DISSERTATION

Part-Based Representations for Robust 3D Object Classification under Domain Shift

conducted in partial fulfillment of the requirements for the degree of a

Doktor der technischen Wissenschaften (Dr. techn.)

supervised by

Ao.Univ. Prof. Dipl.-Ing. Dr. techn. Markus Vincze
E376 Automation and Control Institute

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology

by

Jean-Baptiste Weibel, MSc
DOB 02.05.1992
Matr. Nr.: 01639367

Vienna, March 2022 Jean-Baptiste Weibel

Acknowledgment

Working on a Ph.D. thesis is always a challenge, and one I pondered about for a long time. Which laboratory to study at, which supervisor, and which topic are all questions I considered seriously. And yet, it still was no preparation for the thousands of unexpected doubts and problems along the way. I owe it entirely to the people who surrounded me on this journey to have not abandoned and enjoyed it.

First of all, I would like to thank Prof. Markus Vincze for giving me the opportunity to pursue this doctorate with peace of mind thanks to the trust he put in me, even in times of doubt. He also gave me the chance to broaden my horizons through a variety of projects, making the process all the more enlightening and enjoyable.

I would also like to express my deepest gratitude to Dr. Timothy Patten, who provided continuous support and feedback and always remained overly generous with his time. It is no overstatement to say that I would not have published any paper without his help.

A big thank you also has to go to all the Vision for Robotics lab members, past and present, for keeping me sane through these years. Through stimulating discussions, programming support, endless robot debugging, but also beers, football games, all sorts of silly games, and just generally creating a nice working environment, they made my time wonderful. In particular, I would like to thank my friends Simon for weathering my endless stream of ideas, terrible or not, Mohammad for too many research advice to count and an inspiring work ethic, and Matthias, for all the care he put in bringing us all together and keeping our work-life balance in check.

I am also infinitely indebted to my entire family and especially my parents, Marc and Monique, for keeping me grounded and reminding me what matters in life, and always supporting me even when I decided to run a thousand kilometers away to start a PhD instead of finding a job.

Last but not least, I would like to warmly thank my partner Laura for supporting and caring for me through the combined challenges of a Ph.D. and a global pandemic. I could not have done it alone.

The research leading to these results has received funding from the EC FP7 Program under grant agreement no. 610532, SQUIRREL, the EC Horizon 2020 for Research and Innovation under grant agreement No. 101017089, TraceBot and the Austrian Science Foundation (FWF) under grant agreement No. I3968-N30 HEAP.

Abstract

Scene understanding is a key problem to solve to enable service robots to perform useful actions such as tidying up the room, or bringing a requested item. Methods attempting to solve this problem using 3D data have greatly benefited from the advent of deep learning. Convolutional neural networks in particular have contributed to advancing the field but have mostly been studied when dealing with 2D data. Which learning architecture to use when dealing with 3D data that cannot be reduced to a single view is however still an open problem. This thesis introduces multiple methods that are robust to rotation, scale change and occlusion, which is commonly found in 3D data captured by a robot.

Widely available depth sensors and the progress in camera pose estimation have made real 3D data collection much more accessible at scales never reached before, and artificial 3D models have been produced using CAD software for decades for manufacturing or entertainment purposes. Taking advantage of this large pool of already semantically annotated data requires methods that can deal with both sources indifferently. The domain shift between artificial and real data however remains a major challenge when dealing with 3D data. This thesis introduces a part-based representation to tackle this issue. In particular, the scale of artificial models is often inconsistent, but parts created based on the curvature of the object surface can be be consistent across domains and are easily scaled individually to a canonical space. Similarly, rotation invariance can be achieved by orienting the parts themselves based on their local covariance. Finally, even under occlusion, many parts of the object will remain the same, limiting the dependency of the representation on the entire object being present.

Under certain scenarios, it can be particularly challenging to achieve accurate camera pose estimation, for example when dealing with large scenes. Moreover, each depth sensing modality has its own limitation. Both issues combined can lead to a very large gap between artificial and real data. This thesis introduces a data efficient architecture that takes advantage of the priors introduced by a decade of research in robot vision. Combining the Point Pair Feature representation with a part-based sampling of the pairs and a suitable learning architecture, this method a achieves high level of robustness to noisy data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robots are already part of our everyday life, and it is now common to see them vacuuming, cleaning floors or even cutting the grass. Figure 1.1 shows examples of such robots. We have, however, not yet seen the full potential of service robots come to fruition. The global pandemic that started in 2019 dramatically underlined missed opportunities with overworked health personal that could have used robot assistance for menial tasks in the hospital, or essential workers unable to stay safe and confined during the peak of the pandemic [1]. Combining the abilities to navigate ever-changing environments, to search for specific objects and to manipulate them in order to clean up rooms reliably enough to be applicable in various houses, hospitals or restaurants is still out of reach for today's robots.

That is not to say that the field has not seen major progress. Semantic understanding [2], camera tracking [3] and even sensing with the wide availability of depth sensors, such as the active depth Asus Xtion camera, the active stereo Intel Realsense or LIDAR sensors improved tremendously. These new technologies fueled the development of new Virtual Reality headsets, such as the Occulus Rift and Augmented Reality headsets such as the Microsoft Hololens, which face similar challenges as service robots. Indeed, both fields rely on the ability to understand their environment and their position within it geometrically. Robots, however, will interact with their environment and manipulate objects within it, requiring a much finer semantic understanding. More specifically, because of the interaction between robots and objects, 3D semantic understanding is essential, that is not only understanding the type of object, but also the type of an object's parts and their relations. The geometric information necessary for such a task is most easily obtained through the use of off-the-shelf depth sensors, which directly provide this geometric information, reducing the problem to the inference of the semantic information from the object geometric representation.

Artificial intelligence methods and deep learning in particular provide such semantic information but require a large amount of data to do so. Annotating large amounts of data is however a costly task. This problem is compounded by the fine level of annotation desired for object understanding and manipulation: not only would we need to annotate every object in the scene, but also the parts that should be grasped in a given manipulation sequence. This annotation problem can be alleviated in two ways. First by designing architectures requiring less training data through the use of

Figure 1.1: Top: Examples of commercial robots available today. From left to right: Worx Landroid lawn mower[1], Avidbots Neo[2], iRobot Roomba[3]. Bottom: Examples of service robots used in research labs but struggling to become a commercial product. From left to right: Toyota Human Support Robot[4], PAL Robotics Tiago[5], TU Eindhoven Amigo[6], Fetch Robotics[7]

priors. Second, by using artificial and simulated data. This second option is particularly appealing when working with 3D data as a very large set of artificial models already exists, created through computer-aided design (CAD) for manufacturing, optimization, or video games. There is, however, a domain gap between the artificial models and the objects sensed by the robot in the real-world that needs to be overcome. The remainder of this chapter describes the challenges posed by this domain gap and summarizes our proposed approach to resolve them.

1.1 Problem Statement

Semantic understanding comprises of a variety of task, this work focuses on object classification as it is a fundamental task in the field. The goal of 3D object classification is to predict from the 3D data of the object the correct label out of K possible labels that are fixed and known in advance. Object here is used as a general term that encompasses not only objects manipulable by a robot but also furniture. In

[1]https://www.worx.com/landroid-robotic-lawn-mower
[2]https://www.avidbots.com/avidbots-solutions/the-neo-platform/
[3]https://www.irobot.at/roomba
[4]https://mag.toyota.co.uk/toyota-human-support-robot/
[5]https://pal-robotics.com/robots/tiago/
[6]http://wiki.ros.org/Robots/AMIGO
[7]https://fetchrobotics.com/fetch-mobile-manipulator/

other words, anything found indoors except for walls and other structural elements. 3D data describes data that cannot be reduced to a single depth 2D image. Single-view 3D data is sometimes referred to as 2.5D, and is a special case of 3D data, but this work focuses on the general case.

The challenges of 3D object classification under domain shift is approached by the research community from different complementary angles. Domain adaptation, for example, takes advantage of a large amount of unlabeled data from the real domain to reduce the gap under the condition that this data is available. Simulation is another angle as it is continuously improved through better sensor noise models and light effects. We choose to focus on the architecture of models for 3D representation and their robustness to the diverse sources of divergence between the artificial and real domain.

1.1.1 3D data representation

Many formats are available to represent 3D data. Point clouds represent it as a list of points with position and other properties attached to every point. Triangular meshes add to that list of points a list of triangles connecting three points to approximate the surface between the points defined. Voxel grids on the other hand represent 3D data as a discretized 3D grid where every cell is set to one if crossed by the object surface and set to zero if empty. This variety of representations complicates the design and comparison of classification methods, as the underlying representation bring its own benefits and limitations. This issue is further investigated in Chapter 4.

1.1.2 Artifacts in simulated 3D data

The 3D data created artificially, or CAD models, is generally of higher quality but is not exempt from artifacts. Artificial models are created by humans with the goal of obtaining good rendering, which can diverge from the goal of creating accurate surfaces. Examples of such artifacts are disconnected surfaces, surfaces defined within the object, or surfaces intersecting, as illustrated in Figure 1.2. Disconnected surfaces are used to obtain better lighting effects in the rendering process, while the last two are remnants of the modeling process itself. These artifacts are easily removed by rendering the model with a simulated depth sensor and combining those different views in the same way that we combine the different views obtained from the real-world sensor. While this step reduces the domain gap with real data, differences in surface remain and limit the accuracy of classifier trained and tested on different domains.

1.1.3 Artifacts in real-world 3D data

Real-world 3D data quality is constrained by the quality of the sensor and of the algorithm fusing different views, also called reconstruction. The depth sensor's noise depends on the sensing modality [4]: time-of-flight, structured light, stereo or active stereo setup all suffer from different types of noise. Reconstruction algorithms limit the impact of random and depth-dependent noise from the sensor with proper data

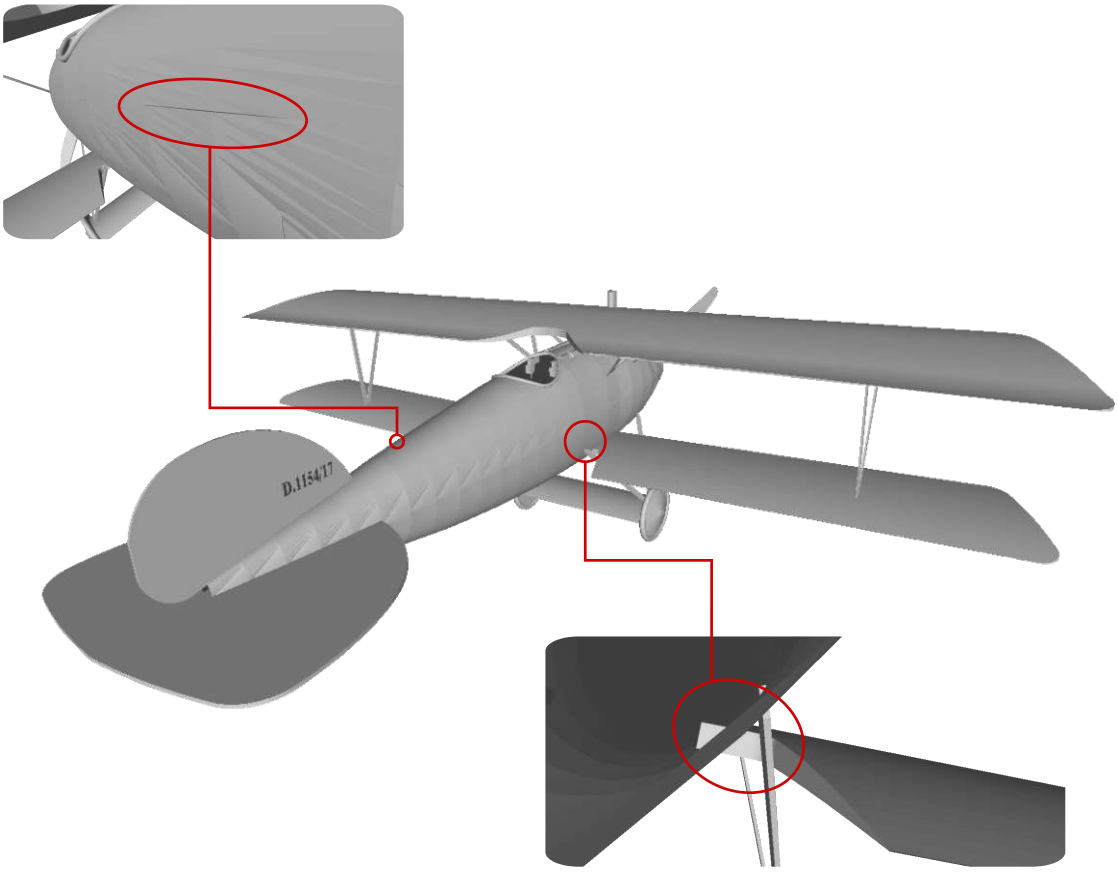

Figure 1.2: Illustration of the modeling artifacts. Top left: a very thin hole is present in the surface. This disconnection does not affect the renderings but changes the surface definition. Bottom right: the wing is crossing through the plane hull. These intersecting meshes define incorrect surfaces but do not affect the renderings.

fusion but brings different artifacts from two sources: the camera pose estimation, and the algorithm used to combine the data once the poses are known. Depending on their reliability, they produce overly smoothed surfaces, or create multiple disconnected surfaces from the same part of a given object (due to poor camera tracking).

Even with state-of-the-art sensors and algorithms, the environment in which the object is perceived remains an unavoidable and major contributor to the gap between artificial and real domain for 3D object classification algorithms. It limits the viewpoints accessible by the robot, and clutter in the area also reduces the parts of the object that is observable both leading to variable degrees of occlusion of the object. The object itself is seen under any possible orientation. While statistical methods such as Principal Component Analysis provide a canonical orientation for a full object, the impact of occlusion on the statistical distribution of the object makes this process significantly more challenging in our case.

Finally, as for any classification task, there exist intra-class variation, and notably regarding object scale. Large-scale dataset of CAD models in particular contain large

object scale fluctuation as they were modeled for different purposes and multiple conventions to define the scale. As such, it is common to create a canonical scale, such as the unit sphere, that is scaling every object to make them fit in a sphere of radius 1. Obtaining a canonical scale for occluded objects in the real world is, however, a much more challenging task, again, because of the impact of the occlusion on the statistical distribution of points and is another contributor to the domain gap.

These challenges of noise, rotation, occlusion and scale are further investigated and suitable architectures are proposed in Chapters 3 5 and 6.

1.1.4 Computational limitations

Due to their autonomous nature, service robots are bound to a limited computational power because they are limited in power supply by their batteries. That computational power is also likely to remain the same throughout their deployment. While this can be partially resolved through the use of cloud computing, that also introduces a host of new challenges in terms of synchronisation, responsiveness and reliability. This computational constraint therefore translates to the need for efficient and fast algorithms to tackle the challenges that the robot will face.

1.2 Contributions and Outline

In this thesis, we focus on understanding and introducing new architectures suitable for the challenges of object classification of real-world 3D data collected by a service robot exploring its environment and under domain shift. To this end, we first investigate the applicability of classical descriptors' design in learned architectures to find a trade-off between robustness and discriminativeness while requiring limited real training data. To best leverage larger artificial datasets, we then analyze the impact of different 3D data low-level representations and best practices to obtain similar robustness to the one obtained by classical descriptors' design. Guided by the findings, we then propose a part-based architecture to deal with the domain shift, and finally a set of improvements on this foundation to make the architecture less sensitive to scale. The general process is described in Figure 1.3. All the contributions will now be described in more detail.

1.2.1 Robust 3D Object Classification by Combining Point Pair Features and Graph Convolution

Classical descriptors have been designed for many years and provide compact and robust representations for point clouds. The advent of deep learning methods [2] brought about a new level of discriminativeness, largely outperforming classical descriptors. This work aims to combine both to obtain robust representation offering competitive classification accuracy. Low-level data is transformed following the design of the Ensemble of Shape Functions [5], but instead of aggregating them using a histogram, the features are directly fed into a suitable deep neural network. The accuracy of this model is further boosted by considering clusters or parts of the objects in the sampling of the pair of points,

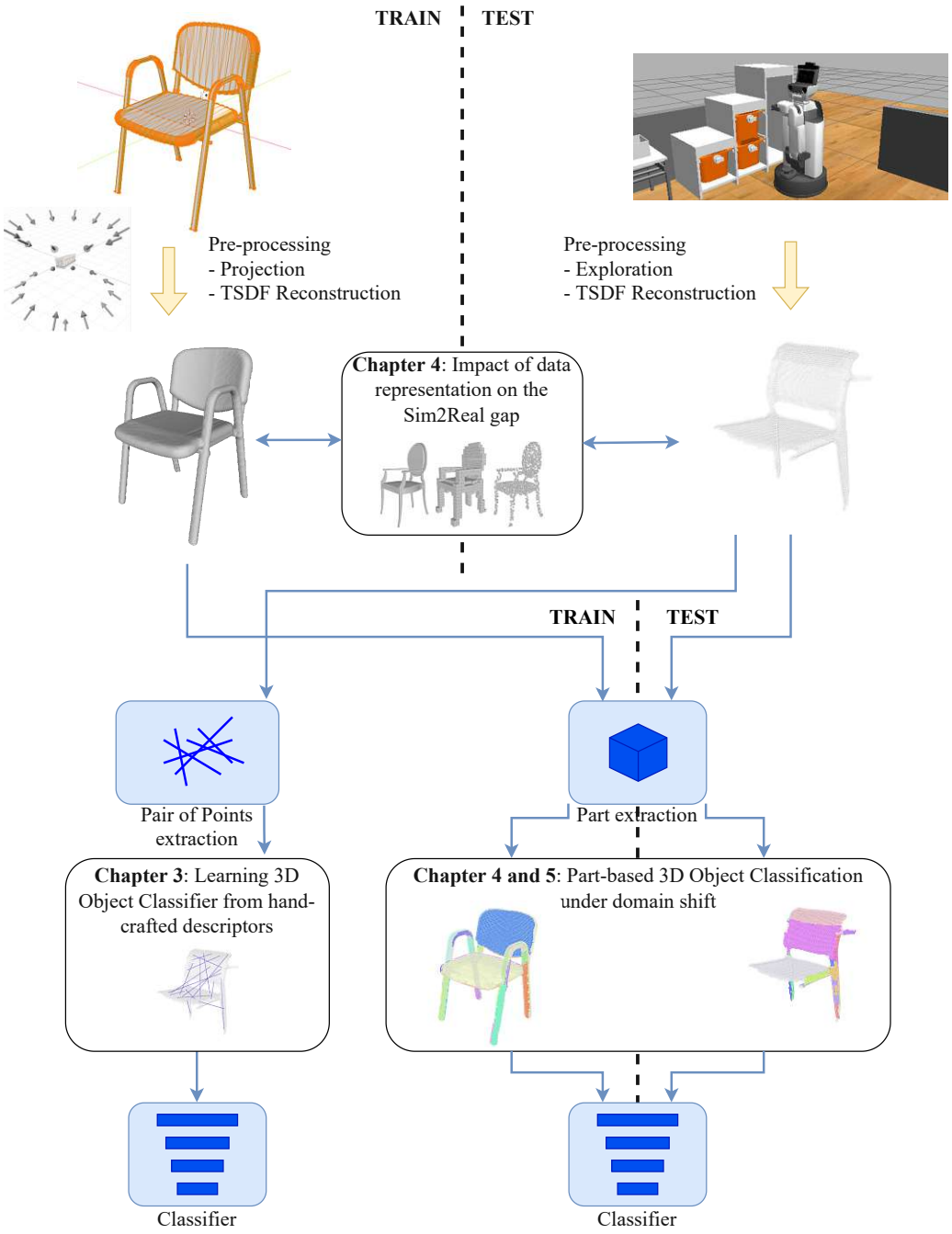

Figure 1.3: Overview of the work presented in this thesis. Artificial data is used for training and reconstructed using a TSDF volume to remove modeling artifacts. Real data is collected by a robot exploring a room, and also reconstructed using a TSDF volume. Objects from both sources are transformed, either by sampling pairs of points, or by segmenting out parts, and classified with an approriate architecture.

and by combining the features obtained for every part using a Graph Convolutional Network [6]. The final architecture provides robustness to rotation and noise present in

reconstruction while providing competitive accuracy for datasets with limited training data. The contributions described here are further detailed and discussed in Chapter 3 and were published in [Weibel, ICRA 2019]

1.2.2 Measuring the Sim2Real gap in 3D Object classification for different 3D data representation

A variety of representations exist for 3D data, all with their trade-off regarding size, precision, flexibility and traversability. These trade-offs also transfer to the learning architecture tailored for each 3D data representation. This work investigates the impact of different artifacts and different types of occlusion on the performance of the aforementioned learning architectures, and compares their individual impacts in the context of domain transfer from simulated to real data. Specifically, the study focuses on PointNet [7] and PointNet++ [8] designed for point clouds, VoxNet [9] designed for voxel grids, and MVCNN [10] designed for multi-view representations. The results underline the importance of hierarchical designs, the benefits of surface representations rather than volumetric ones when dealing with occlusion and sensitivity of all existing architectures to scale differences. The contributions described here are further detailed and discussed in Chapter 4 and were published in [Rohrböck & Weibel, ARW 2021] and [Weibel, ICVS 2021]

1.2.3 Addressing the Sim2Real Gap in Robotic 3D Object Classification

Based on the findings of the previous chapter, the part-based representation seem most appropriate to deal with the challenges of occlusion in the context of 3D object classification under domain shift. This work introduces a simple definition of purely geometric object parts and an appropriate architecture to learn features for each part and combines them to obtain object level representations that are less sensitive to occlusions. The parts are created based on the surface curvature and part features are obtained using PointNet [7]. They are then combined using a Graph Attention Network [11]. Different attention and pooling strategies are evaluated. The approach outperforms other point cloud-based methods when trained on the ModelNet dataset [12] and tested on objects extracted from ScanNet [13]. The contributions described here are further detailed and discussed in Chapter 5 and were published in [Weibel, RA-L 2019]

1.2.4 Robust Sim2Real 3D Object Classification using Graph Representations and a Deep Center Voting Scheme

Graph Neural Networks [6], [11] provide a novel opportunity to deal with the challenges of 3D data. They however do not take into account the relative position of the neighbors in the graph, which is essential information for 3D shapes. Neighborhood-based architectures [14], [15] build on similar concepts but include information about

the relative position of neighbors for improved discriminativeness. Relying on point coordinates, however, introduces a strong dependency on the object scale. This is problematic as the scale of objects in large artificial datasets is generally normalized due to its unreliability and different conventions, but this is a more complicated process for occluded objects that are present in real data as the occlusion affects the statistical distribution and therefore the normalization process. We propose a novel part-based object classification architecture that is less sensitive to scale while being more discriminative than the original Graph Neural Networks. The contributions described here are further detailed and discussed in Chapter 6.

1.3 List of Publications

Parts of the content presented in this dissertation have been previously published in the following papers:

- ***Jean-Baptiste Weibel**, Timothy Patten, and Markus Vincze.* Geometric priors from robot vision in deep networks for 3D object classification. Proceedings of International Conference on Robotics and Automation (Workshop on Multimodal Robot Perception: Perception, Inference and Learning for Joint Semantic, Geometric, and Physical Understanding). 2018.
- ***Jean-Baptiste Weibel**, Timothy Patten, and Markus Vincze.* Robust 3D object classification by combining point pair features and graph convolution. IEEE International Conference on Robotics and Automation (ICRA) 2019.
- ***Jean-Baptiste Weibel**, Timothy Patten, and Markus Vincze.* Addressing the sim2real gap in robotic 3-d object classification. IEEE Robotics and Automation Letters 5.2 (2019): 407-413. Honorable Mention from IEEE Robotics and Automation Letters 2019.
- *Rainer Rohrböck,* ***Jean-Baptiste Weibel**, and Markus Vincze.* Analysis of 3D shape representation in presence of corrupted data. Austrian Robotics Workshop (ARW), 2021.
- ***Jean-Baptiste Weibel**, Rainer Rohrböck, and Markus Vincze.* Measuring the Sim2Real Gap in 3D Object Classification for Different 3D Data Representation. International Conference on Computer Vision Systems. Springer, Cham, 2021.
- ***Jean-Baptiste Weibel**, Timothy Patten, and Markus Vincze.* Robust Sim2Real 3D Object Classification using Graph Representations and a Deep Center Voting Scheme. Under review at IEEE Robotics and Automation Letters (RA-L), 2022.

Chapter 2

Background

This chapter details previous works related to and built upon in this thesis. The different approaches to obtain 3D data, and the available datasets created with those methods are reviewed. An overview of the state-of-the-art of 3D data classification is then given, grouped by the type of approach. Finally, segmentation and part-based approaches are discussed.

2.1 3D data: Depth Sensing and Reconstruction

In this section, we provide an introduction to the sensors and methods needed to create 3D representations of the real world and then give an overview of the available datasets in the field.

2.1.1 Obtaining real 3D data

3D data in this work refers to data that generally cannot be reduced to a single depth image. Indeed, single depth images, sometimes referred to as 2.5D can be seen as a special case of 3D data, and is often processed using techniques developed for RGB images, sometimes computing normals out of the depth to obtain three channels [16], or creating a special mapping to obtain three channels [17]. This approach comes naturally as many depth sensors such as the Asus Xtion, Intel RealSense or Microsoft Kinect Azure provide depth in the image format. Obtaining 3D data requires the combination of multiple images from the sensor.

This brings two challenges. First, the relative robot and therefore camera pose used to capture each view is necessary to combine them meaningfully. This long studied problem in robotics is usually referred to as Simultaneous Localization and Mapping (SLAM). Current state-of-the-art solutions include [3], that developed efficient local features called ORB to match between frames and estimate the relative pose used to capture those frames, [18] that uses image gradient to directly optimize the transformation or [19] that fuses multiple sensors. These solution can be applied with different sets of sensors like monocular or stereo cameras, or directly use the output of depth camera like the Asus Xtion or Intel Realsense, but they take in one image at a time, as they are captured by the robot. In situations where only a collection of unordered images

are available, structure from motion approaches can still be applied [20]. They also rely on feature extraction in every frame, most commonly scale-invariant feature transforms (SIFT) [21], match them across frames and perform a general optimization step of the features position, camera pose and camera intrinsics called bundle adjustment.

The second challenge of depth views combination is related to the data management. While it is possible to simply add points in space obtained from every single view into a large point cloud, this is sub-optimal in that it does not use the redundant observation to reduce the noise in the final observation. Moreover, the final point cloud will have inconsistent point density. Methods either sample a fixed number of points, or learn from local neighborhood, and variable point density will make patterns less consistent in both approaches. Every depth sensor suffers from different types of noise [4], but their systematic errors in depth prediction can be removed by proper calibration, and their random error can be accounted for when fusing views, granted that a noise model exists for the given sensor, like [22] for the Microsoft Kinect. This fusion step comes with its computational challenges, especially when working with larger maps. One of the oldest method is the Truncated Signed Distance Function [23]. The signed distance function represents at every point in space the distance to the closest object surface, being positive when inside the object and negative outside of it. The TSDF stores similar information but only up to a certain distance of the surface, and truncate the value to 1 or -1 when further away from object surfaces. So given a depth measurement from a given pose, the value of the TSDF volume can be updated along the ray of the depth measurement, and set to -1 until a specific distance to the surface is reached and then fused with the previous measurement around the depth measurement. The distance to the object surface where values are updated is chosen according to the depth measurement noise level and the noise level in the camera pose estimation. With proper fusion, the zero-crossings in the TSDF volume define the object surface. This principle has been used in KinectFusion [24] and implemented on GPU to work in real-time. Many methods have built on this idea, notably InfiniTAM [25] that creates many small TSDF volumes and updates all of them to make the method more scalable and usable for larger volumes. BundleFusion [26] optimizes camera poses globally and is able to integrate and deintegrate updates to its TSDF volume to correct errors in the camera poses. Not every reconstruction method relies on TSDF however. ElasticFusion [27] for example directly use a point cloud representation but each point also has a radius attached such that each point represents a small surface (they are referred to as surfels) to avoid storing redundant information. Finally, ScalableFusion [28] directly works on triangular meshes, creating or updating vertex positions. Novel approaches have also emerged in the wake of NeRF [29] that aim to represent the surface implicitly within a neural network set of weights, but are designed for accurate rendering rather than accessing the scene geometry, which is what we are interested in in this work thus making them less applicable here.

2.1.2 3D Datasets

Using the methods described previously, many researchers have worked on creating large-scale datasets of rooms. The NYU Depth Dataset v2 [30] is among the first ones

to also include a complete semantic annotation per frame for a total of 1449 frames. Similarly, ObjectNet3D [31] provides RGB, depth and annotation for an even larger number of frames (90127). But both of these do not include camera poses. Later efforts like the Stanford 3D Indoor dataset [32], SceneNN [33] and ScanNet [13] provided large scale annotated datasets and accurate camera poses enabling scene reconstruction. While the Stanford 3D Indoor dataset focuses on offices with over 70000 frames and includes a limited number of semantic classes, SceneNN and ScanNet cover most rooms and objects found in typical houses with respectively 100 room scans and more than 1500 room scans. Even larger datasets were created recently, chief among them Matterport3D [34] and Gibson [35]. Gibson however does not include complete semantics annotation. While these datasets are extremely useful to researchers and have largely contributed to the progress made in the field, the investment necessary for the larger dataset is considerable, involving dozens of people for many months. Despite that work, they do not cover every indoor scenario and suffer from regional biases, usually including only rooms from a specific area of the world. Moreover, datasets of this scale also create privacy issues in that many people have to accept to open their homes.

To circumvent these issues, many artificial datasets have been created. The most comprehensive effort is the ShapeNet dataset [36] which is a very large collection of CAD models. Two curated subsets have been created: the ShapeNetCore containing 51300 unique models spanning 55 classes, and the ShapeNetSem containing 12000 models spanning 270 classes but also including the real-world dimension of the object among a few other properties. This last set illustrates the difficulty of gathering large sets of scaled CAD models. Due to the existence of various conventions, and because they were created for different purpose, CAD models gathered online need to be manually annotated for scale. The ModelNet [12] dataset, another large-scale CAD model collection, ignores the scale entirely, and provides a 40 class version with 12311 models and a 10 class version with 4899 models. It is commonly used as a benchmark for 3D object classification. A set of the ShapeNet models have also been used in PartNet [37] to provide annotations of the different parts of the objects. Some works like SceneNet [38] also provide artificial scenes created from CAD models. In this case, models have to be gathered, annotated for scale, and combined meaningfully in a floor plan. Habitat [39] goes one step further by providing tools to instantiate agents in these artificial rooms, and can also instantiate agents in the reconstructed rooms of the Matterport3D dataset, Gibson dataset and a few others. SAPIEN [40] builds on top of the PartNet dataset by reusing the same objects and provides simulation tools to interact with the articulated objects of that dataset. Finally, ScanObjectNN [41] is an effort to extract objects from ScanNet and SceneNN that overlap with classes defined in ModelNet40, providing a bridge between artificial and real objects at the scene level.

CAD models are created by humans, and even in curated dataset like ShapeNet and ModelNet, they still contain artifacts from the modeling process. An easy way to remove them and reduce the gap between artificial and real data is to render a set of views around the objects, possibly using a sensor noise model [22] and combine all those views using a TSDF volume [23] (being artificially rendered, the camera pose is always available). While a triangular mesh of constant density is easily obtained from a TSDF volume, it should be noted that a triangular mesh can also be obtained from a point

cloud where normals are computed using a Ball Pivoting technique [42] for example.

2.2 3D Classification

In this section we review the different methods developed for 3D object classification.

2.2.1 Classical methods

There is long history of 2D local descriptors with SIFT [21] being the most widely used. The advent of cheap widely available depth sensors like the Microsoft Kinect significantly sped up the development of 3D local descriptors. One of the earlier example, the Point Feature Histogram (PFH) [43] considers every pair of point in the neighborhood of the point of interest, creates a local reference based on the normals and position of the pair of points. That local reference frame guarantees that the features are rotation invariant. Within that local reference frame, 3 angles are computed. Those 3 angles and the distance between the pair of points are binned into a histogram to obtain the final representation. The Fast PFH (FPFH) [44] was later introduce to keep the useful properties of PFH while speeding up the computation. This speed-up is achieved by only considering the combination of the points of interest with its neighbors instead of every pair in the point of interest's neighborhood. The simplified PFH is then combined with the simplified PFH computed for neighboring points to approximate the complete computation of PFH. The Viewpoint Feature Histogram (VFH) [45] extends the concept and proposes a global descriptor with a viewpoint dependent definition, very suitable for object recognition. The SHOT descriptor [46] also focuses on local changes of the object surface: the neighborhood of a point of interest is oriented with a local reference frame based on the set of points' covariance, which makes the descriptor invariant to object rotations. The points of the neighborhood are then grouped based on their relative position in that oriented spatial grid and the normal angle is considered for each group.

The descriptors described above use a local reference frame, which can be quite sensitive to the quality of normal estimation, which in turn is strongly affected by sensor noise. They are also designed with classification in mind but the Point Pair Features (PPF) [47] uses no local reference frame and is designed for object pose estimation. Two points are randomly sampled in the scene (and not only in the neighborhood of a point of interest) and 4 angles are computed using the point's normal and the vector going from one point to the other. Those angles are then discretized to create the feature representation of that pair of points, which cast a vote for a given object pose if it matches a pair feature of one of the object model. These simple features are rotation-invariant. The Ensemble Shape Feature (ESF) [5] uses similar features for pairs of points and adds to it the ratio of occupied cells in an occupancy grid along the vector connecting the two sampled points. The process is repeated for a given number of pairs to obtain a global descriptor fed to a support vector machine (SVM) [48] used for the object classification. Both of these work have inspired more recent work that try to combine these simple low-level features and more powerful deep learning techniques.

Notably, EPPF [49] uses similar low-level features to ESF but combines them in a 4D histogram, keeping the correlation between features and classifying objects from this histogram using 4D convolutions. PPFNet [50] uses local grouping of PPF features to learn 3D local descriptors using the PointNet architecture [7].

2.2.2 View-based methods

With the advent of modern deep learning and the large improvement in RGB object classification demonstrated by AlexNet [2], a natural idea is to use depth images as input to similar network to obtain powerful 3D object classifier [51]. People have investigated hand-crafted transformations [17] to obtain three channels out of the depth images, similar to the three channels of color images. [16] suggests that the best transformation is to compute the surface normals. Those works focus on single-view depth or 2.5D. To classify 3D data in general, architectures have been developed to deal with sets of views and MVCNN [10] is an early example. All views are fed independently to a 2D convolutional neural network, and the features obtained for each view are max-pooled and fed to a multi-layer perceptron to obtain the final classification. This work was refined and updated with more modern deep learning techniques in [52]. This concept is improved in [53] using an harmonized bilinear pooling. Alternatively to a global pooling step, 3D2SeqViews [54] treats the views as a sequence and uses hierarchical attention aggregation to classify the view sequence. [55] creates a fixed number of clusters from the set of views based on their features, grouping redundant views together before the final classification. [56] jointly learns the object class and the pose of the camera for every view enabling more meaningful pooling. Instead of creating multiple views, it is also possible to explore different projections to include as much information in a single view, as showcased with a panoramic projection with [57], or a stereographic projection with [58]. View-based methods tend to be more sensitive to the object orientation, and the view selection while trivial when working with artificial objects can be a challenging topic when working with real data.

2.2.3 Grid-based methods

Convolutional networks can be extended to work in three dimensions. This comes with an increased memory and computational cost but has been investigated early on with a coarse grid by VoxNet [9]. [59] improves upon that work using orientation boosted voxel networks. [60] tries to address the computational cost by making the network learn from octrees rather than dense grids. A similar idea is used in [61] that learns directly from KD-trees. [62] approaches the same problem from a different angle using binary networks to reduce the computational cost. [63] aggregates multiple points within each cell of the voxel grid using the fisher vector to obtain more meaningful cell features. [64] instead introduce an efficient implementation tailored for sparse voxels grids. While not working on a grid directly, we include in this section methods working on a complete set of points of a point cloud at once. [65] uses as input the point coordinates and considers each point individually before a global pooling step performed using a symmetric function. This is very similar to PointNet [7], which also

uses a spatial transformer network [66] for improved performance. The operation of the network can be understood as a transformation between Euclidean coordinates and position in an implicit learned grid enabling the meaningful combination of each individual point feature using a max pooling function. PointNet++ [8] improves upon PointNet by introducing sampling and grouping modules to enable stacking multiple PointNet modules one after another.

2.2.4 Neighborhood-based methods

Instead of learning from the entire set of points in a point cloud, many works choose to focus on point neighborhoods, emulating the behavior of 2D convolutional neural networks on the irregular domain of object surfaces. Tangent Convolutions[67] recreate 2D surfaces by projecting neighboring points on the tangential plane of the point of interest. SpiderCNN [14] adapts the convolution operation to the irregular domain by using a third-order Taylor decomposition. PointCNN [68] also tries to generalize convolutions to the irregular domain. KPConv [69] creates a set of kernels around the point of interest and aggregates the features of every neighboring point based on their distance to the kernel. This work was extended in [70] to create a rotation invariant representation. DGCNN [15] considers every point in the neighborhood as vertices of a graph connected by edges to the point of interest. Features are learned from those edges, and the graph is dynamically updated in higher levels of the hierarchy of features. [71] creates rigorously rotation invariant features based on the point of interest neighborhood and uses a similar architecture to DGCNN to combine them hierarchically.

2.2.5 Graph-based methods

The neighborhood of points in a point cloud can be retrieved efficiently using a KD-Tree, and both information combined can be seen as a graph with 3D points as vertices and close enough points connected by edges. Triangular meshes are themselves graphs with a specific topology. Deep learning architectures have been developed for general graphs such as Graph Convolutional Networks [6] that learns from the adjacency matrix and a feature matrix. This idea has been extended by GAT [11], which introduces an attention mechanism enabling the weighting of each neighbor of a vertex differently, thus increasing the discriminativeness of the network. These general graph learning architectures have been adapted to 3D data in [72].

There is also a body of work tailored for Triangular Meshes, trying to take advantage of their specific topology. Surface Networks[73] uses a spectral representation of a graph and introduces a learning architecture using the Dirac operator. MeshCNN [74] learns from edges in the graph, introducing a symmetric edge convolution operation and using the edge collapse operation to create a task-driven pooling operation. SpiralNet++ [75] traverses neighboring triangles in a spiral creating a sequence of features that can be learned from with a regular convolution operation. MeshNet [76] relies on the fact that every face of a triangular mesh has three neighbors to create a mesh convolution applied to each face from regular convolution operations and learns from structural and spatial information. Primal-Dual MeshNet [77] creates a primal graph from the mesh faces

and a dual graph from the mesh edges and learns from both graphs alternatively using a GAT network.

2.3 3D Over-Segmentation and Part-based representation

There exists a large body of work dealing with object segmentation from 3D data [78], [79]. A subset of these methods is focused on obtaining an over-segmentation of the scene. Instead of trying to solve the complex and ill-posed problem of object segmentation, these methods focus on obtaining segments as large as possible while still guaranteeing that no segment crosses any object boundary. The segments obtained from such 3D segmentation methods are often referred as supervoxels. This can simplify any later processing step, while also reducing their computational complexity, as the number of supervoxels is much smaller than the number of points. Voxel Cloud Connectivity Segmentation (VCCS) [80] was among the first ones to be designed for point clouds. It creates an adjacency graph from the voxels created from the points in space and grows regions based on normal angles and color differences. This approach is very computationally efficient but produces fairly small supervoxels. Locally Convex Connected Patches (LCCP) [81] builds on this work, and merges supervoxels created by VCCS if they are locally convex and meaningfully connected. The final supervoxels are larger and much more meaningful, and the assumption that convex areas do not contain object boundaries holds true in most situations. As reviewed in [79], there are many methods for part segmentation designed for meshes specifically, as they represent the object surface completely, enabling the development of spectral methods [82] beside the methods based on local geometries [83]. More modern works try to learn more semantically meaningful parts [84] by minimizing the rank of a similarity matrix between points to create patches that are then merged using a Graph Neural Network. [85] proposes an end-to-end Supervised Primitive Fitting Network (SPFN) to extract a set of primitive shapes from a model, while [86] proposes an adapted version of Hough Voting to detect primitives that does not require any learning. VoteNet[87] introduces a neural network and a set of modules such that the network can cast votes for the position of the object center and cluster them. While developed for object detection, the method introduced here is a general approach to combine parts based on the center of the object or of the part. Similarly, Unseen Object Instance Segmentation [88] makes every point cast a vote for the object center, but clusters them with an adapted version of Mean-Shift.

Chapter 3

Robust 3D Object Classification by Combining Point Pair Features and Graph Convolution

3.1 Motivation

Most practical tasks undertaken by service robots require semantic understanding of complex surroundings. Such tasks greatly benefit from the ubiquitous availability of cheap consumer-grade depth sensors, such as the Microsoft Kinect, as they allow for geometric reasoning.

Modern computer vision methods require large amounts of data for training, therefore, it is commonplace to use Computer-Aided Design (CAD) models to simulate the output of common sensors. However, data acquired by robotic platforms in the real world differ in a few characteristic ways (see Fig. 3.1): objects are observed under arbitrary poses, partial occlusion, over- or under-segmentation, and sensor limitations. The fact that objects in the real world have arbitrary poses must be taken into consideration when designing a classifier. Aligning the data is all the more complex when the data also suffer from partial occlusions.

These issues, in addition to the intrinsic limitations of the depth sensors (distance, noise, resolution), can be alleviated by autonomous robots, as they have the ability to explore their environment and accumulate observations. Importantly, reconstruction methods have been successfully developed that fuse depth information using point clouds from multiple viewpoints to generate a globally consistent representation of large environments [27]. Classifying objects directly from reconstructions has many benefits, as more of each object is observed due to multiple viewpoints, and the accumulation of data over many frames reduces sensor noise. However, the unstructured nature of this representation (i.e. no point ordering) is unsuitable for state-of-the-art 2D and 3D convolutional networks. The optimal representation for dealing with 3D data in deep learning is still an open question, but is a very active field of research [7], [10], [61].

In this chapter, we present a novel method that is robust to imperfect data with a deep learning architecture applicable to unstructured point clouds, in particular those that are produced by reconstruction methods. Our model creates a representation that is invariant to rotation around the global vertical axis (z-axis) through the use

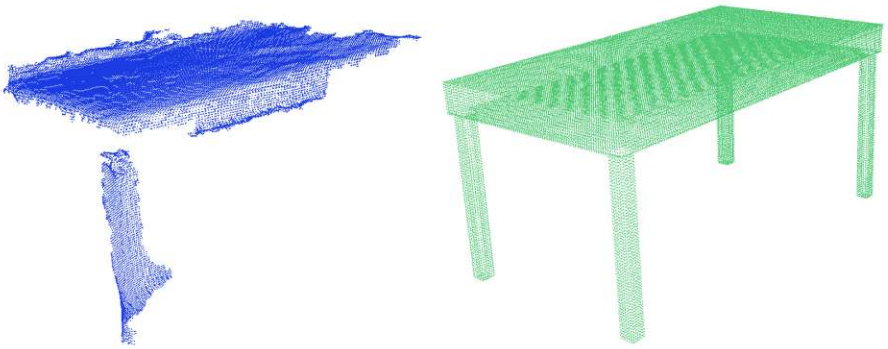

Figure 3.1: Real against artificial data: pervasive noise hinders the performance of methods that are not carefully designed to be robust in the real world.

of features extracted from pairs of points, which avoids the problem of unpredictable object orientation. These low-level features are combined through state-of-the-art set classification methods [7], [65] and a graph convolutional network (GCN) [6].

In summary, the contributions of this chapter are the introduction of

1. a novel attention model to achieve more stable training and
2. a new structured sampling scheme for pair of points enabled by the use of GCNs.

These aspects allow us to significantly improve upon state of the art on the classification task from real-world reconstruction datasets, namely the Stanford 3D indoor dataset [32] and ScanNet [13]. We outperform current methods by up to 22%, which demonstrates the value of incorporating robust features in a deep learning architecture through structured sampling. As an additional test, we show that our method obtains competitive accuracy on a CAD model dataset, ModelNet40 [12], in comparison to other approaches developed for clean data that do not consider real-world robustness.

3.2 Learning from Graphs for Robust 3D Object Classification

Our proposed method builds on the work in [89] and [49] that showed that sets of point pair features are highly discriminative for object classification. We drastically improve the representative power by introducing a graph where each vertex feature encodes information about pairs of points. The resulting structure remains rotation invariant (if the underlying features are rotation invariant) since the graph itself does not impose any strict orientation. The overall architecture is shown in Figure 3.2.

For use in real-world robotics scenarios, we prioritize rotation invariance around the global z-axis by introducing a feature to represent this orientation. The motivation is that most objects have only a small number of stable poses and their representation only depends on the viewpoint around it. A novel attention model is also introduced to stabilize training. This also improves the overall results, particularly when training with fully rotation-invariant features.

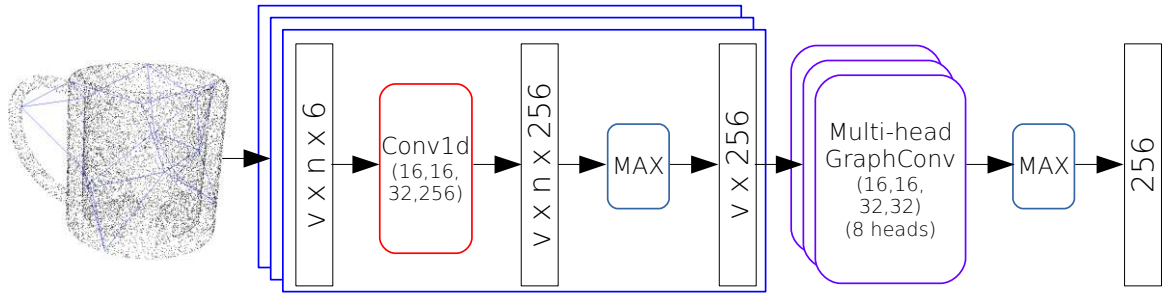

Figure 3.2: System overview of our proposed method where v is the number of vertices and n is the number of pairs sampled for each vertex.

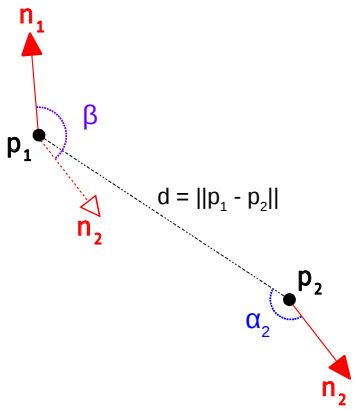

Figure 3.3: Illustration of the point pair features computed at a vertex for two sampled points $\overrightarrow{p_1}$ and $\overrightarrow{p_2}$ (with respective normals $\overrightarrow{n_1}$ and $\overrightarrow{n_2}$). The distance is represented by d, the angle between the normals is represented by β and the angle between the vector $\overrightarrow{p_1} - \overrightarrow{p_2}$ and each of the normals is represented by α_1 and α_2 (only α_2 is shown).

This section describes the proposed architecture. We first explain the point sampling process and the computation of features from each vertex in the graph. We then outline the approach for connecting the vertices in the graph and the design of our graph convolution.

3.2.1 Graph vertices

Vertex features A set of vertices is randomly sampled from the input point cloud where the number of vertices v is significantly smaller than the number of input points N. Each vertex represents a 3D centroid and the sampling ensures that the vertices are spatially separated by a specified distance. A local neighborhood is then defined as a subset of points around each vertex centroid.

For the vertices, features are extracted from pairs of points instead of relying on raw point coordinates by drawing inspiration both from the Point Pair Feature [47], and from the ESF descriptor [5]. A first point $\overrightarrow{p_1}$ is sampled inside the neighborhood

of the vertex and a second point $\overrightarrow{p_2}$ is sampled anywhere in the point cloud. Let us refer to their respective normals (which are assumed to be normalized) as $\overrightarrow{n_1}$ and $\overrightarrow{n_2}$. This sampling scheme is beneficial because pairs of points that are far away tend to be more informative than those that are nearby. As such, sampling from outside of the local neighborhood results in more descriptiveness at an object level than exclusively sampling within the local neighborhood.

Once points are sampled, a number of features are computed for the vertex as shown in Figure 3.3. We extract the distance d between the points

$$d = \|\overrightarrow{p_1} - \overrightarrow{p_2}\|\,, \tag{3.1}$$

the cosine similarity between the normals (where the angle is referred to as β)

$$cos(\beta) = \overrightarrow{n_1} \cdot \overrightarrow{n_2}, \tag{3.2}$$

and the absolute value of the cosine similarity between the vector $\overrightarrow{p_1} - \overrightarrow{p_2}$ and each of the two normals (where the angles are referred to as α_1 and α_2)

$$\left|cos(\alpha_{\{1,2\}}))\right| = \left|\frac{(\overrightarrow{p_1} - \overrightarrow{p_2}).\overrightarrow{n_{\{1,2\}}}}{\|\overrightarrow{p_1} - \overrightarrow{p_2}\|}\right|. \tag{3.3}$$

Finally, as in the ESF descriptor, the occupancy ratio of the vector $\overrightarrow{p_1} - \overrightarrow{p_2}$ is also computed. This quantity is determined by tracing the line in a voxel grid of size 64x64x64 and counting the percentage of filled voxels.

In addition to these features, and retaining invariance to the rotation around the z-axis for robustness, the model includes the difference on the z-axis

$$z_{12} = z_{p_1} - z_{p_2}, \tag{3.4}$$

where z_{p_1} and z_{p_2} are the original z coordinates of the points $\overrightarrow{p_1}$ and $\overrightarrow{p_2}$.

Learning from sets Following the PointNet model, each sextuplet set of features is independently fed to the same convolutional neural network. The feature dimension is enlarged and the set is max-pooled. As described in [89], this amounts to learning the optimal bins over the feature space, taking into account the correlation between the different features.

The complete architecture is shown in Figure 3.2. In difference to the architecture of PointNet, the spatial transformer network is unnecessary as the features of our model do not depend on the orientation because Euclidean coordinates are no longer used. Instead, four 1D convolutional layers are used with a kernel size of 1. A progressively increasing number of filters are then applied before a max-pooling layer to implement the histogram-like feature learning scheme. Each of the layers includes a batch normalization step [90]. A weighted version of the maximum value (over the whole set) of a given filter is subtracted from each output as described in [65]. ReLU is used as an activation function.

3.2.2 Graph edges

Graph connectivity The proposed model is inspired by the GCN model introduced in [6]. The GCN simplifies larger graph convolutional models by only considering neighbors of the first degree. Such models cannot differentiate neighbors from each other. The connectivity of the graph that is fed to the network is therefore particularly important because it introduces more information to partially compensate for the missing order of the neighbors.

In our model, the four nearest vertices of the vertex of interest are connected to strike a good balance between an overconnected or underconnected graph, both of which are detrimental to the learning process. To make the connectivity more similar to the real object shape, connections are only made if the two vertices have an occupancy ratio larger than 0 (due to the noisiness of the feature). Ideally, vertices should be connected according to their geodesic distances, but this is computationally expensive, and would require meshes as input rather than point clouds[1]. Our vertex connection strategy remains computationally cheap and still provides additional information to the model.

Attention model The GCN model can be made more powerful by introducing a sophisticated attention scheme. Instead of considering all neighbors as equal, the neighbors can be weighted according to a criterion that is specific to the attention model chosen. Some examples of attention models have been developed in [72] and [11] that rely solely on the features of the nodes. We will refer to this class of methods as data-driven attention models. To further improve the representational power of a network, multiple attention heads are used for the same layer, and sets of weights are applied per neighbor and per attention head. The attention heads output are concatenated or averaged.

Introducing attention to the GCN model amounts to learning a valid coefficient to replace the normalization factor. For example, in [11], the graph convolution layer becomes

$$h_{v_i}^{l+1} = \sigma\left(\sum_{j \in \mathcal{N}_i} \gamma_{ij} h_{v_j}^{l} W^{l}\right), \tag{3.5}$$

where h_{v_i} is the feature vector of the i-th vertex, σ is the activation function, and W the parameter vector of the layer l. The coefficient γ_{ij} is then defined as

$$\begin{aligned} \gamma_{ij} &= \text{softmax}(e_{ij}), \\ &= \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}, \end{aligned} \tag{3.6}$$

where $e_{ij} = \text{LeakyReLU}(a^T.[h_{v_i}W || h_{v_j}W]))$, $(\cdot)^T$ denotes the transpose operation, $|\cdot|$ denotes the concatenation operation and a is the vector of learned parameter for the attention. This attention model is prone to overfitting when the features of the vertices do not contain information about their relative position, which is the case when using coordinates. To compensate, we introduce an attention model that operates directly on edge features and more specifically on their relative position. We redefine γij as

[1] The point cloud format assumes less information about the object. It is straightforward to transform from a mesh to a point cloud, but the opposite transformation is more difficult.

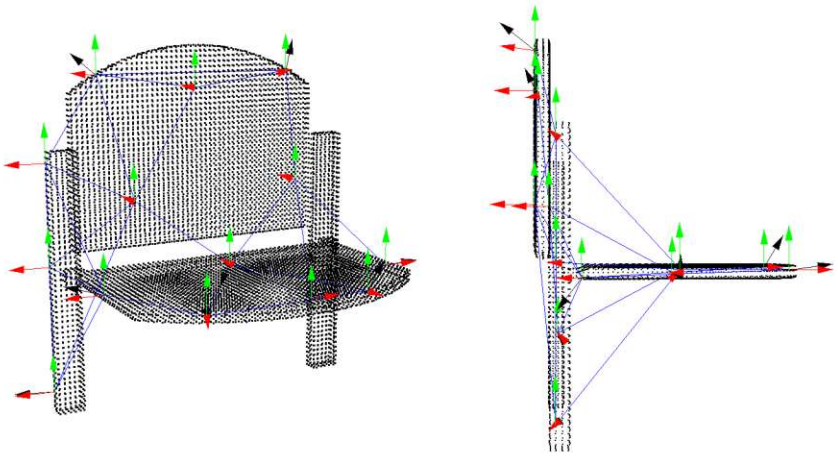

Figure 3.4: Illustration of the local reference frame of a chair under two orientations. The normal vector shown in black, the z-axis (i.e. $\overrightarrow{u}$) shown in green and $\overrightarrow{v}$ shown in red. The edges of the graph are shown in blue.

$$\gamma_{ij} = \frac{\exp(a^T e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(a^T e_{ik})}, \tag{3.7}$$

where e_{ij} is the coordinate of vertex v_j relative to vertex v_i. This approach is more similar to classical convolutions as it allows each attention head to specialize on a specific relative position around the point of interest. Rotation invariance is preserved by defining a local reference frame (LRF) that is specific to the point of interest, while being independent of the orientation around the z-axis. This is achieved by defining three vectors, as shown in Figure 3.4,

$$\begin{aligned} \overrightarrow{u} &= (0, 0, 1), \\ \overrightarrow{v} &= (n_{1x}, n_{1y}, 0), \\ \overrightarrow{w} &= u \times v, \end{aligned} \tag{3.8}$$

where $\overrightarrow{p_i}$ and $\overrightarrow{n_i}$ are the vector and normal vector of the centroid of the i-th vertex in the original reference frame, and the center of the LRF is the point of interest itself. The first vector ($\overrightarrow{u}$) of the LRF is the z-axis itself. The second vector ($\overrightarrow{v}$) is the component of the normal of the point of interest that is in the x-y plane. The third vector ($\overrightarrow{w}$) is obtained by the cross product between the first two vectors. The e_{ij} coefficient in (3.7) is then defined as the coordinates of the j-th vertex in the local reference frame computed around the i-th vertex.

3.3 Experiments

This section presents the experimental results. The first set of experiments compares our proposed approach to state-of-the-art methods for object classification in real-world scenarios by using two reconstruction datasets. The second set of experiments is

performed on a large scale artifical model dataset to put our method in context with other competing deep learning methods that use various data representations. Lastly, we evaluate the contribution of our proposed attention model, showing that its inclusion is very important for achieving high classification accuracy.

3.3.1 Experimental setup

Implementation Our final model is described in Figure 3.2. We sample 800 pairs of points for each of the 16 vertices. The features of each vertex are obtained after feeding them through a series of 1D convolutional layers (kernel size 1 with filters 16, 16, 32 and 256) and max-pooling the whole set. We then use 4 graph convolutional layers. Each of these layers has 8 attention heads with 16, 16, 32 and 32 filters. The output of each attention head is concatenated. Finally, the resulting features are max-pooled over all vertices and the result is passed to the classification layers that consists of two fully connected layers of size 128. All these parameters are a result of a tradeoff between memory and representational power of the network. Under these parameters, the preprocessing time needed to construct the graph and sampled pairs is on average 5ms on a single core of an i7-7700K CPU. Furthermore, the amortized time to train over one batch (both preprocessing and learning) of 32 examples is 249ms on a Nvidia GTX 1070.

Datasets Evaluation is performed on two datasets of real reconstructed scenes, the Stanford 3D Indoor Dataset [32] and the subset of ScanNet [13] that is defined for the classification task. We follow the evaluation procedure described in [49]. For the Stanford dataset, this means using a random 60-40% split for the training and test set (the original split provided is based on which building is explored which is unsuitable here because some classes would not be present in the training or test set). The classes clutter, floor, ceiling and wall are omitted as in [49]. The first version of the ScanNet is used with the same 14 classes in [49] (i.e. basket, bed, cabinet, chair, keyboard, lamp, microwave, pillow, printer, shelf, stove, table and tv). In addition, we presents results on the artificial dataset ModelNet40. This is widely used to evaluate models working on 3D data.

3.3.2 Evaluation on reconstruction datasets

Table 3.1 reports the average classification accuracy and class accuracy on the two reconstruction datasets. We compare our approach with the state-of-the-art methods for point cloud object classification. As the table shows, our proposed approach outperforms the existing methods. On the Stanford dataset our method achieves 9.8% better accuracy and 22.6% better class accuracy over the next best performing approach. On the ScanNet dataset, the improvement over the state of the art is less significant, achieving 1.7% better accuracy and 4.6% class accuracy. Many classes in the Stanford dataset are vertical objects (e.g. board, door and window) so the addition of the z_{12} feature is particularly helpful for this dataset. This feature is a major contributing factor to the larger performance difference. The ScanNet dataset has less vertical objects and

Table 3.1: Object Classification accuracy on Reconstruction Datasets

Dataset	**Metric**	**Stanford** [32]	**ScanNet** [13]
EPPF3D	Accuracy (%)	81.94	70.57
	Class Acc. (%)	66.37	44.35
EPPF4D	Accuracy (%)	83.22	72.1
	Class Acc. (%)	65.11	45.7
PointNet	Accuracy (%)	64.3	63.04
	Class Acc.(%)	42.48	37.5
Ours	Accuracy (%)	**93.0**	73.8
	Class Acc. (%)	**89.0**	50.3
Ours w/ scale	Accuracy (%)	89.1	**78.8**
	Class Acc. (%)	86.8	**58.4**

Table 3.2: Classification accuracy on the ModelNet40 dataset [12]

	MN40	**Input**
VoxNet [9]	83	Voxel Grid
PointNet [7]	89.2	Point Cloud
KD-Networks [61]	**91.8**	KD-Tree
MVCNN [10]	90.1	Views
EPPF4D [49]	82.1	Point Cloud
Ours	87.6	Point Cloud

so the performance gain is less pronounced. Nonetheless, our model still outperforms existing approaches on both these challenging datasets.

These datasets contain reconstructions of *real* objects, therefore, we also evaluate the importance of object scale. This is shown in the bottom row of Table 3.1. The scale information is incorporated by multiplying the features d and z_{12} by the factor by which the point cloud is scaled when fitting it into the unit sphere. This operation recovers the original values of these features while keeping all other aspects the same. The scale information significantly improves the performance on the ScanNet dataset. The performance increases by 5% for accuracy and by 8.1% for class accuracy. Surprisingly, the scale information does not improve the results on the Stanford dataset. This is explained by the fact that the confusion of the Stanford dataset objects mostly happens between classes of similar sizes. These classes have very similar scale, which potentially leads to further confusion between the classes and, therefore, limits the benefits of the added information.

Table 3.3: Evaluation of different configuration on the ModelNet10 dataset [12] (Accuracy / Class average Accuracy)

Attention Model	With z_{12}	Without z_{12}
Data-driven	89.1 / 88.9	-
XYZ Coordinates	92 / 91.7	90.7 / 90.7
LRF Coords	89.8 / 90.1	86 / 86

3.3.3 Evaluation on clean artificial data

Table 3.2 presents the results of our method on ModelNet40. This dataset is widely used to evaluate models operating with 3D data and allows a fair comparison of different data structures. We include a representative subset of the methods developed for, and particularly evaluated on, this dataset. Although our model performs slightly worse than other models, this dataset only presents clean and complete data. Additionally, the data is pre-aligned, which diminishes the importance of rotation invariance. Although KD-Networks [61] performs best, this method needs aligned models and is not robust to rotation or sensor noise. As such, it would fail on reconstructions. This is a key observation, that methods can be tuned for good performance if real-world robustness is not required. Most of the misclassifications by our method are caused by confusing `plant` as `flower_pot` (8.2% of all misclassifications), `table` as `desk` (8.0%), `dresser` as `night_stand` (3.6%) and `dresser` as `wardrobe` (3.5%). Many have not only similar geometric shapes but also similar semantics, which would limit the impact of the mistake in the context of a robotic deployment, or could be resolved by exploiting scale information.

3.3.4 Evaluation of attention models

Two important contributions of our work are the LRF-based attention model and the z-axis feature z_{12}. To evaluate exactly how much performance is gained by their inclusion we perform experiments with various attention models with and without the inclusion of z_{12}. The results for ModelNet10 (a subset of ModelNet40) are shown in Table 3.3.

We evaluate different configurations on the artificial dataset to remove any source of noise to assess our attention model in a "worst-case" scenario. The LRF-based attention needs a given point normal to make a large enough angle with the z-axis for the LRF not to be ambiguous. As such, points cannot be sampled on horizontal planar surfaces. Such surfaces are common in ModelNet10 where many classes are furniture (proportionally much more so than ModelNet40). To give a more complete comparison, we also include the results obtained using the raw coordinates in the original frame (XYZ coordinates attention model) instead of the coordinates in the local reference frame. This is only possible with this dataset because the data is aligned. This approach does not have any restriction on the sampling and can therefore be considered as the

"best-case" for the attention model.

The data-driven attention and our LRF-based attention achieve very similar performance in this setting. Despite its performance, the data-driven approach is susceptible to overfitting when used without the z_{12} feature. In this case the vertices no longer contain information regarding their relative position. When removing the constraint on the vertex sampling, we observe that the XYZ coordinates attention model outperforms the data-driven attention model. This is encouraging as perfectly horizontal planar surfaces are less common in everyday objects that are manipulated by robots. This supports the use of the LRF attention model for real-world applications.

3.4 Discussion

This chapter addresses the important robotics task of object classification from 3D data. We present an approach that combines classical features within a graph convolutional network to retain the robustness of the handcrafted features while benefiting from the power of deep learning. The model is based on pairs of point features that are extracted in a structured manner. The features from the sets are then combined through a graph using our novel attention model. Experiments with our method show high discriminativeness and robustness to noise found in real sensor data. We achieve significant performance gains over current state of the art on two challenging reconstruction datasets while maintaining competitive performance on a clean CAD model dataset. Notably, we empirically showed that building on a representation that is invariant to the rotation around the z-axis provides sufficient robustness for classifying objects in the real world.

Highlights

(1) Point Pair Features are discriminative enough to perform 3D object classification efficiently.
(2) PointNet is a suitable learning architecture to replace and improve upon histograms in classical descriptors.
(3) Graph Convolutional Network, and even more Graph Attention Network, are a powerful architecture for 3D data and its sparsity.

Chapter 4

Measuring the Sim2Real gap in 3D Object classification for different 3D data representation

4.1 Motivation

Understanding the geometry of the environment a robot is operating in is key to its safe operation. Whether it is to identify obstacles or to decide which elements of a scene are indeed objects that will be safely grasped and interacted with, the semantic information provided by 3D object classification is an essential element of this environment understanding.

The state-of-the-art of 3D object classification has continuously improved [8], [15], [52], but new methods are evaluated on standard CAD model datasets, like ModelNet [12]. This is not enough for robotic applications because data captured in the wild suffers from frequent occlusions, smoothed-out surfaces, and over- or under-segmentation. This difference in performance on artificial and on real data is partially demonstrated in [41], but exclusively for point-cloud based methods. Indeed, the evaluation of the Sim2Real gap is made more complex by the co-existence of different 3D data representation in the field. Each of these representations come with their own advantage and drawbacks for object classification.

The contribution of this chapter is to complement this original study by evaluating the Sim2Real gap in multi-view, voxel grid and point cloud based methods. To do so, we select representative deep learning models based on different data representation and use their respective performance as a proxy to evaluate the Sim2Real gap in 3D object classification for each representation. Point clouds (using PointNet [7] and PointNet++ [8] as representatives), multi-view (using MVCNN [52] as representative) and voxel grids (using VoxNet [9] as representative) are the selected representations for evaluation as illustrated in Figure 4.1. We train the deep learning models using a subset of the ModelNet [12] dataset (the entire dataset contains 12,311 artificial CAD models spread in 40 classes), and evaluate them using a subset of the ScanObjectNN dataset [41] (the entire dataset contains around 15,000 real objects spread in 15 classes, with 2902 unique object instances). In a second phase, we perform a set of expefments on modified artificial data to separately evaluate the impact of specific design elements

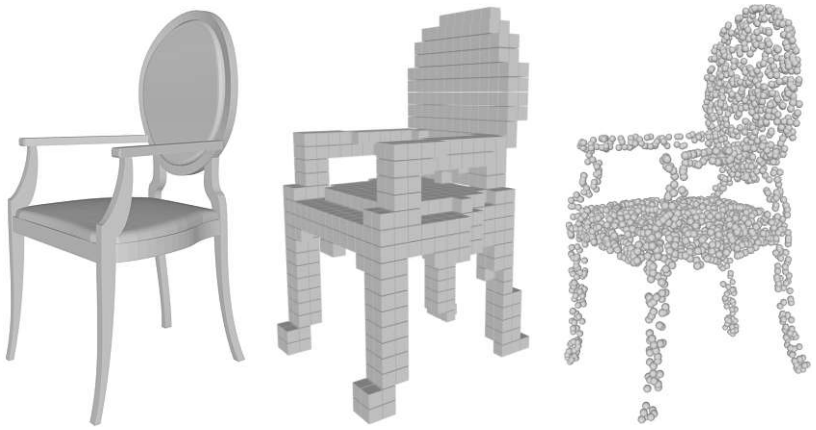

Figure 4.1: 3D data representations: view-, grid and point-based representation

from each deep learning model.

As a results of all these experiments, we formulate a set of design guidelines to improve performance on real-world data in the field of 3D object classification.

4.2 Measuring the Sim2Real gap on ScanObjectNN

In this section, the experimental conditions used to measure the Sim2Real gap are first described, and an analysis of the results is then performed.

4.2.1 Experimental setup

In this set of experiments, models are trained on the subset of 11 classes from ModelNet40 dataset that overlaps with the classes defined in the ScanObjectNN dataset, and then evaluated with objects from the same 11 classes from the ScanObjectNN dataset (as defined in [41]). However, to make the evaluation possible, both datasets have to be transformed.

First, the CAD models of the ModelNet dataset are rendered from multiple viewpoints and the obtained depth maps are used to reconstruct the object using a Truncated Signed Distance Function (TSDF) volume [24]. This step is done to remove artifacts, and more specifically, surfaces defined within the object itself (remnants of the human object modeling process), that would only affect PointNet and VoxNet, but not MVCNN. Object models are first scaled to the unit sphere, and 26 depth maps are created: 12 views in a circle slightly above the object, 12 views in a circle slightly below the object, and the top and bottom views. They are combined into a single TSDF volume from which the object mesh is extracted. The result of the process is illustrated in Figure 4.2

As the ScanObjectNN dataset was originally designed to evaluate point cloud-based object classification methods, a transformation is also necessary, because this format is not suitable for the view rendering necessary for the MVCNN. The surface of the object models is reconstructed from the dense point cloud with pre-computed normals using a ball pivoting method [42].

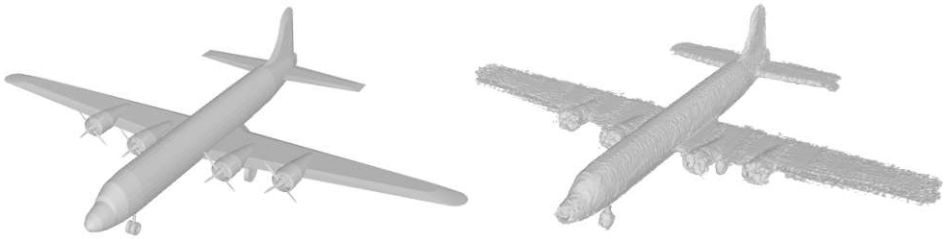

Figure 4.2: CAD Model (left) and TSDF reconstruction (right)

The transformations of both the train and test set make the results presented impossible to directly compare with those presented in [41], but were necessary to fairly compare the different methods evaluated in this chapter.

PointNet [7] and PointNet++ [8] are used for point cloud representation, VoxNet [9] for voxel grids, and MVCNN [52] for multi-views. While these models might not reach the highest accuracy for their respective representation, the focus in this chapter is on the relative change in performance, and their architectural simplicity makes them more representative of the behavior of other models based on the same representation. For every model, the authors' code is used, and the parameters of the chapter are respected. MVCNN is evaluated using the VGG-11 backbone and 12 shaded views (in a circle slightly above the object), PointNet++ and PointNet use 1024 points, and VoxNet is trained on a 32×32×32 grid.

4.2.2 Results

Table 4.1: Impact of the TSDF reconstruction of object models, mean per class accuracy is reported in percent

Train	Test	MVCNN	VoxNet	PointNet++
CAD	CAD	89.4	82.9	88.0
CAD	TSDF	83.4 (-6.0%)	82.2 (-0.7%)	85.6 (-2.4%)
TSDF	TSDF	88.5	83.1	87.1
TSDF	CAD	85.8 (-2.7%)	80.8 (-2.3%)	85.0 (-2.1%)

Before evaluating the gap between artificial and real data, we look at the impact of the transformation performed on the artificial data, that is the rendering of views and fusion of these views using a TSDF volume. Table 4.1 shows that the absolute performance of every method is not significantly affected by the data transformation, but differences do exist as every method accuracy drops a little when applied to the other version. MVCNN is disproportionately affected as minor imperfections in the TSDF reconstructions are made more prevalent by the shading applied to the depth images before being fed to the CNN.

Table 4.2 presents the results of evaluating the different deep learning models on the ScanObjectNN dataset and on the test set of the ModelNet dataset, when trained on the pre-defined training set of the ModelNet dataset. Two ScanObjectNN variants

Table 4.2: Results when training on ModelNet and evaluating on ScanObjectNN, mean class accuracy and relative change is reported in percent

Model	ModelNet	OBJ_ONLY	OBJ_BG
VoxNet	90.08	49.77 (-44.8%)	54.36 (-39.7%)
MVCNN	93.49	62.69 (-33.0%)	58.58 (-37.3%)
PointNet++	93.31	52.49 (-43.8)	55.78 (-40.2%)
PointNet	90.31	54.22 (-40.0%)	55.71 (-38.3%)

are evaluated: the OBJ_ONLY, where only points belonging to the actual object are considered, and the OBJ_BG where all points falling into the bounding box of the object are considered.

4.2.3 Analysis

Overall, the MVCNN is the best performing approach, with a large gap when looking at the OBJ_ONLY variant. Because the MVCNN only relies on object views sampled on a circle slightly above the object, it is less affected than other methods by the absence of data at the bottom and under the object in the ScanObjectNN dataset.

The OBJ_ONLY variant tends to be over-segmented (parts of the objects are missing, for example, decorative cushion occludes parts of a sofa but would not be included), while the OBJ_BG is under-segmented (many background points are included, like the floor). MVCNN is clearly more affected by the under-segmentation than the over-segmentation, which is in line with the experiments in [52] suggesting that the model is sensitive to the object silhouette. On the other hand, VoxNet, PointNet and PointNet++ performs similarly on both variants. Looking further, results are significantly better on the OBJ_ONLY variant for the `table` class (relative improvement 122.2% for VoxNet, 82.4% for MVCNN, 47.6% for PointNet++ and 63.2% for PointNet) which often include the ground floor in the OBJ_BG variant. On the other hand, VoxNet (-10.7%), PointNet++ (-40.6%) and PointNet (-17.5%) performs worse on the `shelf` class. So removing the background points as in OBJ_ONLY seems only beneficial to certain classes while being detrimental on other classes, leading to similar overall performance on both variants for VoxNet, PointNet and Point++. We hypothesize that all methods are sensitive to the addition of background points, but VoxNet, PointNet++ and PointNet are more sensitive to holes than MVCNN, as they break local patterns, whereas MVCNN better learns the overall object appearance from its silhouette.

4.3 Disentangling the impact of design choices

In this section, objects from the ModelNet dataset are altered in specific ways to better understand the impact of design choices of each deep learning model evaluated in this chapter and their consequences on the Sim2Real gap in 3D object classification. The impact of occlusions is studied in three ways. First, a hole is grown in the object model, until a certain percentage of the object surface is reached. second, the set of views used

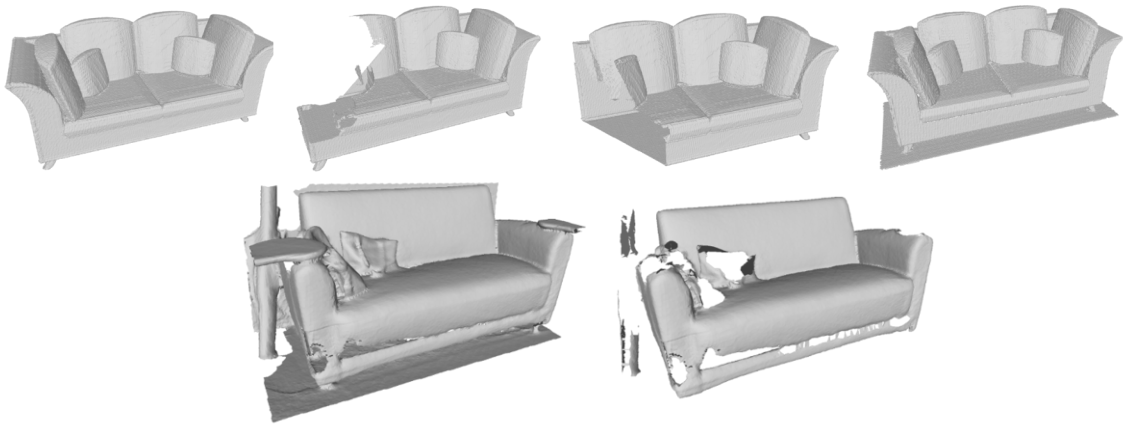

Figure 4.3: Illustration of the alterations of artifical data. Top: complete sofa, with random occlusion (as in Figure 4.4), with a cut (as in Table 4.5), with a bottom plane added (as in Table 4.6). Bottom: an example sofa in the OBJ_BG and in the OBJ_ONLY variant of ScanObjectNN

to reconstruct the object is limited, using only a half or a quarter of them. Third, a section of the object model is cut, removing a fixed volume of the bounding sphere. This second type of occlusion also affects the scale of the object more once normalizing the occluded objects. The impact of under-segmentation is studied by adding a supporting plane under the object. Those alterations are illustrated in Figure 4.3. All deep learning models are trained on reconstructed objects models from the 40 classes of the ModelNet dataset and evaluated on the altered reconstructed test set, according to the experiment performed.

4.3.1 Hierarchical learning from object parts

In this first set of experiments, the impact of hierarchy and subdivision of objects in deep learning models is investigated. Intuitively, learning from parts of the object, and relations between these parts rather than the entire object at once should be more robust to objects occlusions. The performance of deep learning models under increasing amount of random occlusions is therefore evaluated and presented in Figure 4.4. Occlusions here are generated by selecting a random face in the object mesh, and growing the region to be removed until a certain percentage of the object surface area is reached. PointNet performs the worst out of all models, which is consistent with the hypothesis, as it is the only model considering the entire set of points at once. PointNet++ performs the same operation at multiple levels, each level being applied on a larger subset of the object than the previous one. This design mitigates the issue as the performance degrades at the same rate as VoxNet, which also learns hierarchically thanks to the use of classic convolutions. MVCNN degradation relative to the level of occlusion is the slowest, as it benefits from both the hierarchical learning from convolutions in single views and the subdivision of the object in a set of views. We also noticed that while MVCNN performs better with multiple views, when using a single view of a clean artificial object

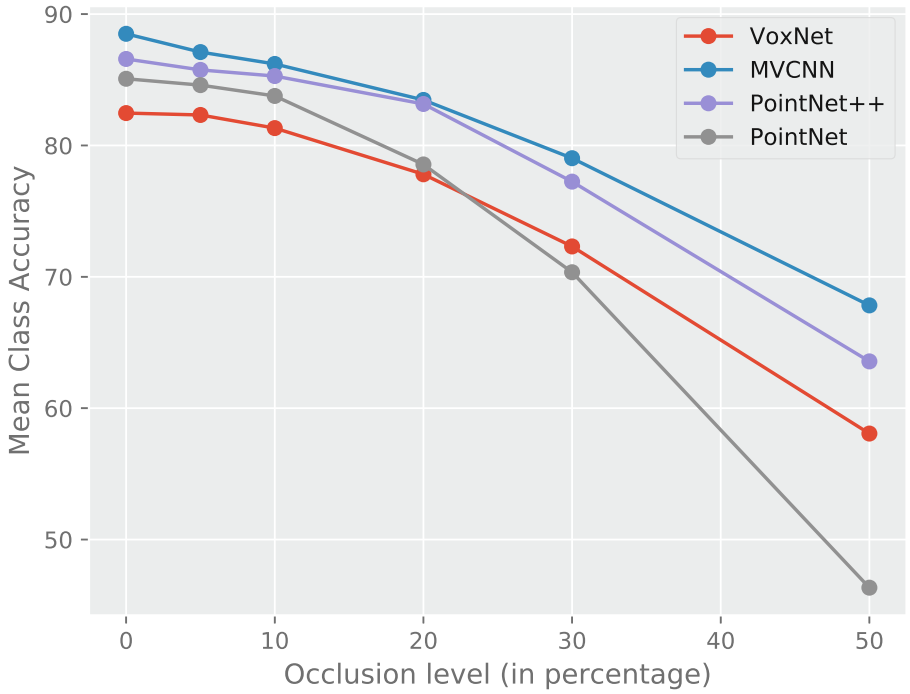

Figure 4.4: Mean class accuracy for various levels of occlusion

(setting the other 11 to black), MVCNN performance only degrades by 13.8%, showing that MVCNN is quite robust to multiple corrupted views.

4.3.2 Impact of Surface-based or Euclidean-based Representation

Table 4.3: Impact of limited viewpoints (full and half models) during model reconstruction. Mean class accuracy and relative change in percent.

Model	Full	Half (Front)	Half (Back)
VoxNet	71.41	67.33 (-5.7%)	62.00 (-13.2%)
MVCNN	78.14	73.45 (-6.0%)	67.63 (-13.5%)
PointNet++	70.58	64.23 (-9.0%)	56.79 (-19.5%)
PointNet	75.27	67.45 (-10.4%)	64.10 (-14.8%)

MVCNN, because of its use of projection, only ever considers one side of an object surface at a time. On the other hand, PointNet++ considers local subset of points defined by the Euclidean distance rather than the co-visibility of points. A robot operating in the wild can rarely observe a given object under every viewpoint. For example the top of the bottom of a table are unlikely to be both accessible. We hypothesize that patterns learned on subsets defined by the Euclidean distance will be

Table 4.4: Impact of limited viewpoints (quarter models) during model reconstruction. Mean class accuracy and relative change in percent.

Model	Quarter (Front)	Quarter (Back)
VoxNet	60.68 (-15.0%)	55.92 (-21.7%)
MVCNN	69.10 (-11.6%)	61.85 (-20.8%)
PointNet++	57.73 (-18.2%)	49.39 (-30.0%)
PointNet	51.62 (-31.4%)	49.14 (-34.7%)

more affected by viewpoints constraints than patterns learned on those defined by the object surface.

The impact of limited views is evaluated in Table 4.3 and Table 4.4. In this experiment, 12 views are generated all around the object on a horizontal circle at half the height of the object (the circle of views is therefore at a equidistant height to the two used to create the training set). We then report the performance on various subsets of views: the half-circle subsets include the 6 views of the front (respectively back) of the object, the quarter circle experiments report the average results when including the 3 left views and when including the 3 right views of the front (respectively back) of the object. The front, back, left and right are defined by the version of ModelNet where objects models are aligned.

PointNet++ performance and even more so, PointNet performance degrades faster than VoxNet and MVCNN. With more and more limited viewpoints available, the difference between surface and Euclidean neighborhood definition grows bigger. While VoxNet theoretically falls into the Euclidean type of representation, we attribute its relatively good performance here to the coarseness of its voxel grid. Indeed, in most situation, front and back surface of an object are likely to fall into the same voxel, mitigating this issue. For example, on the `bathtub` class, PointNet++ and PointNet show respectively a 85.4% and 87.1% decrease compared to -67.1% and -60.0% for the VoxNet and MVCNN respectively on the quarter (back) experiments. This implies a stronger reliance on the presence of both the outer (present) and inner (absent in this experiment) surface of the bathtub. Another example is the `bowl` class, where most models are barely affected, except the PointNet with an average decrease on all quarter experiments of 56.3%. While the class is discriminative enough thanks to its curvature that is uncommon in this dataset, PointNet reliance on both sides of the bowl being present leads to incorrect classification.

4.3.3 Impact of over- and under-segmentation and scale

Many CAD models like those provided in the ModelNet dataset do not have a reliable scale information, as the unit used to create them is not provided. It is common to scale them to the unit sphere. As occlusions of objects will affect their bounding spheres, they affect the final scale of the object. To investigate further over- and under-segmentation and their connection with scale, a new type of occlusion is evaluated in Table 4.5. The object is cut vertically at a certain percentage of the unit sphere diameter, guaranteeing that the resulting object's scale will be changed when re-scaling the remaining points to

Table 4.5: Impact of scale, and over-segmentation. Mean class accuracy and relative change in percent.

Models	30% cut	30% cut scaled	50% cut	50% cut scaled
VoxNet	73.5 (-18.4%)	73.0 (-19.0%)	73.0 (-19.0%)	45.2 (-49.8%)
MVCNN	83.1 (-11.1%)	86.0 (-8.0%)	71.0 (-24.1%)	72.7 (-22.2%)
PointNet++	84.3 (-9.7%)	83.3 (-10.7%)	71.1 (-23.8%)	68.8 (-26.3%)
PointNet	79.7 (-11.7%)	76.4 (-15.4%)	26.0 (-71.2%)	40.4 (-55.3%)

Table 4.6: Impact of under-segmentation. Mean class accuracy and relative change in percent.

Models	Bottom plane
VoxNet	66.5 (-26.2%)
MVCNN	78.3 (-16.2%)
PointNet++	25.3 (-72.9%)
PointNet	23.2 (-74.3%)

the unit sphere. The reliance on the scale information of VoxNet is made clear, as the scaling and centering operation resulting from the occlusion strongly affects the voxel grid. PointNet and PointNet++, thanks to the fixed number of sampled points still extracts discriminative patterns from scaled up occluded objects, and MVCNN already scales every view around the object.

Under-segmentation is also evaluated in Table 4.6 by adding a ground plane under every object of the same width and length as the object. These experiments confirms MVCNN sensitivity to under-segmentation shown in Table 4.2, but also demonstrates the limits of the fixed number of points used in PointNet and PointNet++ representation. Not only does it add outlier point, but it also consequently limits the number of points that can be sampled on the object, forcing them to be further apart, and losing any discriminative local patterns in the process.

4.3.4 Application-specific considerations

While this chapter focused so far on generally applicable guidelines for 3D object classification, knowledge about the end application can also inform the design of the classification method.

For example, the higher absolute performance of the MVCNN in this chapter is partially attributed to its implicit use of surface normals, necessary to the creation of shaded images. They make classes with finer surface details easier to distinguish, making the method more robust to random occlusion. Looking at the difference between performance on complete reconstructed models and models with a 50% random occlusion, the benefit is noticeable for classes like `keyboard` (0% for MVCNN, -60.0% for VoxNet, -40.0% for PointNet++, -65% for PointNet) or `curtain` (-5.3% for MVCNN, -16.0% for VoxNet, -22.2% for PointNet++, -16.7% for PointNet). This benefit is tied to the availability of reliable surface normals in the end application.

The fixed set of viewpoints used by MVCNN can also be tuned to the application, but because of the projection process, it is less suitable to represent concave parts. Looking at the same experiments as before for the `sink` class, MVCNN (-36.8%) is performing worse than VoxNet (-9.3%), PointNet++ (17.6%) or PointNet (-20.0%).

4.4 Discussion

The set of experiments presented here underlines the strengths and weaknesses of point cloud based, voxel grid based and multi-view based representation of 3D data. Through a set of experiments on both real and artificial data, design guidelines have emerged to further reduce the Sim2Real gap in 3D object classification. In particular, based on the results presented here, we advise to learn features hierarchically, as is already common, using object surfaces to define point neighborhood, rather than the Euclidean distance between points, and favoring data creation producing over-segmentation, especially when opting for a fixed number of points representation (which presents significant computational benefits). All these choices mitigate the impact of occlusions present in real data. Moreover, the application and the set of classes that will need to be recognised can further guide the design of the approach, as surface normal, and their implicit use in shaded images help differentiate finer details, but concave elements are harder to represent reliably using a set of views.

Highlights

(1) View-based methods dominate most other approaches both thanks to research already invested in 2D convolutional networks and because they split the object into subparts to better deal with occlusions.
(3) With the exception of view-based methods that benefit from shading, they all struggle with finer details of objects. This could be alleviated by adaptive subparts helping to focus on those smaller structures.
(2) Surface-based neighborhoods are preferable to Euclidean-based neighborhoods when objects are occluded.

Chapter 5

Addressing the Sim2Real Gap in Robotic 3D Object Classification

5.1 Motivation

Whether to recommend the most suitable CAD model to a designer or to enable a service robot to decide where to place objects when tidying a room, 3D object classification is an essential task. Research in this area has greatly benefited from the wide availability of 3D CAD models as well as the accessibility of depth sensors, as this has established a large amount of data to apply geometric reasoning.

Deep learning has profoundly transformed computer vision in recent years, and in particular, object classification has seen spectacular improvements. There has been steady interest for applying these methods for 3D data but introducing geometric reasoning in deep learning is not without its pitfalls. Typical deep learning approaches cannot handle rotated objects and real-world objects might be observed in arbitrary poses. Some methods use the statistical distribution of the data to transform the unknown object to a canonical pose for the deep network [7], [66]. However, inaccessible viewpoints, partial occlusions, supporting surfaces, and over- or under-segmentation observed in real-world data all contribute to modifying the statistical distribution of data that these methods expect, thus hindering their performance. This problem is particularly common for data obtained by indoor service robots. Most deep networks also expect a fixed size input. This is achieved by rescaling, however, applying this to an occluded object can lead to a significant difference in the final fixed size representation. Consider how rescaling and centering a model airplane to the unit sphere and the same model with one wing missing would produce vastly different coordinates. The effect becomes prevalent when transferring from CAD models to real-world objects, as scale information is not available during training since most CAD models are scaleless.

In this work, we develop a method for 3D object classification based on object parts that are reproducible under orientation or scale changes and can be defined for any level of occlusion, as shown in Figure 5.1. It should be noted that we define parts as a *continuous subset* of the original object without any specific semantic meaning. Indeed, semantic parts such as a cup handle or a chair leg are also likely to be occluded and impossible to recover from the original objects, whereas our non-semantic parts

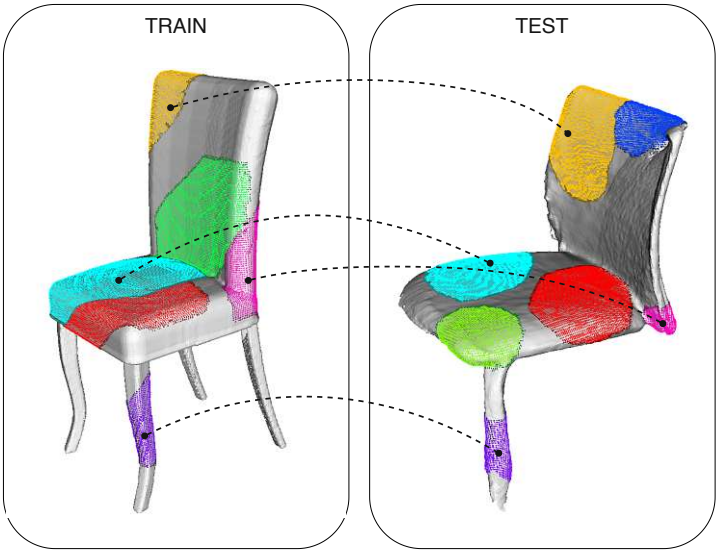

Figure 5.1: Creating reproducible object parts with similar representations on all sources of data enables better transfer from artificial to real objects.

can always be defined. Once parts are extracted, a rotation-invariant representation is computed through the use of a reproducible local reference frame. Finally, a graph-convolution based architecture is used to classify the graph of parts. This approach is chosen because of its ability to incorporate information about the neighborhood of the parts without needing to define a global orientation or to know about the object delineation.

In summary, the contributions of this chapter are the introduction of:

1. a carefully designed angle-based sampling procedure that creates object parts reproducible under various rotation, scale and occlusion and,

2. a general graph-based learning architecture for classification that preserves the relevant properties of 3D object parts.

These aspects allow us to achieve high performance when transferring from artificial to real-world data. In particular, our approach transfers from the ModelNet dataset [12] to objects segmented from the reconstructed scenes in the ScanNet dataset [13] better than previous point cloud-based methods. Our approach also outperforms the baseline 3D method PointNet [7] when training and testing on ScanNet, which further demonstrates the value of careful part design and the inclusion of geometric priors.

5.2 Learning from Object Parts for Robust 3D Object Classification

Our method is developed for over- or under-segmented objects represented by manifold triangle meshes (a mesh is a manifold if each edge is connecting at most two triangles). The reason for using a mesh is that surfaces are preserved. Reconstruction methods

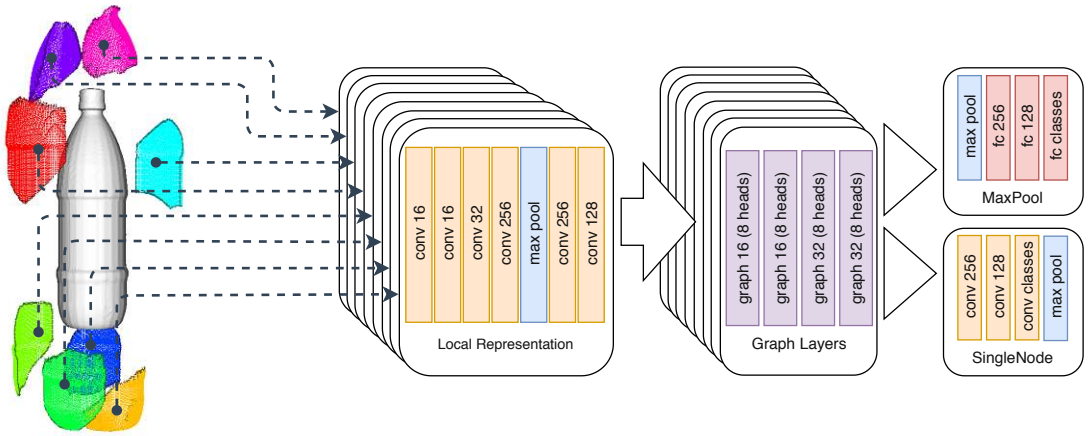

Figure 5.2: Architecture overview of our proposed method. The number of parts is reduced to eight for readability purposes and the connectivity is not displayed. Convolution layers all consider only a single element (kernel size one) and the number of filters is indicated.

generate that representation either directly [28] or by applying a post-processing step such as the marching cube algorithm [25]. However, the method presented here could easily be adapted for dense point clouds by using nearest neighbor approaches to retrieve the neighborhood of each point.

This section describes the proposed method. We first explain the object part sampling process and the part representation. We then outline the learning approach for the graph of object parts and the design of our graph convolution architecture.

5.2.1 Creating object parts

Object parts sampling To transfer from artificial object models to real reconstructed data, object parts should be repeatable under varying orientation, occlusions, scale and point density. Scale-invariance forbids the use of Euclidean distance for sampling parts. To avoid sampling a part that would span through an object, and thus being significantly more sensitive to occlusions, parts are grown by following the surface of the object. The average angle between a triangle and its neighbors on a surface is used when deciding whether a neighboring triangle should be added to the part. Since reconstruction algorithms account for sensor noise and artificial data does not suffer from any random noise, high quality normals can be computed for both type of data. The angle between two neighboring normals is independent of scale and orientation of the object, and in a perfectly noiseless case, even independent of the surface sampling density. Object parts are then extracted by performing a Breadth-First Search (BFS) on the graph defined by the object mesh, or in other words, incrementally adding a one-ring neighborhood around the sampled part center. Due to the strong unpredictability of occlusions, part centers are randomly sampled. Centers are sampled so long as they do not belong to a previously sampled part. The search is stopped when the accumulated angle over the object part reaches a set threshold. The accumulated angle is computed

from the average angle of each triangle, which is simply the average of the angle with each triangle neighbor.

Reconstructed scenes do not provide perfectly smooth surfaces, therefore, we perform low-pass filtering on the normals defined by the triangles. Normals for all points are first computed by averaging the normals of each triangle they belong to. Then triangle normals are computed by averaging the normals of the three points. The resulting normals are smoother than the original mesh.

Object part features The object part representation should maintain the properties of the sampling. In this work, we sample a fixed number of points from the object part to generate a fixed size representation from parts of varying sizes. Orientation-invariance is then achieved by defining a local reference frame (LRF). The center of the LRF is defined by the mean of a set of points and we propose two different orientations.

The first design option is to perform Principal Component Analysis (PCA) with the set of points and use the eigenvectors as the LRF. The first and last eigenvectors (when ordered by decreasing eigenvalues) are kept and the direction of the last eigenvector is flipped in order to follow the average direction of the surface normals of the set of points. This guarantees a different LRF for concave and convex sets. The last vector is the cross-product of the first two vectors. This LRF provides a total orientation invariance and is referred to as PCA-LRF.

The second option is to define the LRF based on the global vertical axis (Z-axis) and the component of the mean surface normal of the part that is orthogonal. This is no longer independent of the orientation of the object but only independent of the orientation of the object around the Z-axis. This local frame of reference is referred to as Z-LRF. Although it is only partially orientation-invariant, it offers a more informative representation. Since many objects have a small number of canonical poses (e.g. most bottles stand upright), it remains beneficial when tested on realistic data.

All point sets are rescaled to the unit sphere to make the representation independent of the scale. In most experiments in Section 5.3, we also add the average angle value as a feature to the point coordinates. When sampling the point, we use the average angle value of the triangle it belongs to. It slightly improves the accuracy without any extra computation overhead because it is computed during sampling.

Finally, the graph is constructed by connecting parts that overlap. In other words, parts are connected if at least one triangle in each of the parts was sampled in the original object.

5.2.2 Model Architecture

General architecture The architecture, as shown in Figure 5.2, follows the PointNet model where each point extracted from a part is independently fed to the same convolutional neural network. In difference to the architecture of PointNet, the spatial transformer network is unnecessary as the points' coordinates are already defined in a LRF. Each of the layers includes a batch normalization step [90]. A weighted version of the maximum value (over the whole set) of a given filter is subtracted from each output as described in [65].

The proposed model includes graph convolution layers inspired by the GCN model introduced in [6]. This is a simplification of larger graph convolutional models because only first-degree neighbors are considered. As a result, the model cannot differentiate neighbors from each other. To address this, we introduce an attention model in our graph convolutions.

Attention model The GCN model is made more powerful by introducing an attention mechanism. Instead of considering all neighbors as equal, the neighbors are weighted according to a criterion that is specific to the attention model chosen. Examples of attention models have been developed in [72] and [11]. To further improve the representational power of a network, multiple attention heads are used for the same layer, and the attention heads output are concatenated at each layer.

Introducing attention to the GCN model amounts to learning a valid coefficient to replace the normalization factor. We follow the model defined in [11] where the graph convolution layer becomes

$$h_{v_i}^{l+1} = \sigma\left(\sum_{j \in \mathcal{N}_i} \gamma_{ij} h_{v_j}^{l} W^{l}\right), \tag{5.1}$$

where h_{v_i} is the feature vector of the i-th vertex, σ is the activation function, and W is the parameter vector of the layer l. The coefficient γ_{ij} is defined as

$$\begin{aligned} \gamma_{ij} &= \text{softmax}(e_{ij}), \\ &= \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}, \end{aligned} \tag{5.2}$$

where $e_{ij} = \text{LeakyReLU}(a^T.[h_{v_i}W || h_{v_j}W]))$, $(\cdot)^T$ denotes the transpose operation, $\cdot||\cdot$ denotes the concatenation operation and a is the vector of learned parameters for the attention. In order to respect the part connectivity, we add a bias matrix to the e_{ij} term before applying the softmax in which disconnected nodes have a value of -10^9 and connected nodes have a value of 0. This model is further discussed in [11] and [89].

Summarizing over object parts In this work, we are interested in predicting the object-level class. The object part representation described so far affords a number of different options for this task. The most straightforward option is to simply perform a max-pooling operation on the feature vectors of each part and then classifying the object (referred to as MaxPool). However, the classifier will be trained expecting all nodes and is therefore less likely to transfer well to real reconstructed data that have missing nodes. A second option is to predict one class per node and average all predictions into an object-level prediction (referred to as SingleNode). Both options are evaluated in Section 5.3. The single node prediction trained on artificial data still provides a representation that assumes perfect connectivity. We therefore propose one last option in which a proportion of nodes are randomly disconnected (except for self-connections).

5.3 Experiments

This section presents the experimental results. The first set of experiments compares our proposed approach to state-of-the-art methods for object classification on artificial

data using the ModelNet dataset [12]. The second set of experiments evaluates the transfer abilities from artificial data (ModelNet) to real-world data (objects extracted from the ScanNet dataset [13]). We also evaluate in more depth the impact of the object part size and the connectivity of the object parts graph. Lastly, our method is evaluated against the PointNet architecture when training and testing on real-world data with objects extracted from the ScanNet dataset.

5.3.1 Experimental setup

Implementation Our final model is described in Figure 5.2. We sample up to 32 parts per object and 250 points per part. The representation of each object part is fed through four 1D convolutional layers and max-pooling is applied over the whole set of points from the object part. The local representation is then passed to four graph convolutional layers. The output of each attention head is concatenated at each layer before being fed to the next. When predicting over single nodes or over a set of nodes, pooling is applied either before or after the classification layers.

Datasets Evaluation is performed on two datasets: The ModelNet dataset [12] and the ScanNet dataset [13] (1513 reconstructed rooms). We use both ModelNet40 (12311 CAD models split between 40 classes) and the ModelNet10 subset (4899 models in 10 of the original 40 classes: `bathtub`, `bed`, `chair`, `desk`, `dresser`, `monitor`, `night_stand`, `sofa`, `table`, `toilet`). The second version of the annotation for ScanNet is used with the train/test/val split defined in the first version. Objects are extracted according to the annotation in the dataset. Afterwards the object classes are mapped to the ModelNet classes. The ModelNet10 mapping is the subset of ScanNet objects that belong to the ModelNet10 classes. The ModelNet40 mapping is the same but uses the ModelNet40 classes.

5.3.2 Evaluation on artificial data

Table 5.1 compares the performance of our method to state-of-the-art methods on the ModelNet [12] dataset. It should be noted that the models of the dataset are non-manifold meshes. To apply our method, we project views of the objects and reconstruct them using a TSDF-based method. As a result, the object models differ slightly. For reference, we provide the accuracy of available methods on both versions and observe a drop in accuracy between one and three percent for the object models created for our approach.

This experiment shows that despite being specifically designed with real-world constraints in mind, our method still shows competitive results on artificial data. The results presented in Table 5.1 correspond to our method with a max pooling and trained with the average angle values as an included feature. Table 5.2 shows the results of the addition of the average angle value as an extra feature. We see that performance slightly improves in all conditions without adding any computation as it is already calculated during the sampling process. Furthermore, the SingleNode type of pooling (i.e. predicting a class for each object part and averaging the prediction over the object) gives similar results to MaxPool on artificial data. An experiment using the GCN model

Table 5.1: Classification accuracy on the ModelNet40 dataset [12] (**MN40Rec** indicates that the method was evaluated on the reconstructed models)

Method	MN40	MN40Rec	Input
VoxNet [9]	83.0	-	Voxel Grid
KD-Networks [61]	91.8	-	KD-Tree
MVCNN [10]	90.1	-	Views
MVCNN-New [52]	**95.0**	91.7	Views
3DmFV-Net [63]	91.6	91.1	Point Cloud
3DCapsules [91]	92.7	-	Point Cloud
PointNet [7]	89.2	88.1	Point Cloud
*Ours (PCA-LRF)	-	86.9	Mesh
*Ours (Z-LRF)	-	89.4	Mesh

Table 5.2: Evaluation of design choices on ModelNet10

Pooling	LRF	Avg ang.	Attention	Acc.
MaxPool	PCA-LRF	✗	✓	85.6
MaxPool	PCA-LRF	✓	✓	86.3
MaxPool	Z-LRF	✗	✓	87.2
MaxPool	Z-LRF	✓	✗	86.9
MaxPool	Z-LRF	✓	✓	89.2
SingleNode	Z-LRF	✓	✓	89.6

[6] without the attention model is also performed. This result shows that the attention model improves the results, which is consistent with [11]. All future results include the average angle as a feature, use the Z-LRF and include the attention model.

Figure 5.3 illustrates the benefits of our design when simulating levels of occlusion on artificial object. The occlusion is simulated by removing a region that is grown on the mesh surface until a set percentage of the object is reached. We observe that the performance degrades significantly better than for the PointNet architecture.

5.3.3 Evaluation of the gap between real and artificial data

This section evaluates the gap when training on artificial data (objects from ModelNet) and testing on real-world data (segmented objects from ScanNet). This is a difficult task because of the domain shift as well as some important characteristics of the ScanNet dataset (see Figure 5.4). ScanNet was first and foremost designed for semantic segmentation, which is more concerned with large-scale structures. Additionally, since it represents natural environments, the dataset is strongly imbalanced. Chair is a highly dominating class, which is seen by the performance of the “chair predictor” baseline (i.e.

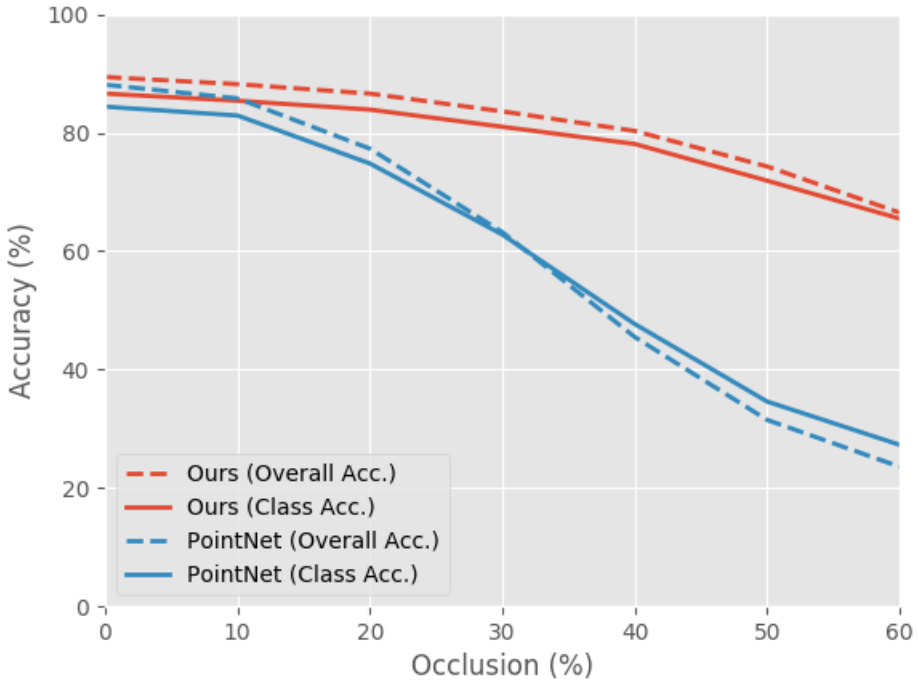

Figure 5.3: Accuracy in percent for various simulated occlusion (ModelNet40 is used, SingleNode model trained with 75% Disconnect).

always predicting the chair class) in Table 5.3. As such, class accuracy is a more relevant metric than overall accuracy. Moreover, most structures tend to be oversmoothed by the reconstruction algorithm. This is a side-effect of the reconstruction algorithm that tries to reconcile various noisy measurements. Subtle differences also exist between ScanNet classes and ModelNet classes (e.g. a pack of bottles in ScanNet is mapped to the bottle class, whereas that class only contains single standing bottle in ModelNet). Finally, the segmentation is often inaccurate for smaller objects. This is due to the fact that scenes are oversegmented and then clusters are annotated. Therefore, many extracted objects include points from the surrounding elements. All these factors make this a very challenging dataset for the task.

As shown in Table 5.3, only MVCNN-New and our model manage to perform some transfer. The result when always predicting the chair class not only demonstrates the imbalance of the dataset but also provides a general reference for evaluating performance. Considering that our object part representation could easily be swapped out in our pipeline, a comparison to PointNet is fairest to our contribution. This local representation was selected for its versatility and proven reliability in various contexts [92], [93]. The good results of the MVCNN-New architecture supports the idea that part-based (in this case parts are defined by projective geometry), and invariance to rotation around the gravity axis is essential to transfer to real reconstruction. MVCNN-New seems to work best if one of the sampled view preserves the outer contour of the reconstructed object. In particular, MVCNN-New is better than our method on

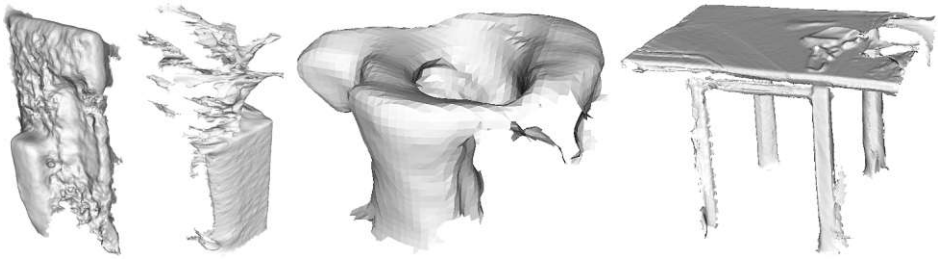

Figure 5.4: Illustration of the noisy objects extracted from the ScanNet dataset. Left to right: monitor (very noisy surface), potted plant (strong occlusions and many disconnected parts), cup (small object) and table (object on top merged with it).

Table 5.3: Evaluation when transferring from ModelNet40 to ScanNet

	Acc.	**Cls Acc.**
Chair Predictor	36.2	2.5
PointNet	2.2	3.3
3DmFV-Net	0.9	4.2
MVCNN-New	**36.6**	**20.0**
Ours	34.6	19.1

the classes `plant`, `sofa` and `toilet`, but worse for `bathtub` `monitor` and `sink`. It is interesting to note that both sink and bathtub are both box-like objects with concave parts that do not affect the outer contour of the object in a projected view. Our model is adversely affected by meshes having many disconnected components such as `plant` and `bookshelf` meshes in the ScanNet dataset.

For our method, the best results are achieved by increasing the sampling threshold by a factor of two compared to training. Also, SingleNode pooling is used with a disconnection rate of 75% during training. In the next section, we evaluate those design choices in more detail.

Influence of the part size in the transfer One significant parameter in the transfer performance is the threshold set for the part sampling algorithm. ScanNet scenes tend to be oversmoothed, which affects the angle-based sampling procedure for objects that have large flat surfaces in artificial models. Also, scenes are reconstructed at a constant density, which means that smaller objects have a smaller density than bigger objects. Combined with the low level of noise, the right trade-off needs to be found for increasing the threshold used in training and testing because classes react differently. Figure 5.5 shows the impact of the sampling threshold on the accuracy when using the mapping from ModelNet40 classes to the ScanNet objects. The best threshold typically increases with the increase of the average size of the class. For small objects, such as `bowl`, the best threshold is close to the training value. For medium-sized objects, such as `toilet`, the best threshold is between three and fours times the original value. For very large

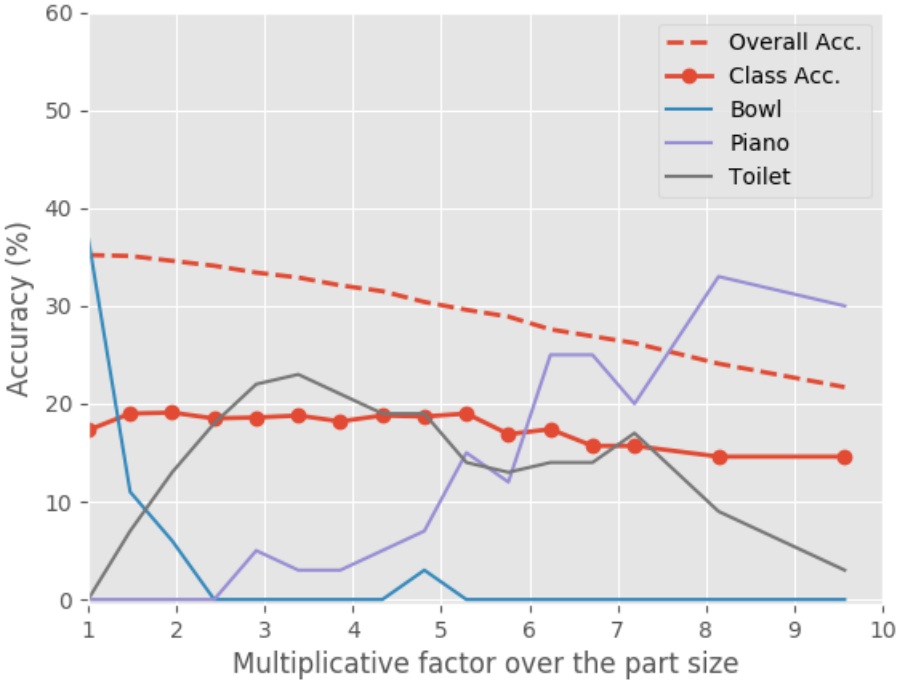

Figure 5.5: Accuracy in percent for various multiplicative factors applied to the sampling threshold (ModelNet40 mapping is used, SingleNode model trained with 75% Disconnect).

objects, such as `piano`, the best best threshold is up to eight times. Larger objects simply have more triangles in the mesh. The threshold is reached much faster due to the noise, therefore, they require a larger threshold in order to reproduce parts similar to those observed during training.

Influence of the connectivity and pooling Table 5.4 shows the impact of different pooling strategies on the transfer performance. It also shows the impact of randomly disconnecting some object parts in the object part graph during training in order to better approximate occluded objects. The experiments are performed using the mapping from ModelNet10 to ScanNet. The significant increase in accuracy is due to the fact that the ModelNet10 mapping contains most of the well-reconstructed objects because classes of ModelNet10 corresponds to objects of larger scale (excluding classes such as `cup`, `keyboard`, `laptop`, `vase` and `xbox`). Clearly, performance improves with disconnection considered. Applying 0% has low accuracy because it does not model the occlusion. At 100% the performance is also low because this over-estimates the level of occlusion. The best compromise is achieved with 75% for the ScanNet dataset. The disconnect experiments are not applied to the MaxPool setup, as it would still take into account each part when max-pooling, thus failing to simulate occlusions.

Table 5.4: Evaluation of design choices for transferring to ScanNet (using the ModelNet10 mapping)

Pooling	Disconnect (%)	Acc.	Cls Acc.
MaxPool	-	44.1	42.35
SingleNode	0	50.1	38.3
SingleNode	50	62.2	39.0
SingleNode	75	**62.9**	**43.2**
SingleNode	100	54.5	33.2
Chair predictor	-	56.0	10.0

Table 5.5: Evaluation when training and testing on ScanNet (using the ModelNet40 mapping)

	Acc.	Cls Acc.
Chair Predictor	36.2	2.5
PointNet	73.6	52.3
PointNet (rescaled obj.)	53.4	17.2
Ours (Z-LRF)	**85.2**	**62.8**

5.3.4 Evaluation on real data

The large size of ScanNet makes it feasible to train methods on real-world data instead of artificial data. This is helpful because it can establish an upper bound for the transfer to this dataset. The results in Table 5.5 are significantly higher than when transferring, which implies the existence of occlusion consistency for a given class. We, however, conjecture that this only holds for larger structures but not for smaller objects, such as household items. The results additionally show that an advantageous side-effect of our design is the ability to better learn from noisy data. Our method achieves an accuracy of 85.2% and class accuracy of 62.8%. The best results are achieved using the Z-LRF frame of reference. Prediction is performed for each node and averaged, and training is performed with a 75% random chance of disconnecting two neighbors. In comparison, the PointNet architecture achieves much lower scores of 73.6% accuracy and 52.3% class accuracy, which is approximately 10% less than our method. Experiments are also performed with PointNet trained on objects rescaled to the unit sphere to prevent the method from taking advantage of scale information. The significant decrease in this case suggests that PointNet finds it more difficult to establish consistent shapes. This supports our argument that even though objects have inconsistent shapes, they always have consistent parts.

5.4 Discussion

This chapter addressed the important robotics task of object classification from 3D data. We present an approach that transfers to real reconstructed objects when trained on clean CAD models only. Results show that our approach significantly outperforms state-of-the-art methods while maintaining competitive performance when training and testing on CAD models. The performance increase in the transfer is achieved through sampling object parts that are reproducible under rotation, occlusion and scale in combination with a graph-based deep learning architecture. Learning with a rotation-invariant and scale-invariant representation of parts enables objects to be recognized with significant portions of missing data.

Hightlights

(1) Part-based representations are useful to deal with the sim2Real gap in 3D object classification, and with occlusion in particular.
(2) Parts can be more reliably scaled and oriented which enables, in combination with graph-based learning architectures, a rotation-invariant and scale-invariant representation.
(3) The benefits of such an approach does not require semantic subdivision of the object, and accomodates itself very well to a simple breadown.

Chapter 6

Robust Sim2Real 3D Object Classification using Graph Representations and a Deep Center Voting Scheme

6.1 Introduction

Service robotics could provide significant relief to overwhelmed healthcare workers or help elderly people remain independent for longer by carrying out menial tasks. To attain this objective, they need to adapt to an ever changing environment and manipulate a constantly evolving set of objects. The majority of state-of-the-art methods use deep learning to achieve this property [8], [14], [15], [68]. However, such approaches require a large flow of annotated data to learn about new objects, which is only practical through simulation and modeling, as manual reconstruction, segmentation and labeling at scale is prohibitively time consuming.

Learning from artificially produced data is an increasingly popular direction of research. The challenge is for the trained model to still operate robustly on data taken by a robot in a realistic setting. The loss of performance, usually referred to as the Sim2Real gap, is still remarkably large in 3D object classification compared to 2D color images [41], even though producing photo-realistic images is more challenging than producing realistic 3D models. Models trained from artificial object models still struggle to classify objects segmented from 3D reconstructions because of occlusions, over-smoothing due to noise in the camera pose, light modeling, CAD models deviation from real objects, and scale differences as seen in Chapter 4. Notably, scale from CAD models is arbitrary, and not necessarily metric, making it difficult to rely on when using a large-scale dataset. It is therefore common to rescale objects to a normalized scale like the unit sphere. For real objects, however, under-segmentation or partial occlusion will affect the rescaling factor. It therefore becomes more challenging to obtain consistent coordinates for object points. Indeed, to obtain object representations at a consistent scale, a very good segmentation of the objects is needed, but to obtain a precise segmentation requires a semantic understanding of the object, which is the objective in the first place. This problem is compounded when designing a method for object segmentation or detection trained on artificial data, but is more easily studied on classification tasks. Relying explicitly or implicitly on coordinates renders these

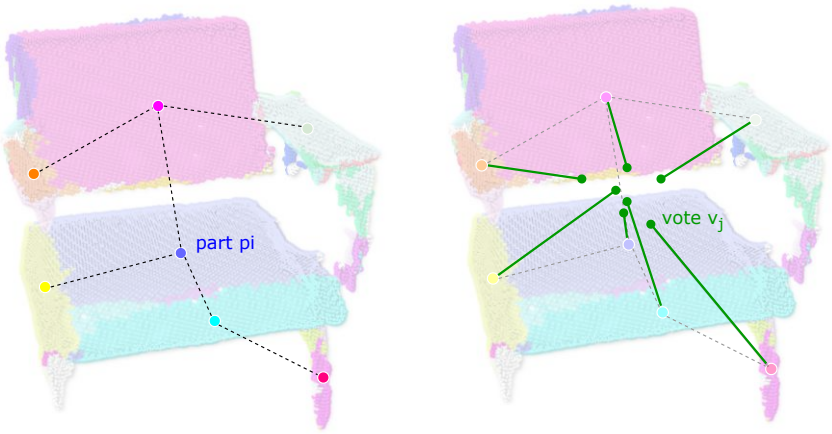

Figure 6.1: Illustration of the voting procedure. Every part of the graph learns from its own shape and neighborhood (left). They each cast a vote for the object center that then get clustered (right). Votes from background points are more scattered than votes from the object, which makes the separation more marked.

representations more sensitive to scale, which in turn leads to misclassification on real data.

In this work, we propose to learn representations significantly more robust to scale from a graph of reproducible parts and their connectivity. Specifically, only the relative direction between neighboring parts is considered, instead of their actual coordinates. In addition, we introduce a voting mechanism to further robustify our method to under-segmentation as illustrated in Figure 6.1. Each part votes for its corresponding object center such that background and foreground parts are separated through their center prediction.

In summary, the contributions of this chapter are:

- 3D object classification evaluation showcasing the sensitivity of state-of-the-art methods to scale.
- A segmentation method for creating a robust graph of parts for 3D object classification.
- DirEdgeConv - a scale-invariant adaptation of EdgeConv [15].
- The introduction of a voting procedure for separating foreground from background.
- Improvement of up to 16% on the noisiest variation of real object dataset ScanObjectNN[41] when training on artificial models dataset ModelNet.

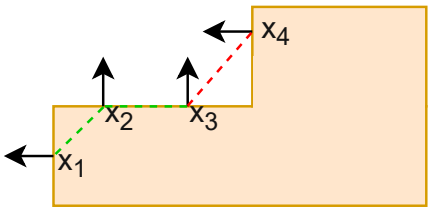

Figure 6.2: Illustration of the convexity criterion. The connection between x_1, x_2 and x_3 are convex or flat so they are merged in the same part.

6.2 Sim2Real Classification of Under-segmented Objects

The key idea of the proposed method is to increase the robustness to scale change. To achieve this, we represent the object as a graph of object parts. Its creation and the learning architecture used to preserve these properties is described in the following sections.

6.2.1 Building a graph of parts

Our method creates a graph of parts from object reconstructions. As we intend to learn an object representation based on object parts, they should be reproducible between artificial models and real data of reconstructed objects to facilitate the transfer from one domain to the other. Since scale is inconsistent between the two domains, parts are defined in a scale-invariant way, depending on the surface changes themselves. They need to cover a limited portion of an object such that most parts are unaffected by partial occlusion. Finally, parts need to respect object boundaries and not span over multiple objects such that representations remain consistent in the presence of background points.

Locally Convex Connected Patches (LCCP) [81] is shown to create an over-segmentation that respects object boundaries well, based on experiments in [78]. Starting from a set of supervoxels (sets of voxels guaranteed to not cross object boundaries), LCCP computes a convexity criterion between every pair of neighboring supervoxels. Based on this connectivity, a region growing procedure is performed to obtain the final clusters. Formally, for a pair of connected supervoxels with centroids $\mathbf{x_1}$ and $\mathbf{x_2}$ and their respective normals $\mathbf{n_1}$ and $\mathbf{n_2}$, the convexity criterion (CC) in [81] is used to decide whether two parts should be merged, with an extra tolerance threshold α_{thresh} (set to the cosine of 10Â° in all our experiments):

$$CC = \begin{cases} \text{true if} & (\mathbf{n_1} - \mathbf{n_2}).\frac{\mathbf{x_1}-\mathbf{x_2}}{\|\mathbf{x_1}-\mathbf{x_2}\|} > 0 \vee \mathbf{n_1}.\mathbf{n_2} < \alpha_{thresh} \\ \text{false} & \text{otherwise} \end{cases} \tag{6.1}$$

Two supervoxels are neighbors if they have a convex edge or if their angular difference is smaller than the arccosine of α_{thresh}. This is further illustrated in Figure 6.2. A small angular difference occurs in mostly flat areas, and the threshold is set to accommodate

noise in the surface orientation. In combination with the sanity criterion described in [81], object parts are grown based on this newly obtained connectivity.

Following a similar pipeline, we start by extracting supervoxels as in [80]. Since we work with 3D data, and not a single frame depth map, both sides of an object can be observed. For thin object parts, such as a chair backrest seen from the front and the back, points from both sides could fall into the same voxel. Because the connectivity is derived from a voxel grid, a connection would be created between parts through the object volume instead of the object surface. This is problematic for the convexity criterion, as the front and the back will have opposite normals leading to very different segmentations depending on which side is used. Moreover, creating parts according to the object surface is a desirable property for occlusion robustness because two surfaces with opposite orientation are unlikely to both be observable when viewpoints are limited.

To circumvent this issue, we allow our approach to create up to two supervoxels per voxel if the angular difference is more than 90Â°. Supervoxels are refined exclusively based on the angular difference.

When applied to object reconstructions, parts created by LCCP will cover large portions of the object. Indeed, supervoxels will belong to the same part as long as there is a path linking them in the convexity graph, or in other words, every connection between two parts need to be concave in order for them to be split. These very large parts are detrimental to the robustness of the approach since large object parts are unlikely to be fully visible once deployed because of occlusions. The growth of a part is therefore constrained and a supervoxel s_i (with normal $\mathbf{n_{s_i}}$) can be added to a part $P = \{s_j | j \in 1...k\}$ only if

$$\mathbf{n_{s_i}}.\mathbf{n_{s_j}} < \beta_{thresh} \; \forall \; s_j \in P \tag{6.2}$$

We set β_{thresh} to the cosine of 120Â° and use this criterion to approximate, in a computationally efficient way, the fact that there should be a projection where every supervoxel of a part is visible, since reconstructions are obtained from a collection of views. This makes the region growing procedure dependent on the seed initialization. As such, seeds are randomly sampled among the supervoxels that do not belong to a part.

The connectivity between object parts is derived from the connectivity of the supervoxels, where any voxel sharing a face, edge or a vertex in the grid is considered to be a neighbor.

This segmentation is sensitive to scale only through the original voxel resolution chosen. We argue that the part growing compensates for this aspect, and as long as the original voxel grid is fine enough, the combination of supervoxels based on their normals will result in a similar part.

6.2.2 Learning object representations from object graphs

We now introduce our method to learn from a graph of parts while preserving the properties of the parts. Firstly, since parts cover many points, we describe how to learn features from the parts' shapes independently of their scales. Secondly, we describe how

to learn object-level representations from a graph of parts while preserving the scale change robustness.

Local parts representation Parts created using the method described in the previous section represent different surfaces and cover many points. A part-specific representation is introduced to learn the shape information of each part. A fixed number of points within each part is sampled. Each set of points is re-centered and scaled to fit the unit sphere. The parts representation is learned using an architecture (Figure 6.3) similar to PointNet [7], except that at every one-dimensional convolution layer the maximum of the input feature is subtracted, weighted by a learned parameter as described in [65] for added representational power. Batch normalization [90] is used at every layer.

Representing parts in context using Graph representations Relying purely on the graph structure and its connectivity as in chapter 5 loses the information about the relative position of graph nodes. However, positions are sensitive to scale changes. Specifically, the graph produced by the method described in Section 6.2 creates parts of variable size and the relative positions can therefore have different magnitudes. We therefore only consider the relative direction of each neighboring part by adapting the EdgeConv layer. It should be noted that analogous adaptation can be made to all neighborhood-based representations.

In the original EdgeConv layer introduced in [15], $f_i^{l+1} \in \mathbb{R}^F$, the feature vector of length F of part i at layer $l+1$, is obtained based on its neighborhood $\mathcal{N}_i$ according to

$$f_i^{l+1} = \max_{j \in \mathcal{N}_i} \left(\text{concat} \left(f_i^l,\, f_i^l - f_j^l \right) \right) \tag{6.3}$$

where *concat* represents the concatenation of the two vectors. Features are learned directly from point coordinates and the neighborhood is defined by a nearest neighbor search on the set of feature vectors.

We introduce a new layer based on EdgeConv, called DirEdgeConv for clarity, that can learn from object parts without introducing any sensitivity to scale change. The local parts representation only encodes shape and not position (which is the case for point coordinates). To re-introduce this information without depending on the coordinates scale, we use the relative direction of neighbors defined as $\frac{x_j - x_i}{\|x_j - x_i\|}$ where x_i and x_j are the centers of the two neighboring parts. This is in contrast to using their relative position $x_j - x_i$ as is done implicitly in the original EdgeConv layer. Putting everything together, with $f_i^{l+1} \in \mathbb{R}^F$ the feature vector of length F of part i at layer $l+1$, x_i its center and $\mathcal{N}_i$ its neighborhood, we propose the following DirEdgeConv layer

$$f_i^{l+1} = \max_{j \in \mathcal{N}_i} \left(\text{concat} \left(f_i^l,\, f_i^l - f_j^l,\, \frac{x_j - x_i}{\|x_j - x_i\|} \right) \right) \tag{6.4}$$

In Section 6.3, DirEdgConv is evaluated both when using the fixed neighborhood defined in the previous section and when recomputing the neighborhood dynamically based on the nearest neighbor search between parts features as in [15]. In the case of the dynamic connectivity, the first DirEdgeConv layer still uses the fixed neighborhood defined previously because parts features do not encode their position.

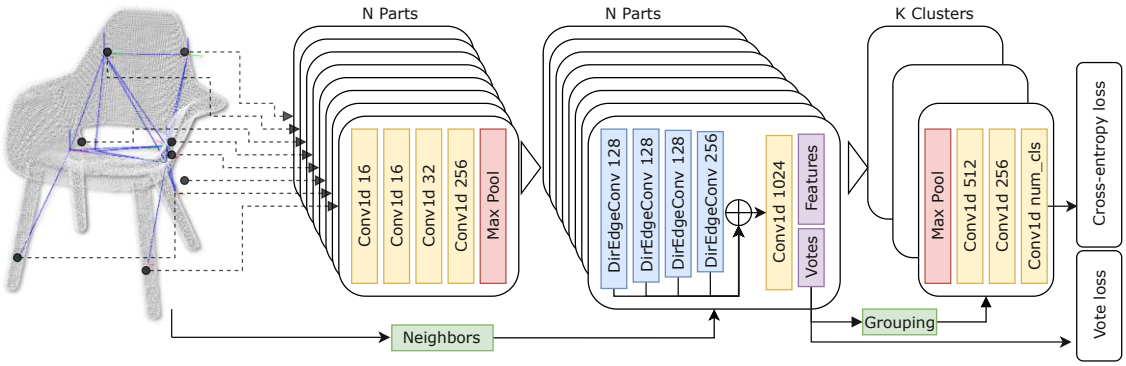

Figure 6.3: Overview of our approach. A local representation is learned for the N parts extracted from the object. Taking advantage of their connectivity and relative position, four DirEdgeConv layers are used and concatenated to learn the final part representation and vote for the object center. Parts' features are then grouped based on the position of votes to create the final prediction.

6.2.3 Separating objects from the background using a voting scheme

In the context of Sim2Real classification, objects are also often under-segmented, meaning that not every point belongs to the object of interest. As good classification results require good segmentation and good segmentation masks require good classification, we propose to frame this problem of under-segmented object classification as a detection problem. Based on the findings of [87], we conjecture that background parts and object parts can be separated using object center prediction as object parts will have more consistent votes.

Each part predicts an offset Δp_i from the part center to the object center. Using farthest point sampling, a fixed number of cluster centers is sampled among the votes cast. Every vote within a sphere of fixed radius is considered a part of the cluster. Using a max-pooling layer, we aggregate the features of all parts whose vote was cast within a cluster. This cluster representation is then classified. The full architecture is illustrated in Figure 6.3.

During training, only CAD models are used, which can easily be centered and thus avoids the need for any extra annotation. The voting procedure is supervised by an L2-loss, which, for a part of center p_i and offset predicted Δp_i, is given by

$$L_{vote} = \|\Delta p_i + p_i\|^2 \tag{6.5}$$

Keeping the representation unaware of scale makes the voting procedure more uncertain, but this remains an effective foreground-background strategy if the object classes considered have sufficiently similar scales.

The complete loss of the network is the weighted sum $L = L_{cross-entropy} + \lambda L_{vote}$ with $\lambda = 0$ when using the max-pooling structure and $\lambda = 2$ when using the vote pooling structure.

6.3 Experiments

This section presents our experimental results. We first consider the performance of our approach on the standard ModelNet benchmark, a set of artificial CAD models, and look at the impact of scale on a selection of state-of-the-art methods. We then present our results when learning from artificial data and testing on real data. Finally, the impacts of our design choices are evaluated in an ablation study.

6.3.1 Experimental setup

Experiments are performed using the ModelNet dataset [12], a set of 12311 CAD models from 40 classes, and the ScanObjectNN dataset [41], a set of 2902 unique real object instances. The ScanObjectNN dataset is composed of objects extracted from real reconstructions of 3D scenes. The original dataset referred to as OBJ, and the T25 and T50_RS variations are used to showcase different amounts of error in the object points extraction. To obtain meshes more similar to the ones obtained from real reconstructions from artificial CAD models, a set of views is sampled around and above the object. Those views are combined using a TSDF (Truncated Signed Distance Function) volume. This reconstruction step provides dense mesh representations that are free from artifacts of CAD models (e.g., variable density, triangles disconnected for sharper rendering, etc.) and closer to the reconstruction objects as the bottom of objects are ignored. Since the resulting meshes have constant density, point clouds can be trivially obtained by only keeping the vertices of the mesh. The models are all scaled to the unit sphere because the ModelNet dataset does not provide accurate scales for its CAD models.

The graph of object parts is created with up to 200 parts (the exact number of parts is driven by the geometry of the object), and 128 points are sampled within each part. Votes are grouped into five clusters. Since the task at hand is classification, only one prediction per object is necessary, so only the predictions from the most confident cluster, as defined by the softmax, are used. As in the original implementation [15], we use a multi-scale architecture concatenating the output of every DirEdgeConv and classifying using this feature vector. The number of layers and number of features used at each layer is given in Figure 6.3. The network is trained using a cross-entropy loss. The vote loss introduced in Section 6.2 is added with a factor of 2 in the version of the network using the voting scheme (further referred to as “with voting”).

The part segmentation computation time depends on the number of points and geometry of object. Our measurements vary between 25ms for 23000 points and 523ms for 1 million points. The network is trained in 15 hours on a NVidia 1070Ti and a single inference step for one example takes 10.4ms.

6.3.2 Evaluation on artificial data

In this section, the performance of our method is compared to the four best performing methods on the ScanObjectNN dataset [41]. We report their performance on artificial data for completeness. They are all state-of-the-art methods developed specifically for

Table 6.1: Evaluation on ModelNet40 [12] and ScanObjectNN (original scale, and re-scaled). Results when training with the raw models in OBJ*, and with the reconstruction step in OBJ.

Method	MN40	OBJ*	OBJ	OBJ scaled
PointNet [7]	89.2	42.3	50.7	56.2 (10.8%)
PointNet++ [8]	90.7	43.6	50.3	59.8 (18.9%)
DGCNN [15]	92.2	49.3	55.6	63.2 (13.7%)
SpiderCNN [14]	90.0	44.2	48.8	57.5 (17.8%)
KPConv [69]	92.9	41.4	44.8	54.4 (21.4%)
Ours	85.8	-	58.6	-
Ours (with voting)	86.9	-	59.2	-

point clouds. The results on the 40 classes of the ModelNet dataset [12] are presented in Table 6.1.

We also report the results when training on the reconstructed ModelNet dataset, and evaluating on the ScanObjectNN dataset, with and without rescaling the objects in Table 6.1. This evaluation is done without any of the background points. In a realistic scenario, this rescaling would not be possible, as a proper re-scaling of the object requires an accurate segmentation. This would require a good understanding of the object at hand, which depends on the scale of the representation. The evaluation on re-scaled objects is intended to highlight the sensitivity of methods to scale change and the performance "lost" due to this sensitivity in a realistic scenario. In particular, the relative difference in percentage reported underlines the particular sensitivity of KPConv, PointNet++ and SpiderCNN.

6.3.3 Evaluation of the gap between artificial and real data

In this section, models are trained on a subset of ModelNet based on the classes that overlap with the classes defined in ScanObjectNN. The evaluation is then performed on the original version of ScanObjectNN, with and without background points. Background points are points within the bounding box around the original object. The evaluation is also performed on perturbed versions of ScanObjectNN. Noise is added to the object bounding boxes before extracting objects from the scene reconstructions, and only points within the noisy bounding boxes are kept. Specifically in ScanObjectNN_T25, bounding boxes from the original ScanObjectNN are translated by up to 25% of the object size. In ScanObjectNN_T50_RS, bounding boxes are translated by up to 50% and objects are rescaled and rotated randomly. Those perturbations provide variable amounts of noise in the object pre-segmentation, and provide a test bed to estimate the robustness of methods to over-segmentation (translating the bounding the box before extraction leads to less object points remaining) and under-segmentation (background points remaining). All results reported are obtained using the test set defined by ScanObjectNN. More details on the perturbation and split definition can be found

Table 6.2: Evaluation when training on ModelNet and testing on ScanObjectNN without any background points

	OBJ	**T25**	**T50_RS**
PointNet [7]	50.7	40.5	32.7
PointNet++ [8]	50.3	42.8	40.0
DGCNN [15]	55.6	44.8	36.2
SpiderCNN [14]	48.8	38.7	28.5
KPConv [69]	44.8	40.1	37.8
GAT-PointNet similar to Chapter 5	38.4	36.7	33.8
GAT-PointNet, our seg.	50.0	43.9	38.8
Ours	58.6	50.6	**45.2**
Ours (with voting)	**59.2**	**51.1**	44.4

in [41]. Example objects are shown in Figure 6.4. All the results presented in Table 6.2 and Table 6.3 use a dynamic graph for the method without voting and a fixed graph when using the voting scheme.

Table 6.2 and 6.3 clearly show the benefit of our approach on the task of transferring from artificial data to real data as it outperforms every other method on every variant of the ScanObjectNN dataset. We report the performance of our method both with and without the voting scheme. Overall, the more perturbed the input, the larger the improvement compared to other methods. In Table 6.2, the relative increase in accuracy is, respectively, 5.4% (6.5% with voting), 12.9% (14.1% with voting) and 13.0% (11.0% with voting) over the state-of-the-art for the original, the T25 and T50_RS variants of ScanObjectNN.

Table 6.3 illustrates the benefit of our voting scheme as we test on ScanObjectNN while keeping the background points. The difference in accuracy with respect to the state-of-the-art is -3.0% (2.0% with voting), 3.4% (6.3 % with voting) and 14.2% (16.4% with voting), for the original, T25 and T50_RS variants of ScanObjectNN.

We also report the results when using [11] in combination with the same parts features as our method similarly to the method presented in Chapter 5 (referred to as GAT-PointNet) to learn from the graph of parts. This method only learns from the graph structure underlining the importance of the relative position of nodes in 3D object classification.

6.3.4 Ablation study

In this section, different design choices are evaluated independently and results are presented in Table 6.4. Specifically, DirEdgeConv is compared to the original EdgeConv [15], with and without re-scaling of the data. This demonstrates that DirEdgeConv reaches the same maximal performance as EdgeConv, but without the sensitivity to scale changes, since it only relies on the direction of neighbors. The results when using DirEdgeConv without normalizing the relative position in equation (6.4) indicated by †

Table 6.3: Evaluation when training on ModelNet and testing on ScanObjectNN including background points

	OBJ	**T25**	**T50_RS**
PointNet [7]	52.0	40.4	34.7
PointNet++ [8]	43.2	34.9	33.7
DGCNN [15]	53.9	44.7	36.6
SpiderCNN [14]	44.6	36.7	30.5
KPConv [69]	40.5	35.4	34.68
GAT-PointNet similar to Chapter 5	31.8	28.5	26.0
GAT-PointNet, our seg.	41.2	34.3	30.7
Ours	52.3	46.2	41.8
Ours (with voting)	**55.0**	**47.5**	**42.6**

Figure 6.4: Example segmentations. Top row: Objects from ModelNet40, bottom row: objects from ScanObjectNN. Colors are assigned randomly.

also demonstrate that the perfomance increase comes from only using the direction, and not the introduction of local coordinates information at every layer by the DirEdgeConv layer. The voting scheme presented in Section 6.2 is referred to as Vote pooling and is evaluated against max-pooling over the features of every part (Max in Table 6.4). These results show the benefit of the voting scheme when background points are included.

In Table 6.4, we also report the results using the dynamic and the fixed graph configuration. The dynamic graph outperforms its fixed graph counterpart for Max pooling but is worse for Vote pooling. Fixed connectivity is indeed important to meaningfully predict the object center. These results highlight the importance of connectivity in graph representations, and is a good indicator that fully learned segmentation and connectivity is a promising research direction.

Finally, Figure 6.5 showcases the robustness of the method presented to different voxel sizes. This result supports the claim that as long as the voxel size is small enough, the final parts will be similar.

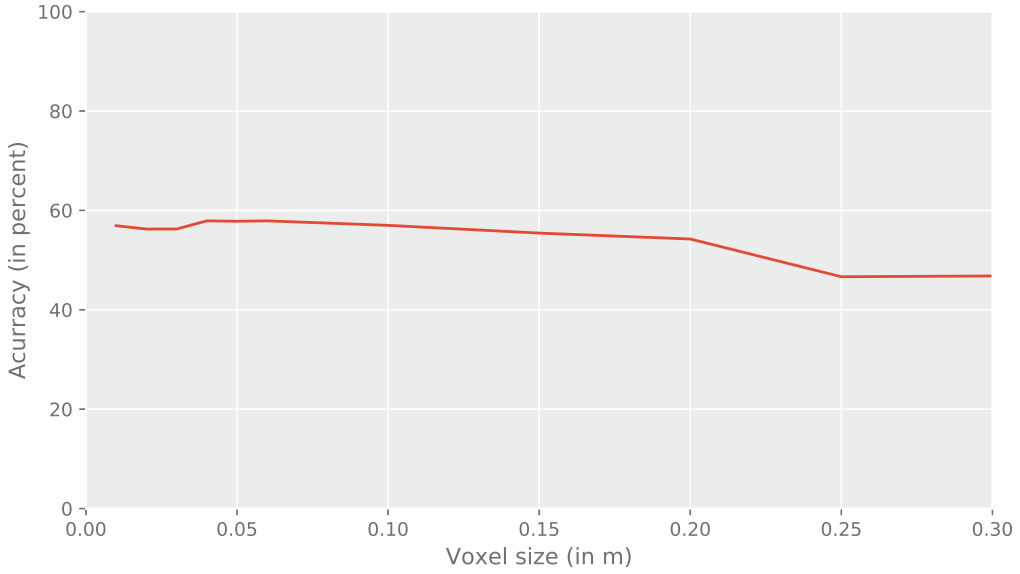

Figure 6.5: Influence of the voxel size on the accuracy. In the range [0.01, 0.1] the absolute difference in accuracy of our method is less than 1%.

Table 6.4: Ablation study. Results are given on the unperturbed version of ScanObjectNN. * indicates that the results are obtained when scaling the objects. † indicates that the relative position are not normalized in Equation 6.4

Layer	Graph	Pooling	Without bg.		With bg.	
EdgeConv	Dyn.	Max	47.4	47.9	41.7	40.7
EdgeConv*	Dyn.	Max	58.4	57.3	52.0	50.4
DirEdgeConv †	Fixed	Max	49.9	48.1	42.4	40.9
DirEdgeConv	Fixed	Max	57.0	56.2	49.4	50.0
DirEdgeConv	Dyn.	Max	58.6	**58.4**	52.3	53.2
DirEdgeConv	Fixed	Vote	**59.2**	57.7	**55.0**	**54.1**
DirEdgeConv	Dyn.	Vote	58.2	56.6	54.4	53.2

6.4 Discussion

We presented a novel method that reduces the Sim2Real gap in 3D object classification. This is achieved both through a better robustness to under segmentation using a voting scheme and a better robustness to scale change through reproducible parts and a scale-invariant graph learning architecture. Our conjecture that methods tested on real scenes are negatively impacted when relying on point coordinates is supported by our experimental results on the ScanObjectNN dataset [41]. The performance of our method, that does not rely on point coordinates, better handles the challenges

of imperfect segmentation and scale. The impact of scale change is a particularly important issue as even within a given class, scale can vary greatly.

Hightlights

(1) Neighborhood-based methods can easily be adapted to not require points coordinates. This makes them significantly less sensitive to scale change, while more discriminative than graph-based representations that are unaware of the relative position of their neighbors.
(2) While the part definition does not require parts to have a particular semantic meaning, it is beneficial to the classification accuracy if they are respectful of the object boundaries.
(3) Voting-based approaches can help further robustify methods to under-segmented inputs.

Chapter 7

Conclusion

A service robot that can work anytime and anywhere is still out-of-reach given the current state of research. There is, therefore, a need to adapt to the environment the robot will be deployed in. Besides potential re-training and modeling efforts, the methods have to be robust enough to the variations the robot will face. While it is commonly accepted that more data and larger learning architectures can deal with most artifacts and variations and noise sources, it is not an adequate solution for resource-constrained robots. In particular, 3D understanding, which remains a key part in creating helpful service robots, has greatly progressed, thanks notably to the advent of deep learning. This progress, however, has come at the expense of the amount of data required. Leveraging artificial data helps to alleviate this challenge, but a gap remains between the artificial and real domain due to sensor noise, pervasive occlusions, and variable orientations.

This chapter summarizes how the methods introduced in this thesis integrate long-standing design ideas from robot vision in deep learning architectures and increase the robustness and reduce the gap between artificial and real domains without needing more data. Finally, insights are given about potential future works in 3D object classification.

7.1 Summary

In each chapter, different part-based representations are introduced to achieve different robustness, and in particular to occlusion, rotation, and scale changes. The different methods are evaluated on a variety of standard artificial and real-world datasets and achieve state-of-the-art performance.

A novel architecture inspired by the Ensemble Shape Function (ESF) and the Point Pair Feature (PPF) is introduced in Chapter 3. Sampling of pair of points with one point within a neighborhood and one anywhere in the object, a set of clusters is obtained. The features extracted from the pairs of points make the representation rotation-invariant, and feature vectors are obtained for each vector using a PointNet-like architecture. The features of each cluster are combined into an object-level feature representation using a graph convolutional network to obtain a data-efficient network that is more discriminative than previous work.

Chapter 4 explores the impact of the sim2real gap in 3D object classification on

different 3D representations. In particular, the robustness of standard point cloud-based, view-based, and voxel grid-based neural networks is investigated when trained on artificial data and tested on real data. From this set of experiments, general guidelines for architecture design are obtained. In particular, over-segmentation and hierarchical approaches are more robust to occlusion. Furthermore, representations that are based on the object surface rather than the Euclidean neighborhood also handle occlusions better. Simple artificial data transformation also helps reduce the gap. While the problem of view selection remains, view-based approaches tend to outperform methods designed specifically for 3D data, in good parts because 2D architectures have been studied and improved over a longer period of time.

Chapter 5 builds on the lessons learned in the previous chapters and introduces a part-based method that tackles the challenges of learning from artificial 3D data to classify real 3D data. The parts are grown following the object's surface based on the surface curvature, such that featureless areas create larger parts. The part representation is based on the PointNet architecture and is rotation-invariant. The parts features are further combined using a graph convolutional network to achieve better discriminativeness. The method achieves results on par with state-of-the-art view-based methods when training on artificial data and testing on real data and outperforms state-of-the-art point cloud based methods. In addition, the method also outperforms state-of-the-art point cloud based methods when training and testing on real data.

Chapter 6 further improves upon the architecture introduced in Chapter 5. Specifically, the parts are created by adapting the LCCP segmentation method to 3D instead of 2.5D, reducing the likelihood that a given part would cross an object boundary. The parts are combined using a method similar to DGCNN instead of the graph convolutional network such that the features can take advantage of the relative position of each neighbor without being sensitive to scale, whereas the graph convolutional network is unaware of their relative position. Finally, each part votes for the object center, further separating object parts from noisy background parts, making the overall method more robust to under-segmentation. Altogether, the method outperforms state-of-the-art methods when training on artificial data and testing on real data through much better robustness to scale change.

7.2 Outlook

Reducing the gap between artificial and real data is ongoing work, and further improvements are expected in the coming years. It creates the opportunity to benefit from the power of deep learning without needing massive, costly data annotation work, especially when dealing with 3D data that has been simulated for many years and that is easier to simulate than RGB data. This also opens up new opportunities and new challenges, bringing us closer to having a robot perform useful tasks.

7.2.1 3D object detection and segmentation with limited annotation

One natural extension of the work presented in this thesis is to consider the tasks of object detection and segmentation. 3D object classification is a canonical task in the field and serves as a meaningful test-bed for any other semantic task performed on 3D data, but more work is needed to adapt the ideas presented here for 3D object segmentation. Building into the architecture the robustness described in this thesis could be one step. An even more interesting direction would be to improve the graph-based part understanding such that object detectors could be trained directly from examples of single objects rather than complete annotated scenes. This task, while trivial for humans, is very challenging for current learning approaches but could further lower the annotation effort. Indeed, while simpler to achieve in artificial data than with real-world data, creating artificial scenes still requires a lot of manual work to scale and arrange objects in a semantically meaningful way. Using this part-based approach, the segmentation is transformed into an edge prediction problem, creating an edge when two parts belong to the same object and no edge otherwise.

7.2.2 3D object classification of articulated and deformable objects

Given the progress of state-of-the-art 3D object classification, a natural extension is to consider non-rigid objects. Learning architectures relying on point coordinates are sub-optimal to represent object deformation as any deformation would drastically change the resulting features. Part-based approaches could alleviate this issue when dealing with articulated objects, as long as the parts do not cross any joint, and rotation-invariant representation can be created for every part. This part representation could therefore help make networks more data-efficient as they would not have to learn a different representation for every possible angle of the articulation. The challenge is then to manage the creation of meaningful object-level representations despite different potential relative orientations of the parts.

7.2.3 The place of classical robot vision in a deep learning world

Robot vision and machine vision have produced many useful methods, and while using priors in the learning architecture is bound to be outperformed by prior-free approaches given enough data and computation, there are many opportunities to integrate some of these ideas in current architectures. These priors can help create more computationally efficient methods that require less data and more robust prediction within the domain of application. Furthermore, many classical methods provide confidence estimates, and replicating these in deep learning network would be very beneficial and enable a more meaningful connection between learned and learning-free blocks in a robot complex system. A robot needs to act on its environment, and an understanding of risk and

uncertainty could help avoid catastrophic failures and potential harm to humans and the robot itself. Bayesian Deep Learning and Evidential Deep Learning both provide solid frameworks to obtain such estimates.

Bibliography

[1] C. Esterwood and L. Robert, "Robots and covid-19: Re-imagining human–robot collaborative work in terms of reducing risks to essential workers," *Available at SSRN 3767609*, 2021.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] G. Halmetschlager-Funek, M. Suchi, M. Kampel, and M. Vincze, "An empirical evaluation of ten depth cameras: Bias, precision, lateral noise, different lighting conditions and materials, and multiple sensor setups in indoor environments," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 67–77, 2018.

[5] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *Proceedings of IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 2987–2992.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of International Conference on Learning Representations*, 2017.

[7] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[8] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proceedings of Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108.

[9] D. Maturana and S. Scherer, "Voxnet: A 3D convolutional neural network for real-time object recognition," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 922–928.

[10] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proceedings of IEEE International Conference on Computer Vision*, 2015, pp. 945–953.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, (accepted as poster), 2018.

[12] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[13] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5828–5839.

[14] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 87–102.

[15] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[16] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 681–687.

[17] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European Conference on Computer Vision*, Springer, 2014, pp. 345–360.

[18] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[19] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[20] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[22] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *2012 second international conference on 3D imaging, modeling, processing, visualization & transmission*, IEEE, 2012, pp. 524–530.

[23] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96, New York, NY, USA: Association for Computing Machinery, 1996, 303â€"312, ISBN: 0897917464. [Online]. Available: https://doi.org/10.1145/237170.237269.

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

[25] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3d reconstruction with loop closure," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, 2016, pp. 500–516.

[26] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration," *ACM Transactions on Graphics 2017 (TOG)*, 2017.

[27] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "ElasticFusion: Dense Slam without a pose graph," in *Robotics: Science and Systems*, 2015.

[28] S. Schreiberhuber, J. Prankl, T. Patten, and M. Vincze, "Scalablefusion: High-resolution mesh-based real-time 3D reconstruction," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2019, pp. 140–146.

[29] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European conference on computer vision*, Springer, 2020, pp. 405–421.

[30] P. K. Nathan Silberman Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[31] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "Objectnet3d: A large scale database for 3d object recognition," in *European Conference Computer Vision (ECCV)*, 2016.

[32] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-Semantic Data for Indoor Scene Understanding," *ArXiv e-prints*, Feb. 2017. arXiv: 1702.01105 [cs.CV].

[33] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "Scenenn: A scene meshes dataset with annotations," in *International Conference on 3D Vision (3DV)*, 2016.

[34] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017.

[35] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: Real-world perception for embodied agents," in *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*, IEEE, 2018.

[36] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[37] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[38] A. Handa, V. Pătrăucean, S. Stent, and R. Cipolla, "Scenenet: An annotated model generator for indoor scene understanding," in *ICRA*, 2016.

[39] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[40] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, "SAPIEN: A simulated part-based interactive environment," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[41] M. A. Uy, Q.-H. Pham, B.-S. Hua, D. T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *International Conference on Computer Vision (ICCV)*, 2019.

[42] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, 1999.

[43] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Learning informative point classes for the acquisition of object model maps," in *2008 10th International Conference on Control, Automation, Robotics and Vision*, IEEE, 2008, pp. 643–650.

[44] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2009, pp. 1848–1853.

[45] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 2155–2162.

[46] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proceedings of European Conference on Computer Vision*, 2010, pp. 356–369.

[47] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 998–1005.

[48] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[49] D. Bobkov, S. Chen, R. Jian, M. Z. Iqbal, and E. Steinbach, "Noise-resistant deep learning for object classification in three-dimensional point clouds using a point pair descriptor," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 865–872, 2018.

[50] H. Deng, T. Birdal, and S. Ilic, "Ppfnet: Global context aware local features for robust 3d point matching," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 195–205.

[51] F. M. Carlucci, P. Russo, and B. Caputo, "A deep representation for depth images from synthetic data," in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 1362–1369.

[52] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, "A deeper look at 3d shape classifiers," in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds., Cham: Springer International Publishing, 2019, pp. 645–661, ISBN: 978-3-030-11015-4.

[53] T. Yu, J. Meng, and J. Yuan, "Multi-view harmonized bilinear network for 3d object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 186–194.

[54] Z. Han, H. Lu, Z. Liu, C.-M. Vong, Y.-S. Liu, M. Zwicker, J. Han, and C. P. Chen, "3d2seqviews: Aggregating sequential views for 3d global feature learning by cnn with hierarchical attention aggregation," *IEEE Transactions on Image Processing*, vol. 28, no. 8, pp. 3986–3999, 2019.

[55] C. Wang, M. Pelillo, and K. Siddiqi, "Dominant set clustering and pooling for multi-view 3d object recognition," in *Proceedings of British Machine Vision Conference (BMVC)*, vol. 12, 2017.

[56] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[57] K. Sfikas, I. Pratikakis, and T. Theoharis, "Ensemble of panorama-based convolutional neural networks for 3d model classification and retrieval," *Computers & Graphics*, vol. 71, pp. 208–218, 2018.

[58] M. Yavartanoo, E. Y. Kim, and K. M. Lee, "Spnet: Deep 3d object classification and retrieval using stereographic projection," in *Asian conference on computer vision*, Springer, 2018, pp. 691–706.

[59] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, "Orientation-boosted voxel nets for 3D object recognition," in *Proceedings of British Machine Vision Conference*, 2017.

[60] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3577–3586.

[61] R. Klokov and V. Lempitsky, "Escape from cells: Deep KD-Networks for the recognition of 3D point cloud models," in *Proceedings of IEEE International Conference on Computer Vision*, 2017, pp. 863–872.

[62] C. Ma, W. An, Y. Lei, and Y. Guo, "Bv-cnns: Binary volumetric convolutional networks for 3d object recognition.," in *BMVC*, vol. 1, 2017, p. 4.

[63] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3145–3152, 2018.

[64] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084.

[65] S. Ravanbakhsh, H. Su, J. Schneider, and B. Poczos, "Deep learning with sets and point clouds," in *Proceedings of International Conference on Learning Representations*, 2017.

[66] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proceedings of Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[67] M. Tatarchenko*, J. Park*, V. Koltun, and Q.-Y. Zhou., "Tangent convolutions for dense prediction in 3D," *CVPR*, 2018.

[68] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on χ-transformed points," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 828–838.

[69] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," *arXiv preprint arXiv:1904.08889*, 2019.

[70] H. Thomas, "Rotation-invariant point convolution with multiple equivariant alignments.," in *2020 International Conference on 3D Vision (3DV)*, 2020, pp. 504–513.

[71] C. Chen, G. Li, R. Xu, T. Chen, M. Wang, and L. Lin, "Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[72] N. Verma, E. Boyer, and J. Verbeek, "Feastnet: Feature-steered graph convolutions for 3d shape analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2598–2606.

[73] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and J. Bruna, "Surface networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Oral Presentation)*, Jun. 2018.

[74] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: A network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, 90:1–90:12, 2019.

[75] S. Gong, L. Chen, M. Bronstein, and S. Zafeiriou, "Spiralnet++: A fast and highly efficient mesh convolution operator," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct. 2019.

[76] Y. Feng, Y. Feng, H. You, X. Zhao, and Y. Gao, "Meshnet: Mesh neural network for 3d shape representation," *AAAI 2019*, 2018.

[77] F. Milano, A. Loquercio, A. Rosinol, D. Scaramuzza, and L. Carlone, "Primal-dual mesh convolutional neural networks," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. [Online]. Available: `https://github.com/MIT-SPARK/PD-MeshNet`.

[78] M. Suchi, T. Patten, D. Fischinger, and M. Vincze, "Easylabel: A semi-automatic pixel-wise object annotation tool for creating robotic rgb-d datasets," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6678–6684.

[79] R. S. Rodrigues, J. F. Morgado, and A. J. Gomes, "Part-based mesh segmentation: A survey," in *Computer Graphics Forum*, Wiley Online Library, vol. 37, 2018, pp. 235–274.

[80] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter, "Voxel cloud connectivity segmentation - supervoxels for point clouds," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2027–2034.

[81] S. C. Stein, M. Schoeler, J. Papon, and F. Wörgötter, "Object partitioning using local convexity," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 304–311.

[82] R. Liu and H. Zhang, "Segmentation of 3d meshes through spectral clustering," in *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, IEEE, 2004, pp. 298–305.

[83] O. V. Kaick, N. Fish, Y. Kleiman, S. Asafi, and D. Cohen-Or, "Shape segmentation by approximate convexity analysis," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 1, pp. 1–11, 2014.

[84] X. Wang, X. Sun, X. Cao, K. Xu, and B. Zhou, "Learning fine-grained segmentation of 3d shapes without part labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 276–10 285.

[85] L. Li, M. Sung, A. Dubrovina, L. Yi, and L. J. Guibas, "Supervised fitting of geometric primitives to 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2652–2660.

[86] C. Sommer, Y. Sun, E. Bylow, and D. Cremers, "Primitect: Fast continuous hough voting for primitive detection," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 8404–8410.

[87] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9277–9286.

[88] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "Unseen object instance segmentation for robotic environments," *IEEE Transactions on Robotics*, 2021.

[89] J.-B. Weibel, T. Patten, and M. Vincze, "Geometric priors from robot vision in deep networks for 3D object classification," in *Proceedings of International Conference on Robotics and Automation (Workshop on Multimodal Robot Perception: Perception, Inference and Learning for Joint Semantic, Geometric, and Physical Understanding)*, 2018.

[90] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of International Conference on Machine Learning*, 2015, pp. 448–456.

[91] A. Cheraghian and L. Petersson, "3dcapsule: Extending the capsule architecture to classify 3d point clouds," *CoRR*, vol. abs/1811.02191, 2018. arXiv: `1811.02191`. [Online]. Available: `http://arxiv.org/abs/1811.02191`.

[92] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust & efficient point cloud registration using pointnet," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[93] L. Ge, Y. Cai, J. Weng, and J. Yuan, "Hand pointnet: 3d hand pose estimation using point sets," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

Appendix A: Open Resources

A.1 Codes and Tools

- Robust 3D classification (Chapter 3): `https://github.com/jibweb/robust_3d_object_clf`
- Addressing the Sim2Real gap (Chapter 5): `https://github.com/jibweb/addressing_sim2real`
- 3D Object Classification using Graph representations and Deep Center Voting (Chapter 6): `https://github.com/jibweb/dgcnn_voting`