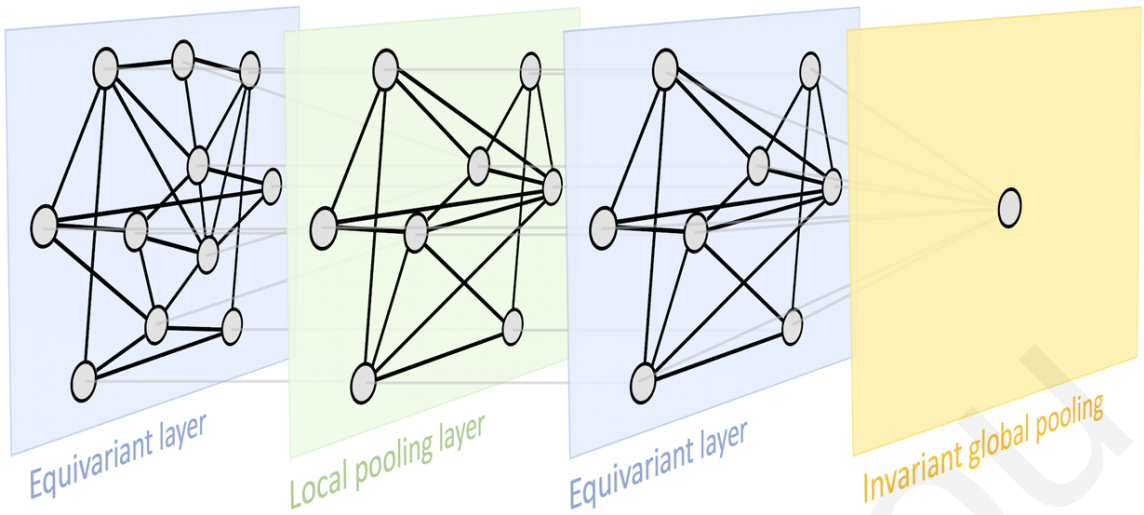

Figure 2.1: Geometric Deep Learning blueprint, exemplified on a graph. A typical Graph Neural Network architecture may contain permutation equivariant layers (computing node-wise features), local pooling (graph coarsening), and a permutation-invariant global pooling layer (readout layer). *This image is taken from [19].*

pooling operation combines all node representations to generate a global graph representation. Figure 2.1 illustrates a generic design blueprint of a GNN incorporating these principles.

Building upon the previous graph operations, we can formulate a generic GNN layer that computes a permutation-equivariant function. This is achieved through the application of shared permutation-invariant functions, typically realized by shared MLPs over local neighborhoods, followed by an aggregation function. Broadly, GNN layers can be classified into three distinct "flavours", based on the extent to which the layer transforms the neighborhood features [19] (Figure 2.2 illustrates the dataflow for the three flavours of GNN layers):

- **Convolutional flavour:** In this approach, features are directly aggregated with fixed weights, specifying the significance of the sender node to the receiver node's representation [39, 97, 236].
- **Attentional flavour:** Here, interactions among nodes are implicit, with a self-attention mechanism computing importance weights based on node features [218, 160, 261].
- **Message-passing flavour:** This approach computes vectors (or "messages")

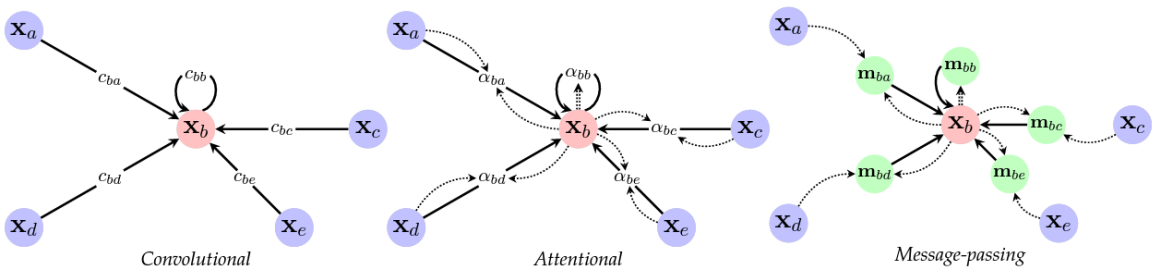

Figure 2.2: A visualisation of the dataflow for the three flavours of GNN layers. *Left-to-right:* **convolutional**, where sender node features are multiplied with a constant; **attentional**, where this multiplier is implicitly computed via an attention mechanism of the receiver over the sender; and **message-passing**, where vector-based messages are computed based on both the sender and receiver. *This image is taken from [19].*

> across neighborhood edges, with learnable "message" functions determining vectors between sender and receiver nodes. These vectors are then aggregated to the receiver node via message passing throughout the graph [59, 8, 96].

Each method presented in this thesis aligns with one of the GNN flavours outlined previously. In Chapter 3, we employ DGCNN [231], featuring GNN layers that adhere to the convolutional flavour. Here, each sender node is initially centered based on the centroid of its local neighborhood, and features are directly aggregated using shared weights across the point cloud graph. For more insights, refer to Section 3.1.

In Chapter 4, the Cross-ShapeNet method propagates features across shapes, by following the attentional flavour paradigm. We utilize self-shape attention representations of each shape to compute compatibility between pairs of shapes. This scalar value moderates the influence of sender shape point-wise representations on receiver shape features. For a detailed analysis of our approach, see Section 4.1.1.

Finally, in Chapter 5, we establish a graph over the building's mesh subgroups, representing the graph's nodes. Edges are established between subgroups, capturing various structural and spatial relationships. Edge representations are initially updated by leveraging the features of their endpoints (sender and receiver nodes) and are then aggregated to update the receiver node's features, adhering to the message-passing paradigm. Section 5.2 provides an in-depth analysis of our GNN on 3D buildings.

Chapter 3

Learning Part Boundaries from 3D Point Clouds

This chapter presents a neural network approach that learns to detect part boundaries in point clouds of 3D shapes[1]. Although there has been significant amount of research in detecting contours and object boundaries in natural images with neural networks [11, 12, 13, 26, 74, 105, 119, 132, 138, 139, 149, 198, 232, 243], detecting boundaries in 3D point clouds is largely an unexplored area. Despite the significant advances in the area of 3D deep learning for processing unstructured point clouds, most research has focused so far on assigning part tags to individual points. The resulting segmentations often suffer from artifacts at areas that lie near the boundaries of parts, since the point assignments become highly uncertain at these areas (see also Figure 1.1).

There is a number of technical challenges to overcome in developing an approach that addresses this problem. First, the notion of an object part is often ambiguous and usually depends on the task. For example, in semantic segmentation, parts follow label definitions (e.g., leg, back, seat for chairs), while for 3D modeling tasks, shapes are often modeled as collections of geometric primitives (e.g., spheres, cylinders, surfaces of extrusion, NURBS, and so on). We show that an effective boundary detector can be trained from semantic segmentation datasets to accurately localize boundaries of labeled parts, and also from shape datasets with segmented geometric patches. Second, boundaries are usually sparse; only

[1]The work presented in this chapter is also published in Computer Graphics Forum, vol. 39, no. 5, 2020. Project page: https://marios2019.github.io/learning_part_boundaries. This is the author's version of the work. The definitive version of the article was published in Computer Graphics Forum, vol. 39, no. 5, 2020, https://doi.org/10.1111/cgf.14078.

a small percentage of points in a point cloud lie near boundaries. During training, we employ a sampling procedure to gather a sufficient amount of boundary points for training, and use a classification loss function robust to the imbalance of the number of boundary versus non-boundary points. Furthermore, in contrast to semantic segmentation networks that often rely on points expressed in global coordinate frames, we found that learning features from points expressed in local frames aligned with surface normals are better suited for boundary extraction. The output of our method is probabilistic: it assigns a probability for each point belonging to a part boundary or not. We demonstrate pairwise terms that can easily adopt these probabilities within graph cuts formulations.

Our method draws inspiration from contour detection techniques used in object segmentation within natural images. Early approaches to contour detection in natural images relied on local gradient estimates [181, 170, 150, 100, 21]. With the appearance of the first datasets with manual annotations of object segmentations [3], machine learning approaches enhanced contour detection through learned classifiers [151, 47, 104, 176, 129, 48]. Most recent approaches achieve state-of-the-art performance on contour detection by training deep convolutional or recurrent neural networks [105, 11, 12, 13, 198, 139, 149, 26, 243, 232, 74, 138, 132, 119]. In contrast to the regular 2D grid structure of images, point clouds are unorganized and non-uniformly sampled. Our method adapts neural networks for point cloud processing and is trained on datasets for 3D semantic or geometric segmentation of shapes.

We conducted a number of experiments to validate our approach. First, we compare our extracted boundaries with annotated ones in geometric and semantic segmentation tasks. We found that the boundaries produced by our architecture are much closer to ground-truth ones compared to alternatives. For example, we observed that the error was reduced by 61.2% compared to the best alternative edge detector we adapted for our task (EC-Net [254]), measured based on Chamfer distance between detected and ground-truth boundaries in the ABC dataset [103]. We also show that our boundary detector, when combined with graph cuts, offers a small, but noticeable boost in terms the semantic segmentation performance: an increase of $+2.6\%$ in shape Intersection over Union (IoU), and $+0.5\%$ in part IoU on average in PartNet [158] compared to using a neural network (DGCNN [231]) that assigns tags to points without explicitly considering boundaries.

The contributions of the present chapter can be summarized as follows:

- a neural network module, called LocalEdgeConv (inspired by DGCNN), that operates on point cloud neighborhoods expressed in local frames (in constrast to global frames used in [231]). We found that this adaptation is more suitable for the task of 3D part boundary detection.
- a network training procedure that robustly samples and weights boundary data of either semantic parts or geometric primitives.
- a graph cuts formulation that uses our probabilistic boundary detector to improve semantic shape segmentation, especially near part boundaries.

3.1 Method

The PB-DGCNN architecture takes as input a point cloud $\mathbf{P} = \{\mathbf{p}_i, \mathbf{n}_i\}_{i=1}^{N}$, where $\mathbf{p}_i$ are 3D point coordinates and $\mathbf{n}_i$ are 3D normals, and outputs a scalar $b_i \in [0,1]$ for each point. The output b_i represents the probability for a part boundary to lie on the point i. Our architecture is shown in Figure 3.1.

3.1.1 Architecture

Our architecture, follows the concept of graph edge convolution (EdgeConv) introduced in the DGCNN network [231]. To implement edge convolution, a graph first needs to be formed over the point cloud. In the first EdgeConv layer, each point i is connected to its K neighbors in Euclidean space, where K is a hyper-parameter of the network. In the original DGCNN formulation, the first EdgeConv layer processes the input point coordinates $\mathbf{p}_i$ of each point i along with the coordinates of its neighbors $\{\mathbf{p}_j\}_{j \in \mathcal{N}_e(i)}$, where $\mathcal{N}_e(i)$ is the Euclidean neighborhood of the point i. The output representation for each point is computed as follows:

$$\mathbf{y}_i = \max_{j \in \mathcal{N}_e(i)} MLP(\mathbf{p}_i, \mathbf{p}_j - \mathbf{p}_i) \tag{3.1}$$

where MLP represents a learned Multi-Layer Perceptron operating on the above input feature vector of point coordinates and differences (concatenated and flattened). In this manner, each edge encodes the input coordinates at a point i along with the coordinates of its neighbors expressed relative to it. The max operator

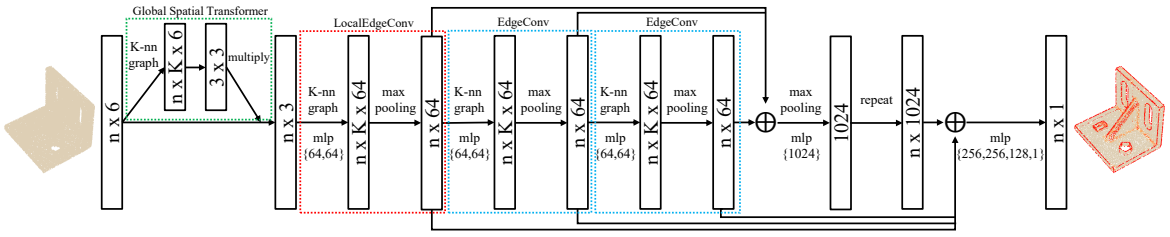

Figure 3.1: Architecture of our network (PB-DGCNN) for probabilistic boundary detection. It consists of three main blocks: the LocalEdgeConv layer, EdgeConv layers [231], and a Global Spatial Transformer. The LocalEdgeConv layer constructs a K-NN graph for each input point i in Euclidean space and expresses the coordinates of its K nearest neighbors in the local coordinate frame at point i. Then a feature transformation is applied to the edge features of the graph through a 2-layer MLP, and output representations are aggregated through max-pooling. These representations are further processed by the two EdgeConv layers that operate on the K-nn graphs constructed in the feature space of the previous layer. Finally, a global descriptor is aggregated and concatenated with the point-wise descriptors of the three previous layers, which are transformed through a 4-layer MLP, and the boundary probability is produced by a sigmoid function. The Global Spatial Transformer [85] operates on the Euclidean space of the input points and helps to align the point cloud to a canonical space.

(max pooling) is used to aggregate all edge representations per point and guarantees invariance to point and edge permutations.

3.1.2 LocalEdgeConv layer

The relative point coordinate differences $(\mathbf{p}_j - \mathbf{p}_i)$ help capturing local neighborhood structure in the original DGCNN. However, these differences are still expressed with respect to the global coordinate frame axes. As a result, if the local neighborhood is rotated, the input edge features to the MLP will change. In turn, the output of the MLP may also change. This effect might be desirable in the case of semantic segmentation, since changing the orientation of a part in a shape may change its functionality and its semantic label, especially for man-made objects (e.g. rotating a horizontal tailplane 90 degrees in an airplane would make it look like a vertical stabilizer). However, the part boundaries are more likely to remain unaffected by such local rotations e.g., one would still want to label points

between the fuselage and the tailplane, or the stabilizer, as boundaries. Thus, in our architecture, we make the following modification to the first edge convolution layer:

$$\mathbf{y}_i = \max_{j \in \mathcal{N}_e(i)} MLP\big(\mathbf{p}_i, \mathbf{R}_i^T(\mathbf{p}_j - \mathbf{p}_i)\big) \tag{3.2}$$

where $\mathbf{R}_i$ is a rotation matrix responsible for expressing the relative point coordinate differences in a local coordinate frame at point i (instead of a global one). Note that the transpose of the rotation is used to perform the coordinate transformation. The local frame is formed from the point normal $\mathbf{n}_i$ and two tangent vectors $\mathbf{u}, \mathbf{v}$ randomly selected on the tangent plane of the point i: $\mathbf{R}_i = [\mathbf{u}_i \ \mathbf{v}_i \ \mathbf{n}_i]$. Since the tangent vectors are chosen randomly, rotational invariance is not guaranteed, however, the use of the point normal decreases the variance of inputs that the network needs to handle. We call the above edge convolution of Equation 3.2 as LocalEdgeConv. Experimentally, we observed a significant improvement in boundary detection due to LocalEdgeConv (see our evaluation in Section 3.4). We note that we also experimented with using principal curvature directions as tangent directions, and also treating their sign ambiguity through max pooling, yet the gain was still smaller than LocalEdgeConv (see results section).

3.1.3 LocalEdgeConv layer with normals as features

Another variant of LocalEdgeConv we experimented with was to include point normals as additional input features to this layer. Specifically, we horizontally concatenate point positions and normals per point ($\mathbf{x}_i = [\mathbf{p}_i, \mathbf{n}_i]$), then transform them through a MLP in a local coordinate system:

$$\mathbf{y}_i = \max_{j \in \mathcal{N}_e(i)} MLP\big(\mathbf{x}_i, \mathbf{R}_i^T(\mathbf{x}_j - \mathbf{x}_i)\big) \tag{3.3}$$

In this manner, the network also considers differences of normal coordinates in a neighborhood around each point transformed in a local coordinate frame. We note that we also experimented with processing points together with normals as input to the original EdgeConv as first layer (instead of LocalEdgeConv). However, the gain was smaller compared to using LocalEdgeConv with normals as input features.

3.1.4 PB-DGCNN architecture

After using a LocalEdgeConv layer (with or without normals as additional features), our architecture stacks two EdgeConv layers (Figure 3.1) that sequentially process the representations extracted based on our local coordinate frames. At each EdgeConv layer, each point is connected its K nearest neighbors dynamically updated from the input feature space of the layer, as done in DGCNN [231]. The point-wise representations extracted from the LocalEdgeConv and the two EdgeConv layers are concatenated, then processed through a max pooling layer, which produces a global shape descriptor. The global descriptor is tiled and horizontally concatenated with the point-wise representations (Figure 3.1), so that the resulting point representations encode both local and global shape information. These are passed into a MLP, followed by a sigmoid transformation that outputs a boundary probability b_i for each point i.

3.2 Datasets

To train our network, we make use of datasets that provide shape segmentations. We made use of two datasets for training and evaluation: a geometric segmentation dataset and a semantic segmentation one, described below. We train and test our architecture on each dataset separately.

3.2.1 Geometric segmentation dataset

The ABC dataset [103] introduced a large repository of 3D geometric models, each defined by parametric surfaces and ground truth information on their decomposition into individual patches. This dataset is a good source to learn segmentation boundaries between geometric primitives and patches. Another advantage of this dataset is that the patch boundaries are provided in parametric curve format, which allows us to extract very accurate boundaries for training and evaluation.

Since our goal is to detect boundaries for input point clouds, we first convert the geometric models into point-sampled surfaces for training. Specifically, we first sample the surface of 3D models with $10K$ points based on Poisson-Disk sampling [50], to create an initial point cloud. Since it is rather unlikely to sample points

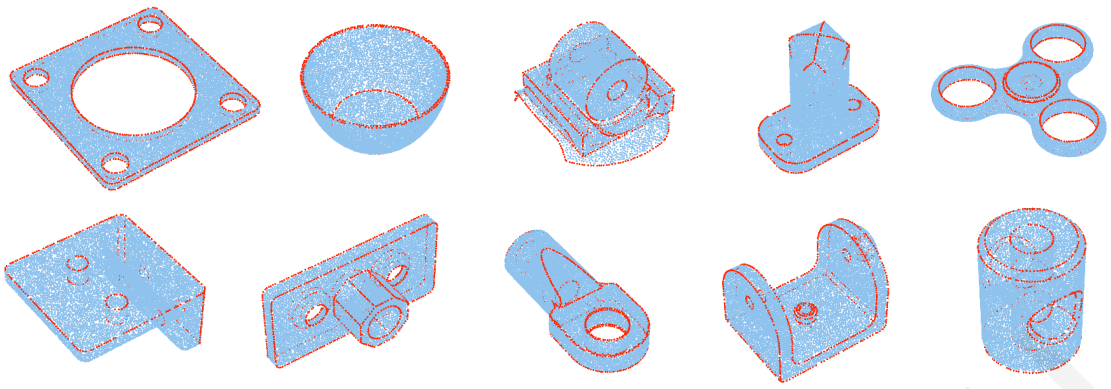

Figure 3.2: Marked (with red) boundaries on ABC point clouds for training.

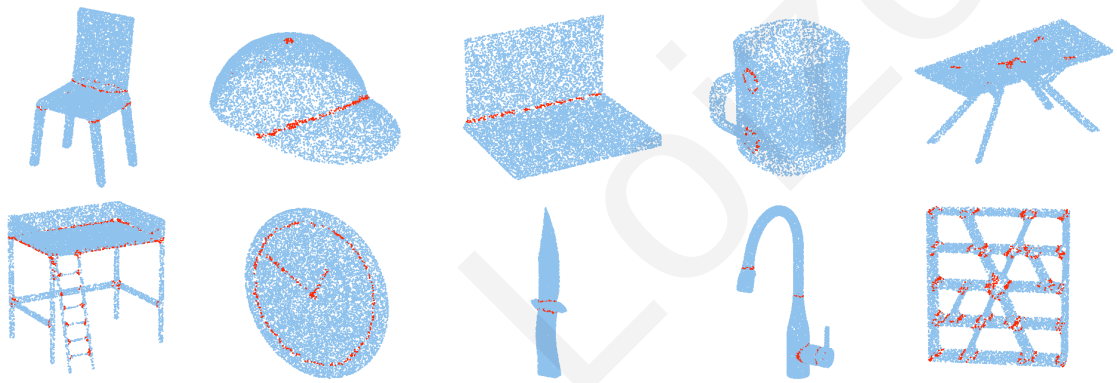

Figure 3.3: Marked boundaries on PartNet point clouds for training.

lying exactly on boundary curves with this sampling procedure, we perform a second pass where we randomly sample another $10K$ points along boundary curves, specifically based on their underlying parametric representation. Concatenating the surface point samples of the first pass with the boundary point samples of the second pass tends to create higher point cloud density near the boundary regions. To avoid this higher density bias during training, we perform a third pass where for each boundary point, we remove any surface samples within a distance equal to ϵ, which is computed by measuring the distance of each point sample of the first pass to its nearest neighbor, then setting it to the maximum distance over the point cloud. Finally, we observed that some ABC shapes sometimes contain adjacent patches of same local geometry (e.g., two adjacent planes forming a flat boundary), where the boundary between them can be ignored. We filtered out such boundaries. All shapes are centered in the origin and scaled so they lie inside the unit sphere.

The result of this procedure is the generation of a point cloud for each ABC shape with surface points carrying a binary label: boundary or non-boundary point. Figure 3.2 shows examples of such point clouds colored according to the binary label. We created a training set of 16,291 labeled point clouds from ABC based on the above procedure.

To monitor the training procedure, we also need a hold-out validation set. In addition, for evaluation, we need a test set. We gathered an additional set of 2,327 shapes for hold-out validation and 4,655 shapes for testing i.e., in total we had 23,273 point clouds from ABC, and a 70%-10%-20% proportion for training, validation and testing respectively. It is also important to note that the hold-out validation and testing point clouds are generated with Poisson point sampling from the original surfaces without adding boundary points (i.e., without the second and third pass used in training shapes). In this manner, we avoid biasing our testing procedure with point samples that are exactly at the geometric boundaries, and which may exhibit particular regular patterns due to their sampling from the underlying parametric representation of boundary curves. As discussed in our results section, the goal of our evaluation metrics is to detect boundaries up to a certain distance tolerance i.e., find points whose distance to the ground-truth parametric curve boundaries is up to distance equal to the maximum point sampling distance ϵ. Finally, to simulate noisy point clouds for validation and testing, we add isotropic Gaussian noise to point coordinates with $\mu = 0$ and $\sigma = 0.005$. Normals are also perturbed from their original direction, by an angle sampled from a normal distribution trimmed within an interval $[-3, 3]$ degrees.

3.2.2 Semantic segmentation dataset

To learn boundaries for semantic segmentation, we use the recent PartNet dataset [158]. This dataset provides hierarchical segmentations of 26,671 shapes into labeled parts in 24 categories. The shapes are provided in the form of polygon meshes split into parts according to their label. We use the segmentations from the last hierarchy level in each category (i.e., the "fine-grained" segmentations). To generate the training point clouds with boundaries, we follow the following procedure. First, we sample $10K$ points based on Poisson-Disk sampling. PartNet does not provide boundary curves. Furthermore, neighboring parts in PartNet meshes

are topologically disconnected from each other in their mesh representation, often inter-penetrate each other, or even do not touch each other. We instead mark as boundaries all points of triangles that have neighboring points labeled with a different part label within a radius equal to ϵ set to the largest distance between all-pairs of nearest neighboring point samples. We add the same noise profile in the validation and test shapes, as in our ABC dataset. Figure 3.3 shows examples of the resulting point clouds, colored according to the binary label. The boundaries are more fuzzy and spread compared to the ABC dataset, yet are still clearly indicating zones separating semantic parts. PartNet provides training, hold-out validation, and test splits, thus we follow the same splits in our case. For the validation and test point clouds, we only perform the first pass to randomly sample 10K points. As in the case of ABC dataset, the goal of our evaluation is to detect boundaries up to a certain distance tolerance.

3.3 Training procedure

To train our architecture, we use the marked boundary and non-boundary points from either of the above training datasets as supervisory signal. We treat the problem as binary classification, and we use binary cross-entropy as our loss function. However, since the number of boundary points is extremely small compared to the number of non-boundary ones (i.e., they represent less than 1% of the total points on average), we use a weighted cross entropy loss that emphasizes the error on boundary points more:

$$L = \sum_{s \in \mathcal{D}} \sum_{i=1}^{N_s} w_b \cdot \hat{t}_i \cdot \log(b_i) + (1 - \hat{t}_i) \cdot \log(1 - b_i) \tag{3.4}$$

where $\hat{t}_i = 1$ for marked boundary points and $\hat{t}_i = 0$ for non-boundary points, and w_b weights the cross-entropy terms for boundary points. Specifically, we set the weight according to the ratio of the number of non-boundary points and the number of boundary points: $w_b = (\sum_i [\hat{t}_i == 0]) / (\sum_i [\hat{t}_i == 1])$. In this manner, we penalize more misclassifications of boundary points. During training, as a form of data augmentation, and to also increase robustness, we add random noise in the points and normals (same noise distributions used in the validation and test sets of our datasets).

Implementation details. Training is done through the Adam optimizer [95] with learning rate 0.001, beta coefficients set to $(0.9, 0.999)$, batch normalization with momentum set to 0.5 and batch normalization decay set to 0.5 every 10 epochs. The batch size is set to 8 point clouds. The PB-DGCNN method is implemented[2] in TensorFlow [1].

3.4 Evaluation

We now discuss experimental evaluation of our method. First, we introduce evaluation metrics for part boundary detection, and present results on the ABC dataset for geometric boundary detection. Then we present an application of our method to semantic segmentation, and present evaluation on the PartNet dataset.

3.4.1 Evaluation metrics

Our evaluation metrics are inspired by the literature on line drawing and segmentation for 3D meshes. Cole *et al.* [35] introduced metrics that evaluate similarity of human-annotated line drawings with computer-generated ones based on precision and recall. Liu *et al.* [133] extended these metrics to include Intersection over Union (IoU). Our part boundaries can be thought of as point-sampled lines in 3D, thus we also use precision, recall, and IoU inspired by these works. Chen *et al.* [32] introduced various metrics for evaluating segmentation for 3D meshes. In the case of boundaries, they propose cut discrepancy that measures distances of annotated and predicted boundaries on the surface. Following the above works, we introduce the following metrics for the evaluation of boundaries:

Precision is defined as the fraction of predicted boundary points in a point cloud that are "near" any annotated boundary. The proximity is computed by measuring Euclidean distance of points to boundary curves in ABC dataset, or boundary point samples in PartNet. The definition of "near" requires a distance threshold indicating tolerance to small errors. We define this tolerance as the maximum point sampling distance ϵ (largest distance between all-pairs of nearest neighboring point samples per point cloud). We also examine performance under varying levels of

[2]The implementation is available at https://github.com/marios2019/learning_part_boundaries.

tolerance (multiples of ϵ).

Recall is defined as the fraction of annotated boundary points that are "near" any predicted boundary point. We follow the same definition of nearness as above. In the ABC dataset, we densely sample the parametric boundary curves to evaluate recall.

F1-score is the harmonic mean of precision and recall, often used to combine them both in one metric.

Boundary IoU (bIoU) is the Intersection over Union that measures "overlap" between annotated boundaries and predicted ones. A boundary and predicted point "overlap" if they are near to each other, based on the same definition of nearness as above.

Chamfer distance (CD) measures Euclidean distance from annotated boundary samples to nearest predicted boundary points, and vice versa (i.e., we use the symmetric Chamfer distance).

It is important to note that in order to evaluate the above metrics, the probabilistic boundaries must be binarized first. In the ABC dataset, we use thresholding (i.e., a point becomes boundary if its probability is above a threshold). To select the threshold, we perform dense grid search in our hold-out validation dataset, and select the value that minimizes the Chamfer Distance. In the PartNet dataset, the probabilistic boundaries are used in a pairwise term in graph cuts - the points crossed by the cut are marked as boundaries. Finally, we note that the metrics are computed for each test point cloud shape, then averaged over the test shapes.

3.4.2 Geometric part boundary detection

We now discuss evaluation for detection of part boundaries between geometric primitives on ABC [103] based on the dataset described in Section 3.2. The primitives in ABC include plane, cone, cylinder, sphere, torus, surface of revolution or extrusion or NURBS patch. We compare our method with the edge detection network called EC-Net from [254]. The method was introduced for detecting edges on point clouds for 3D reconstruction. It upsamples the original point cloud, while we also producing a value per point corresponding to its distance to the nearest edge.

Model	Input Features		Metrics				
	position	normal	CD	bIoU	F_1	P	R
EC-Net	✓		4.9	52.7	64.6	88.5	50.9
	✓	✓	7.5	56.9	67.2	85.7	55.3
PB-DGCNN w/ EdgeConv	✓		3.0	81.6	85.2	89.8	81.1
	✓	✓	2.1	89.8	90.3	90.9	89.7
PB-DGCNN w/ LocalEdgeConv-curv	✓		2.6	85.2	88.0	91.3	85.8
	✓	✓	2.0	89.2	90.5	**91.8**	89.1
PB-DGCNN w/ LocalEdgeConv	✓		2.4	90.0	89.7	89.2	90.1
	✓	✓	**1.9**	**92.0**	**91.9**	**91.8**	**92.1**

Table 3.1: Boundary classification results on the ABC dataset (CD: Chamfer Distance - %, bIoU: Boundary IoU - %, F_1: F_1 score - %, P: Precision - %, R: Recall - %)

By thresholding the value, the method detects edges. We adapted their method for our task. We trained their method on our dataset, tuned their hyper-parameters (weights of losses) in our hold-out validation set, tuned the threshold for edge detection using hold-out validation to optimize Chamfer distance, and used the same augmentation as in our method. Since their method is based on sampling individual patches from the point cloud, we experimentally verified that the sampled patches fully cover the ABC shapes by setting their number to 50.

Table 3.1 reports our five evaluation metrics for EC-Net and our method. We evaluated two version of EC-Net: one with points only as input features (EC-Net w/o normals), and another with points and normals as input features (EC-Net w/ normals). As indicated by all metrics, our method produces boundaries that are much closer to the annotated ones compared to EC-Net. For example, the EC-Net without normals has 2.04 times higher error than our method without normals (see PB-DGCNN w/ LocalEdgeConv w/o normals) in terms of Chamfer Distance, and 2.58 times higher error than our method with normals (see PB-DGCNN w/ LocalEdgeConv w/ normals). The EC-Net with normals seems to have even higher error in Chamfer Distance, yet better Recall and IoU profile than EC-net without normals. It seems that the EC-Net with normals makes better predictions near ground-truth boundaries, but also produces additional boundaries away from

ground-truth ones, which results in higher Chamfer Distance. In any case, our PB-DGCNN with LocalEdgeConv offers much better performance compared to both versions of EC-Net according to all our evaluation metrics.

Figure 3.4 provides a visual demonstration for some example point clouds from ABC. We find that the EC-Net boundaries are highly noisy and inconsistent, while ours tend to agree with the ground-truth more.

Application to geometric decomposition. We found that our boundaries can be used for segmentation of several ABC shapes using a simple flood-filling, watershed segmentation approach (Figures 1.1 and 3.5). We first construct a K-NN graph ($K = 4$) over the point cloud, then we perform a BFS starting from a random seed point and stopping at predicted boundary points. All visited points result in a segment. Then we start the same procedure by using a random seed point from the rest of the non-visited points. We note, however, that this simple flood-filling approach can fail in cases where small gaps exist in boundaries (Figure 3.8).

3.4.3 Ablation study

In Table 3.1, we also report the performance of our method under the following variants: (a) "PB-DGCNN w/ EdgeConv" where we use the original EdgeConv layer of DGCNN [231] as first layer instead of LocalEdgeConv. We include the performance of this variant with and without using normals as input features (b) "PB-DGCNN w/ LocalEdgeConv-curv" where we use the principal curvature directions as tangent vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ to define the local coordinate frame $\mathbf{R}_i$ per point (we note that since the curvature directions are defined up to a sign, the max operator in Equation 3.3 is applied to MLPs that also include coordinate transformations based on the opposite principal directions). Curvature is estimated based on the method proposed in [91]. Finally, we include the performance of of our method "PB-DGCNN w/LocalEdgeConv" with and without normals as input features. Based on the numerical results, we observe that "PB-DGCNN w/LocalEdgeConv w/ normals" has the best performance on average. Its achieved Chamfer Distance is lowest compared to all variants, and the bIoU and F1 score are the highest. Using principal directions did not seem to help the performance of LocalEdgeConv. The LocalEdgeConv w/ normals has consistently better performance compared to us-

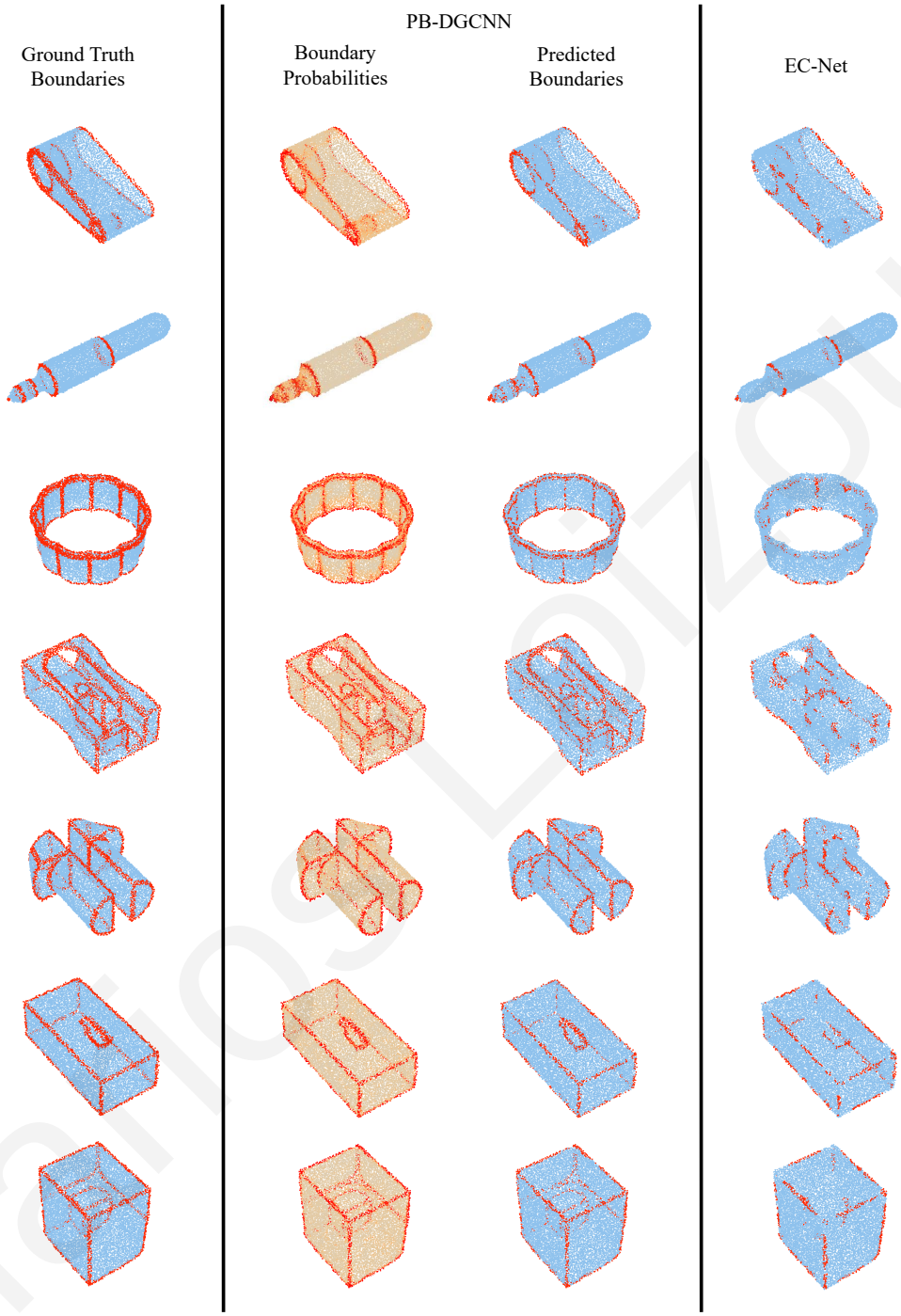

Figure 3.4: Visual comparison of the boundaries detected by our method PB-DGCNN, and EC-Net on some example ABC point clouds. The first column on the left shows the ground truth boundaries. The second column shows boundary probabilities produced by PB-DGCNN, and the third column shows boundaries predicted by PB-DGCNN after thresholding. The last column shows the boundaries predicted by EC-Net.

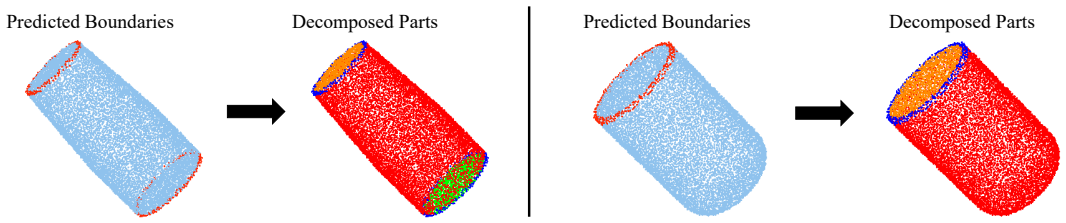

Figure 3.5: Examples of watershed (flood-filling) segmentation. In these cases well-defined predicted boundaries between geometric parts, enable their decomposition to individual segments through simple BFS-based flood-filling.

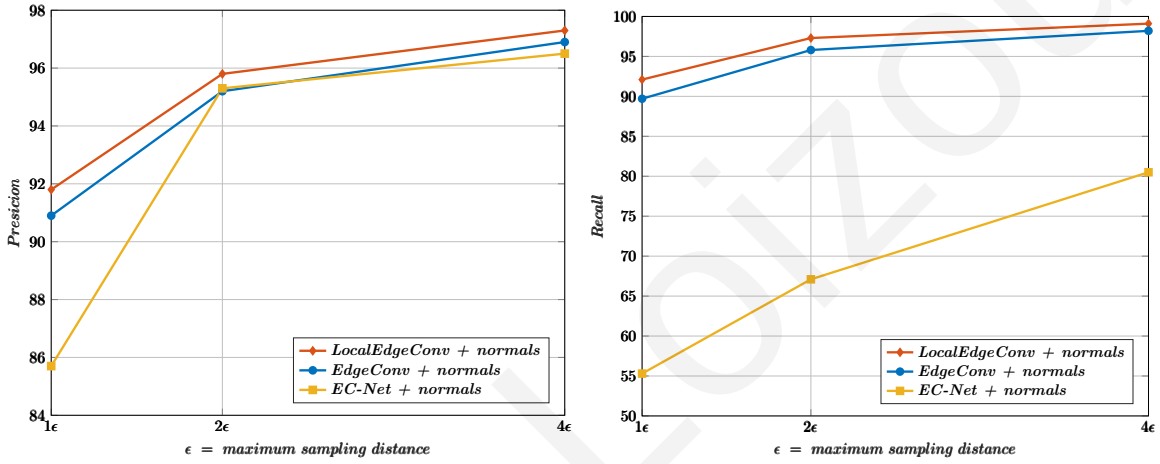

Figure 3.6: Boundary detection evaluation wrt. precision and recall for different tolerance levels.

ing EdgeConv w/ normals according to all metrics. Similarly LocalEdgeConv w/o normals is better than using EdgeConv w/o normals on average. The precision of EdgeConv w/o normals is a bit higher than LocalEdgeConv w/o normals, yet note that its recall is much lower.

Figure 3.6 shows precision and recall for LocalEdgeConv, EdgeConv and EC-Net (here, we use points and normals as input to all methods). Increasing the tolerance results in increasing the precision for all methods, since more predicted boundaries are classified as correct by increasing the boundary distance tolerance threshold, as expected. Similarly, recall is also increased. Most importantly, our method based on LocalEdgeConv has better behavior than the rest, since it demonstrates both higher precision and recall for all tolerance levels we examined.

Category	Bag	Bed	Bott	Bowl	Chai	Cloc	Dish	Disp	Door	Ear	Fauc	Hat	Key	Knif	Lamp	Lap	Micr	Mug	Frid	Scis	Stor	Tabl	Tras	Vase	Avg
Shape IoU																									
Unary only	75.9	25.7	59.1	75.5	50.8	43.9	53.9	84.0	44.0	52.9	55.8	64.3	**62.4**	43.3	43.5	95.9	59.9	88.4	51.6	76.8	52.1	52.9	52.4	80.8	60.2
GC normal diff	76.1	25.9	60.6	81.3	55.0	**44.1**	54.1	85.2	45.4	**53.0**	58.0	65.2	**62.4**	45.6	47.2	**96.0**	60.7	89.5	**52.9**	76.9	55.2	55.7	54.0	82.5	61.8
GC PB-DGCNN	76.1	26.1	61.4	83.2	54.6	43.7	54.0	**86.1**	48.9	52.9	57.5	70.3	**62.4**	47.4	48.5	94.6	**60.9**	**90.4**	51.6	77.2	54.1	56.3	**55.0**	83.5	62.4
GC both	**76.2**	**26.2**	**61.6**	**84.6**	**56.3**	43.7	**54.4**	85.9	**49.3**	52.9	**58.4**	**72.2**	**62.4**	**48.0**	**49.9**	94.6	60.8	88.7	52.6	**78.0**	**55.3**	**57.0**	54.7	**83.8**	**62.8**
Part IoU																									
Unary only	49.9	26.8	39.9	64	40.6	**24.6**	46.2	84.3	32.2	**42.4**	46.1	62.7	**61.1**	39.5	24.1	95.6	**54.4**	81.5	37.7	76.5	43.1	33.4	**45.5**	55.9	50.3
GC normal diff	50.0	**27.0**	39.9	64.2	41.5	**24.6**	46.5	**84.6**	32.7	**42.4**	47.1	63.0	**61.1**	38.5	24.3	**95.7**	52.3	80.4	**38.1**	76.6	**43.3**	**33.6**	42.8	**56.8**	50.3
GC PB-DGCNN	**50.2**	**27.0**	44.3	66.7	41.2	24.0	46.7	**84.6**	32.6	**42.4**	46.7	63.6	**61.1**	**39.7**	**24.4**	93.8	53.9	**82.7**	37.7	76.8	43.2	33.4	43.8	56.6	50.7
GC both	**50.2**	**27.0**	**45.0**	**69.4**	**41.6**	24.0	**47.0**	84.4	**33.1**	**42.4**	**47.3**	**65.7**	**61.1**	38.4	**24.4**	93.9	52.2	80.2	**38.1**	**77.6**	43.2	33.4	42.0	56.7	**50.8**

Table 3.2: Point labeling evaluation of fine-grained semantic segmentation on the PartNet dataset (Part IoU, Shape IoU - %). "Unary alone" represents using the per point part probabilities produced by DGCNN, "GC normal diff" represents graph cuts using normal angles as pairwise term, "GC PB-DGCNN" represents graph cuts using our predicted boundary confidences as pairwise term, and "GC both" represents graph cuts using the weighted combination of pairwise terms based on both normal angles and PB-DGCNN.

3.4.4 Semantic shape segmentation

We now discuss evaluation on semantic shape segmentation based on the PartNet dataset. Here, we train our network on the PartNet training split for each of its categories, as described in Section 3.2. To take advantage of semantic part labels, here we first use a network that predicts a probability for each part per point. Specifically, we use the DGCNN network for this task [231], operating on 10K number of point samples per shape. We then incorporate a graph cuts formulation, where the above per-point part probability is used as a unary term, and the output boundary probabilities from our method ("PB-DGCNN w/ LocalEdgeConv w/normals") are used as a pairwise term:

$$E(\mathbf{c}) = \sum_{i \in \mathcal{P}} \psi(c_i) + \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}(i)} \phi(c_i, c_j) \tag{3.5}$$

where $\mathbf{c} = \{c_i\}$ are the label assignments we wish to compute by minimizing the above energy, $\mathcal{P}$ is the set of points in a test point cloud, and $\mathcal{N}(i)$ is the neighborhood of each point i formed by its $K = 4$ nearest Euclidean neighbors. The unary term is expressed as follows $\psi(c_i) = -logP(c_i)$, where $P(c_i)$ is the probability distribution over part labels associated with the point i produced by the DGCNN point labeling network. The pairwise term uses the maximum of our PB-DGCNN boundary probabilities for the two points: $\phi(c_i, c_j) = -\lambda \cdot log(\max(b_i, b_j))$ for $c_i \neq c_j$,

Category	Bag	Bed	Bott	Bowl	Chai	Cloc	Dish	Disp	Door	Ear	Fauc	Hat	Key	Knif	Lamp	Lap	Micr	Mug	Frid	Scis	Stor	Tabl	Tras	Vase	**Avg**
Chamfer Distance																									
Unary alone	**6.3**	**3.1**	**6.9**	38.8	4.0	**5.4**	6.6	7.8	37.3	**9.9**	6.0	4.1	**2.3**	9.4	7.2	**1.0**	**3.3**	3.6	**3.9**	9.4	1.5	3.2	**3.8**	**10.9**	8.2
GC both	6.4	**3.1**	7.1	**37.0**	**3.7**	**5.4**	**6.2**	**7.7**	**36.4**	**9.9**	**5.4**	**3.1**	**2.3**	**7.3**	**5.9**	1.3	3.5	**3.2**	**3.9**	**8.7**	**1.3**	**2.7**	5.9	11.0	**7.9**
Boundary IoU																									
Unary alone	**61.5**	72.3	48.0	56.0	67.0	59.0	64.0	73.5	54.4	**49.1**	54.4	77.3	**76.5**	38.7	57.1	90.5	**79.1**	66.7	69.1	44.7	88.4	75.9	**77.6**	73.4	65.6
GC both	60.9	**73.0**	**48.4**	**60.7**	**70.2**	**63.9**	**66.0**	**74.9**	**57.9**	49.0	**63.2**	**82.1**	76.1	**47.0**	**67.0**	**90.9**	76.4	**75.3**	**69.7**	**50.4**	**90.0**	**79.2**	74.8	**75.4**	**68.4**
Precision																									
Unary alone	74.3	70.5	52.3	58.7	65.3	57.8	67.1	72.1	56.9	56.6	51.8	78.4	81.8	34.8	56.2	91.2	81.4	75.0	76.4	46.2	88.6	78.0	78.4	72.8	67.6
GC both	**77.6**	**73.7**	**60.4**	**68.0**	**80.2**	**73.9**	**75.7**	**81.4**	**65.7**	**57.6**	**71.2**	**89.2**	**82.1**	**54.7**	**81.9**	**93.8**	**81.7**	**87.3**	**80.3**	**54.8**	**94.4**	**90.1**	**89.6**	**78.8**	**76.8**
Recall																									
Unary alone	**57.9**	**79.5**	**48.9**	**62.8**	**74.7**	**68.0**	**68.0**	**79.1**	**56.1**	**47.8**	**61.8**	**78.9**	**75.9**	**51.6**	**72.5**	**91.0**	**78.0**	65.1	**66.9**	45.3	**90.0**	**81.1**	**79.9**	**79.9**	**69.2**
GC both	55.9	76.8	44.8	62.1	67.1	63.4	63.2	72.7	55.9	46.8	60.2	78.0	75.3	46.2	63.9	90.0	73.3	**70.6**	65.4	**48.9**	87.5	76.2	68.9	76.4	66.2
F_1-score																									
Unary alone	**65.1**	74.7	50.5	60.7	69.7	62.5	67.5	75.4	56.5	**51.8**	56.4	78.7	**78.8**	41.5	63.3	91.1	**79.7**	69.7	71.3	45.7	89.3	79.5	**79.1**	76.2	68.1
GC both	65.0	**75.2**	**51.4**	**64.9**	**73.1**	**68.3**	**68.9**	**76.8**	**60.4**	51.6	**65.2**	**83.2**	78.5	**50.1**	**71.8**	**91.9**	77.2	**78.0**	**72.1**	**51.7**	**90.8**	**82.5**	77.9	**77.6**	**71.0**

Table 3.3: Evaluation of fine-grained semantic segmentation boundaries (Chamfer distance, Boundary IoU, Precision, Recall, F_1 score - %) on the PartNet dataset. "Unary alone" represents using the per point part probabilities produced by DGCNN, "GC both" represents a weighted combination of pairwise terms based on normal angles and PB-DGCNN.

and 0 otherwise. The weighting parameter λ is adjusted through grid search in the hold-out validation set per shape category. To avoid infinite costs, we add a small $\epsilon = 10^{-3}$ to the above log expressions.

We also experimented with another pairwise term variant as baseline that considers angles between between point normals: $\phi'(c_i, c_j) = -\lambda' \cdot log(min(\omega_{i,j}/90^o, 1))$, for $c_i \neq c_j$, where $\omega_{i,j}$ is the angle between the point normals. The term results in zero cost for right angles between normals indicating a strong edge. The weighting parameter λ' is adjusted through grid search in the hold-out validation set per shape category. We finally experimented with a combination of using both the above pairwise terms in Eq. 3.5: $\phi(c_i, c_j) + \phi'(c_i, c_j)$.

We first report point labeling performance in Table 3.2 based on the standard part IoU and shape IoU metrics in PartNet [158]. In here $y(\cdot)$ and $\hat{y}(\cdot)$ are functions that map the points of shape s, P_s, to their predicted and ground truth values respectively. Also, L_s is the labels set of s that are either annotated or predicted by the network and finally S_C are all the shapes that belong to category C, of the PartNet dataset. Thus, shape IoU ($sIoU$) measures the overlap of predicted and ground truth labelled parts within a single shape, and is defined as:

$$sIoU(s) = \frac{1}{|L_s|} \sum_{l \in L_s} \frac{\{y(P_s) == l\} \cap \{\hat{y}(P_s) == l\}}{\{y(P_s) == l\} \cup \{\hat{y}(P_s) == l\}} \tag{3.6}$$

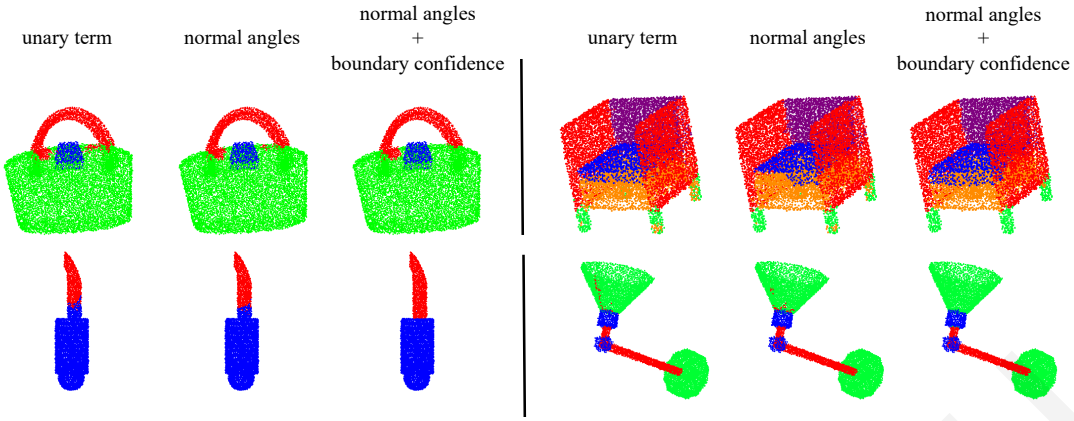

Figure 3.7: Visual comparison of semantic segmentation for example PartNet point clouds, using DGCNN alone (unary), a graph-cut formulation with normal angles in the pairwise term, and a graph-cut formulation with a combination of normal angles and boundary probabilities produced by PB-DGCNN in the pairwise term. Boundary confidence can help to diminish small semantic segments, which are falsely predicted from the semantic segmentation model (unary). These are the cases of Bag (top row, left) and Lamp (bottom row, right), where wrong annotated parts are present even after applying the graph cuts method with normal angles as a pairwise term. Moreover, it can further smooth out semantic parts as it is been illustrated in the case of Chair (top row, right) and Knife (bottom row, left).

On the other hand, part IoU ($pIoU$), measures the overlap of a single predicted and ground truth labelled part, across all shapes that belong to category C, and is defined as:

$$pIoU(l) = \sum_{s \in S_C} \frac{\{y(P_s) == l\} \cap \{\hat{y}(P_s) == l\}}{\{y(P_s) == l\} \cup \{\hat{y}(P_s) == l\}} \tag{3.7}$$

The average shape and part IOU for the whole dataset are calculated as:

$$avg\{sIoU\} = \frac{1}{|S_C|} \sum_{s \in S_C} sIoU(s) \tag{3.8}$$

$$avg\{pIoU\} = \frac{1}{|L_C|} \sum_{l \in L_C} pIoU(l) \tag{3.9}$$

where L_C are all the labelled parts of category C.

We examine the performance of using DGCNN alone as unary term ("unary alone"), then using graph cuts based on the normal angle baseline described above ("GC normal diff"), graph cuts based on the predicted boundary probabilities of PB-DGCNN ("GC PB-DGCNN"), and finally graph cuts using the summation of pairwise terms from normal angles and PB-DGCNN ("GC both"). We observe

Category	Bag	Bed	Bott	Bowl	Chai	Cloc	Dish	Disp	Door	Ear	Fauc	Hat	Key	Knif	Lamp	Lap	Micr	Mug	Frid	Scis	Stor	Tabl	Tras	Vase	**Avg**
Chamfer Distance																									
Unary alone	**6.3**	**3.1**	**6.9**	38.8	4.0	**5.4**	6.6	7.8	37.3	**9.9**	6.0	4.1	**2.3**	9.4	7.2	**1.0**	**3.3**	3.6	**3.9**	9.4	1.5	3.2	**3.8**	**10.9**	8.2
GC both	6.4	**3.1**	7.1	**37.0**	**3.7**	**5.4**	**6.2**	**7.7**	**36.4**	**9.9**	**5.4**	**3.1**	**2.3**	**7.3**	**5.9**	1.3	3.5	**3.2**	**3.9**	**8.7**	**1.3**	**2.7**	5.9	11.0	**7.9**
Boundary IoU																									
Unary alone	**61.5**	72.3	48.0	56.0	67.0	59.0	64.0	73.5	54.4	**49.1**	54.4	77.3	**76.5**	38.7	57.1	90.5	**79.1**	66.7	69.1	44.7	88.4	75.9	**77.6**	73.4	65.6
GC both	60.9	**73.0**	**48.4**	**60.7**	**70.2**	**63.9**	**66.0**	**74.9**	**57.9**	49.0	**63.2**	**82.1**	76.1	**47.0**	**67.0**	**90.9**	76.4	**75.3**	**69.7**	**50.4**	**90.0**	**79.2**	74.8	**75.4**	**68.4**
Precision																									
Unary alone	74.3	70.5	52.3	58.7	65.3	57.8	67.1	72.1	56.9	56.6	51.8	78.4	81.8	34.8	56.2	91.2	81.4	75.0	76.4	46.2	88.6	78.0	78.4	72.8	67.6
GC both	**77.6**	**73.7**	**60.4**	**68.0**	**80.2**	**73.9**	**75.7**	**81.4**	**65.7**	**57.6**	**71.2**	**89.2**	**82.1**	**54.7**	**81.9**	**93.8**	**81.7**	**87.3**	**80.3**	**54.8**	**94.4**	**90.1**	**89.6**	**78.8**	**76.8**
Recall																									
Unary alone	**57.9**	**79.5**	**48.9**	**62.8**	**74.7**	**68.0**	**68.0**	**79.1**	**56.1**	**47.8**	**61.8**	**78.9**	**75.9**	**51.6**	**72.5**	**91.0**	**78.0**	65.1	**66.9**	45.3	**90.0**	**81.1**	**79.9**	**79.9**	**69.2**
GC both	55.9	76.8	44.8	62.1	67.1	63.4	63.2	72.7	55.9	46.8	60.2	78.0	75.3	46.2	63.9	90.0	73.3	**70.6**	65.4	**48.9**	87.5	76.2	68.9	76.4	66.2
F_1-score																									
Unary alone	**65.1**	74.7	50.5	60.7	69.7	62.5	67.5	75.4	56.5	**51.8**	56.4	78.7	**78.8**	41.5	63.3	91.1	**79.7**	69.7	71.3	45.7	89.3	79.5	**79.1**	76.2	68.1
GC both	65.0	**75.2**	**51.4**	**64.9**	**73.1**	**68.3**	**68.9**	**76.8**	**60.4**	51.6	**65.2**	**83.2**	78.5	**50.1**	**71.8**	**91.9**	77.2	**78.0**	**72.1**	**51.7**	**90.8**	**82.5**	77.9	**77.6**	**71.0**

Table 3.4: Evaluation of fine-grained semantic segmentation boundaries (Chamfer distance, Boundary IoU, Precision, Recall, F_1 score - %) on the PartNet dataset. "Unary alone" represents using the per point part probabilities produced by DGCNN, "GC both" represents a weighted combination of pairwise terms based on normal angles and PB-DGCNN.

small but noticeable average performance increases for both shape IoU and part IoU when using all variants of graph cuts. The best performance is achieved on average (see last row, last column) when combining both pairwise terms. Specifically, we observe an increase of average shape IoU by 2.6% and part IoU 0.5% compared to using the unary term alone. We believe that using both pairwise terms offers the best performance because our boundary probabilities are more fuzzy in PartNet - we note that the training boundary data were also slightly fuzzy in PartNet (see Figure 3.3) in the first place. Using normal angles further sharpens our probabilistic boundaries. Nevertheless, the metrics seem improved with the use of our probabilistic boundaries in the pairwise term alone. We note that graph cuts is executed in a deterministic manner (i.e., there is no variance in the above increases given a fixed unary term). The performance increase is not dramatic: this is not surprising, since refining boundaries changes relatively few point labels near boundaries.

Table 3.4 reports our evaluation metrics in terms of boundary quality. We note that compared to the ABC dataset, the evaluation of boundaries here is less reliable. In contrast to ABC, where the ground-truth boundaries were parametric curves and were highly accurate, the ground-truth boundaries in PartNet are marked approximately using the heuristic search described in Section 3.2. We report the performance of our best variant of graph cuts (using both terms), and also the unary

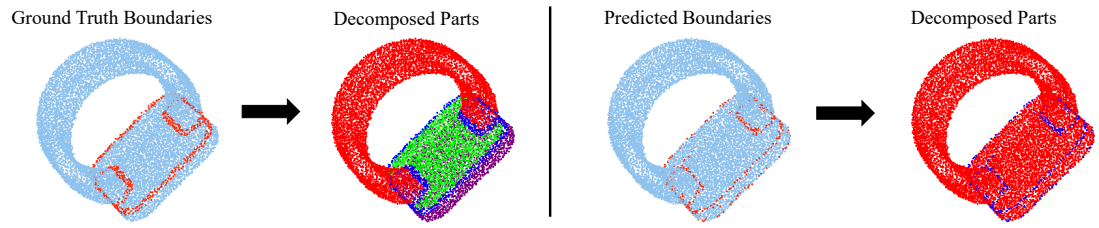

Figure 3.8: The figure on the left depicts part decomposition of the point cloud from ground truth boundaries, with BFS flood-filling. It successfully segments the point cloud into three parts. This is not the case on the right figure, where the predicted boundaries fail to enclose points into separate segments, which results to only one part after the watershed segmentation procedure.

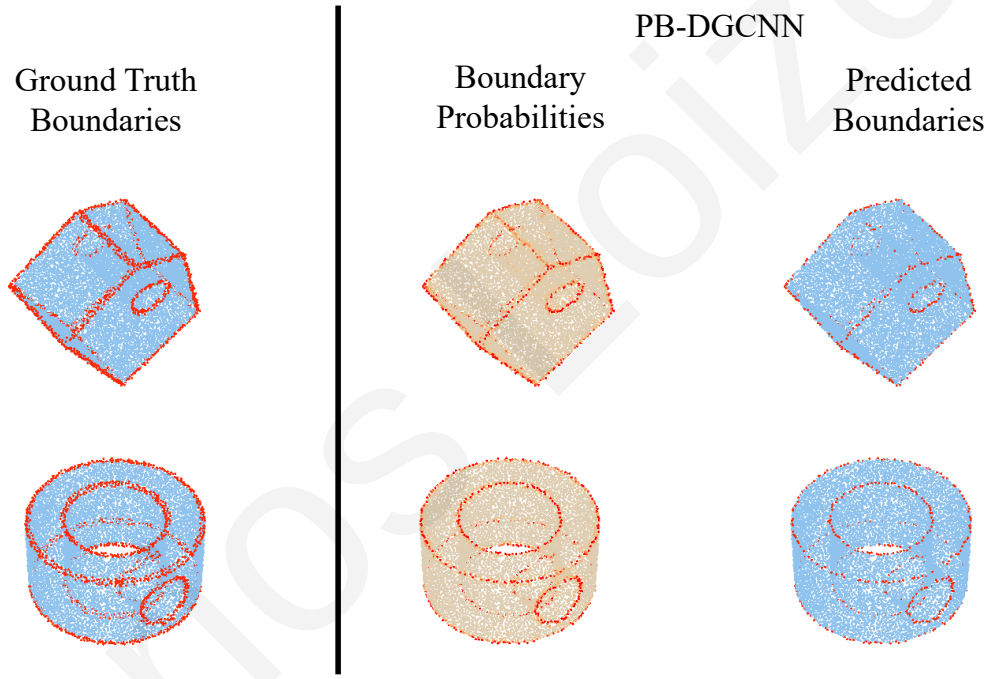

Figure 3.9: Predicted boundary confidences (middle column) is sometimes low resulting in sparsely labeled boundary points (right column).

term alone. We observe that graph cuts result in boundaries that are more consistent with ground-truth. In particular, we observe an improvement of 2.8% for bIoU, and 2.9% for F1-score on average. We note that although the recall is lower, the precision is significantly much higher. Figures 1.1 and 3.7 show semantic segmentation results for a few examples from PartNet.

3.5 Discussion

In this chapter we presented a method for detecting probabilistic boundaries in point clouds. It utilizes a neural module that performs graph edge convolutions within point cloud neighborhoods, expressed in local frames. The proposed architecture is trained with a classification loss and sampling strategy, that are robust to the class imbalance between boundary and non-boundary points. In addition, a graph cuts formulation was used, that takes into account our predicted boundary probabilities, to promote more coherent boundaries in segmentation tasks. Our evaluation showed that our boundaries are closer to ground-truth in geometric decomposition tasks, and also improve the quality of cuts in semantic segmentation tasks.

Limitations. Our method also has limitations that could inspire future research. First, our method currently extracts probabilities of part boundaries over points. Sometimes these probabilities seem too low (Figure 3.9), resulting in sparsely labeled boundary points, which makes it harder to extract a continuous boundary curve. It would be interesting to investigate robust fitting of parametric curves or lines to probabilistic boundaries to localize them more accurately. This could in turn be combined with neural patch fitting [197], and also result in geometric decomposition of point clouds to primitives with more accurately trimmed boundaries. For semantic segmentation, jointly optimizing the unary and pairwise term with the rest of the network in an end-to-end manner could further improve results. Finally, it would be interesting to extend our method to handle polygon mesh segmentations based on our detected boundaries.

Chapter 4

Cross-Shape Attention for Part Segmentation of 3D Point Clouds

In the present chapter, we propose a *cross-shape attention* mechanism that enables interaction and propagation of point-wise feature representations across shapes of an input collection[1]. Developing such a cross-shape attention mechanism is challenging. Performing cross-attention across all pairs of shapes becomes prohibitively expensive for large input collections of shapes. Our method learns a measure that allows us to select a small set of other shapes useful for such cross-attention operations with a given input shape. For example, given an input office chair, it is more useful to allow interactions of its points with points of another structurally similar office chair rather than a stool. During training, we maintain a sparse graph (Figure 1.2), whose nodes represent training shapes and edges specify which pairs of shapes should interact for training our cross-shape attention mechanism. At test time, the shape collection graph is augmented with additional nodes representing test shapes. New edges are added connecting them to training shapes for propagating representations from relevant training shapes.

Our approach is inspired by recent cross-attention models proposed for video classification, image classification, keypoint recognition, and image-text matching. Wang *et al.* [230] introduced non-local networks that allow any image query position to perceive features of all the other positions within the same image or across

[1]The work presented in this chapter is also published in Computer Graphics Forum, vol. 42, no. 5, 2023. Project page: https://marios2019.github.io/CSN. This is the author's version of the work. The definitive version of the article was published in Computer Graphics Forum, vol. 42, no. 5, 2023, https://doi.org/10.1111/cgf.14909.

frames in a video. To avoid huge attention maps, Huang *et al.* [83] proposes a "criss-cross" attention module to maintain sparse connections for each position in image feature maps. Cao *et al.* [24] simplifies non-local blocks with query-independent attention maps. Lee *et al.* [121] propose cross-attention between text and images to discover latent alignments between image regions and words in a sentence. Hou *et al.* [75] models the semantic relevance between class and query feature maps in images through cross-attention to localize more relevant image regions for classification and generate more discriminative features. Sarlin *et al.* [188] learns keypoint matching between two indoor images from different viewpoints by leveraging self-attention and cross-attention to boost the receptive field of local descriptors and allow cross-image communication. Chen *et al.* [29] propose cross-attention between multiscape representations for image classification. Finally, Doersch *et al.* [46] introduced a CrossTransformer model for few-shot learning on images. Give an unlabeled query image, their model computes local cross-attention similarities with a number of labeled images and then infers class membership. Our method instead introduces attention mechanisms across 3D shapes. In contrast to cross-attention approaches in the above domains, we do not assume any pre-existing paired data. The usefulness of shape pairs is determined based on a learned shape compatibility measure.

We tested our cross-shape attention mechanism on two different backbones to extract the initial point-wise features per shape for the task of part segmentation: a sparse tensor network based on MinkowskiNet [34] and the octree-based network MID-FC [226]. For both backbones, we observed that our mechanism significantly improves the point-wise features for segmentation. Compared to the MinkowskiNet baseline, we found an improvement of 3.1% in mean part IoU in the PartNet benchmark [158]. Compared to MID-FC, we found an improvement of $+1.3\%$ in mean part IoU, achieving a new state-of-the-art result in PartNet (MID-FC: 60.8% $\rightarrow$ Ours: 62.1%).

In summary, the contributions of this[2] chapter are:

- an attention-based mechanism that enables point-wise feature interaction and propagation within and across shapes for more consistent segmentation.
- our experiments show state-of-the-art performance on the recent PartNet dataset.

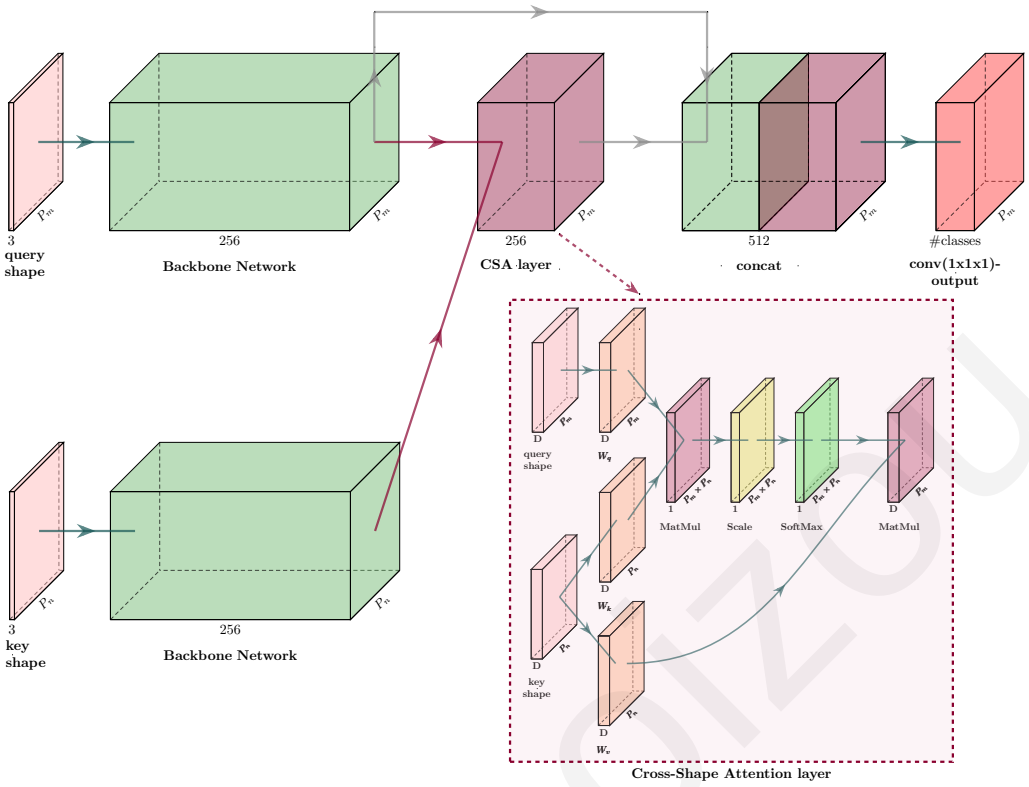

Figure 4.1: Our cross-shape network architecture: given an input test shape ("query shape") represented as an input point set, we first extract initial point-wise features through a backbone (our MinkowskiNet variant, called "MinkNetHRNet", or alternatively the MID-FC network [226]). Then our proposed cross-attention layer, called CSA layer, propagates features extracted from another shape of the input shape collection ("key shape") to the query shape such that their semantic segmentation becomes more synchronized. The output point-wise features of the CSA layer are concatenated with the original features of the query shape, then they are passed to a classification layer for semantic segmentation. Note that the illustrated CSA layer in the inlet figure uses only one head ($H = 1$).

4.1 Method

Given an input collection of 3D shapes represented as point clouds, the goal of our method is to extract and propagate point-based feature representations from one shape to another, and use the resulting representations for 3D semantic segmentation. To perform the feature propagation, we propose a Cross-Shape Attention (CSA) mechanism. The mechanism first assesses the degree of interaction between

[2]Marios Loizou, Siddhant Garg and Dmitry Petrov contributed equally to this work.

pairs of points on different shapes. Then it allows point-wise features on one shape to influence the point-wise features of the other shape based on their assessed degree of interaction. In addition, we provide a mechanism that automatically selects shapes ("key shapes") to pair with an input test shape ("query shape") to execute these cross-shape attention operations. In the following sections, we first discuss the CSA layer at test time (Section 4.1.1). Then we discuss our retrieval mechanism to find key shapes given a test shape (Section 4.1.2), our training (Section 4.1.3), test stage (Section 4.1.4), and finally our network architecture details (Section 4.1.5).

4.1.1 Cross-shape attention for a shape pair

The input to our CSA layer is a pair of shapes represented as point clouds: $\mathcal{S}_m = \{\mathbf{p}_i\}_{i=1}^{P_m}$ and $\mathcal{S}_n = \{\mathbf{p}_j\}_{j=1}^{P_n}$ where $\mathbf{p}_i, \mathbf{p}_j \in \mathcal{R}^3$ represent 3D point positions and P_m, P_n are the number of points for each shape respectively. Our first step is to extract point-wise features for each shape.

In our implementation, we experimented with two backbones for point-wise feature extraction: a sparse tensor network based on a modified version of MinkowskiNet [34], and an octree-based network, called MID-FC [226] (architecture details for the two backbones are provided in Section 4.1.5 and Appendix A). The output from the backbone is a per-point D-dimensional representation stacked into a matrix for each of the two shapes respectively: $\mathbf{X}_m \in \mathcal{R}^{P_m \times D}$ and $\mathbf{X}_n \in \mathcal{R}^{P_n \times D}$. The CSA layer produces new D-dimensional point-wise representations for both shapes:

$$\boldsymbol{X}'_m = f\left(\mathbf{X}_m, \mathbf{X}_n; \boldsymbol{\theta}\right), \;\; \boldsymbol{X}'_n = f\left(\mathbf{X}_n, \mathbf{X}_m; \boldsymbol{\theta}\right) \tag{4.1}$$

where f is the cross-shape attention function with learned parameters θ described in the next paragraphs.

Key and query intermediate representations. Inspired by Transformers [217], we first transform the input point representations of the first shape in the pair to intermediate representations, called "query" representations. The input point representations of the second shape are transformed to intermediate "key" representations. The keys will be compared to queries to determine the degree of influence of one point on another. Specifically, these transformations are expressed as follows:

$$\boldsymbol{q}_{m,i}^{(h)} = \boldsymbol{W}_q^{(h)} \boldsymbol{x}_{m,i}, \;\; \boldsymbol{k}_{n,j}^{(h)} = \boldsymbol{W}_k^{(h)} \boldsymbol{x}_{n,j} \tag{4.2}$$

where $\boldsymbol{x}_{m,i}$ and $\boldsymbol{x}_{n,j}$ are point representations for the query shape $\mathcal{S}_m$ and key shape $\mathcal{S}_n$, $\boldsymbol{W}_q^{(h)}$ and $\boldsymbol{W}_k^{(h)}$ are $D' \times D$ learned transformation matrices shared across all points of the query and key shape respectively, and h is an index denoting each different transformation ("head"). The dimensionality of the key and query representations D' is set to $\lfloor D/H \rfloor$, where H is the number of heads. These intermediate representations are stacked into the matrices $\boldsymbol{Q}_m^{(h)} \in \mathcal{R}^{P_m \times D'}$ and $\boldsymbol{K}_n^{(h)} \in \mathcal{R}^{P_n \times D'}$. Furthermore, the point representations of the key shape $\mathcal{S}_n$ are transformed to value representations as:

$$\boldsymbol{v}_{n,j}^{(h)} = \boldsymbol{W}_v^{(h)} \boldsymbol{x}_{n,j} \tag{4.3}$$

where $\boldsymbol{W}_v^{(h)}$ is a learned $D' \times D$ transformation shared across the points of the key shape. These are also stacked to a matrix $\boldsymbol{V}_n^{(h)} \in \mathcal{R}^{P_n \times D'}$.

Pairwise point attention. The similarity of key and query representations is determined through scaled dot product [217]. This provides a measure of how much one shape point influences the point on the other shape. The similarity of key and query representations is determined for each head as:

$$\boldsymbol{A}_{m,n}^{(h)} = softmax\left(\frac{\boldsymbol{Q}_m^{(h)} \cdot \left(\boldsymbol{K}_n^{(h)}\right)^\top}{\sqrt{D'}}\right) \tag{4.4}$$

where $\boldsymbol{A}_{m,n}^{(h)} \in \mathcal{R}^{P_m \times P_n}$ is a cross-attention matrix between the two shapes for each head.

Feature representation updates. The cross-attention matrix is used to update the point representations for the query shape $\mathcal{S}_m$:

$$\boldsymbol{z}_{m,i}^{(h)} = \sum_{j=1}^{P_n} \boldsymbol{A}_{m,n}^{(h)}[i,j] \boldsymbol{W}_v^{(h)} \boldsymbol{x}_{n,j} \tag{4.5}$$

The point-wise features are concatenated across all heads, then a linear transformation layer projects them back to D-dimensional space and they are added back to the original point-wise features of the query shape:

$$\boldsymbol{x}'_{m,i} = \boldsymbol{x}_{m,i} + \mathbf{W}_d \cdot [\boldsymbol{z}_{m,i}^{(1)}, \boldsymbol{z}_{m,i}^{(2)}, ..., \boldsymbol{z}_{m,i}^{(H)}] \tag{4.6}$$

where H is the number of heads and $\mathbf{W}_d$ is another linear transformation. The features are stacked into a matrix $\boldsymbol{X}'_m \in \mathcal{R}^{P_m \times D}$, followed by layer normalization [7].

Self-shape attention. The pairwise attention of Equation 4.4 and update operation of Equation 4.5 can also be applied to a pair that consists of the shape and itself. In this case, the CSA layer is equivalent to Self-Shape Attention (SSA), enabling long-range interactions between shape points within the same shape.

Cross-shape attention for multiple shapes. We can further generalize the cross-shape operation in order to handle multiple shapes and also combine it with self-shape attention. Given a selected set of key shapes, our CSA layer outputs point representations for the query shape $\mathcal{S}_m$ as follows:

$$\boldsymbol{X}'_m = \sum_{n \in \{\mathcal{C}(m), m\}} c(m,n) \boldsymbol{A}_{m,n} \boldsymbol{V}_n \tag{4.7}$$

where $\mathcal{C}(m)$ is a set of key shapes deemed compatible for cross-shape attention with shape $\mathcal{S}_m$ and $c(m,n)$ is a learned pairwise compatibility function between the query shape $\mathcal{S}_m$ and each key shape $\mathcal{S}_n$. The key idea of the above operation is to update point representations of the query shape $\mathcal{S}_m$ as a weighted average of attention-modulated representations computed by using other key shapes as well as the shape itself. The compatibility function $c(m,n)$ assesses these weights that different shapes should have for cross-shape attention. It also implicitly provides the weight of self-shape attention when $\mathcal{S}_m = \mathcal{S}_n$.

Compatibility function. To compute the compatibility function, we first extract a global, D-dimensional vector representation $\boldsymbol{y}_m$ and $\boldsymbol{y}_n$ for the query shape $\mathcal{S}_m$ and each key shape $\mathcal{S}_n$ respectively through mean-pooling on their self-shape attention representations:

$$\boldsymbol{y}_m^{(SSA)} = \text{avg}_i \boldsymbol{X}'^{(SSA)}_{m,i} = \text{avg}_i \left(\boldsymbol{A}_{m,m} \boldsymbol{V}_m \right) \tag{4.8}$$

$$\boldsymbol{y}_n^{(SSA)} = \text{avg}_i \boldsymbol{X}'^{(SSA)}_{n,i} = \text{avg}_i \left(\boldsymbol{A}_{n,n} \boldsymbol{V}_n \right) \tag{4.9}$$

In this manner, the self-attention representations of both shapes provide cues for the compatibility between them expressed using their scaled dot product similarity [217]:

$$\begin{aligned} \boldsymbol{u}_m &= \boldsymbol{U}_q \boldsymbol{y}_m^{(SSA)} \\ \boldsymbol{u}_n &= \boldsymbol{U}_k \boldsymbol{y}_n^{(SSA)} \\ s(m,n) &= \hat{\boldsymbol{u}}_m \cdot \hat{\boldsymbol{u}}_n^\top \end{aligned} \tag{4.10}$$

where $\boldsymbol{U}_q$ and $\boldsymbol{U}_k$ are learned $D \times D$ transformations for the query and key shape respectively, and $\hat{\boldsymbol{u}}_m = \boldsymbol{u}_m / ||\boldsymbol{u}_m||, \hat{\boldsymbol{u}}_n = \boldsymbol{u}_n / ||\boldsymbol{u}_n||$. The final compatibility function $c(m,n)$ is computed as a normalized measure using a softmax transformation of compatibilities of the shape m with all other shapes in the set $\mathcal{C}(m)$, including the self-compatibility:

$$c(m,n) = \frac{exp\big(s(m,n)\big)}{\sum_{n \in \{C(m),m\}} exp\big(s(m,n)\big)} \tag{4.11}$$

4.1.2 Key shape retrieval

To perform cross-shape attention, we need to retrieve one or more key shapes for each query shape. One possibility is to use the measure of Equation 4.10 to evaluate the compatibility of the query shape with each candidate key shape from an input collection. However, we found that this compatibility is more appropriate for the particular task of weighting the contribution of each selected key shape for cross-shape attention, rather than retrieving key shapes themselves (see Appendix B for additional discussion). We instead found that it is better to retrieve key shapes whose point-wise representations are on average more similar to the ones of the query shape. To achieve this, we perform the following steps:

(i) We compute the similarity between points of the query shape and the points of candidate key shapes in terms of cosine similarity of their SSA representations:

$$\boldsymbol{S}_{m,n} = \boldsymbol{X}'^{(SSA)}_m \cdot (\boldsymbol{X}'^{(SSA)}_n)^\top \tag{4.12}$$

where $\boldsymbol{S}_{m,n} \in \mathcal{R}^{P_m \times P_n}$.

(ii) Then for each query point, we find its best matching candidate key shape point yielding the highest cosine similarity:

$$r_i(m,n) = \max_j \boldsymbol{S}_{m,n}[i,j] \tag{4.13}$$

(iii) Finally, we compute the average of these highest similarities across all query points:

$$r(m,n) = \underset{i}{avg}\, r_i(m,n) \tag{4.14}$$

The retrieval measure $r(m,n)$ is used to compare the query shape $\mathcal{S}_n$ with candidate key shapes from a collection.

4.1.3 Training

The input to our training procedure is a collection of point clouds with part annotations along with a smaller annotated collection used for hold-out validation. We first train our backbone including a layer that implements self-shape attention alone according to Equations 4.3-4.6 (i.e., $\mathcal{S}_m = \mathcal{S}_n$ in this case). The resulting output features are passed to a softmax layer for semantic segmentation. The network is trained according to softmax loss. Based on the resulting SSA features, we construct a graph (Figure 1.2), where each training shape is connected with K shapes, deemed as "key" shapes, according to our retrieval measure of Equation 4.14. One such graph is constructed for the training split, and another for the validation split. We then fine-tune our backbone and train a layer that implements our full cross-shape attention involving all K key shapes per training shape using the same loss. During training, we measure the performance of the network on the validation split, in terms of part IoU [158], and if it reaches a plateau state, we recalculate the K-neighborhood of each shape based on the updated features. We further fine-tune our backbone and CSA layer. This iteration of graph update and fine-tuning of our network is performed two times in our implementation.

4.1.4 Inference

During inference, we create a graph connecting each test shape with K training shapes retrieved by the measure of Equation 4.14. We then perform a feed-forward pass through our backbone, CSA layer, and classification layer to assess the label probabilities for each test shape.

4.1.5 Architecture

Here we describe the two backbones (MinkNetHRNet, MID-FC) we used to provide point-wise features to our CSA layer (see Figure 4.1).

MinkNetHRNet. The first backbone is a variant of the sparse tensor network based on MinkowskiNet [34]. We note that our variant performed better than the original MinkowskiNet for 3D segmentation [34], as discussed in our experiments. In a pre-processing step, we normalize the point clouds to a unit sphere

and convert them to a sparse voxel grid (voxel size $= 0.05$). After two convolutional layers, the network branches into three stages inspired by the HRNet [221], a network that processes 2D images in a multi-resolution manner. In our case, the first stage consists of three residual blocks processing the sparse voxel grid in its original resolution. The second stage downsamples the voxel grid by a factor of 2 and processes it through two other residual blocks. The third stage further downsamples the voxel grid by a factor of 2 and processes it through another residual block. The multi-resolution features from the three stages are combined into one feature map through upsampling following [221]. The resulting feature map is further processed by a $1D$ convolutional block. The sparse voxel features are then mapped back to points as done in the original MinkowskiNet [34]. Details about the architecture of this backbone are provided in Section A.1 (Appendix A).

MID-FC. The second variant utilizes an octree-based architecture based on the MID-FC network [226]. This network also incorporates a three-stage HRNet [221] to effectively maintain and merge multi-scale resolution feature maps. To implement this architecture, each point cloud is first converted into an octree representation with a resolution of 64^3. To train this network, a self-supervised learning approach is employed using a multi-resolution instance discrimination pretext task with ShapeNetCore55 [189]. The training process involves two losses: a shape instance discrimination loss to classify augmented copies of each shape instance and a point instance discrimination loss to classify the same points on the augmented copies of a shape instance. This joint learning approach enables the network to acquire generic shape and point encodings that can be used for shape analysis tasks. Finally, the pre-trained network is combined with two fully-connected layers and our CSA layer. During training for our segmentation task, the HRNet is frozen, while we train only the two fully-connected layers and CSA layer for efficiency reasons. Details about the architecture of this backbone are provided in Section A.2 (Appendix A).

4.1.6 Implementation details

We train our Cross Shape Network for each object category of PartNet [158] separately, using the standard cross entropy loss, for 200 epochs. We set the batch

Variant	**avg part IoU**
MinkResUNet	46.8
MinkHRNet	48.0
MinkHRNetCSN-SSA	48.7
MinkHRNetCSN-K1	**49.9**
MinkHRNetCSN-K2	49.7
MinkHRNetCSN-K3	47.2
MID-FC	60.8
MID-FC-SSA	61.8
MID-FC-CSN-K1	61.9
MID-FC-CSN-K2	61.9
MID-FC-CSN-K3	62.0
MID-FC-CSN-K4	**62.1**
MID-FC-CSN-K5	62.0

Table 4.1: Ablation study for all our variants in PartNet.

size equal to 8 for all variants (SSA, K=1,2,3). For optimization we use the SGD optimizer [184] with a learning rate of 0.5 and momentum $= 0.9$. We scale learning rate by a factor of 0.5, whenever the loss of the hold-out validation split saturates (patience $= 10$ epochs, cooldown $= 10$ epochs). For updating the shape graph for the training and validation split, we measure the performance of the validation split in terms of Part IoU. If it reaches a saturation point (patience $= 10$ epochs, cooldown $= 5$ epochs), we load the best model up to that moment, based on Part IoU performance, and update the graph for both splits. The graph is updated twice throughout our training procedure. For all layers we use batch normalization [84] with momentum $= 0.02$, except for the CSA module, where the layer normalization [7] is adopted. Our method is implemented[3] in PyTorch [167].

Category	Bed	Bott	Chai	Cloc	Dish	Disp	Door	Ear	Fauc	Knif	Lamp	Micr	Frid	Stor	Tabl	Tras	Vase	**avg.**	**#cat.**
SpiderCNN [245]	36.2	32.2	30.0	24.8	50.0	80.1	30.5	37.2	44.1	22.2	19.6	43.9	39.1	44.6	20.1	42.4	32.4	37.0	0
PointNet++ [173]	30.3	41.4	39.2	41.6	50.1	80.7	32.6	38.4	52.4	34.1	25.3	48.5	36.4	40.5	33.9	46.7	49.8	42.5	0
ResGCN-28 [122]	35.9	49.3	41.1	33.8	56.2	81.0	31.1	45.8	52.8	44.5	23.1	51.8	34.9	47.2	33.6	50.8	54.2	45.1	0
PointCNN [126]	41.9	41.8	43.9	36.3	58.7	82.5	37.8	48.9	60.5	34.1	20.1	58.2	42.9	49.4	21.3	53.1	58.9	46.5	0
CloserLook3D [140]	49.5	49.4	48.3	49.0	65.6	84.2	56.8	53.8	62.4	39.3	24.7	61.3	55.5	54.6	44.8	56.9	58.2	53.8	0
MinkResUNet [34]	39.4	44.2	42.3	35.4	57.8	82.4	33.9	45.8	57.8	46.7	25.0	53.7	40.5	45.0	35.7	50.6	58.8	46.8	0
MinkHRNetCSN-K1 (ours)	42.1	54.0	42.5	42.9	58.2	83.2	43.5	51.5	59.4	47.8	27.9	57.4	43.7	46.2	36.8	51.5	60.0	49.9	0
MID-FC [226]	51.6	56.5	**55.7**	55.3	75.6	91.3	56.6	53.8	64.6	55.4	31.2	78.7	63.1	62.8	45.7	65.8	69.3	60.8	1
MID-FC-CSN-K4 (ours)	**52.2**	**58.6**	**55.7**	**57.7**	**76.4**	**91.4**	**58.9**	**54.5**	**65.2**	**62.2**	**33.1**	**79.2**	**64.0**	**62.9**	**46.0**	**67.2**	**69.9**	**62.1**	**16**

Table 4.2: Comparisons with other methods reporting performance in PartNet. The column "avg." reports the mean Part IoU (averaged over all 17 categories). The last column "#cat" counts the number of categories that a method wins over others.

4.2 Results

We evaluated our method for fine-grained shape segmentation qualitatively and quantitatively. In the next sections, we discuss the used dataset, evaluation metrics, comparisons, and an analysis considering the computation time and size of our CSA layer.

4.2.1 Dataset

We use the PartNet dataset [158] for training and evaluating our method according to its provided training, validation, and testing splits. Our evaluation focuses on the fine-grained level of semantic segmentation, which includes 17 out of the 24 object categories present in the PartNet dataset. We trained our network and competing variants separately for each object category.

4.2.2 Evaluation metrics

For evaluating the performance of our method and variants, we used the standard Part IoU metric (see Section 3.4.4, Equations 3.7 and 3.9), as also proposed in the PartNet benchmark [158]. The goal of our evaluation is to verify the hypothesis that our self-attention and cross-shape attention mechanisms yield better features for segmentation than the ones produced by any of the two original backbones on the task of semantic shape segmentation.

[3]The implementation is available at https://github.com/marios2019/CSN.

4.2.3 Ablation

Table 4.1 reports the mean part IoU performance averaged the PartNet's part categories for the original backbones ("MinkHRNet") and ("MID-FC"). We first observe that our backbone variant "MinkHRNet" improves over the original "MinkResUNet" proposed in [34], yielding an improvement of 1.2% in mean Part IoU. Our variant based on self-shape attention alone ("MinkHRNetCSN-SSA") further improves our backbone by 0.7% in Part IoU. We further examined the performance of our cross-shape attention (CSA layer) tested in the variants "MinkHRNetCSN-K1", "MinkHRNetCSN-K2", and "MinkHRNetCSN-K3", where we use $K = 1,2,3$ key shapes per query shape. Our CrossShapeNet with $K = 1$ ("MinkHRNetCSN-K1") offers the best performance on average by improving Part IoU by another 1.2% with respect to using self-shape attention alone. When using $K = 2$ key shapes in cross-shape attention, the performance drops slightly (-0.2% in Part IoU on average) compared to using $K = 1$, and drops even more when using $K = 3$. Thus, for the MinkowskiNet variants, it appears that the optimal number of key shapes is $K = 1$; we suspect that the performance drop for higher K is due to the issue that the chance of retrieving shapes that are incompatible to the query shape is increased with larger numbers of retrieved key shapes.

We also observe improvements using the MID-FC backbone. Note that this backbone has higher performance than the MinkowskiNet variants due to its pretraining and fine-tuning strategies [226]. Our variant based on self-shape attention alone ("MID-FC-SSA") further improves the original MID-FC backbone by 1.0% in mean Part IoU. When using cross-shape attention, the optimal performance is achieved when using $K = 4$ key shapes ("MID-FC-CSN-K4"), which improves Part IoU by another 0.3% with respect to using self-shape attention alone. We note that the above improvements are quite stable – by repeating all experiments 15 times, the standard deviation of mean Part IoU is $\sigma = 0.03\%$. This means that the above differences are significant – even the improvement of 0.3% of "MID-FC-CSN-K4" over "MID-FC-SSA" is of scale 10σ.

4.2.4 Comparisons with other methods

Table 4.2 includes comparisons with other methods reporting their performance on PartNet per category [245, 173, 122, 126, 140, 226] along with our best performing variants ("MinkHRNetCSN-K1" and "MID-FC-CSN-K4"). Compared to the original MinkowskiNet ("MinkResUNet"), our "MinkHRNetCSN-K1" variant achieves an improvement of 3.1% in terms of mean Part IoU in PartNet. Compared to "MID-FC", our best variant ("MID- FC-CSN-K4") also offers a noticeable improvement of 1.3% in mean Part IoU. To the best of our knowledge, the result of our best variant represents the highest mean Part IoU performance achieved in the PartNet benchmark so far[4]. As it can been in the last column of Table 4.2, our method improves performance for 16 out of 17 categories.

4.2.5 Qualitative Results

Figures 4.2 shows qualitative comparisons for MinkowskiNet-based variants – specifically our best variant in this case using cross-shape attention with $K = 1$, self-shape attention, our MinkHRNet backbone, and the original MinkowskiNet. Our backbone often improves the labeling relative to the original MinkowskiNet (e.g., see bed mattress). Our cross-shape attention tends to further improve upon fine-grained details in the segmentation e.g., see the top of the bottle, the armrests in the chair, and the bottom of the faucet, pushing the segmentation to be more consistent with the retrieved key shape shown in the inlet images. Figure 4.3 shows comparisons for the MID-FC-based variants, including using cross-shape attention with $K = 4$, self-shape attention, and the original MID-FC. We can drive similar conclusions – our method improves the consistency of segmentation especially for fine-grained details e.g., the lower bars of the table, the bottom of the lamp, the handle of the sword, and the sides of the seat.

[4]As of March 20, 2024, according to the latest leaderboard for *3D Semantic Segmentation on PartNet*, available on Papers with Code at the following link: https://paperswithcode.com/sota/3d-semantic-segmentation-on-partnet?p=cross-shape-graph-convolutional-networks

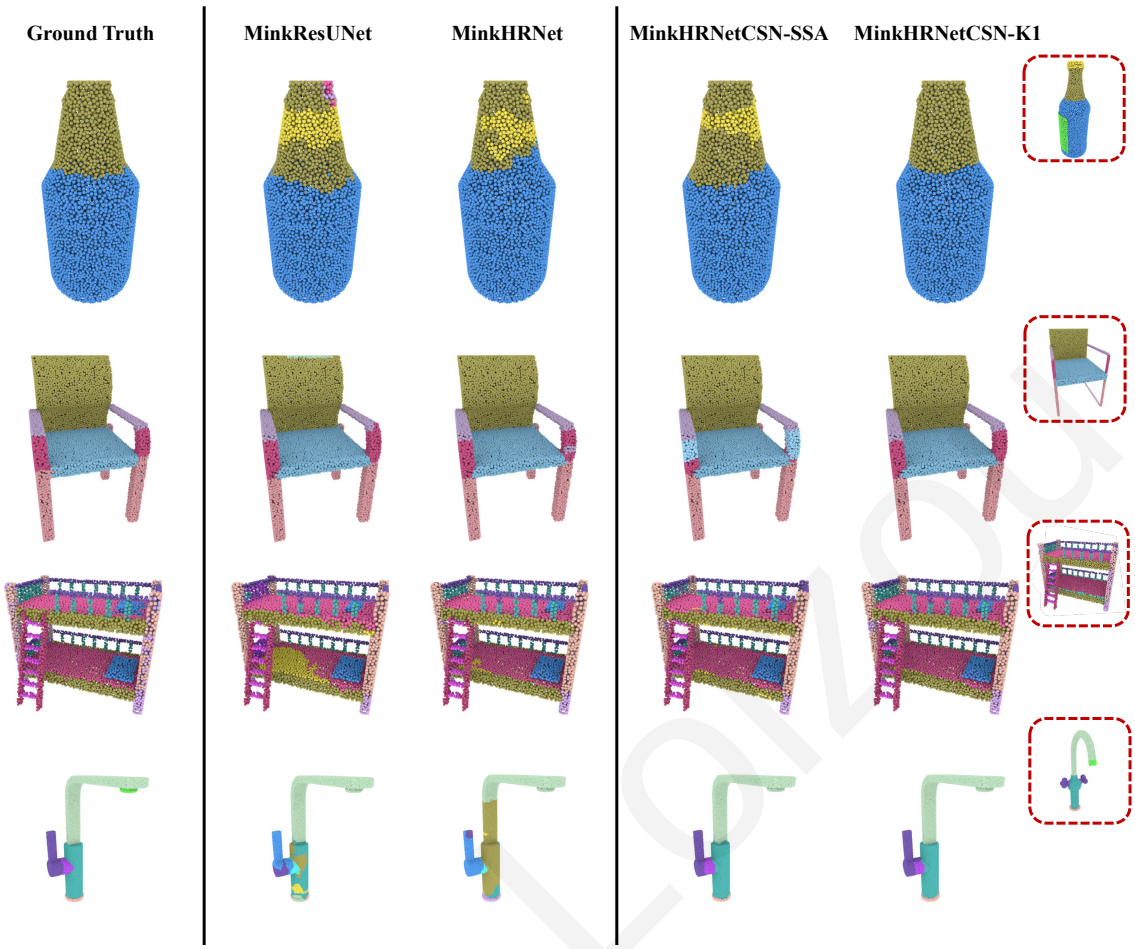

Figure 4.2: Qualitative comparisons for a few characteristic test shapes of PartNet between the original MinkowskiNet for 3D shape segmentation ("MinkResUNet"), our backbone ("MinkHRNet"), and CrossShapeNet (CSN) in case of using self-shape attention alone ("MinkHRNetCSN-SSA") and using cross-shape attention with $K = 1$ key shape per query shape ("MinkHRNetCSN-K1"). The inlet images (red dotted box) show this key shape retrieved for each of the test shapes.

4.2.6 Number of parameters

Our CSA layer adds a relatively small overhead in terms of number of parameters. The MID-FC backbone has $1.8M$ parameters, the MinkHRNet has $24.8M$ parameters, while the CSA layer adds $0.4M$ parameters.

4.2.7 Computation

The cross-shape operation (Equation 4.7) has linear time complexity wrt the number of used key shapes (K) and quadratic time complexity wrt the number of points per shape during both training and testing. To accelerate computation, subsets of

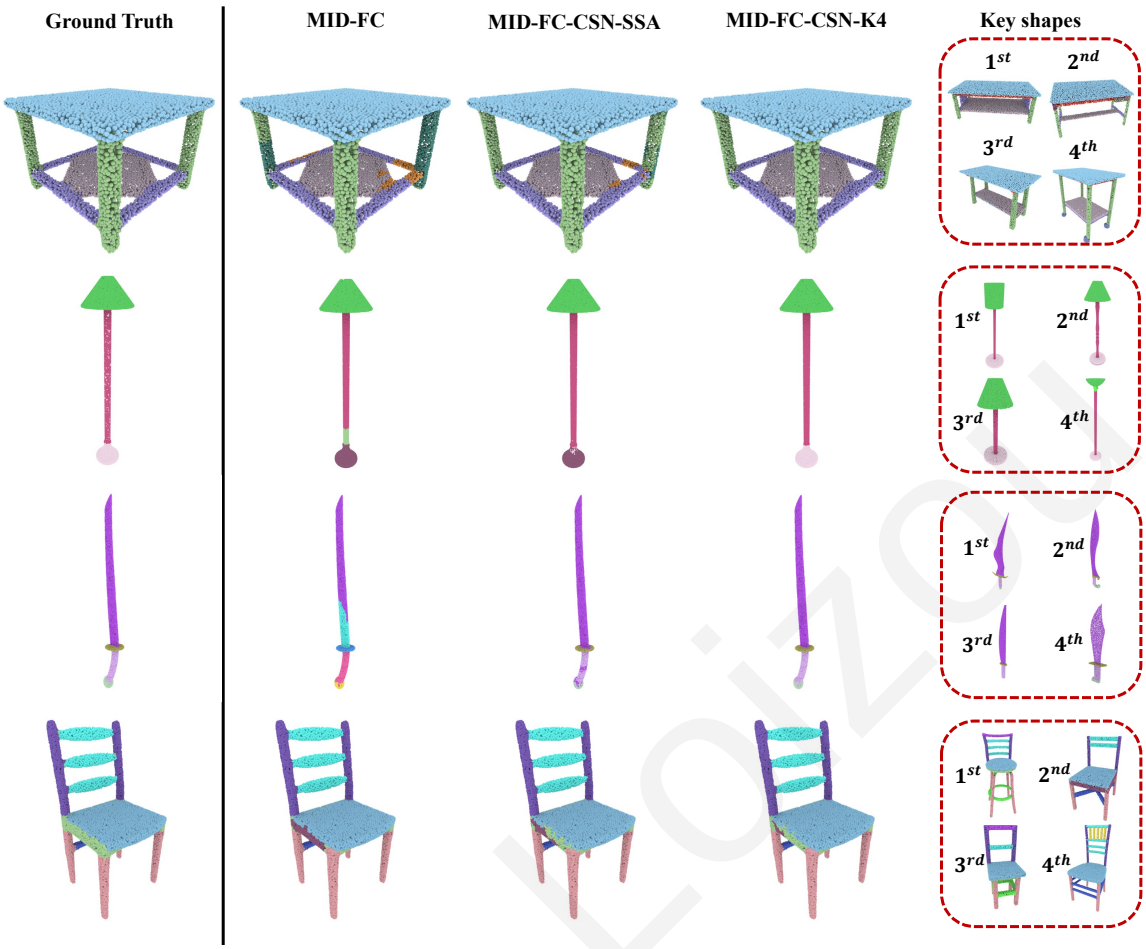

Figure 4.3: Qualitative comparisons for a few characteristic test shapes of PartNet between the original MID-FC network for 3D shape segmentation ("MID-FC") [226], and CrossShapeNet (CSN) in case of using self-shape attention alone ("MID-FC-CSN-SSA") and using cross-shape attention with $K = 4$ key shape per query shape ("MID-FC-CSN-K4"). The last column shows the key shapes and their ordering, retrieved for each test shape.

points could be used as key points, as done in sparsified formulations of attention [257, 134]. The construction of the shape graph during training has quadratic time complexity wrt the number of the training shapes in the input collection. This is because the retrieval measure (Equation 4.14) must be evaluated for all pairs of training shapes in our current implementation. For example, our CSA layer for $K = 1$ requires approximately 105 hours on a NVidia $V100$ to train for the largest PartNet category (3.5× more compared to training either backbone alone) due to the iterative graph construction and fine-tuning discussed in Section 4.1.3. A more efficient implementation could involve a more hierarchical approach e.g., perform clustering to select only a subset of training shapes as candidate key shapes for

the retrieval measure. During inference, the time required per test shape exhibits linear complexity wrt the number of training shapes used for retrieval. In our experiments, testing time ranges from 0.7 sec ("Dish" class with the smallest number of shapes) to 7.8 sec ("Table" class with the largest number of shapes).

4.3 Discussion

This chapter presents a method that enables interaction of point-wise features across different shapes in a collection. The interaction is mediated through a new cross-shape attention mechanism, that assesses the degree of feature interaction among shapes. In addition, a retrieval measure was introduced, that leverages self-shape representations to retrieve key shapes, that are compatible for cross-shape operations. Our experiments show improvements of this interaction in the case of fine-grained shape segmentation.

Limitations. Our method has also limitation. The performance increase comes with a higher computational cost at training and test time. It would be interesting to explore if further performance gains can be achieved through self-supervised pre-training [241, 226, 195] that could in turn guide our attention mechanism. Sparsifying the attention and accelerating the key shape retrieval mechanism would also be important to decrease the time complexity. Another future research direction is to explore how to generalize the cross-shape attention mechanism from single shapes to entire scenes.

Chapter 5

BuildingNet: Learning to Label 3D Buildings

Architecture is a significant application area of 3D vision. There is a rich body of research on autonomous perception of buildings, led in large part by digital map developers seeking rich annotations and 3D viewing capabilities for building exteriors [62], as well as roboticists who design robots to operate in building interiors (e.g. [193]). Recent advances in AR/VR also rely on computer-aided building analysis [31]. Early work on digital techniques for architectural design, including freeform design explorations as well as full-fledged constructions [63], led to the current ubiquity of computational design tools in architectural studios. In addition, computers can automate the processing of architectural data such as photographs, satellite images and building plans, for archival and analytical purposes (e.g. [259, 148]).

Thus, there is significant incentive to apply modern data-driven geometry processing to the analysis of buildings. However, while buildings are bona fide geometric objects with well-established design principles and clear ontologies, their structural and stylistic complexity is typically greater than, or at least markedly *different from*, those of shapes in common 3D datasets like ShapeNet [28] and ScanNet [37]. This makes them challenging for standard shape analysis pipelines, both for discriminative tasks such as classification, segmentation and point correspondences, as well as for generative tasks like synthesis and style transfer. Further, data-driven methods demand data, and to the best of our knowledge there are no large-scale, consistently-annotated, public datasets of 3D building models.

In this chapter, we present BuildingNet[1], the first publicly available large-scale dataset of annotated 3D building models whose exteriors and surroundings are consistently labeled. The dataset provides 513K annotated mesh primitives across

2K building models. We include a benchmark for mesh and point cloud labeling, and evaluate several mesh and point cloud labeling networks. These methods were developed primarily for smaller single objects or interior scenes and are less successful on architectural data. This benchmark featured also as a workshop challenge in the Second Workshop on Structural and Compositional Learning on 3D Data.

In addition, we introduce a graph neural network (GNN) that labels building meshes by analyzing spatial and structural relations of their geometric primitives. Our GNN treats each subgroup as a node, and takes advantage of relations, such as adjacency and containment, between pairs of nodes. Neural message passing in the graph yields the final mesh labeling. Our experiments show that this approach yields significantly better results for 3D building data than prior methods. To summarize, the contributions[2] of this chapter are:

- The first large-scale, publicly available 3D building dataset with annotated parts covering several common categories, in addition to a benchmark.
- A graph neural network that leverages pre-existing noisy subgroups in mesh files to achieve state-of-the-art results in labeling building meshes.

5.1 Dataset and Benchmark

In contrast to 3D models of small and mid-scale objects, such as tools, furniture, and vehicles encountered in existing 3D shape segmentation benchmarks, such as ShapeNet [249, 250] and PartNet [158], buildings tend to contain much richer structure, as indicated by their mesh metadata. For example, one common type of metadata are groupings of polygon faces, commonly known as *mesh subgroups* [158], which correspond to geometric primitives and modeling operations used by modelers while designing shapes. These subgroups often correspond to "pieces" of semantic parts e.g., a window is made of subgroups representing individual

[1]The work presented in this chapter is also published in the proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021. Project page: https://buildingnet.org. This is the authors' version of the work.

[2]This research was carried out in collaboration with Pratheba Selvaraju, Ph.D. student at the University of Massachusetts Amherst, who has made an equal contribution to this work.

Category	# build.	# train.	# val.	# test
Residential	1266	1007 (62.9%)	133 (66.5%)	126 (63.0%)
Religious	469	386 (24.1%)	38 (19.0%)	45 (22.5%)
Commercial	131	104 (6.5%)	16 (8.0%)	11 (5.5%)
Military	73	58 (3.6%)	5 (2.5%)	10 (5.0%)
Public	61	45 (2.8%)	8 (4.0%)	8 (4.0%)
Total	2,000	1600 (80%)	200 (10%)	200 (10%)

Table 5.1: *From left to right:* number of models per basic building category, number and percentage of training, hold-out validation and test buildings

horizontal and vertical frame pieces or glass parts. The average number of mesh subgroups per object at the last level of group hierarchy in the largest shape segmentation benchmark (PartNet [158]) is 24.4, and the median is 11. In our dataset, the average number of mesh subgroups per building is 25.5x larger (623.6 subgroups), while the median is 44x larger (497.5 subgroups; see Table 5.2 for more subgroup statistics per building category). We note that these numbers include only building exteriors i.e., without considering building interiors (e.g, indoor furniture).

5.1.1 BuildingNet dataset

To create our dataset, we mined building models from the 3D Warehouse repository [215]. Mining was driven by various quality checks e.g., excluding low-poly, incomplete, untextured meshes, and meshes with no or too few subgroups. We also categorized them into basic building categories following the Wikipedia's article on "list of building types" [235] and an Amazon MTurk questionnaire. Table 5.1 provides statistics per basic building category in our dataset and its splits. Since we aimed to gather annotations of building exteriors, during a pre-processing step we removed interior structure from each building. This was done by performing exhaustive ray casting originating from mesh faces of each subgroup and checking if the rays were blocked. We also used ray casting to orient faces such that their normals are pointing outwards [211].

Category	num# models	avg# subgrps	med# subgrps	min# subgrps	max# subgrps	avg# un. subgrps	med# un. subgrps	min# un. subgrps	max# un. subgrps	avg# un. l.subgrps	med# un. l.subgrps	min# un. l.subgrps	max# un. l.subgrps
Residential	1,424	678.7	547	83	1989	167.1	144	61	920	61.4	50.0	7	613
Religious	540	487.0	348	93	1981	139.9	129	65	667	47.2	45.0	7	139
Commercial	153	723.4	606	90	1981	159.8	139	70	907	49.4	44.0	3	223
Military	85	609.8	485	125	1786	193.0	166	76	590	38.6	37	2	92
Public	67	628.8	480	118	1822	144.4	123	75	618	43.0	43.0	8	106
Total	2,000	623.6	497.5	83	1989	160.5	140	61	920	55.9	47.0	2	613

Table 5.2: Statistics for each building category regarding mesh subgroups *From left to right:* building category, total number of models, average/median/minimum/maximum number of mesh subgroups over the category's models, average/median/minimum/maximum number of unique (non-duplicate) subgroups, average/median/minimum/maximum number of annotated unique mesh subgroups.

Category	num# models	avg# l.comp.	med# l.comp.	min# l.comp.	max# l.comp.	avg# un. l.comp.	med# un. l.comp.	min# un. l.comp.	max# un. l.comp.
Residential	1,424	321.8	243.0	13	1970	46.1	42.0	8	371
Religious	540	272.2	184.0	18	1469	37.7	35.0	6	135
Commercial	153	408.0	296.0	4	1680	44.6	39.0	3	247
Military	85	295.3	210.0	40	1200	30.5	28.0	2	107
Public	67	378.4	263.0	36	1667	39.3	33.0	7	252
Total	2,000	316.6	231.0	4	1970	43.2	39.0	2	371

Table 5.3: Statistics per building category regarding components (merged adjacent mesh subgroups). *From left to right:* building category, total number of models, average/median/minimum/maximum number of annotated components per model, average/median/minimum/maximum number of annotated unique (non-duplicate) components per model.

Part labels. To determine a set of *common* labels required to annotate building exteriors, we launched an initial user study involving a small subset of 100 buildings across all classes and 10 participants with domain expertise (graduate students in civil engineering and architecture). We selected a list of 31 frequently entered tags to define our label set (see Table 5.4).

Data gathering. We gathered annotations for 2K buildings. Each building was annotated by 5 different, qualified MTurkers (10K annotations in total). We accepted a label for each subgroup if a majority of at least 3 MTurkers out of 5 agreed on it. Figure 5.1 shows a histogram displaying the distribution of buildings (vertical axis) for different bins of percentage of surface area labeled with achieved majority (horizontal axis). All buildings in our dataset have labeled area more than 50%, and most have $>$ 80% area labeled. In terms of annotator consistency, i.e., the percentage of times that the subgroup label selected by a qualified MTurker agreed with the majority, we found that it is 92.0%, indicating that the workers were highly consistent. Our resulting 2K dataset has 513,087 annotated mesh subgroups, and 291,998 annotated components (after merging adjacent subgroups with the same label). The number of unique annotated subgroups and components are 111,832 and 86,492 respectively. Table 5.2 reports statistics on the number of subgroups per building category, unique subgroups (counting repeated subgroups with exactly the same mesh geometry as one unique subgroup), and number of annotated subgroups. We note that there were often subgroups that represented tiny, obscure pieces (e.g., subgroups with a few triangles covering a tiny area of a wall, beam, or frame), and these were often not labeled by annotators. Moreover, Table 5.3 presents more statistics on the labeled components (merged, adjacent subgroups with the same label) of the 2K building dataset per each basic category. Table 5.4 shows labeled component statistics per part label.

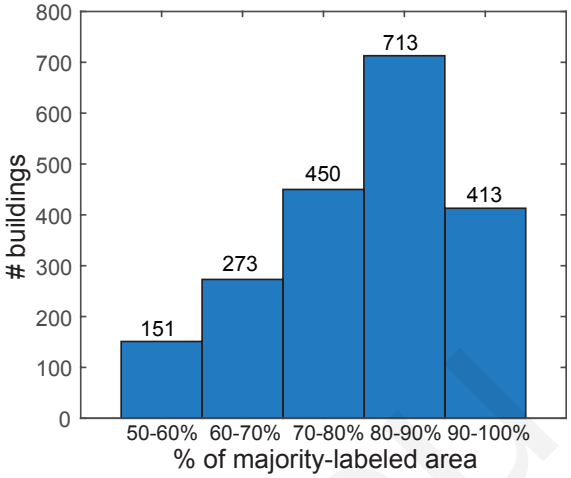

Figure 5.1: Distribution of buildings with achieved labeling majority

Splits. We split our dataset into 1600 buildings for training, 200 for validation, 200 for testing (80/10/10% proportion). We created the splits such that (a) the distribution of building classes and parts is similar across the splits (Tables 5.1 and 5.4) and (b) test buildings have high majority-labeled area ($>$ 85%) i.e., more complete labelings for evaluation.

Label	# labeled comp.	# in training split (%)	# in validation split (%)	# in test split (%)
Window	140,972	109,218 (47.8%)	15,740 (55.1%)	16,014 (46.0%)
Plant	26,735	20,974 (9.2%)	1,870 (6.5%)	3,891 (11.2%)
Wall	22,814	18,468 (8.1%)	2,270 (7.9%)	2,076 (6.0%)
Roof	12,881	10,342 (4.5%)	1,396 (4.9%)	1,143 (3.3%)
Banister	13,954	9,678 (4.2%)	1,467 (5.1%)	2,809 (8.1%)
Vehicle	8,491	7,421 (3.2%)	716 (2.5%)	354 (1.0%)
Door	9,417	7,363 (3.2%)	785 (2.7%)	1,269 (3.6%)
Fence	5,932	5,637 (2.5%)	88 (0.3%)	207 (0.6%)
Furniture	6,282	5,000 (2.2%)	575 (2.0%)	707 (2.0%)
Column	6,394	4,870 (2.1%)	623 (2.2%)	901 (2.6%)
Beam	6,391	4,814 (2.1%)	437 (1.5%)	1,140 (3.3%)
Tower	4,478	3,873 (1.7%)	286 (1.0%)	319 (0.9%)
Stairs	4,193	2,960 (1.3%)	472 (1.7%)	761 (2.2%)
Shutters	2,275	1,908 (0.8%)	77 (0.3%)	290 (0.8%)
Ground	2,057	1,572 (0.7%)	229 (0.8%)	256 (0.7%)
Garage	1,984	1,552 (0.7%)	182 (0.6%)	250 (0.7%)
Parapet	1,986	1,457 (0.6%)	153 (0.5%)	376 (1.1%)
Balcony	1,847	1,442 (0.6%)	199 (0.7%)	206 (0.6%)
Floor	1,670	1,257 (0.5%)	205 (0.7%)	208 (0.6%)
Buttress	1,590	1,230 (0.5%)	53 (0.2%)	307 (0.9%)
Dome	1,327	1,098 (0.5%)	114 (0.4%)	115 (0.3%)
Corridor	1,257	1,008 (0.4%)	113 (0.4%)	136 (0.4%)
Ceiling	1,193	903 (0.4%)	111 (0.4%)	179 (0.5%)
Chimney	1,090	800 (0.4%)	103 (0.4%)	187 (0.5%)
Gate	827	737 (0.3%)	65 (0.2%)	25 (0.1%)
Lighting	921	702 (0.3%)	51 (0.2%)	168 (0.5%)
Dormer	798	601 (0.3%)	48 (0.2%)	149 (0.4%)
Pool	742	544 (0.2%)	78 (0.3%)	120 (0.3%)
Road	590	444 (0.2%)	55 (0.2%)	91 (0.3%)
Arch	524	393 (0.2%)	11 (0.03%)	120 (0.3%)
Awning	386	295 (0.1%)	19 (0.1%)	72 (0.2%)
Total	291,998	228,561	28,591	34,846

Table 5.4: Number of labeled components per part label in our dataset, along with their number and frequency in the training split, hold-out validation, and test split.

5.1.2 BuildingNet benchmark

We provide two tracks in our benchmark. In the first track, called "BuildingNet-Mesh", algorithms can access the mesh data, including subgroups. In this aspect, they can take advantage of any pre-existing mesh structure common in 3D building models. The algorithms are evaluated in two conditions: when the RGB texture is available, and when it is not. In the second condition, algorithms must label the building using only geometric information. The second track, called "BuildingNet-Points", is designed for large-scale point- based processing algorithms that must deal with unstructured point cloud data without access to mesh structure or subgroups, which is still challenging even in the noiseless setting. To this end, for each mesh, we sample 100K points with Poisson disc sampling [50], to achieve a near-uniform sampling similarly to PartNet [158]. The point normals originate from triangles. There are also two evaluation conditions: with and without RGB color for points.

5.2 BuildingGNN

We now describe a graph neural network[3] for labeling 3D meshes by taking advantage of pre-existing mesh structure in the form of subgroups. The main idea of the network is to take into account spatial and structural relations between subgroups to promote more coherent mesh labeling. The input to our network is a 3D building mesh with subgroups $\mathcal{C} = \{c_i\}_{i=1}^{N}$, where N is the number of subgroups, and the output is a label per subgroup. In the next section, we describe how the graph representing a building is created, then we discuss our GNN architecture operating on this graph.

5.2.1 Graph nodes

For each 3D building model, we create a node for each mesh subgroup. Nodes carry an initial raw representation of the subgroup. Specifically, we first sample the mesh with 100K points (same point set used in the "BuildingNet-Points" track), then process them through the 3D sparse convolutional architecture of

[3]The graph neural network was implemented by Pratheba Selvaraju with the help and feedback from other authors, including the author of this thesis.

Minkowski network (MinkowskiUNet34 variant [34]). We also experimented using PointNet++ [173]. We extract per-point features from the last layer of these nets, then perform average pooling over the points originating from the faces of the subgroup to extract an initial node representation. We concatenate this representation with the 3D barycenter position of the subgroup, its mesh surface area, and the coordinates of the opposite corners of its Oriented Bounding Box (OBB) so that we capture its spatial dimensions explicitly. The combination of the above features in the resulting 41D node representation $\mathbf{n}_i$ yielded better performance in our experiments.

5.2.2 Proximity edges

Driven by the observation that nearby subgroups tend to have the same label (e.g., adjacent pieces of glass or frame are labeled as "window"), or related labels (e.g., windows are often adjacent to walls), we create edges for pairs of subgroups that capture their degree of proximity. To avoid creating an overly dense graph, which would pose excessive memory overheads for the GNN, we created edges for pairs of subgroups whose distance was up to 10% of the average of their OBB diagonals. Relaxing this bound did not improve results. To avoid a hard dependency on a single threshold, and to capture the degree of subgroup proximity at multiple scales, we computed the percentage of point samples of each subgroup whose distance to the other subgroup is less than 1%, 2.5%, 5%, and 10% of the average of their OBB diagonals. Given a pair of subgroups (c_i, c_j), this results in a $4D$ edge raw representation $\mathbf{e}_{i,j}^{(prox)}$, where each entry approximates the surface area percentage of c_i proximal to c_j at a different scale. Similarly, we compute a $4D$ representation $\mathbf{e}_{j,i}^{(prox)}$ for the opposite edge direction.

5.2.3 Support edges

Certain arrangements of labels are often expected along the upright axis of the building e.g., the roof is on top of walls. We create a "supporting" edge for each subgroup found to support another subgroup, and "supported-by" edges of opposite direction for each subgroup found to be supported by another subgroup. The edges are created by examining OBB spatial relations. Specifically, as in the case of proximity edges, we compute a multi-scale 4D edge raw representation $\mathbf{e}_{i,j}^{(ontop)}$

measuring the area percentage of c_i's bottom OBB face lying above the c_j's top OBB face for different distances 1%, 2.5%, 5%, 10% of the average of the two OBB's heights. We also compute a 4D edge raw representation $\mathbf{e}_{j,i}^{(below)}$ corresponding to the the surface area percentage of c_j's top OBB face lying beneath the c_i's bottom OBB face.

5.2.4 Similarity edges

Subgroups placed under a symmetric arrangement often share the same label (e.g., repeated windows along a facade). We create an edge per pair of subgroups capturing repetition. For each pair of subgroups, we compute the bidirectional Chamfer distance between their sample points after rigid alignment. To promote robustness to any minor misalignment, or small geometric differences between subgroups, we create similarity edges if the Chamfer distance $d_{i,j}$ is less than 10% of the average of their OBB diagonals. Increasing this bound did not improve results. We normalize it within $[0,1]$, where 1.0 corresponds to the above upper bound, and use $\mathbf{e}_{i,j}^{(symm)} = 1 - d_{i,j}$ as raw similarity edge representation. We also use the same representation for this opposite direction: $\mathbf{e}_{j,i}^{(symm)} = \mathbf{e}_{i,j}^{(symm)}$.

5.2.5 Containment edges

Driven by the observation that parts, such as doors or windows, are enclosed by, or contained within other larger parts, such as walls, we create edges for pairs of subgroups capturing their degree of containment. For each pair of subgroups, we measure the amount of c_i's volume contained within the c_j's OBB and also their volume Intersection over Union as a 2D edge representation $\mathbf{e}_{i,j}^{(contain)}$ (and similarly for the opposite edge direction).

5.2.6 Network architecture

The network updates node and edge representations at each layer inspired by neural message passing [96]. Figure 5.2 shows one such layer of message passing. Below we explain our architecture at test time.

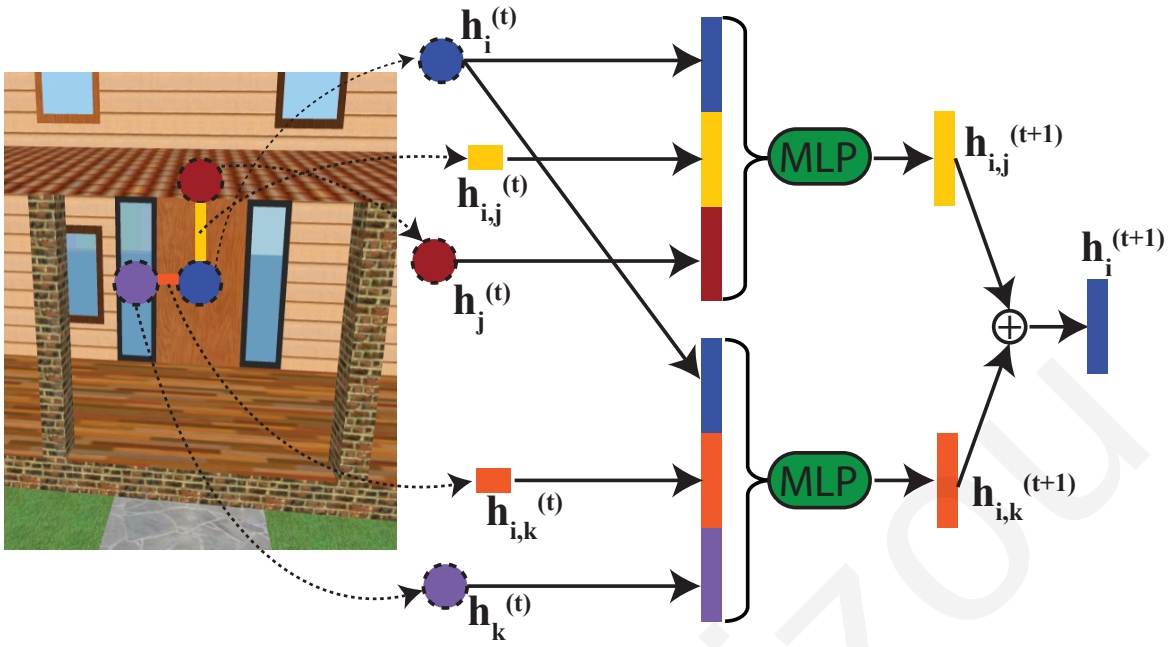

Figure 5.2: Architecture of the message passing layer. The door representation (blue node) is updated from a support edge (yellow edge) to a roof component (red node) and a proximity edge (orange edge) to a window (purple node).

Initialization. Given a pair of subgroups c_i and c_j, we first concatenate their edge representations across all types:

$$\mathbf{e}_{i,j} = \{\mathbf{e}_{i,j}^{(prox)}, \mathbf{e}_{i,j}^{(ontop)}, \mathbf{e}_{i,j}^{(below)}, \mathbf{e}_{i,j}^{(contain)}, \mathbf{e}_{i,j}^{(sim)}\} \tag{5.1}$$

We note that some of the edge types might not be present between two subgroups based on our graph construction. The entries of our edge representations indicate degree of proximity, support, containment, or similarity, and are normalized between $[0,1]$ by definition. Zero values for an edge representation of a particular type indicate non-existence for this type. Each raw edge representation $\mathbf{e}_{i,j}$ is initially processed by a MLP to output a learned representation $\mathbf{h}_{i,j}^{(0)} = MLP(\mathbf{e}_{i,j}; \mathbf{w}^{(0)})$, where $\mathbf{w}^{(0)}$ are learned MLP parameters. The initial node representation is $\mathbf{h}_i^{(0)} = \mathbf{n}_i$.

Node and edge updates. Each of the following layers process the node and edge representations of the previous layer through MLPs and mean aggregation respectively:

$$\mathbf{h}_{i,j}^{(l+1)} = MLP(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{h}_{i,j}^{(l)}; \mathbf{w}^{(l)}) \tag{5.2}$$

$$\mathbf{h}_i^{(l+1)} = \frac{1}{|N(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{h}_{i,j}^{(l+1)} \tag{5.3}$$

where $\mathbf{w}^{(l)}$ are learned MLP parameters. We use 3 layers of node/edge updates. Finally, the last GNN layer processes the node representations of the third layer, and decodes them to a probability per label using a MLP and softmax. Details about the architecture are in Section C.1 (Appendix C).

Training loss. Since some parts are more rare than others, as shown in Table 5.4, we use a weighted softmax loss to train our network, where weights are higher for rarer parts to promote correct labeling for them (i.e., higher mean Part IoU). For each building, the loss is $L = -\sum_{c_i \in \mathcal{L}} w_l \cdot \hat{\mathbf{q}}_i \log \mathbf{q}_i$, where $\mathcal{L}$ is the set of all annotated subgroups in the building, $\hat{\mathbf{q}}_i$ is the ground-truth one-hot label vector for subgroup c_i, $\mathbf{q}_i$ is its predicted label probabilities, and w_l is the weight for the label empirically set to be the log of inverse label frequency (i.e., a smoothed version of inverse frequency weights similarly to [156]). We use the same loss to train the MinkowskiNet used in our node representation: the loss is simply applied to points instead of subgroups. We experimented with other losses, such as the focal loss [130] and the class-balanced loss [36], but we did not find significant improvements in our dataset (see Section C.2, Appendix C).

Implementation details. Training of the BuildingGNN is done through the Adam optimizer [95] with learning rate 0.0001, beta coefficients are $(0.9, 0.999)$ and weight decay is set to 10^{-5}. We pick the best model and hyper-parameters based on the performance in the holdout validation split. Our method is implemented[4] in PyTorch [167].

5.3 Results

We now discuss our evaluation protocol, then show qualitative and quantitative results for our benchmark tracks.

5.3.1 Evaluation protocol

Since most part classes are commonly encountered across different building categories (e.g., walls, doors, windows), all evaluated methods are trained across all

[4]The implementation is available at https://github.com/buildingnet/buildingnet_dataset.

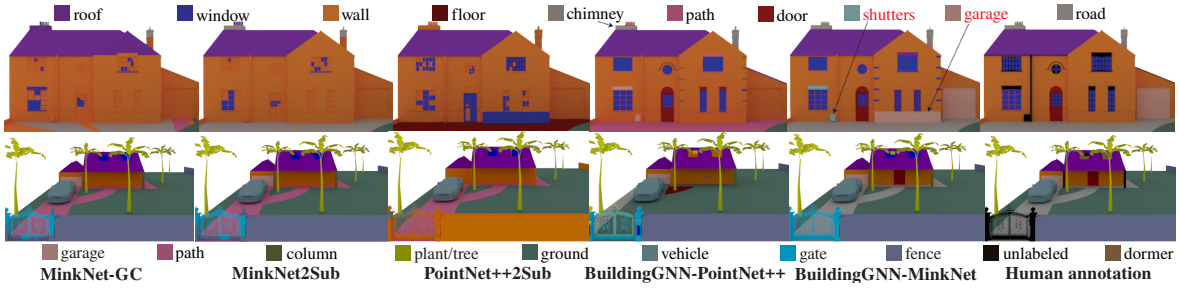

Figure 5.3: Comparisons with other methods. Despite a few errors (red text), the BuildingGNN is closer to human annotations.

five building categories (i.e., no category-specific training). Methods must also deal with the part class imbalance of our dataset. For evaluation in the point cloud track ("BuildingNet-Points"), we use the metrics of mean shape IoU and part IoU (see Equations 3.8 and 3.9), as in PartNet [158]. We also report the per-point classification accuracy. For the mesh track ("BuildingNet-Mesh"), the same measures are applied on triangles. However, since triangles may differ in area, we propose the following IoU variations, where the contribution of each triangle is weighted by its face area. Given all the annotated triangles across all buildings of the test dataset T_D, the part IoU for a label l is measured as:

$$IoU(l) = \frac{\sum_{t \in T_D} a_t \cdot ([y_t == l] \wedge [\hat{y}_t == l])}{\sum_{t \in T_D} a_t \cdot ([y_t == l] \vee [\hat{y}_t == l])} \tag{5.4}$$

where $\hat{y}_t$ is the majority-annotated (ground-truth) label for a triangle $t \in T_d$, y_t is the predicted label for it, and $[\cdot]$ evaluates the above binary expressions. The shape IoU for a shape s with a set of annotated triangles T_s is measured as:

$$IoU(s) = \frac{1}{|L_s|} \sum_{l \in L_s} \frac{\sum_{t \in T_s} a_t \cdot ([y_t == l] \wedge [\hat{y}_t == l])}{\sum_{t \in T_s} a_t \cdot ([y_t == l] \vee [\hat{y}_t == l])} \tag{5.5}$$

where L_s is the set of all labels present in the annotations or predictions for that shape. We also report the per-triangle classification accuracy weighted by face area [89].

5.3.2 "BuildingNet-Points" track

As an initial seed for the leaderboard of this track, we evaluated three popular nets able to handle our $100K$ point sets: PointNet++ [173], MID-FC [226], and MinkowskiUNet34 [34]. We also tried other point-based networks e.g., DGCNN [231], but were unable to handle large point clouds due to excessive memory requirements (see Section D.2, Appendix D for more discussion). All networks were

Method	**n**?	**c**?	Part IoU	Shape IoU	Class acc.
PointNet++	✓	×	8.8%	12.2%	52.7%
MID-FC(nopre)	✓	×	20.9%	19.0%	59.4%
MinkNet	✓	×	**26.9%**	**22.2%**	**62.2%**
PointNet++	✓	✓	14.1%	16.7%	59.5%
MID-FC(nopre)	✓	✓	25.0%	22.3%	63.2%
MinkNet	✓	✓	**29.9%**	**24.3%**	**65.5%**

Table 5.5: "BuildingNet-Points" track results. The column '**n**?' means whether networks use point normals, and the column '**c**?' means whether they use RGB color as input.

trained under the same augmentation scheme (12 global rotations per building and small random translations). For all networks, we experimented with SGD, Adam [95], with and without warm restarts [144], and selected the best scheduler and hyperparameters for each of them based on the validation split. We did not use any form of pre-training. Table 5.5 reports the results. We observe that the MinkowskiNet offers the best performance. We also observe that the inclusion of color tends to improve performance e.g., we observe a 3% increase in Part IoU for MinkowskiNet. Another observation is that compared to PartNet classes, where the Part IoU ranges between $\sim 30 - 70\%$ for PointNet++, the performance in our dataset is much lower: PointNet++ has 14.1% Part IoU. Even for the best performing method (MinkowskiNet), the part IoU is still relatively low (29.9%), indicating that our building dataset is substantially more challenging.

5.3.3 "BuildingNet-Mesh" track

For our mesh track, we first include a number of baselines which rely on networks trained on the point cloud track, then transferring their results to meshes. One strategy for this transfer is to build correspondences between mesh faces and nearest points. Specifically, for each point we find its nearest triangle. Since some triangles might not be associated with any points, we also build the reverse mapping: for each triangle, we find its closest point. In this manner, every triangle t has a set of points P_t assigned to it with the above bi-directional mapping. Then we perform

Method	**n**?	**c**?	Part IoU	Shape IoU	Class acc.
PointNet++2Triangle	✓	×	8.8%	13.1%	54.7%
MidFC2Triangle	✓	×	23.1%	22.1%	42.9%
MinkNet2Triangle	✓	×	28.8%	26.7%	64.8%
PointNet++2Sub	✓	×	9.5%	16.0%	57.9%
MidFC2Sub	✓	×	26.4%	28.4%	46.2%
MinkNet2Sub	✓	×	33.1%	36.0%	69.9%
MinkNet-GC	✓	×	29.9%	28.3%	66.0%
BuildingGNN-PointNet++	✓	×	29.0%	33.5%	67.9%
BuildingGNN-MinkNet	✓	×	**40.0%**	**44.0%**	**74.5%**
PointNet2Triangle	✓	✓	14.0%	18.0%	60.7%
MidFC2Triangle	✓	✓	27.3%	26.2%	45.6%
MinkNet2Triangle	✓	✓	32.8%	29.2%	68.1%
PointNet2Sub	✓	✓	16.1%	23.5%	64.8%
MidFC2Sub	✓	✓	30.3%	33.1%	48.6%
MinkNet2Sub	✓	✓	37.0%	39.1%	73.2%
MinkNet-GC	✓	✓	33.8%	31.1%	68.9%
BuildingGNN-PointNet++	✓	✓	31.5%	35.9%	73.9%
BuildingGNN-MinkNet	✓	✓	**42.6%**	**46.8%**	**77.8%**

Table 5.6: "BuildingNet-Mesh" results. PointNet++2Triangle means triangle-pooling with PointNet++ (similarly for others). PointNet2Sub means subgroup-pooling. MinkNet-GC means graph cuts with MinkowskiUNet34 unary terms.

average pooling of the point probabilities per triangle: $\mathbf{q}_t = \sum_{p \in P_t} \mathbf{q}_p / |P_t|$ where $\mathbf{q}_p$ and $\mathbf{q}_t$ are point and triangle probabilities respectively. We report results of these baselines in Table 5.6. We note that we tried max pooling, yet average pooling had better performance (see Section C.3, Appendix C). Another strategy is to aggregate predictions based on mesh subgroups instead of triangles i.e., average probabilities of points belonging to each subgroup. This strategy takes advantage of mesh structure and improves results. Another baseline is Graph Cuts (GC) on the mesh, which has been used in mesh segmentation [89] (see Section C.4, Appendix C for the GC energy). Finally, we report results from our GNN ("BuildingGNN"), using Point-

Net++ or MinkowskiNet node features. The BuildingGNN significantly improves the respective baselines e.g., with color as input, BuildingGNN with PointNet++ features improves Part IoU by 15.4% over the best PointNet++ variant, while BuildingGNN with MinkowskiNet features improves Part IoU by 5.6% over the best MinkowskiNet variant. The BuildingGNN with MinkowskiNet features performs the best with or without color. Section D.1, Appendix D includes an ablation study showing that each edge type in the BuildingGNN improves performance over using node features alone, while the best model is the one with all edges.

5.3.4 Qualitative results

Figure 5.3 shows comparisons of BuildingGNN with other methods. We observe that its predictions are closer to human annotations compared to others. Figure 1.3 presents more results from BuildingGNN.

5.4 The BuildingNet challenge

During the Second Workshop on Structural and Compositional Learning on 3D Data[5], held at the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023[6], we organized the BuildingNet challenge[7]. The objective of this challenge was to advance the field of part segmentation in the context of 3D buildings, represented either as 3D meshes or point clouds. Our aim was to promote the development of robust and efficient algorithms, that can accurately segment different parts of 3D buildings. Through the BuildingNet challenge, participants had the opportunity to benchmark their models against a carefully curated dataset and contribute to the advancement of part segmentation techniques in the domain of 3D data analysis. To facilitate the BuildingNet challenge, we leveraged the power of EvalAI [246], an open-source platform designed for evaluating and comparing machine learning and artificial intelligence algorithms at scale. EvalAI provided us with a robust infrastructure to host the challenge and streamline the evaluation process. Using EvalAI, we were able to create multiple evaluation phases and dataset splits, accommodating the diverse requirements of the challenge. This flex-

[5]The workshop's page is available at https://struco3d.github.io/cvpr2023.

[6]The conference's page is available at https://cvpr.thecvf.com/Conferences/2023.

[7]The challenge's page is available at https://eval.ai/web/challenges/challenge-page/1938.

Category	# build.	# train.	# val.	# test
Residential	1225	976 (65.9%)	129 (69.0%)	120 (66.3%)
Religious	445	366 (24.7%)	36 (19.3%)	43 (23.8%)
Commercial	123	96 (6.5%)	16 (8.6%)	11 (6.1%)
Public	56	43 (2.9%)	6 (3.2%)	7 (3.9%)
Total	1,849	1,481 (80.1%)	187 (10.1%)	181 (9.8%)

Table 5.7: BuildingNet v1 dataset - *From left to right:* number of models per basic building category, number and percentage of training, hold-out validation and test buildings

ibility allowed participants to train and fine-tune their models using the training and validation splits, while keeping the test split concealed until the final evaluation.

5.4.1 Challenge dataset

For the purpose of this challenge, we released a new version (v1) of the BuildingNet dataset, so that participants do not benefit from any pre-existing models trained on the existing data. The BuildingNet v1 dataset includes 1,849 building models with more complete and improved labelings, as the number of annotated mesh primitives and semantic part components (merged adjacent subgroups sharing the same label) is increased to 554,095 and 304,650, respectively. The new version of the dataset is split into 1,481 buildings for training, 187 for validation and 181 for testing (see Table 5.7). These splits maintain the 80/10/10% proportion, as established in the previous iteration of the dataset. Furthermore, the distribution of building classes and parts is preserved similar across the splits (Tables 5.7 and 5.8).

5.4.2 Challenge results

Here we first discuss the evaluation criteria of the challenge and then provide several challenge statistics.

Label	# labeled comp.	# in training split (%)	# in validation split (%)	# in test split (%)
Window	116,936	94,238 (39.3%)	14,331 (48.5%)	8,367 (23.6%)
Plant	55,773	40,696 (17.0%)	4,548 (15.4%)	10,529 (29.8%)
Banister	17,979	13,183 (5.5%)	1,728 (5.8%)	3,068 (8.7%)
Furniture	14,516	11,190 (4.7%)	1,073 (3.6%)	2,253 (6.4%)
Vehicle	13,376	11,104 (4.6%)	722 (2.4%)	1,550 (4.4%)
Wall	13,127	11,068 (4.6%)	1,245 (4.2%)	814 (2.3%)
Fence	12,441	11,210 (4.7%)	496 (1.7%)	735 (2.1%)
Roof	11,117	9,003 (3.8%)	1,396 (4.7%)	718 (2.0%)
Door	7,756	6,315 (2.6%)	567 (1.9%)	874 (2.5%)
Column	6,676	5,093 (2.1%)	645 (2.2%)	938 (2.7%)
Beam	5,703	3,668 (1.5%)	759 (2.6%)	1,276 (3.6%)
Shutters	4,740	3,961 (1.7%)	223 (0.8%)	556 (1.6%)
Tower	3,463	2,951 (1.2%)	143 (0.5%)	369 (1.0%)
Stairs	3,139	2,304 (1.0%)	273 (0.9%)	562 (1.6%)
Balcony	1,941	1,561 (0.7%)	153 (0.5%)	227 (0.6%)
Ground	1,762	1,357 (0.6%)	184 (0.6%)	221 (0.6%)
Garage	1,551	1,229 (0.5%)	166 (0.6%)	156 (0.4%)
Buttress	1,430	1,128 (0.5%)	22 (0.1%)	280 (0.8%)
Gate	1,369	915 (0.4%)	68 (0.2%)	386 (1.1%)
Parapet	1,352	920 (0.4%)	130 (0.4%)	302 (0.9%)
Dome	1,334	1,130 (0.5%)	105 (0.4%)	99 (0.3%)
Floor	1,068	839 (0.4%)	114 (0.4%)	115 (0.3%)
Corridor	1,010	815 (0.3%)	77 (0.3%)	118 (0.3%)
Ceiling	972	724 (0.3%)	90 (0.3%)	158 (0.4%)
Lighting	961	726 (0.3%)	22 (0.1%)	213 (0.6%)
Chimney	834	648 (0.3%)	83 (0.3%)	103 (0.3%)
Dormer	642	489 (0.2%)	45 (0.2%)	108 (0.3%)
Pool	542	403 (0.2%)	51 (0.2%)	88 (0.2%)
Road	427	327 (0.1%)	48 (0.2%)	52 (0.1%)
Arch	402	276 (0.1%)	24 (0.1%)	102 (0.3%)
Awning	311	241 (0.1%)	23 (0.1%)	47 (0.1%)
Total	304,650	239,712	29,554	35,384

Table 5.8: BuildingNet v1 dataset - Number of labeled components per part label in our dataset, along with their number and frequency in the training split, hold-out validation, and test split.

Evaluation criteria. Following the BuildingNet benchmark, as described in Section 5.1.2, this challenge offers two main evaluation phases:

- **BuildingNet-Mesh:** In this phase algorithms can access the mesh data, including subgroups. In this aspect, they can take advantage of any pre-existing mesh structure common in 3D building models.
- **BuildingNet-Points:** This phase is designed for large-scale point-based processing algorithms that must deal with unstructured point cloud data without access to mesh structure or subgroups. For each mesh, 100K points with Poisson disc sampling [50] are obtained, to achieve a near-uniform sampling. For each point cloud, point normals and RGB texture information, that originate from triangles, are also available.

For both phases, participants can submit their predictions either on the validation split (Dev phases) in order to familiarize with the online submission platform, or upload their predictions on the test split (Main phases).

Moreover, this challenges uses the same evaluation protocol as in Section 5.3.1, where for the evaluation of the "BuildingNet-Points" phase, the metrics of mean shape and part IoU are used, along with the per-point classification accuracy. For the "BuildingNet-Mesh" phase, a variation of the previous evaluation metrics is applied on triangles, where the contribution of each triangle is weighted by its face area. The winning team for each phase is selected based on their performance according to the part IoU metric.

Challenge statistics. The challenge took place from March 15 to May 24 of 2023, providing adequate time for participants to familiarize with the dataset and to showcase their expertise in part segmentation of 3D buildings. We received participation from 10 teams, each bringing their unique approache to the challenge. In the "BuildingNet-Points" evaluation phase, we received a total of 5 valid submissions from the participating teams. Table 5.9 reports the best performing submission of each participating team. The winning submission relies on the XL variant of the PointNeXt [174] method. This approach introduces improved training and model scaling strategies in order to boost the performance of the PointNet++ architecture [173] to the state-of-the-art level.

Participant team	Part IoU	Shape IoU	Class acc.
Thea (PointGPT-S_10k [30])	25.5%	20.2%	59.8%
FOO (Minkowski CNN [34])	31.0%	23.2%	64.3%
Host_23798_Team (MinkRes16UNet34C [34])	31.2%	**24.1%**	64.9%
Shicheng Xu (PointNeXt XL [174])	**31.3%**	22.8%	**65.2%**

Table 5.9: "BuildingNet-Points" evaluation phase leaderboard[8].

5.5 Discussion

In this chapter we presented BuildingNet, the first publicly available large-scale dataset of annotated 3D building models whose exteriors and surroundings are consistently labeled. Moreover, we introduces a graph neural network that utilizes the buildings metadata in their mesh representation to improve labeling. We also highlighted the significance of the dataset in advancing methods for part segmentation of 3D buildings, through the BuildingNet Challenge.

Limitations. A future avenue of research is to automatically discover segments in point clouds and embed them into a GNN like ours. Currently, edges between mesh subgroups are extracted heuristically based on structural and spatial relationships. Learning edges and features in an end-to-end manner may improve results. Finally, mesh cutting and hierarchical labeling can lead to richer future dataset versions.

[8]The leaderboard is publicly available at https://eval.ai/web/challenges/challenge-page/1938/leaderboard/4590.

Chapter 6

Conclusion

6.1 Summary

This thesis presents three neural approaches for 3D shape segmentation. Each method employs a graph-based neural network to tackle challenges in segmenting 3D shapes, including enhancing segmentation accuracy around part boundaries, learning more effective point representations for shape understanding, and leveraging relationships between structural parts of large-scale shapes, such as buildings, to further enhance segmentation accuracy. To address these challenges, each method adopts a graph representation of the input, whether at the level of individual points or mesh subgroups within shapes, or at the scale of entire collections of shapes.

First, the PB-DGCNN method is designed to detect part boundaries in 3D shapes represented as point clouds. It achieves this by assigning a probability to each point indicating its likelihood of lying on a boundary between parts within the shape. This approach utilizes graph edge convolutions to integrate local neighborhood information effectively. Additionally, the method expresses each point neighborhood in the local coordinate frame of its centroid, enhancing its ability to handle local variations under different orientations while ensuring that part boundaries remain unaffected. To facilitate the training process of PB-DGCNN, two datasets were preprocessed and annotated with part boundaries, tailored to address two distinct segmentation tasks: geometric segmentation and semantic segmentation. The extracted per-point boundary probabilities can be used in the geometric decomposition of point clouds into regions enclosed by sharp bound-

aries. Alternatively, they can be employed as pairwise terms in a graph-cuts formulation, to enhance existing graph-based semantic segmentation techniques.

Second, the Cross-ShapeNet presents a novel cross-shape attention (CSA) mechanism, enhancing segmentation consistency across a collection of shapes represented as 3D point clouds. This mechanism evaluates the interaction between points across different shapes, allowing features from one shape to influence those of another based on their interaction degree. Moreover, a retrieval mechanism automatically selects shapes compatible for cross-shape feature propagation. This approach significantly enhances segmentation consistency, particularly in capturing fine-grained details, leading to state-of-the-art performance on the PartNet [158] benchmark.

Third, BuildingNet introduces the first large-scale, publicly available dataset of annotated 3D building models whose exteriors and surroundings are consistently labeled. Unlike common 3D datasets which primarily contain simple man-made objects, this dataset provides shapes of higher structural complexity. It comprises 513K annotated mesh primitives across 2K building models spanning various common building categories. Additionally, a benchmark for mesh and point cloud labeling is provided, that also featured as a workshop challenge in the Second Workshop on Structural and Compositional Learning on 3D Data. Moreover, a graph neural network that labels building meshes is presented. BuildingGNN utilizes pre-existing mesh structure in the form of subgroups taking into account their spatial and structural relationships, resulting in more coherent mesh labeling.

These methods address the problem of 3D shape segmentation, utilizing graph-based neural networks, and provide solid foundations for future studies.

6.2 Future work

Although each method presented in this thesis excels within its own context, they still exhibit individual limitations that could be the focal point for future research.

Starting with the analysis of buildings using a graph neural network, our current approach leverages pre-existing mesh subgroups and establishes hand-crafted dependencies based on various spatial and structural relationships like similarity, proximity, support, and containment. While this strategy has improved segmentation results, it relies on the assumption that the 3D representation of the input

shape is already segmented into coherent regions, as seen with mesh primitives common in 3D building models. Moreover, the extraction of graph edges is done in a heuristic manner. A promising future direction would be to explore methods that can automatically extract parts from the 3D shape and then learn edge connections between them in an end-to-end manner, enhancing the robustness of the segmentation process.

These challenges can be addressed through a two-stage procedure. First, a neural network processing a 3D point cloud of a building shape can learn to group points into coherent parts by leveraging foundational image segmentation models like the Segment Anything Model (SAM) [98]. Training this model could follow a self-supervised knowledge distillation process similar to [25], where a point cloud processing network acts as a student network updated based on the teacher network (SAM), aligning the 3D segmentation masks extracted by the student with the 2D part proposals from SAM. During inference, the 3D module will group the building shape into meaningful parts without the need for rendering multi-view images as in [248].

The second stage could involve a pretext task, such as auto-regressive generation similar to [30], allowing for the extraction of dependencies between proposed parts in a learnable manner using architectures that adhere to the attentional flavour of GNNs. The semantic labeling of parts can then be addressed as a low-shot or fully supervised downstream task.

In Cross-ShapeNet we have addressed the propagation of point-wise features between shapes by proposing a cross-shape attention mechanism, which assesses the degree of interaction between them. Shapes compatible for cross-shape attention are retrieved from an input collection utilizing their self-shape attention representations. This method was able to achieve state-of-the-art results in fine-grained part segmentation, but with a high computational cost during training and at test time. Additionally, the current approach is limited to single objects.

First, it would be interesting to investigate more recent attention implementations such as Flash Attention [38], FlatFormer [142] and Point Transformer V3 [238], as they offer reduced computational and space complexity compared to traditional attention layer architectures [217]. Moreover, the shape retrieval module can benefit from recent findings [2] by leveraging deep vision transformer features through a 2D-to-3D lifting process, to extract 3D semantic correspondences between the

query shape and shapes from the input collection.

Recent developments have introduced open-vocabulary approaches that co-embed deep features of 3D points with image and text CLIP features [168], guided by 2D instance masks [164] for zero-shot segmentation of 3D scenes. Integrating such methods could extend Cross-ShapeNet to handle 3D scenes by identifying similar objects suitable for cross-shape attention operations. A more ambitious goal would be to adopt zero-guidance methods [178], eliminating the need for text prompts during test time.

Finally, PB-DGCNN is able to improve segmentation boundaries between semantic parts and geometric primitives, by extracting per-point boundary probabilities. Ideally, boundaries could be extracted in the form of continuous curves instead, such as parametric curves. Following the ideas of PIE-Net [228] and ComplexGen [65], we can use PB-DGCNN as a part boundaries proposal network for guiding the process of extracting parametric curves. In addition, since our boundary detector can also extract boundaries between semantic parts, we can extend the extraction of parametric curves to man-made objects as well, which can facilitate the 3D shape assembly process [128].

These exciting research projects have the potential to push the boundaries and advance the state-of-the-art in 3D shape and scene understanding. They could revolutionize how we perceive and interact with 3D data, opening up new possibilities and applications in various fields.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/` (cit. on p. 28).

[2] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel, "Deep ViT Features as Dense Visual Descriptors", in *Proceedings of the European Conference on Computer Vision Workshops (ECCV Workshops)*, 2022 (cit. on p. 77).

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik, "Contour Detection and Hierarchical Image Segmentation", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 33, no. 5, 2011 (cit. on p. 20).

[4] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese, "3D Semantic Parsing of Large-Scale Indoor Spaces", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2016 (cit. on pp. 1, 15).

[5] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo, "Hierarchical mesh segmentation based on fitting primitives", *The Visual Computer*, vol. 22, 2006 (cit. on p. 10).

[6] Matan Atzmon, Haggai Maron, and Yaron Lipman, "Point Convolutional Neural Networks by Extension Operators", *ACM Transactions on Graphics*, vol. 37, no. 4, 2018 (cit. on p. 12).

[7] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton, "Layer Normalization", *arXiv preprint arXiv:1607.06450*, 2016 (cit. on pp. 44, 49).

[8] Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehrej, Francis Songj, Andy Ballard, Justin Gilmer, George E Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyerj, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu, "Relational inductive biases, deep learning, and graph networks", *arXiv preprint arXiv:1806.01261*, 2018 (cit. on p. 18).

[9] Jensa Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019 (cit. on pp. 1, 15).

[10] Alexander Belyaev, Elena Anoshkina, Ralph Martin, Helmut Bez, and Malcolm Sabin, "Detection of Surface Creases in Range Data", *Mathematics of Surfaces XI. Lecture Notes in Computer Science*, vol. 3604, 2005 (cit. on p. 13).

[11] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani, "DeepEdge: A multi-scale bifurcated deep network for top-down contour detection", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015 (cit. on pp. 19, 20).

[12] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani, "High-for-Low and Low-for-High: Efficient Boundary Detection from Deep Object Features and its Applications to High-Level Vision", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015 (cit. on pp. 19, 20).

[13] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani, "Semantic Segmentation with Boundary Neural Fields", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2016 (cit. on pp. 19, 20).

[14] Paul J. Besl and Ramesh C. Jain, "Segmentation through variable-order surface fitting", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 10, no. 2, 1988 (cit. on p. 9).

[15] Igor Bogoslavskyi and Cyrill Stachniss, "Fast Range Image-Based Segmentation of Sparse 3D Laser Scans for Online Operation", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016 (cit. on p. 12).

[16] Davide Boscaini, Jonathan Masci, Simone Melzi, Michael M Bronstein, Umberto Castellani, and Pierre Vandergheynst, "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks", *Computer Graphics Forum*, vol. 34, no. 5, 2015 (cit. on p. 12).

[17] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2016 (cit. on pp. 12, 13).

[18] Yuri Boykov, Olga Veksler, and Ramin Zabih, "Fast Approximate Energy Minimization via Graph Cuts", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 23, no. 11, 2001 (cit. on pp. 11, 111).

[19] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković, "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges", *arXiv preprint arXiv:2104.13478*, 2021 (cit. on pp. 2, 3, 8, 16–18).

[20] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst, "Geometric Deep Learning: Going beyond Euclidean data", *IEEE Signal Processing Magazine*, vol. 34, no. 4, 2017 (cit. on p. 12).

[21] John Canny, "A Computational Approach to Edge Detection", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 8, no. 6, 1986 (cit. on p. 20).

[22] Anh-Quan Cao, Gilles Puy, Alexandre Boulch, and Renaud Marlet, "PCAM: Product of Cross-Attention Matrices for Rigid Registration of Point Clouds", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021 (cit. on p. 12).

[23] Yan-Pei. Cao, Tao Ju, Jie Xu, and Shin-Min Hu, "Extracting Sharp Features from RGB-D Images", *Computer Graphics Forum*, vol. 36, no. 8, 2017 (cit. on p. 13).

[24] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu, "GCNet: Non-local networks meet squeeze-excitation networks and beyond", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference Workshops (CVPR Workshops)*, 2019 (cit. on p. 41).

[25] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin, "Emerging Properties in Self-Supervised Vision Transformers", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021 (cit. on p. 77).

[26] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler, "Annotating Object Instances with a Polygon-RNN", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on pp. 19, 20).

[27] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang, "Matterport3D: Learning from RGB-D Data in Indoor Environments", in *Proceedings of the IEEE International Conference on 3D Vision (3DV)*, 2017 (cit. on pp. 1, 15).

[28] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu, "ShapeNet: An Information-Rich 3D Model Repository", *arXiv preprint arXiv:1512.03012*, 2015 (cit. on pp. 1, 56).

[29] Chun-Fu (Richard) Chen, Quanfu Fan, and Rameswar Panda, "CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021 (cit. on p. 41).

[30] Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue, "PointGPT: Auto-regressively Generative Pre-training from Point Clouds", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2023 (cit. on pp. 12, 74, 77).

[31] Long Chen, Wen Tang, Nigel John, Tao Ruan Wan, and Jian Jun Zhang, "Context-Aware Mixed Reality: A Learning-Based Framework for Semantic-Level Interaction", *Computer Graphics Forum*, vol. 39, no. 1, 2020 (cit. on p. 56).

[32] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser, "A Benchmark for 3D Mesh Segmentation", *ACM Transactions on Graphics (also in the Proceedings of SIGGRAPH)*, vol. 28, no. 3, 2009 (cit. on pp. 10, 14, 28).

[33] Changhyun Choi, Alexander J. B. Trevor, and Henrik I. Christensen, "RGB-D edge detection and edge-based registration", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013 (cit. on p. 13).

[34] Christopher Choy, JunYoung Gwak, and Silvio Savarese, "4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on pp. 41, 43, 47, 48, 50, 51, 63, 67, 74).

[35] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz,

"Where do people draw lines?", *ACM Transactions on Graphics*, vol. 27, no. 3, 2008 (cit. on p. 28).

[36] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie, "Class-Balanced Loss Based on Effective Number of Samples", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on pp. 66, 107).

[37] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner, "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on pp. 1, 11, 15, 56).

[38] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré, "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2022 (cit. on p. 77).

[39] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2016 (cit. on p. 17).

[40] Ilke Demir, Daniel G. Aliaga, and Bedrich Benes, "Procedural Editing of 3D Building Point Clouds", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015 (cit. on p. 14).

[41] Ilke Demir, Daniel G. Aliaga, and Bedrich Benes, "Coupled Segmentation and Similarity Detection for Architectural Models", *ACM Transactions on Graphics*, vol. 34, no. 4, 2015 (cit. on p. 14).

[42] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi, "CvxNet: Learnable Convex Decomposition", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 12).

[43] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas Guibas, "Vector Neurons: a general framework for SO(3)-equivariant networks", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021 (cit. on p. 11).

[44] James Diebel and Sebastian Thrun, "An Application of Markov Random Fields to Range Sensing", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2005 (cit. on p. 10).

[45] Andrey Dimitrov and Mani Golparvar-Fard, "Segmentation of building point cloud models including detailed architectural/structural features and MEP systems", *Automation in Construction*, vol. 51, 2015 (cit. on p. 1).

[46] Carl Doersch, Ankush Gupta, and Andrew Zisserman, "Crosstransformers: spatially-aware few-shot transfer", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2020 (cit. on p. 41).

[47] Piotr Dollar, Zhuowen Tu, and Serge Belongie, "Supervised Learning of Edges and Object Boundaries", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2006 (cit. on p. 20).

[48] Piotr Dollár and C. Lawrence Zitnick, "Structured Forests for Fast Edge Detection", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2013 (cit. on p. 20).

[49] Richard O Duda and Peter E Hart, "Use of the Hough transformation to detect lines and curves in pictures", *Communications of the ACM*, vol. 15, no. 1, 1972 (cit. on p. 9).

[50] Mohamed S. Ebeida, Andrew A. Davidson, Anjul Patney, Patrick M. Knupp, Scott A. Mitchell, and John D. Owens, "Efficient Maximal Poisson-Disk Sampling", *ACM Transactions on Graphics*, vol. 30, no. 4, 2011 (cit. on pp. 24, 62, 73).

[51] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer, "Point transformer", *IEEE Access*, vol. 9, 2021 (cit. on p. 12).

[52] Pedro F. Felzenszwalb and Daniel P. Huttenlocher, "Efficient Graph-Based Image Segmentation", *International Journal of Computer Vision*, vol. 59, no. 2, 2004 (cit. on p. 10).

[53] Sagi Filin, "Surface clustering from airborne laser scanning data", in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Archives)*, 2002 (cit. on p. 9).

[54] Sagi Filin and Norbert Pfeifer, "Segmentation of airborne laser scanning data using a slope adaptive neighborhood", *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Archives)*, vol. 60, no. 2, 2006 (cit. on p. 9).

[55] Martin A. Fischler and Robert C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Communications of the ACM*, vol. 24, no. 6, 1981 (cit. on p. 9).

[56] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao, "3D-FUTURE: 3D Furniture shape with TextURE", *International Journal of Computer Vision*, vol. 129, no. 12, 2021 (cit. on p. 15).

[57] Kunihiko Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition", *Neural Networks*, vol. 1, no. 2, 1988 (cit. on p. 2).

[58] Natasha Gelfand and Leonidas J. Guibas, "Shape segmentation using local slippage analysis", in *Proceedings of the Symposium on Geometry Processing (SGP)*, 2004 (cit. on p. 9).

[59] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl, "Neural message passing for Quantum chemistry", in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017 (cit. on p. 18).

[60] Aleksey Golovinskiy and Thomas Funkhouser, "Randomized cuts for 3D mesh analysis", *ACM Transactions on Graphics*, vol. 27, no. 5, 2008 (cit. on pp. 10, 12).

[61] Aleksey Golovinskiy and Thomas Funkhouser, "Min-Cut Based Segmentation of Point Clouds", in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCV Workshops)*, 2009 (cit. on pp. 9, 12).

[62] Google, *Google Maps*, 2024 (cit. on p. 56).

[63] Jon Arteta Grisaleña, *The Paradigm of Complexity in Architectural and Urban Design (PhD Thesis)*. University of Alcala, 2017 (cit. on p. 56).

[64] Fabian Groh, Patrick Wieschollek, and Hendrik P. A. Lensch, "Flex-Convolution (Million-Scale Point-Cloud Learning Beyond Grid-Worlds)", in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2018 (cit. on p. 12).

[65] Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo, "ComplexGen: CAD Reconstruction by B-Rep Chain Complex Generation", *ACM Transactions on Graphics (also in the Proceedings of SIGGRAPH)*, vol. 41, no. 4, 2022 (cit. on p. 78).

[66] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu, "PCT: Point cloud transformer", *Computational Visual Media*, vol. 7, no. 2, 2021 (cit. on p. 12).

[67] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun, "Deep Learning for 3D Point Clouds: A Survey", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 43, no. 12, 2021 (cit. on p. 1).

[68] Yun-Chih Guo, Tzu-Hsuan Weng, Robin Fischer, and Li-Chen Fu, "3D semantic segmentation based on spatial-aware convolution and shape completion for augmented reality applications", *Computer Vision and Image Understanding*, vol. 224, 2022 (cit. on p. 1).

[69] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan D. Wegner, Konrad Schindler, and Marc Pollefeys, "SEMANTIC3D.NET: A new large-scale point cloud classification benchmark", in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Archives)*, 2017 (cit. on pp. 1, 15).

[70] Wenkai Han, Chenglu Wen, Cheng Wang, Xin Li, and Qing Li, "Point2node: Correlation learning of dynamic-node for point cloud feature modeling", in *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2020 (cit. on p. 12).

[71] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or, "MeshCNN: A Network with an Edge", *ACM Transactions on Graphics*, vol. 38, no. 4, 2019 (cit. on p. 13).

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2016 (cit. on pp. 2, 103).

[73] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski, "Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds", *ACM Transactions on Graphics*, vol. 37, no. 6, 2018 (cit. on p. 12).

[74] Qibin Hou, Jiang-Jiang Liu, Ming-Ming Cheng, Ali Borji, and Philip H. S. Torr, "Three Birds One Stone: A General Architecture for Salient Object Segmentation, Edge Detection and Skeleton Extraction", *arXiv preprint arXiv:180-3.09860*, 2018 (cit. on pp. 19, 20).

[75] Ruibing Hou, Hong Chang, MA Bingpeng, Shiguang Shan, and Xilin Chen, "Cross Attention Network for Few-shot Classification", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2019 (cit. on p. 41).

[76] Qingyong Hu, Bo Yang, Sheikh Khalid, Wen Xiao, Niki Trigoni, and Andrew Markham, "Sensaturban: Learning semantics from urban-scale photogrammetric point clouds", *International Journal of Computer Vision*, vol. 130, no. 2, 2022 (cit. on p. 1).

[77] Ruizhen Hu, Lubin Fan, and Ligang Liu, "Co-Segmentation of 3D Shapes via Subspace Clustering", *Computer Graphics Forum*, vol. 31, no. 5, 2012 (cit. on p. 14).

[78] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung, "SceneNN: A Scene Meshes Dataset with aNNotations", in *Proceedings of the IEEE International Conference on 3D Vision (3DV)*, 2016 (cit. on p. 15).

[79] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung, "Point-wise Convolutional Neural Network", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018 (cit. on p. 12).

[80] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger, "Densely Connected Convolutional Networks", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on p. 2).

[81] Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G. Kim, and Ersin Yumer, "Learning Local Shape Descriptors from Part Correspondences with Multiview Convolutional Networks", *ACM Transactions on Graphics*, vol. 37, no. 1, 2017 (cit. on p. 12).

[82] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao (Richard) Zhang, "Edge-Aware Point Set Resampling", *ACM Transactions on Graphics*, vol. 32, no. 1, 2013 (cit. on p. 13).

[83] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu, "CCNet: Criss-cross attention for semantic segmentation", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 41).

[84] Sergey Ioffe and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015 (cit. on p. 49).

[85] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu, "Spatial Transformer Networks", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2015 (cit. on p. 22).

[86] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia, "Hierarchical Point-Edge Interaction Network for Point Cloud Semantic Segmentation", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019 (cit. on p. 12).

[87] Oliver van Kaick, Noa Fish, Yanir Kleiman, Shmuel Asafi, and Daniel Cohen-Or, "Shape Segmentation by Approximate Convexity Analysis", *ACM Transactions on Graphics*, vol. 34, no. 1, 2014 (cit. on p. 12).

[88] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri, "3D Shape Segmentation with Projective Convolutional Networks", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on pp. 12, 13).

[89] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh, "Learning 3D Mesh Segmentation and Labeling", *ACM Transactions on Graphics*, vol. 29, no. 4, 2010 (cit. on pp. 11, 12, 14, 67, 69, 111).

[90] Evangelos Kalogerakis, Derek Nowrouzezahrai, Patricio Simari, James McCrae, Aaron Hertzmann, and Karan Singh, "Data-driven curvature for real-time line drawing of dynamic scene", *ACM Transactions on Graphics*, vol. 28, no. 1, 2009 (cit. on p. 13).

[91] Evangelos Kalogerakis, Patricio Simari, Derek Nowrouzezahrai, and Karan Singh, "Robust statistical estimation of curvature on discretized surfaces", in *Proceedings of the Symposium on Geometry Processing (SGP)* (cit. on p. 31).

[92] Andrej Karpathy, Stephen Miller, and Li Fei-Fei, "Object Discovery in 3D Scenes via Shape Analysis", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013 (cit. on p. 12).

[93] Sagi Katz, George Leifman, and Ayellet Tal, "Mesh segmentation using feature point and core extraction", *The Visual Computer*, vol. 21, 2005 (cit. on p. 10).

[94] Sagi Katz and Ayellet Tal, "Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts", *ACM Transactions on Graphics*, vol. 22, no. 3, 2003 (cit. on p. 12).

[95] Diederik Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015 (cit. on pp. 28, 66, 68).

[96] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel, "Neural Relational Inference for Interacting Systems", in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018 (cit. on pp. 18, 64).

[97] Thomas N. Kipf and Max Welling, "Semi-Supervised Classification with Graph Convolutional Networks", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017 (cit. on p. 17).

[98] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick, "Segment Anything", *arXiv preprint arXiv:2304.02643*, 2023 (cit. on p. 77).

[99] Hannes Kisner and Ulrike Thomas, "Segmentation of 3D Point Clouds using a New Spectral Clustering Algorithm Without a-priori Knowledge", in *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, 2018 (cit. on p. 12).

[100] Josef Kittler, "On the accuracy of the Sobel edge detector", *Image and Vision Computing*, vol. 1, no. 1, 1983 (cit. on p. 20).

[101] Roman Klokov and Victor Lempitsky, "Escape from Cells: Deep Kd-Networks for The Recognition of 3D Point Cloud Models", 2017 (cit. on p. 11).

[102] Nikolay Kobyshev, Hayko Riemenschneider, Andras Bodis-Szomoru, and Luc Van Gool, "Architectural decomposition for 3D landmark building understanding", in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2016 (cit. on p. 14).

[103] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo, "ABC: A Big CAD Model Dataset For Geometric Deep Learning", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on pp. 1, 20, 24, 29).

[104] Iasonas Kokkinos, "Boundary Detection Using F-Measure-, Filter- and Feature- (F3) Boost", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010 (cit. on p. 20).

[105] Iasonas Kokkinos, "Pushing the Boundaries of Boundary Detection using Deep Learning", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015 (cit. on pp. 19, 20).

[106] Michael Kolomenkin, Ilan Shimshoni, and Ayellet Tal, "On edge detection on surfaces", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2009 (cit. on p. 13).

[107] Artem Komarichev, Zichun Zhong, and Jing Hua, "A-CNN: Annularly Convolutional Neural Networks on Point Clouds", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 12).

[108] Klaus Koster and Michael Spann, "MIR: an approach to robust clustering-application to range image segmentation", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 22, no. 5, 2000 (cit. on p. 9).

[109] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2012 (cit. on p. 2).

[110] Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru, "Virtual Multi-view Fusion for 3D Semantic Segmentation", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020 (cit. on p. 13).

[111] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", in *Proceedings of the International Conference on Machine Learning (ICML)*, 2001 (cit. on pp. 10, 11).

[112] Alon Lahav and Ayellet Tal, "MeshWalker: Deep Mesh Understanding by Random Walks", *ACM Transactions on Graphics (also in the Proceedings of SIGGRAPH Asia)*, vol. 39, no. 6, 2020 (cit. on p. 13).

[113] Yu-Kun Lai, Shi-Min Hu, Ralph R. Martin, and Paul L. Rosin, "Fast mesh segmentation using random walks", in *Proceedings of the ACM Symposium on Solid and Physical Modeling (SPM)*, 2008 (cit. on p. 10).

[114] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia, "Stratified transformer for 3d point cloud segmentation", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2022 (cit. on p. 12).

[115] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S Davis, "Modeling local geometric structure of 3D point clouds using Geo-CNN", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 12).

[116] Loic Landrieu and Mohamed Boussaha, "Point cloud oversegmentation with graph-structured deep metric learning", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 12).

[117] Loic Landrieu and Martin Simonovsky, "Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018 (cit. on p. 14).

[118] Eric-Tuan Le, Iasonas Kokkinos, and Niloy J. Mitra, "Going Deeper with Lean Point Networks", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2020 (cit. on p. 11).

[119] Truc Le, Yuyan Li, and Ye Duan, "RED-NET: A Recursive Encoder-Decoder Network for Edge Detection", *arXiv preprint arXiv:1912.02914*, 2019 (cit. on pp. 19, 20).

[120] Yann Lecun and Yoshua Bengio, "Convolutional Networks for Images, Speech and Time Series", in *The Handbook of Brain Theory and Neural Networks*. The MIT Press, 1995 (cit. on p. 2).

[121] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He, "Stacked cross attention for image-text matching", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018 (cit. on p. 41).

[122] Guohao Li, Matthias Müller, Guocheng Qian, Itzel Carolina Delgadillo Perez, Abdulellah Abualshour, Ali Kassem Thabet, and Bernard Ghanem, "DeepGCNs: Making GCNS Go as Deep as CNNs", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 45, no. 6, 2023 (cit. on pp. 12, 50, 52).

[123] Jiaxin Li, Ben Chen, and Gim Lee, "SO-Net: Self-Organizing Network for Point Cloud Analysis", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018 (cit. on p. 11).

[124] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas Guibas, "Supervised Fitting of Geometric Primitives to 3D Point Clouds", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 12).

[125] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger, "InteriorNet: Mega-scale Multi-sensor Photo-realistic Indoor Scenes Dataset", in *Proceedings of the BMVA British Machine Vision Conference (BMVC)*, 2018 (cit. on p. 15).

[126] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen, "PointCNN: Convolution On X-Transformed Points", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2018 (cit. on pp. 12, 50, 52).

[127] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra, "GlobFit: Consistently Fitting Primitives by Discovering Global Relations", *ACM Transactions on Graphics*, vol. 30, no. 4, 2011 (cit. on pp. 9, 12).

[128] Yichen Li, Kaichun Mo, Yueqi Duan, He Wang, Jiequan Zhang, Lin Shao, Wojciech Matusik, and Leonidas Guibas, "Category-Level Multi-Part Multi-Joint 3D Shape Assembly", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2024 (cit. on p. 78).

[129] Joseph J. Lim, C. Lawrence Zitnick, and Piotr Dollar, "Sketch Tokens: A Learned Mid-Level Representation for Contour and Object Detection", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2013 (cit. on p. 20).

[130] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár, "Focal Loss for Dense Object Detection", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017 (cit. on pp. 66, 107).

[131] Weiyang Lin, Ali Anwar, Zhan Li, Mingsi Tong, Jianbin Qiu, and Huijun Gao, "Recognition and Pose Estimation of Auto Parts for an Autonomous Spray Painting Robot", *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, 2019 (cit. on p. 1).

[132] Drew Linsley, Junkyung Kim, Vijay Veerabadran, Charles Windolf, and Thomas Serre, "Learning long-range spatial dependencies with horizontal gated recurrent units", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2018 (cit. on pp. 19, 20).

[133] Difan Liu, Mohamed Nabail, Aaron Hertzmann, and Evangelos Kalogerakis, "Neural Contours: Learning to Draw Lines from 3D Shapes", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2020 (cit. on p. 28).

[134] Difan Liu, Sandesh Shetty, Tobias Hinz, Matthew Fisher, Richard Zhang, Taesung Park, and Evangelos Kalogerakis, "ASSET: Autoregressive Semantic Scene Editing with Transformers at High Resolutions", *ACM Transactions on Graphics*, vol. 41, no. 4, 2022 (cit. on p. 54).

[135] Yifan Liu, Wuyang Li, Jie Liu, Hui Chen, and Yixuan Yuan, "GRAB-Net: Graph-Based Boundary-Aware Network for Medical Point Cloud Segmentation", *IEEE Transactions on Medical Imaging*, vol. 42, no. 9, 2023 (cit. on p. 1).

[136] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan, "DensePoint: Learning Densely Contextual Representation for Efficient Point Cloud Processing", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019 (cit. on p. 12).

[137] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan, "Relation-Shape Convolutional Neural Network for Point Cloud Analysis", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 12).

[138] Yun Liu, Ming-Ming Cheng, Deng-Ping Fan, Le Zhang, JiaWang Bian, and Dacheng Tao, "Semantic Edge Detection with Diverse Deep Supervision", *arXiv preprint arXiv:1804.02864*, 2018 (cit. on pp. 19, 20).

[139] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai, "Richer Convolutional Features for Edge Detection", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on pp. 19, 20).

[140] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong, "A Closer Look at Local Aggregation Operators in Point Cloud Analysis", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020 (cit. on pp. 11, 50, 52).

[141] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han, "Point-Voxel CNN for Efficient 3D Deep Learning", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2019 (cit. on p. 11).

[142] Zhijian Liu, Xinyu Yang, Haotian Tang, Shang Yang, and Song Han, "FlatFormer: Flattened Window Attention for Efficient Point Cloud Transformer", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2023 (cit. on p. 77).

[143] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie, "A ConvNet for the 2020s", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2022 (cit. on p. 2).

[144] Ilya Loshchilov and Frank Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017 (cit. on p. 68).

[145] Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer, "Elements of Style: Learning Perceptual Shape Style Similarity", *ACM Transactions on Graphics*, vol. 34, no. 4, 2015 (cit. on p. 15).

[146] Shitong Luo, Jiahan Li, Jiaqi Guan, Yufeng Su, Chaoran Cheng, Jian Peng, and Jianzhu Ma, "Equivariant Point Cloud Analysis via Learning Orientations for Message Passing", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2022 (cit. on p. 11).

[147] Jong Won Ma, Thomas Czerniawski, and Fernanda Leite, "Semantic segmentation of point clouds of building interiors with deep learning: Augmenting training datasets with synthetic BIM-based point clouds", *Automation in Construction*, vol. 113, 2020 (cit. on p. 1).

[148] Jisan Mahmud, True Price, Akash Bapat, and Jan-Michael Frahm, "Boundary-Aware 3D Building Reconstruction From a Single Overhead Image", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2020 (cit. on p. 56).

[149] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool, "Convolutional Oriented Boundaries: From Image Segmentation to High-Level Tasks", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 40, no. 4, 2018 (cit. on pp. 19, 20).

[150] David Marr and Ellen C. Hildreth, "Theory of Edge Detection", *Proceedings of the Royal Society of London Series B*, vol. 207, no. 1167, 1980 (cit. on p. 20).

[151] David R. Martin, Charless C. Fowlkes, and Jitendra Malik, "Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 26, no. 5, 2004 (cit. on p. 20).

[152] Andelo Martinovic, Jan Knopp, Hayko Riemenschneider, and Luc Van Gool, "3D All The Way: Semantic Segmentation of Urban Scenes From Start to End in 3D", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015 (cit. on p. 14).

[153] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst, "Geodesic convolutional neural networks on Riemannian manifolds", in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCV Workshops)*, 2015 (cit. on p. 13).

[154] Daniel Maturana and Sebastian Scherer, "Voxnet: A 3D convolutional neural network for real-time object recognition", in *Proceedings of the IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), 2015 (cit. on p. 11).

[155] Kirill Mazur and Victor Lempitsky, "Cloud Transformers: A Universal Approach to Point Cloud Processing Tasks", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021 (cit. on p. 12).

[156] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, "Distributed Representations of Words and Phrases and Their Compositionality", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2013 (cit. on p. 66).

[157] Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly, "Partial and Approximate Symmetry Detection for 3D Geometry", *ACM Transactions on Graphics*, vol. 25, no. 3, 2006 (cit. on p. 14).

[158] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su, "PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on pp. 1, 14, 20, 26, 35, 41, 47, 48, 50, 57, 58, 62, 67, 76).

[159] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on pp. 12, 13).

[160] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein, "Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on p. 17).

[161] Daniel Munoz, J. Andrew Bagnell, Nicolas Vandapel, and Martial Hebert, "Contextual classification with functional Max-Margin Markov Networks", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2009 (cit. on p. 15).

[162] Anh Nguyen and Bac Le, "3D point cloud segmentation: A survey", in *Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, 2013 (cit. on pp. 1, 8, 9).

[163] Duc Thanh Nguyen, Binh-Son Hua, Lap-Fai Yu, and Sai-Kit Yeung, "A Robust 3D-2D Interactive Tool for Scene Segmentation and Annotation", *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 12, 2018 (cit. on p. 15).

[164] Phuc D. A. Nguyen, Tuan Duc Ngo, Chuang Gan, Evangelos Kalogerakis, Anh Tran, Cuong Pham, and Khoi Nguyen, "Open3DIS: Open-vocabulary 3D Instance Segmentation with 2D Mask Guidance", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2023 (cit. on p. 78).

[165] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel, "Ridge-valley lines on meshes via implicit surface fitting", *ACM Transactions on Graphics*, vol. 23, no. 3, 2004 (cit. on p. 13).

[166] Yatian Pang, Wenxiao Wang, Francis E. H. Tay, Wei Liu, Yonghong Tian, and Li Yuan, "Masked Autoencoders for Point Cloud Self-supervised Learning",

in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022 (cit. on p. 12).

[167] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2019 (cit. on pp. 49, 66).

[168] Songyou Peng, Kyle Genova, Chiyu "Max" Jiang, Andrea Tagliasacchi, Marc Pollefeys, and Thomas Funkhouser, "OpenScene: 3D Scene Understanding with Open Vocabularies", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2023 (cit. on p. 78).

[169] Quang-Hieu Pham, Duc Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung, "JSIS3D: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 12).

[170] Judith M.S. Prewitt, "Object enhancement and extraction", *Picture Processing and. Psychopictorics*, 1970 (cit. on p. 20).

[171] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas, "Volumetric and multi-view cnns for object classification on 3d data", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2016 (cit. on p. 12).

[172] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on pp. 1, 2, 11).

[173] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2017 (cit. on pp. 1, 11, 50, 52, 63, 67, 73).

[174] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem, "Pointnext: Revisiting pointnet++ with improved training and scaling strategies", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2022 (cit. on pp. 1, 11, 73, 74).

[175] Yi-Ling Qiao, Lin Gao, Jie Yang, Paul L. Rosin, Yu-Kun Lai, and Xilin Chen, "Learning on 3D Meshes with Laplacian Encoding and Pooling", *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 2, 2022 (cit. on p. 13).

[176] Xiaofeng Ren and Liefeng Bo, "Discriminatively Trained Sparse Code Gradients for Contour Detection", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2012 (cit. on p. 20).

[177] Dario Rethage, Johanna Wald, Jürgen Sturm, Nassir Navab, and Federico Tombari, "Fully-Convolutional Point Networks for Large-Scale Point Clouds",

in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018 (cit. on p. 11).

[178] Pitchaporn Rewatbowornwong, Nattanat Chatthee, Ekapol Chuangsuwanich, and Supasorn Suwajanakorn, "Zero-guidance Segmentation Using Zero Segment Labels", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023 (cit. on p. 78).

[179] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on p. 11).

[180] Hayko Riemenschneider, András Bódis-Szomorú, Julien Weissenberg, and Luc Van Gool, "Learning Where to Classify in Multi-view Semantic Segmentation", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014 (cit. on p. 15).

[181] Lawrence G. Roberts, *Machine Perception of Three-Dimensional Solids*. Massachusetts Institute of Technology, 1963 (cit. on p. 20).

[182] Frank Rosenblatt, "The Perceptron: A Probabilistic Model For Information Storage And Organization in the Brain", *Psychological Review*, vol. 65, no. 6, 1958 (cit. on p. 2).

[183] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette, "Paris-Lille-3D: A large and high-quality ground truth urban point cloud dataset for automatic segmentation and classification", *The International Journal of Robotics Research*, vol. 37, no. 6, 2018 (cit. on pp. 1, 15).

[184] Sebastian Ruder, "An overview of gradient descent optimization algorithms", *arXiv preprint arXiv:1609.04747*, 2016 (cit. on p. 49).

[185] Radu Bogdan Rusu and Steve Cousins, "3D is here: Point Cloud Library (PCL)", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011 (cit. on p. 1).

[186] Radu Bogdan Rusu, Andreas Holzbach, Nico Blodow, and Michael Beetz, "Fast geometric point labeling using conditional random fields", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009 (cit. on p. 10).

[187] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz, "Towards 3D Point cloud based object maps for household environments", *Robotics and Autonomous Systems*, vol. 56, no. 11, 2008 (cit. on p. 9).

[188] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich, "SuperGlue: Learning Feature Matching with Graph Neural Networks", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 41).

[189] Manolis Savva, Fisher Yu, Hao Su, Asako Kanezaki, Takahiko Furuya, Ryutarou Ohbuchi, Zhichao Zhou, Rui Yu, Song Bai, Xiang Bai, Masaki Aono, Atsushi Tatsuma, S. Thermos, A. Axenopoulos, G. Th. Papadopoulos, P. Daras, Xiao Deng, Zhouhui Lian, Bo Li, Henry Johan, Yijuan Lu, and Sanjeev Mk, "Large-Scale 3D Shape Retrieval from ShapeNet Core55", in *Proceedings of the Eurographics Workshop on 3D Object Retrieval (3DOR)*, 2017 (cit. on p. 48).

[190] Ruwen Schnabel, Roland Wahl, and Reinhard Klein, "Efficient RANSAC for Point-Cloud Shape Detection", *Computer Graphics Forum*, vol. 26, no. 2, 2007 (cit. on p. 9).

[191] Jonathan R. Schoenberg, Aaron Nathan, and Mark Campbell, "Segmentation of dense range information in complex urban scenes", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010 (cit. on p. 10).

[192] Jonas Schult, Francis Engelmann, Theodora Kontogianni, and Bastian Leibe, "DualConvMesh-Net: Joint Geodesic and Euclidean Convolutions on 3D Meshes", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2020 (cit. on pp. 12, 13).

[193] Gabriel Sepulveda, Juan Carlos Niebles, and Alvaro Soto, "A Deep Learning Based Behavioral Approach to Indoor Autonomous Navigation", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018 (cit. on p. 56).

[194] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or, "Consistent mesh partitioning and skeletonisation using the shape diameter function", *The Visual Computer*, vol. 24, 2008 (cit. on p. 11).

[195] Gopal Sharma, Bidya Dash, Aruni RoyChowdhury, Matheus Gadelha, Marios Loizou, LiangLiang Cao, Rui Wang, Learned-Miller Erik, Subhransu Maji, and Kalogerakis Evangelos, "PRIFIT: Learning to Fit Primitives Improves Few Shot Point Cloud Segmentation", *Computer Graphics Forum (also in the Proceedings of the Symposium on Geometry Processing)*, vol. 41, no. 5, 2022 (cit. on p. 55).

[196] Gopal Sharma, Evangeloss Kalogeraki, and Subhransu Maji, "Learning Point Embeddings from Shape Repositories for Few-Shot Segmentation", in *Proceedings of the IEEE International Conference on 3D Vision (3DV)*, 2019 (cit. on p. 11).

[197] Gopal Sharma, Difan Liu, Evangelos Kalogerakis, Subhransu Maji, Siddhartha Chaudhuri, and Radomír Měch, "ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020 (cit. on pp. 12, 39).

[198] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang, "DeepContour: A Deep Convolutional Feature Learned by Positive-sharing Loss for Contour Detection", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015 (cit. on pp. 19, 20).

[199] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian, "Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018 (cit. on p. 12).

[200] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li, "PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 11).

[201] Shymon Shlafman, Ayellet Tal, and Sagi Katz, "Metamorphosis of Polyhedral Surfaces using Decomposition", *Computer Graphics Forum*, vol. 21, no. 3, 2002 (cit. on p. 10).

[202] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015 (cit. on p. 2).

[203] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015 (cit. on p. 1).

[204] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser, "Semantic Scene Completion from a Single Depth Image", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on p. 15).

[205] Nitish Srivastava, Hanlin Goh, and Ruslan Salakhutdinov, "Geometric Capsule Autoencoders for 3D Point Clouds", *arXiv preprint arXiv:1912.03310*, 2019 (cit. on p. 11).

[206] Simon Christoph Stein, Florentin Wörgötter, Markus Schoeler, Jeremie Papon, and Tomas Kulvicius, "Convexity based object partitioning for robot applications", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014 (cit. on p. 12).

[207] Johannes Strom, Andrew Richardson, and Edwin Olson, "Graph-based segmentation for colored 3D laser point clouds", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010 (cit. on p. 10).

[208] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz, "SPLATNet: Sparse Lattice Networks for Point Cloud Processing", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018 (cit. on p. 11).

[209] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015 (cit. on p. 12).

[210] Martin Sunkel, Silke Jansen, Michael Wand, Elmar Eisemann, and Hans-Peter Seidel, "Learning Line Features in 3D Geometry", *Computer Graphics Forum*, vol. 30, no. 2, 2011 (cit. on p. 13).

[211] Kenshi Takayama, Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung, "A Simple Method for Correcting Facet Orientations in Polygon Meshes Based on Ray Casting", *Journal of Computer Graphics Techniques*, vol. 3, no. 4, 2014 (cit. on p. 58).

[212] Weikai Tan, Nannan Qin, Lingfei Ma, Ying Li, Jing Du, Guorong Cai, Ke Yang, and Jonathan Li, "Toronto-3D: A Large-scale Mobile LiDAR Dataset for Semantic Segmentation of Urban Roadways", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference Workshops (CVPR Workshops)*, 2020 (cit. on pp. 1, 15).

[213] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas, "KPConv: Flexible and Deformable Convolution for Point Clouds", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019 (cit. on p. 12).

[214] Alexander Toshev and Ben Taskar, "Detecting and parsing architecture at city scale from range data", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2010 (cit. on p. 14).

[215] Trimble, *3D Warehouse*, 2020 (cit. on p. 58).

[216] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung, "Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019 (cit. on pp. 1, 14).

[217] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is All you Need", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2017 (cit. on pp. 43–45, 77).

[218] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio, "Graph Attention Networks", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018 (cit. on p. 17).

[219] George Vosselman, Ben Gorte, George Sithole, and Tehreem Rabbani, "Recognising structure in laser scanning point clouds", in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Archives)*, 2004 (cit. on p. 9).

[220] Chu Wang, Babak Samari, and Kaleem Siddiqi, "Local Spectral Graph Convolution for Point Set Feature Learning", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018 (cit. on p. 12).

[221] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao, "Deep High-Resolution Representation Learning for Visual Recognition", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 43, no. 10, 2021 (cit. on pp. 48, 101).

[222] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie, "PlanIT: Planning and Instantiating Indoor Scenes with Relation Graph and Spatial Prior Networks", *ACM Transactions on Graphics*, vol. 38, no. 4, 2019 (cit. on p. 14).

[223] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan, "Graph Attention Convolution for Point Cloud Semantic Segmentation", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 12).

[224] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong, "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis", *ACM Transactions on Graphics (also in the Proceedings of SIGGRAPH)*, vol. 36, no. 4, 2017 (cit. on pp. 11, 103).

[225] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong, "Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes", *ACM Transactions on Graphics (also in the Proceedings of SIGGRAPH Asia)*, vol. 37, no. 6, 2018 (cit. on p. 11).

[226] Peng-Shuai Wang, Yu-Qi Yang, Qian-Fang Zou, Zhirong Wu, Yang Liu, and Xin Tong, "Unsupervised 3D Learning for Shape Analysis via Multiresolution Instance Discrimination", in *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2021 (cit. on pp. 11, 41–43, 48, 50–52, 54, 55, 67).

[227] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun, "Deep parametric continuous convolutional neural networks", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018 (cit. on p. 12).

[228] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang, "PIE-NET: Parametric Inference of Point Cloud Edges", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2020 (cit. on p. 78).

[229] Xiaogang Wang, Bin Zhou, Haiyue Fang, Xiaowu Chen, Qinping Zhao, and Kai Xu, "Learning to Group and Label Fine-Grained Shape Components", *ACM Transactions on Graphics*, vol. 37, no. 6, 2018 (cit. on pp. 13, 14).

[230] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He, "Non-local neural networks", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018 (cit. on p. 40).

[231] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds", *ACM Transactions on Graphics*, vol. 38, no. 5, 2019 (cit. on pp. 1, 12, 18, 20–22, 24, 31, 34, 67, 114).

[232] Yupei Wang, Xin Zhao, Yin Li, and Kaiqi Huang, "Deep Crisp Boundaries: From Boundaries to Higher-level Tasks", *IEEE Transactions on Image Processing*, vol. 28, no. 3, 2018 (cit. on pp. 19, 20).

[233] Zeyu Wang, Cuong Nguyen, Paul Asente, and Julie Dorsey, "PointShopAR: Supporting Environmental Design Prototyping Using Point Cloud in Augmented Reality", in *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, 2023 (cit. on p. 1).

[234] Zihao Wang and Lei Wu, "Theoretical Analysis of the Inductive Biases in Deep Convolutional Networks", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2023 (cit. on p. 2).

[235] Wikipedia, *List of building types*, 2018 (cit. on p. 58).

[236] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger, "Simplifying Graph Convolutional Networks", in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019 (cit. on p. 17).

[237] Wenxuan Wu, Zhongang Qi, and Fuxin Li, "PointConv: Deep Convolutional Networks on 3D Point Clouds", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 12).

[238] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao, "Point Transformer V3: Simpler, Faster, Stronger", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2024 (cit. on pp. 12, 77).

[239] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao, "3D ShapeNets: A deep representation for

volumetric shapes", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015 (cit. on p. 11).

[240] Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han, "SnowflakeNet: Point Cloud Completion by Snowflake Point Deconvolution With Skip-Transformer", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021 (cit. on p. 12).

[241] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas Guibas, and Or Litany, "PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020 (cit. on pp. 12, 55).

[242] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu, "Attentional shapecontextnet for point cloud recognition", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018 (cit. on p. 12).

[243] Saining Xie and Zhuowen Tu, "Holistically-Nested Edge Detection", *International Journal of Computer Vision*, vol. 125, no. 1-3, 2017 (cit. on pp. 19, 20).

[244] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann, "Grid-GCN for Fast and Scalable Point Cloud Learning", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on p. 12).

[245] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao, "SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018 (cit. on pp. 12, 50, 52).

[246] Deshraj Yadav, Rishabh Jain, Harsh Agrawal, Prithvijit Chattopadhyay, Taranjeet Singh, Akash Jain, Shiv Baran Singh, Stefan Lee, and Dhruv Batra, "EvalAI: Towards Better Evaluation Systems for AI Agents", *arXiv preprint arXiv:1902.03570*, 2019 (cit. on p. 70).

[247] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo, "Swin3D: A Pretrained Transformer Backbone for 3D Indoor Scene Understanding", *arXiv preprint arXiv:2304.06906*, 2023 (cit. on pp. 1, 12).

[248] Yunhan Yang, Xiaoyang Wu, Tong He, Hengshuang Zhao, and Xihui Liu, "SAM3D: Segment Anything in 3D Scenes", *arXiv preprint arXiv:2306.03908*, 2023 (cit. on p. 77).

[249] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas, "A Scalable Active Framework for Region Annotation in 3D Shape Collections", *ACM Transactions on Graphics*, vol. 35, no. 6, 2016 (cit. on pp. 1, 14, 57).

[250] Li Yi, Lin Shao, Manolis Savva, Haibin Huang, Yang Zhou, Qirui Wang, Benjamin Graham, Martin Engelcke, Roman Klokov, Victor S. Lempitsky, Yuan Gan, Pengyu Wang, Kun Liu, Fenggen Yu, Panpan Shui, Bingyang Hu, Yan Zhang, Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Minki Jeong, Jaehoon Choi, Changick Kim, Angom Geetchandra, Narasimha Murthy, Bhargava Ramu, Bharadwaj Manda, M. Ramanathan, Gautam Kumar, P. Preetham, Siddharth Srivastava, Swati Bhugra, Brejesh Lall, Christian Häne, Shubham Tulsiani, Jitendra Malik, Jared Lafer, Ramsey Jones, Siyuan Li, Jie Lu, Shi Jin,

Jingyi Yu, Qixing Huang, Evangelos Kalogerakis, Silvio Savarese, Pat Hanrahan, Thomas A. Funkhouser, Hao Su, and Leonidas J. Guibas, "Large-Scale 3D Shape Reconstruction and Segmentation from ShapeNet Core55", *arXiv preprint arXiv:1710.06104*, 2017 (cit. on pp. 1, 57).

[251] Li Yi, Hao Su, Xingwen Guo, and Leonidas Guibas, "SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on pp. 12, 13).

[252] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu, "PartNet: A Recursive Part Decomposition Network for Fine-Grained and Hierarchical Shape Segmentation", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019 (cit. on pp. 14, 15).

[253] Jianhui Yu, Chaoyi Zhang, Heng Wang, Dingxin Zhang, Yang Song, Tiange Xiang, Dongnan Liu, and Weidong (Tom) Cai, "3D Medical Point Transformer: Introducing Convolution to Attention Networks for Medical Point Cloud Analysis", *arXiv preprint arXiv:2112.04863*, 2021 (cit. on p. 1).

[254] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng, "EC-Net: an Edge-aware Point set Consolidation Network", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018 (cit. on pp. 13, 20, 29).

[255] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu, "Point-bert: Pre-training 3d point cloud transformers with masked point modeling", in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2022 (cit. on p. 12).

[256] Shi Yunxiao, Fang Haoyu, Zhu Jing, and Fang Yi, "Pairwise Attention Encoding for Point Cloud Feature Learning", in *Proceedings of the IEEE International Conference on 3D Vision (3DV)*, 2019 (cit. on p. 12).

[257] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed, "Big Bird: Transformers for Longer Sequences", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2020 (cit. on p. 54).

[258] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R. Salakhutdinov, and Alexander J. Smola, "Deep Sets", in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2017 (cit. on pp. 2, 11).

[259] Matthias Zeppelzauer, Miroslav Despotovic, Muntaha Sakeena, David Koch, and Mario Döller, "Automatic Prediction of Building Age from Photographs", in *Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR)*, 2018 (cit. on p. 56).

[260] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos, "Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017 (cit. on p. 1).

[261] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung, "GaAN: Gated Attention Networks for Learning on Large and Spa-

tiotemporal Graphs", in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018 (cit. on p. 17).

[262] Junming Zhang, Weijia Chen, Yuping Wang, Ram Vasudevan, and Matthew Johnson-Roberson, "Point Set Voting for Partial Point Cloud Analysis", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, 2020 (cit. on pp. 11, 12).

[263] Ling Zhang and Zhigang Zhu, "Unsupervised Feature Learning for Point Cloud by Contrasting and Clustering With Graph Convolutional Neural Network", in *Proceedings of the IEEE International Conference on 3D Vision (3DV)*, 2019 (cit. on p. 12).

[264] Chen Zhao, Jiaqi Yang, Xin Xiong, Angfan Zhu, Zhiguo Cao, and Xin Li, "Rotation invariant point cloud analysis: Where local geometry meets global topology", *Pattern Recognition*, vol. 127, no. C, 2022 (cit. on p. 11).

[265] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun, "Point transformer", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021 (cit. on pp. 1, 12).

[266] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou, "Structured3D: A Large Photo-realistic Dataset for Structured 3D Modeling", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020 (cit. on p. 15).

[267] Yang Zhou, Zachary While, and Evangelos Kalogerakis, "SceneGraphNet: Neural Message Passing for 3D Indoor Scene Augmentation", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019 (cit. on p. 14).

[268] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem, "3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017 (cit. on p. 12).

Appendix A

Cross-shape network backbone architecture details

A.1 MinkHRNetCSN architecture details

In Table A.1 we describe the overall Cross-Shape Network architecture for $K = 1$ key shapes per query shape, based on the HRNet [221] backbone ("MinkHRNetCSN-K1"). For $K = 2, 3$ and SSA variants, we use the same architecture. First, for an input query-key pair of shapes $\mathcal{S}_m \in \mathcal{R}^{P_m \times 3}$ and $\mathcal{S}_n \in \mathcal{R}^{P_n \times 3}$, point-wise features $\boldsymbol{X}_m$ and $\boldsymbol{X}_n$ are extracted using the "Mink-HRNet" backbone (Layers 2 and 3). For each set of point features their self-shape attention representations are calculated via the *Cross-Shape Attention* layer (Layers 4 and 5). The query shape point self-shape attention representations are then aggregated into a global feature using mean-pooling and undergo two separate linear transformations (*Linear-Q* and *Linear-K* in Layers 6 and 7, respectively). Leveraging these, the self-shape similarity is computed, using the *scaled dot product* (Layer 9). For the key shape point self-shape attention representations, we use only the *Linear-K* transformation on the key shape's global feature (Layer 8), and calculate the query-key similarity (Layer 10). The compatibility for the cross-shape attention is computed as the softmax transformation of these two similarity measures (Layer 11). The cross-shape point representations of the query shape, propagating point features from the key shape, are extracted by our CSA module in Layer 12. The self-shape (Layer 4) and cross-shape (Layer 12) point representations are combined together, weighted by the pairwise compatibility, resulting in the cross-shape attention representations $\boldsymbol{X}'_m$ (Layer 13). Finally, part

	Cross-shape network architecture	$\leftarrow$ CSN(*query* $\mathcal{S}_m$, *key* $\mathcal{S}_n$, *#classes* K)
Index	Layer	Out
1	*Input*: $\mathcal{S}_m$,$\mathcal{S}_n$	$P_m \times 3, P_n \times 3$
2	*Mink-HRNet*($\mathcal{S}_m$, 3, 256)	$P_m \times 256$ - query point repr.
3	*Mink-HRNet*($\mathcal{S}_n$, 3, 256)	$P_n \times 256$ - key point repr.
4	*CSA*(Out(2), Out(2), 256, 4)	$P_m \times 256$ - query SSA repr.
5	*CSA*(Out(3), Out(3), 256, 4)	$P_n \times 256$ - key SSA repr.
6	*Linear-Q*(*avg-pool*(Out(4)), 256,256)	1×256 - query global repr.
7	*Linear-K*(*avg-pool*(Out(4)), 256,256)	1×256 - query global repr.
8	*Linear-K*(*avg-pool*(Out(5)), 256,256)	1×256 - key global repr.
9	*ScaledDotProduct*(Out(6), Out(7))	1×1 - query-query similarity
10	*ScaledDotProduct*(Out(6), Out(8))	1×1 - query-key similarity
11	*Softmax*(Out(9), Out(10))	2×1 - compatibility
12	*CSA*(Out(2), Out(3), 256, 4)	$P_m \times 256$ - query CSA repr.
13	Out(4) $*$ *compatbility*[0] + Out(12) $*$ *compatbility*[1]	$P_m \times 256$ - cross-shape attention
14	*Softmax*(*Conv*(*ConCat*(Out(2), Out(13)), 512, K))	$P_m \times K$ - per-point part label probabilities

Table A.1: Cross-shape network architecture for $K = 1$ key shapes per query shape.

label probabilities are extracted per point, through a $1 \times 1 \times 1$ convolution and a softmax transformation (Layer 14), based on the concatenation of the query shape's backbone representations $\boldsymbol{X}_m$ and cross-shape attention representations $\boldsymbol{X}'_m$.

The architecture of our backbone network, "Mink-HRNet", is described in Table A.3. Based on an input shape, our backbone first extracts point representations through two consecutive convolutions (Layers 2-5). Then, three multi-resolution branches are deployed. The first branch, called *High-ResNetBlock* (Layers 6, 8 and 15), operates on the input shape's resolution, while the other two, *Mid-ResNetBlock* (Layers 9 and 16) and *Low-ResNetBlock* (Layer 17), downsample the shape's resolution by a factor of 2 and 4, respectively. In addition, feature representations are exchanged between these branches, through downsampling and upsampling modules (Layers 7, 10-14). The point representations of the two low-resolution branches are upsampled to the original resolution (Layers 18-20) and by concatenating them with point features of the high-resolution branch, point representations are extracted for the input shape, through a full-connected layer (Layers 21 and 22).

The *Cross-Shape Attention* (CSA), *Downsampling* and *Upsampling* layers, along with *Residual Basic Block* are described in more detail in Table A.4.

	MID-FC-Cross-shape network architecture	← MID-FC-CSN(*query* $\mathcal{S}_m$, *key* $\mathcal{S}_n$, *#classes* K)
Index	Layer	Out
1	*Input*: $\mathcal{S}_m$,$\mathcal{S}_n$	$P_m \times 3, P_n \times 3$
2	*MID-Net*($\mathcal{S}_m$, 3, 256)	$P_m \times 256$ - query point repr.
3	*MID-Net*($\mathcal{S}_n$, 3, 256)	$P_n \times 256$ - key point repr.
4	*CSA*(Out(2), Out(2), 256, 8)	$P_m \times 256$ - query SSA repr.
5	*CSA*(Out(3), Out(3), 256, 8)	$P_n \times 256$ - key SSA repr.
6	*Linear-Q*(*avg-pool*(Out(4)), 256,256)	1×256 - query global repr.
7	*Linear-K*(*avg-pool*(Out(4)), 256,256)	1×256 - query global repr.
8	*Linear-K*(*avg-pool*(Out(5)), 256,256)	1×256 - key global repr.
9	*ScaledDotProduct*(Out(6), Out(7))	1×1 - query-query similarity
10	*ScaledDotProduct*(Out(6), Out(8))	1×1 - query-key similarity
11	*Softmax*(Out(9), Out(10))	2×1 - compatibility
12	*CSA*(Out(2), Out(3), 256, 8)	$P_m \times 256$ - query CSA repr.
13	Out(4) $*$ *compatbility*[0] + Out(12) $*$ *compatbility*[1]	$P_m \times 256$ - cross-shape attention
14	*Softmax*(*FC*(Out(13), 256, K))	$P_m \times K$ - per-point part label probabilities

Table A.2: MID-FC-CSN architecture for $K = 1$ key shapes per query shape.

A.2 MID-FC-CSN architecture details

Similar to the MinkHRNetCSN, the MID-FC-CSN variant also follows a comparable architecture (see Table A.2). To extract point features $\boldsymbol{X}_m$ and $\boldsymbol{X}_n$ for the input query-key pair of shapes, the "MID-Net" backbone is utilized (Layers 2 and 3). This backbone also adopts a three-stage HRNet architecture, which is built on an octree-based CNN framework [224]. ResNet blocks with a bottleneck structure [72] are used in all multi-resolution branches, and feature sharing is achieved using downsample and upsample exchange blocks, implemented by max-pooling and tri-linear up-sampling, respectively. The CSA module (Layers 4 and 12) is employed to construct the self-shape and cross-shape attention features for the query shape. These are then weighted by the learned pairwise compatibility (Layers 4-11) and aggregated to generate the final cross-shape attention representations $\boldsymbol{X}'_m$ (Layer 13). Part label probabilities are extracted per point using a fully-connected layer and a softmax transformation based on the cross-shape attention representations (Layer 14).

	Mink-HRNet backbone	← Mink-HRNet(*shape repr.* $\mathbf{X}_m$, *in_feat* D_{in}, *out_feat* D_{out})
Index	Layer	Out
1	*Input*: $\mathbf{X}_m$	$P_m \times D_{in}$
2	*Conv*($\mathbf{X}_m$, D_{in}, 32)	$P_m \times 32$
3	*ReLU*(*BatchNorm*(Out(2)))	$P_m \times 32$
4	*Conv*(Out(3), 32, 64)	$P_m \times 64$
5	*ReLU*(*BatchNorm*(Out(4)))	$P_m \times 64$
6	*High-ResNetBlock*(3× *BasicBlock*(Out(5), 64))	$P_m \times 64$
7	*Downsampling*(Out(6), 64, 128)	$P_m/2 \times 128$
8	*High-ResNetBlock*(3× *BasicBlock*(Out(6), 64))	$P_m \times 64$
9	*Mid-ResNetBlock*(3× *BasicBlock*(*ReLU*(Out(7)), 128))	$P_m/2 \times 128$
10	*Downsampling*(Out(8), 64, 128)	$P_m/2 \times 128$
11	*ReLU*(*Downsampling*(Out(8), 64, 128))	$P_m/2 \times 128$
12	*Downsampling*(Out(11), 128, 256)	$P_m/4 \times 256$
13	*Upsampling*(Out(9), 128, 64)	$P_m \times 64$
14	*Downsampling*(Out(9), 128, 256))	$P_m/4 \times 256$
15	*High-ResNetBlock*(3× *BasicBlock*(*ReLU*(Out(8)+Out(13)), 64))	$P_m \times 64$
16	*Mid-ResNetBlock*(3× *BasicBlock*(*ReLU*(Out(9) + Out(10)), 128))	$P_m/2 \times 128$
17	*Low-ResNetBlock*(3× *BasicBlock*(*ReLU*(Out(12) + Out(14)), 256))	$P_m/4 \times 256$
18	*ReLU*(*Upsampling*(Out(16), 128, 128))	$P_m \times 128$
19	*ReLU*(*Upsampling*(Out(17), 256, 256))	$P_m/2 \times 256$
20	*ReLU*(*Upsampling*(Out(19), 256, 256))	$P_m \times 256$
21	*Conv*(*ConCat*(Out(3), Out(15), Out(18), Out(20)), 480, D_{out})	$P_m \times D_{out}$
22	*ReLU*(*BatchNorm*(Out(21)))	$P_m \times D_{out}$

Table A.3: Mink-HRNet backbone architecture. High, mid and low-resolution ResNet blocks consist of 3 consecutive residual basic blocks, each. Point representations are exchanged between multi-resolution branches via downsampling and upsampling layers (see Table A.4 for a more detailed description of their architecture). The convolution kernel of Layer 2 is of size $5 \times 5 \times 5$, in order to increase its receptive field, while for Layer 4 is of size $3 \times 3 \times 3$. For Layer 21 we used a kernel of $1 \times 1 \times 1$, since this acts as a fully-connected layer.

	Cross-Shape Attention Layer	$\leftarrow$ CSA(*query* $\boldsymbol{X}_m$, *key* $\boldsymbol{X}_n$, *#feats* D, *#heads* H)
Index	Layer	Out
1	*Input*: $\boldsymbol{X}_m$,$\boldsymbol{X}_n$	$P_m \times D$,$P_n \times D$
2	$H\times$ *Linear-Q*($\boldsymbol{X}_m$,D,$\lfloor D/H \rfloor$)	$P_m \times H \times D'$
3	$H\times$ *Linear-K*($\boldsymbol{X}_n$,D,$\lfloor D/H \rfloor$)	$P_n \times H \times D'$
4	$H\times$ *Linear-V*($\boldsymbol{X}_n$,D,$\lfloor D/H \rfloor$)	$P_n \times H \times D'$
5	*Attention*(Out(2), Out(3))	$H \times P_m \times P_n$
6	*MatMul*(Out(5), Out(4))	$P_m \times H \times D'$
7	*Linear*(*ConCat*(Out(6)), D, D)	$P_m \times D$
8	*LayerNorm*($\boldsymbol{X}_m$+ Out(7))	$P_m \times D$
	Downsampling Layer	$\leftarrow$ Downsampling(*shape repr.* $\boldsymbol{X}_m$, *in_feat* D_{in}, *out_feat* D_{out})
1	*Input*: $\boldsymbol{X}_m$	$P_m \times D_{in}$
2	*Conv*($\boldsymbol{X}_m$, D_{in}, D_{out}, *stride* $= 2$)	$P_m/2 \times D_{out}$
3	*BatchNorm*(Out(2))	$P_m/2 \times D_{out}$
	Upsampling Layer	$\leftarrow$ Upsampling(*shape repr.* $\boldsymbol{X}_m$, *in_feat* D_{in}, *out_feat* D_{out})
1	*Input*: $\boldsymbol{X}_m$	$P_m \times D_{in}$
2	*TrConv*($\boldsymbol{X}_m$, D_{in}, D_{out}, *stride* $= 2$)	$2 * P_m \times D_{out}$
3	*BatchNorm*(Out(2))	$2 * P_m \times D_{out}$
	Residual Basic Block	$\leftarrow$ BasicBlock(*shape repr.* $\boldsymbol{X}_m$, *#feats* D)
1	*Input*: $\boldsymbol{X}_m$	$P_m \times D$
2	*Conv*($\boldsymbol{X}_m$, D, D)	$P_m \times D$
3	*ReLU*(*BatchNorm*(Out(2)))	$P_m \times D$
4	*Conv*(Out(3), D, D)	$P_m \times D$
5	*ReLU*($\boldsymbol{X}_m$ + *BatchNorm*(Out(4)))	$P_m \times D$

Table A.4: Cross-shape network basic layers. All convolution kernels are of size $3 \times 3 \times 3$.

Appendix B

Key shape retrieval measure comparison

Category	Bed	Bott	Chai	Cloc	Dish	Disp	Door	Ear	Fauc	Knif	Lamp	Micr	Frid	Stor	Tabl	Tras	Vase	**avg.**	**#cat.**
Part IoU																			
MinkHRNetCSN-K1 (Eq. 4.14)	**42.1**	**54.0**	**42.5**	**42.9**	**58.2**	**83.2**	**43.5**	**51.5**	**59.4**	**47.8**	**27.9**	**57.4**	43.7	**46.2**	**36.8**	**51.5**	**60.0**	**49.9**	**16**
MinkHRNetCSN-K1 (Eq. 4.10)	38.7	47.0	41.9	40.8	55.7	82.3	41.3	50.5	57.9	37.3	24.7	56.2	**44.1**	45.9	32.3	51.4	58.8	47.4	1

Table B.1: Comparison of shape retrieval measures based on point-wise (Equation 4.14) and global (Equation 4.10) representations of a query-key pair of shapes, in terms of Part IoU and Shape IoU.

As an additional ablation, we evaluated the performance of our "MinkHRNetCSN-K1" variant for two key shape retrieval measures (see Section 4.1.2 in Chapter 4). The first relies on the point-wise representations between a query and a key shape and retrieves key shapes that are on average more similar to their query counterparts (Equation 4.14). The second measure, takes into account only the global representations of a query-key pair of shapes (Equation 4.10). In Table B.1 we report the performance for both measures, in terms of Part IoU and Shape IoU. Our default variant, "MinkHRNetCSN-K1 (Eq. 4.14)", achieves better performance according to Part IoU (+2.5%), and it outperforms the other variant ("MinkHRNetCSN-K1, Eq. 4.10)" in 16 out 17 object categories. This is a strong indication that the key shape retrieval measure based in Equation 4.14 is more effective in retrieving key shapes for cross-shape attention.

Appendix C

BuildingGNN architecture and experiments details

C.1 BuildingGNN architecture details

We provide more details about the structure of the BuildingGNN network architecture in Table C.1. Table C.2 presents statistics on the number of edges per type used in BuildingGNN for our training set.

C.2 Experiments with different losses

We experimented with different losses for our MinkowskiNet variants for the "BuildingNet-Points" and "BuildingNet-Mesh" tracks. Specifically, we experimented with the Weighted Cross-Entropy Loss (WCE) described in Section 5.2.6 (Chapter 5), Cross-Entropy Loss (CE) without label weights, the Focal Loss (FL) [130], α-balanced Focal Loss (α-FL) [130], and Class-Balanced Cross Entropy Loss (CB) [36]. Table C.3 and Table C.4 show results for the "BuildingNet-Points" and "BuildingNet-Mesh" tracks respectively. We observe that (a) in the case that color is not available, WCE is slightly better than alternatives according to all measures for both tracks (b) when color is available, CB is a bit better in terms of Part IoU, but worse in terms of Shape IoU than WCE in the case of the point cloud track. For the mesh track, CB is slightly better according to all measures. In general, WCE and CB behave the best on average, yet their difference is small. For the rest of our experiments, we use WCE.

	Layers	Output
Edge	(MLP(11×41, layer=1)))	41
Node	(6D(OBB)+1D(SA)+3D(C)+31D(MN)	41
Input	($Node_i$ + $Edge_{ij}$ + $Node_j$)	41
Encoder	(MLP(Input×256, layer=1)))	64
	GN(LeakyReLU(0.2)))	64
	(MLP(64*3×128, layer=3)))	64
	GN(LeakyReLU(0.2)))	64
	(MLP(64*3×128, layer=5)))	64
	GN(LeakyReLU(0.2)))	64
Decoder	(MLP(128×64, layer=1)))	31
	softmax	31

Table C.1: BuildingGNN architecture: The Node representation combines the OBB - (Object Oriented Bounding Box), SA - (Surface area), C - (centroid) and MN - (MinkowskiNet pre-trained features) for each sub group. The GNN is composed of (a) an encoder block made of three MLPs having 1, 3 and 5 hidden layers respectively, and (b) a decoder block with one MLP having 1 hidden layer followed by softmax.

Label	max # edges	min # edges	mean # edges	# median edges
Proximity	16317	81	778.0	489.0
Similarity	762156	5	26452.1	4875.5
Containment	26354	71	2,054.5	1,390.0
Support	7234	7	687.5	492.0
All	772878	259	29972.1	7818.0

Table C.2: Statistics for the number of BuildingGNN edges per type present in the graphs of the training buildings.

Method	Loss	**n**?	**c**?	Part IoU	Shape IoU	Class acc.
MinkNet	WCE	✓	×	**26.9%**	**22.2%**	**62.2%**
	CE	✓	×	24.5%	21.2%	61.3%
	FL	✓	×	26.1%	21.8%	61.2%
	α-FL	✓	×	22.3%	19.8%	61.5%
	CB	✓	×	26.4%	20.9%	61.4%
MinkNet	WCE	✓	✓	29.9%	24.3%	**65.5%**
	CE	✓	✓	28.5%	24.5%	65.3%
	FL	✓	✓	28.7%	24.9%	65.2%
	α-FL	✓	✓	30.1%	**25.3%**	65.2%
	CB	✓	✓	**30.4%**	24.0%	**65.5%**

Table C.3: "BuildingNet-Points" track results using the Weighted Cross-Entropy Loss (WCE), Cross-Entropy Loss (CE), Focal Loss (FL), α-balanced Focal Loss (α-FL) and finally Class-Balanced Cross Entropy Loss (CB). All these were used to train the MinkowskiUNet34 architecture. For the FL and α-FL experiments the γ hyper-parameter was set to 2.0 and for the α-FL the same weights were used as the weighted cross entropy loss (see Section 5.2.6). For the CB experiments we set $\beta = 0.999999$.

C.3 Average vs max pooling

As discussed in the experiments Section 5.3 of Chapter 5, one possibility to aggregate probabilities of points associated per triangle or component is average pooling: $\mathbf{q}_t = \sum_{p \in P_t} \mathbf{q}_p / |P_t|$ where $\mathbf{q}_p$ and $\mathbf{q}_t$ are point and triangle probabilities respectively. An alternative is to use max pooling (i.e., replace sum with max above). We experimented with average vs max pooling also per component. As shown in Table C.5, average pooling works better for both triangle- and component-based pooling (we experimented with MinkowskiNet per-point probabilities).

Method	Loss	**n**?	**c**?	Part IoU	Shape IoU	Class acc.
MinkNet2Sub	WCE	✓	×	**33.1%**	**36.0%**	**69.9%**
	CE	✓	×	30.7%	32.7%	68.8%
	FL	✓	×	31.0%	33.4%	67.9%
	α-FL	✓	×	27.2%	28.3%	66.7%
	CB	✓	×	32.9%	34.3%	69.1%
MinkNet2Sub	WCE	✓	✓	37.0%	39.1%	73.2%
	CE	✓	✓	35.6%	39.2%	73.5%
	FL	✓	✓	35.1%	38.4%	73.2%
	α-FL	✓	✓	36.0%	38.2%	72.4%
	CB	✓	✓	**38.0%**	**39.7%**	**73.9%**

Table C.4: "BuildingNet-Mesh" results using different loss functions

Method	Pool.	**n**?	**c**?	Part IoU	Shape IoU	Class acc.
MinkNet2Triangle	Avg	✓	×	**28.8%**	**26.7%**	**64.8%**
	Max	✓	×	28.6%	26.1%	64.4%
	Avg	✓	✓	**32.8%**	**29.2%**	**68.1%**
	Max	✓	✓	31.5%	28.1%	66.8%
MinkNet2Sub	Avg	✓	×	**33.1%**	**36.0%**	**69.9%**
	Max	✓	×	30.4%	32.4%	65.6%
	Avg	✓	✓	**37.0%**	**39.1%**	**73.2%**
	Max	✓	✓	32.7%	34.8%	67.4%

Table C.5: "BuildingNet-Mesh" results using average and max pooling aggregation over triangles and components (weighted cross-entropy loss was used for all these experiments).

C.4 MinkNet-GC

As mentioned in the experiments Section 5.3 of Chapter 5, we implemented a simple graph-cuts variant, called MinkNet-GC, that incorporates label probabilities from MinkowskiUNet34 as unary terms, and a pairwise term that depends on an-

gles between triangles, inspired by [89]. Specifically, we use the following energy that we minimize using [18]:

$$E(\boldsymbol{y}) = \sum_{i \in \mathcal{F}} \psi(y_i) + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{N}(i)} \phi(y_i, y_j) \tag{C.1}$$

where $\boldsymbol{y} = \{y_i\}$ are the label assignments we wish to compute by minimizing the above energy, $\mathcal{F}$ is the set of faces in a mesh, and $\mathcal{N}(i)$ are the adjacent faces of each face i. The unary term is expressed as follows: $\psi(y_i) = -\log f(y_i)$, where $f(y_i)$ is the probability distribution over part labels associated with the face i produced through average pooling of probabilities computed from MinkowskiUNet34 on the triangle's associated points. The pairwise term uses angles between face normals, $\phi'(y_i, y_j) = -\lambda' \cdot \log(\min(\omega_{i,j}/90^o, 1))$, for $y_i \neq y_j$, where $\omega_{i,j}$ is the angle between the normals of faces i, j. The term results in zero cost for right angles between normals indicating a strong edge. The parameter λ is adjusted with grid search in the hold-out validation set.

C.5 Performance for each part label

Chapter 5 reports mean Part IoU performance in the experiments Section 5.3. Table C.6 reports the BuildingGNN-PointNet++ and BuildingGNN-MinkNet part IoU performance for each label. We also report the performance of MinkowskiNet and PointNet++ for the point cloud track. We observe that networks do better for common part labels, such as window, wall, roof, plant, vehicle, while the performance degrades for rare parts (e.g., awning, arch), or parts whose shape can easily be confused with other more dominant parts (e.g., garage is often confused with door, wall, or window).

Label	BuildingGNN MinkNet(n+c)	BuildingGNN PointNet++(n+c)	MinkNet (n+c)	PointNet++ (n+c)	BuildingGNN MinkNet(n)	BuildingGNN PointNet++(n)	MinkNet (n)	PointNet++ (n)
Window	70.5%	71.1%	44.1%	34.8%	70.4%	68.3%	35.6%	0.0%
Plant	81.0%	69.8%	79.6%	70.3%	79.8%	69.8%	79.7%	48.4%
Vehicle	83.7%	77.3%	77.1%	29.7%	82.7%	72.4%	75.8%	19.2%
Wall	78.1%	77.5%	64.5%	57.9%	76.0%	74.4%	63.2%	54.4%
Banister	50.0%	19.9%	44.9%	0.0%	56.5%	22.0%	45.6%	0.0%
Furniture	59.7%	37.0%	56.0%	0.0%	58.3%	43.5%	54.9%	0.0%
Fence	55.5%	34.7%	71.3%	16.5%	64.1%	19.7%	49.5%	9.6%
Roof	78.9%	72.1%	65.3%	58.2%	70.2%	69.0%	67.0%	56.4%
Door	41.7%	37.6%	21.7%	0.0%	39.2%	37.7%	23.8%	0.0%
Tower	53.4%	41.2%	46.5%	2.3%	50.8%	37.5%	43.4%	4.8%
Column	61.5%	27.6%	49.5%	0.6%	53.6%	34.7%	42.9%	1.1%
Beam	24.9%	22.4%	13.8%	0.02%	30.3%	21.5%	17.2%	0.0%
Stairs	38.6%	25.6%	26.9%	0.0%	41.0%	24.1%	27.8%	0.0%
Shutters	1.0%	1.3%	0.0%	0.0%	1.7%	0.0%	0.0%	0.0%
Garage	9.0%	10.6%	3.6%	0.0%	10.6%	8.4%	6.8%	0.0%
Parapet	24.9%	3.9%	11.6%	0.0%	28.6%	2.5%	21.0%	0.0%
Gate	14.0%	16.5%	6.4%	0.0%	7.9%	12.3%	7.9%	0.0%
Dome	53.8%	10.1%	48.0%	1.9%	54.3%	14.2%	54.5%	16.3%
Floor	51.5%	37.7%	47.8%	36.9%	51.2%	30.9%	46.8%	30.0%
Ground	75.0%	65.1%	77.4%	64.1%	61.8%	55.5%	60.8%	42.6%
Buttress	23.8%	9.6%	15.6%	0.0%	38.7%	12.3%	6.1%	0.0%
Balcony	19.6%	9.5%	15.0%	0.0%	15.5%	15.6%	17.3%	0.0%
Chimney	70.0%	50.9%	57.9%	0.0%	53.6%	49.5%	60.1%	0.0%
Lighting	6.4%	9.1%	16.8%	0.0%	24.9%	3.5%	23.3%	0.0%
Corridor	16.3%	10.5%	15.9%	4.2%	7.2%	4.1%	7.2%	0.0%
Ceiling	28.0%	23.8%	22.1%	4.6%	28.0%	20.5%	17.4%	4.6%
Pool	70.8%	53.0%	78.7%	77.8%	38.1%	33.0%	43.0%	0.0%
Dormer	27.3%	20.4%	9.6%	0.0%	22.1%	23.3%	6.8%	0.0%
Road	46.2%	24.1%	53.5%	40.0%	1.9%	16.3%	21.5%	0.0%
Arch	8.4%	5.2%	0.9%	0.0%	3.2%	2.9%	0.8%	0.0%
Awning	1.5%	0%	3.8%	0.0%	1.6%	0.0%	0.0%	0.0%

Table C.6: Part IoU performance for each label. BuildingGNN-MinkNet and BuildingGNN-PointNet++ are tested on the mesh track, while MinkNet and PointNet++ are tested on the point cloud track. The left half of the table reports performance when color is available ("n+c"), while the right half reports performance when it is not available ("n").

Appendix D

BuildingGNN ablation study

D.1 Node features ablation

We conducted an ablation study involving different node features, and also experimenting with different types of edges in our BuildingGNN. Table D.1 present the results for different experimental conditions of our BuildingGNN based on PointNet++ as node features. We first experimented using no edges and processing node features alone through our MLP structure. We experimented with using only OBB-based features ("Node-OBB"), using features from PointNet++ alone ("Node-PointNet++"), and finally using both node features concatenated ("Node-OBB+PointNet++"). We observe that using all combinations of node features yields better performance compared to using either node feature type alone. Then we started experimented with adding each type of edges individually to our network (e.g., "w/ support edges" in Table D.1 means that we use node features with support edges only). Adding each type of edge individually further boosts performance compared to using node features alone. Using all edges ("BuildingGNN-PointNet") yields a noticeable 7.1% Part IoU increase and 8.1% Shape IoU increase compared to using node features alone. Table D.2 shows the same experiments using MinkowskiNet-based features. We observe that combined node features perform better than using either node feature type alone. Adding each type of edges helps, except for proximity edges that seem to have no improvement when used alone. Using all edges still yields a noticeable 2.6% Part IoU increase and 6.2% Shape IoU increase compared to using node features alone.

Variant	**n**?	**c**?	Part IoU	Shape IoU	Class acc.
Node-OBB	✓	✓	10.0%	17.1%	56.5%
Node-PointNet++	✓	✓	14.0%	19.1%	52.2%
Node-OBB+PointNet++	✓	✓	24.4%	27.8%	71.7%
w/ support edges	✓	✓	26.7%	29.2%	71.5%
w/ containment edges	✓	✓	27.9%	30.6%	72.6%
w/ proximity edges	✓	✓	26.4%	29.4%	71.4%
w/ similarity edges	✓	✓	23.1%	28.5%	69.8%
BuildingGNN-PointNet++	✓	✓	**31.5%**	**35.9%**	**73.9%**

Table D.1: BuildingGNN ablation study based on PointNet++ node features.

Variant	**n**?	**c**?	Part IoU	Shape IoU	Class acc.
Node-OBB	✓	✓	10.0%	17.1%	56.5%
Node-MinkNet	✓	✓	35.6%	35.9%	67.7%
Node-OBB+MinkNet	✓	✓	40.0%	40.6%	75.8%
w/ support edges	✓	✓	42.0%	43.5%	77.8%
w/ containment edges	✓	✓	41.1%	42.0%	76.8%
w/ proximity edges	✓	✓	39.9%	40.6%	75.6%
w/ similarity edges	✓	✓	41.2%	43.0%	75.8%
BuildingGNN-MinkNet	✓	✓	**42.6%**	**46.8%**	**77.8%**

Table D.2: BuildingGNN ablation study based on MinkowskiNet node features.

D.2 DGCNN experiments

We also experimented with DGCNN [231] as a backbone in our GNN for extracting node features. Unfortunately, DGCNN could not directly handle our large points clouds (100K points). It runs out of memory even with batch size 1 on a 48GB GPU card. We tried to downsample the point clouds (10K points) to pass them to DGCNN, then propagated the node features back to the 100K points using nearest neighbor upsampling. The part IoU was 32.5% in the mesh track with color input and using all edges (i.e., the performance is comparable to BuildingGNN-PointNet++, but much lower than BuildingGNN-MinkNet). Still, since other meth-

ods were able to handle the original resolution without downsampling, this comparison is not necessarily fair, thus we excluded it from the tables showing the track results in Chapter 5, Section 5.3.