Label Efficient 3D Scene Understanding

David Griffiths

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

University College London

October 20, 2022

I, David Griffiths, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

3D scene understanding models are becoming increasingly integrated into modern society. With applications ranging from autonomous driving, Augmented Reality, Virtual Reality, robotics and mapping, the demand for well-behaved models is rapidly increasing. A key requirement for training modern 3D models is high-quality manually labelled training data. Collecting training data is often the time and monetary bottleneck, limiting the size of datasets. As modern data-driven neural networks require very large datasets to achieve good generalisation, finding alternative strategies to manual labelling is sought after for many industries.

In this thesis, we present a comprehensive study on achieving 3D scene understanding with fewer labels. Specifically, we evaluate 4 approaches: existing data, synthetic data, weakly-supervised and self-supervised. Existing data looks at the potential of using readily available national mapping data as coarse labels for training a building segmentation model. We further introduce an energy-based active contour snake algorithm to improve label quality by utilising co-registered LiDAR data. This is attractive as whilst the models may still require manual labels, these labels already exist. Synthetic data also exploits already existing data which was not originally designed for training neural networks. We demonstrate a pipeline for generating a synthetic Mobile Laser Scanner dataset. We experimentally evaluate if such a synthetic dataset can be used to pre-train smaller real-world datasets, increasing the generalisation with less data.

A weakly-supervised approach is presented which allows for competitive performance on challenging real-world benchmark 3D scene understanding datasets with up to 95% less data. We propose a novel learning approach where the loss

function is learnt. Our key insight is that the loss function is a local function and therefore can be trained with less data on a simpler task. Once trained our loss function can be used to train a 3D object detector using only unlabelled scenes. Our method is both flexible and very scalable, even performing well across datasets.

Finally, we propose a method which only requires a single geometric representation of each object class as supervision for 3D monocular object detection. We discuss why typical $\mathcal{L}_2$-like losses do not work for 3D object detection when using differentiable renderer-based optimisation. We show that the undesirable local-minimas that the $\mathcal{L}_2$-like losses fall into can be avoided with the inclusion of a Generative Adversarial Network-like loss. We achieve state-of-the-art performance on the challenging 6DoF LineMOD dataset, without any scene level labels.

Impact Statement

The work presented in this thesis addresses fundamental problems in 3D computer vision. Specifically, this thesis looks at how to train machine learning models with a relaxed requirement on collection of ground truth 3D data. Access to high-quality ground truth data is a fundamental bottleneck for both academic researchers as well as industry companies looking to deploy 3D machine learning systems. As such, the potential impact of the work presented spans both academia and industry alike.

Within academia, several pipelines are established which can act as an initial starting point for new and existing researchers. The methods described within this thesis clearly outline practical steps which can be taken to train 3D machine learning models without undertaking large-scale ground truth data labelling exercises. The potential impact to an academic researcher here is two-fold. Firstly, the methods presented can undoubtedly be improved and refined. This offers an established starting point to further address the problems discussed. Secondly, academic researchers often suffer from a lack of ground truth training data. This is due to most large-scale data collections being undertaken by private companies, which retain the data for internal use. By employing the methodologies laid out in the subsequent chapter's, researchers can overcome some of the data limitations, allowing for accelerated research on other areas where data is not the main focal point.

Outside of academia this thesis has a potential impact in the training of production 3D machine learning models. Whilst large companies can often afford large-scale ground truth labelling exercises, this is usually not the case for smaller, and especially new, companies. By utilising the research presented in this thesis companies can potentially compete with companies that have an order of magnitude larger

labelling budget. This would be especially true for non-governmental organisations which due to their non-profit nature often cannot afford to undertake expensive data collections.

The potential impact of this thesis is not specific to any industry, or geographically bound. As such, the impacts described above could be applied across multiple industries as well as internationally.

Acknowledgements

I would first and foremost like to thank my supervisor, Jan Boehm. Jan introduced me into the topic of this thesis and has remained by my side as we have progressed and developed the ideas presented throughout. He has been both a supervisor, mentor and friend and is responsible for all the positive memories I have for my time at UCL. I could not have asked for a better supervisor.

Secondly, I would like to thank Tobias Ritschel. Tobias shaped much of the latter research projects during my PhD. He adopted me into his group, which gave me invaluable exposure to amazing researchers and friends. Without his mentoring I would not be the researcher I am today and much of my research was only made possible because of his dedication. Like Jan, Tobias has been a supervisor, mentor and friend.

For my early academic interests, I would like to thank my MSc supervisor Helene Burningham. Helene introduced me to research when I undertook my first serious research project with her. This project was also my first exposure to 3D vision. It was because this experience was such a positive one that I decided to pursue research further, leading to this PhD thesis. Throughout my time as a post-graduate at UCL Helene has a remained a great friend and mentor.

Throughout my PhD I have had the privilege to study alongside amazing PhD students. The list would be too long to name here, but I would like to thank everyone for all the moral support, ideas and lunch-time chats which made the entire process of the PhD thoroughly enjoyable.

This PhD was funded by Bentley Systems, I wish to thank them for their financial support and discussions throughout the process.

Outside of the academic environment I have been fortunate to have been whole heartily supported by my parents. Throughout my entire studies my parents have supported me in every possible way, always being my biggest supporters and taking a genuine interest in the work I carried out at UCL. Without their tireless support this thesis would not have been possible.

Lastly, I would like to thank Jan Dirk Wegner and Lourdes Agapito, for taking the time out their very busy schedules to review my work and serve on the committee.

Acronyms

ACM Active Contour Models. 36–38

ALS Airborne Lidar Scanning. 32–34, 54, 57

AR Augmented Reality. iii, 1

CAD Computer Aided Design. 28, 29

CNN Convolutional Neural Network. 17, 19, 22, 41, 46, 47, 49, 53, 92, 94–96, 123

DoF Degree of Freedom. 4, 8, 28

DR Differentiable Renderer. 28, 29, 92, 94, 95, 97, 103, 109, 110, 114

DSM Digital Surface Model. xvii, 35, 36, 39, 40, 48

GAC Geodesic Active Contours. 36–39, 42, 43, 46–48, 50

GAN Generative Adversarial Network. iv, 10, 11, 24, 29, 96, 114, 120, 125

GIS Geographical Information System. 23, 33, 34, 49

IoU Intersection over Union. 43, 45, 47, 49, 80

LiDAR Light Detection and Ranging. iii, 3, 8, 9, 11, 23–25, 28, 29, 33, 54, 117, 124

mAP mean Average Precision. 43–47, 49, 51, 52, 80–82, 84, 86–89

MLP multi-layer perceptron. 16, 18–20, 92, 100

MLS Mobile Laser Scanner. iii, 10, 11, 30, 32, 54, 55, 57, 60, 69, 124

NMS Non-maximum Suppression. 21, 80, 81

NN Neural Network. 58

OS Ordnance Survey. 35, 36, 45, 47

PDE Partial Differential Equation. 37–39

RG-D Red, Green and Depth. 36, 42, 44–47, 49

RGB Red, Green and Blue. 11, 20, 25, 34–36, 42, 44–47, 49, 117

SDF Signed Distance Function. 28

SfM Structure-from-Motion. 2, 4

SIFT Scale-Invariant Feature Transform. 2

sim2real Synthetic to Real. 24, 54, 66

TLS Terrestrial Laser Scanner. 30–32, 54

VR Virtual Reality. iii, 1

Publications

The work presented in this thesis is based on the following publications.

***Griffiths David**, Boehm Jan.* Improving public data for building segmentation from Convolutional Neural Networks (CNNs) for fused airborne lidar and image data using active contours // ISPRS Journal of Photogrammetry and Remote Sensing. 2019. 154. 70–83.

***Griffiths David**, Boehm Jan.* SynthCity: A large scale synthetic point cloud // arXivpreprint arXiv:1907.04758. 2019.

***Griffiths David**, Boehm Jan, Ritschel Tobias.* Curiosity-driven 3D Object Detection Without Labels // 2021 International Conference on 3D Vision (3DV), 2021, pp. 525-534.

***Griffiths David**, Boehm Jan, Ritschel Tobias.* Finding Your (3D) Center: 3D Object Detection Using a Learned Loss // Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVIII 16.2020. 70–85.

The following works were also published during the course of study.

***Griffiths David**, Boehm Jan*. Rapid object detection systems, utilising deep learningand unmanned aerial systems (uas) for civil engineering applications

// International Archives of the Photogrammetry, Remote Sensing and Spatial InformationSciences-ISPRS Archives. 2018. 391–398.

Griffiths David*, Boehm Jan*. Weighted Point Cloud Augmentation for Neural Net-work Training Data Class-Imbalance. // International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives. 2019. 981–987.

Eric Sanchez Castillo, ***Griffiths David****, Boehm Jan*. Semantic Segmentation of Terrestrial LiDAR Data Using Cp-Registered RGB Data // International Archives of the Photogrammetry,Remote Sensing and Spatial Information Sciences-ISPRS Archives. 2021. 223–229.

David Griffiths*, Tobias Ritschel, Julien Philip*. OutCast: Outdoor Single Image Depth Relighting with Cast Shadows. // Eurographics 2022: 43rd Annual Conference of the European Association for Computer Graphics. April 25-29. 2022.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Over the past decade there has been a significant increase in the use of deep learning to solve visual perception tasks. This is largely owed to advancements in algorithm development, computer hardware and the ability to store and transfer very large quantities of data. As a result, technologies such as facial recognition, automated medical imaging and large-scale image retrieval are commonplace in society. A key commonality in visual perception tasks that are experiencing a high level of success is that they tend to operate in the 2D domain. More specifically, they assume that the 3D world in which they represent has been projected onto a 2D plane. There is still, however, a very large number of tasks that require processing 3D geometric representations. Such problems are typically sub-categorised as 3D computer vision.

3D computer vision is concerned with the long-standing problem of how to understand and infer meaningful information from 3D representations of the world. Popular tasks include perception and mapping problems in robotics and autonomous driving, Augmented Reality (AR), Virtual Reality (VR), automated national mapping and 3D reconstruction. In a world where 3D computer vision is ”solved” we would expect the entire world to be completely mapped, knowing exactly where everything is in a 3D space, for both real-time on-board applications (e.g. robotics, autonomous vehicles, AR/VR) and off-board applications such as national and in-

ternational asset databases. Whilst this is still an elusive goal, progress is being made thanks to advancements in 3D data capture.

Capturing 3D data through passive sensors is (almost) as old as capturing 2D data (e.g., a photograph). In 1851 French inventor Aimé Laussedat utilised the newly invented camera for mapmaking and surveying (Encyclopedia Britannica, 2021). The technique of using images to determine 3D information, known as photogrammetry, became the standard approach for national relief mapping. The term photogrammetry is dervied from three Greek works: *photo* meaning "light", *gram* meaning "drawn" and **metry** meaning "measuring". This technique that was both invaluable and rapidly developed during both World Wars 1 and 2. Whilst photogrammetry is still widely used, traditional approaches which require large and heavy machinery (e.g., stereoplotters) and trained human operators have been superseded by fully autonomous computer vision-based pipelines. Although the underlying mathematics remains relevant, arguably the largest advancement allowing for automated photogrammetry is the ability to detect the same world feature in two images, something that until recently could only be achieved by humans. The seminal work by Lowe (2004) expressed image points as vectors, known as feature descriptors, describing the context of the pixel. Feature descriptors are then matched across images by finding descriptors with a close (Euclidean) distance in feature space. The feature descriptor known as Scale-Invariant Feature Transform (SIFT) has proved to be robust against mild variances expected between image capture (e.g., light, perspective change etc.). Advancements in feature point matching have enabled the now widely adopted framework of Structure-from-Motion (SfM) (Ullman, 1979; Schonberger, Frahm, 2016; Andrew, 2001). Broadly speaking, SfM is a photogrammetric technique which samples 2D surfaces of 3D geometry from either a sequence of images from a single camera, or a single image from a sequence of cameras. Each 3D measurement $p \in \mathbb{R}^3$ contains an x, y and z coordinate in an arbitrary 3D space. A set of points p is called a Point Cloud $\mathcal{P} = \{p_1, p_2 \dots p_n\}$. In practice much more per-point information is stored for example the red, green and blue values from pixels used to determine the spatial coordinates of the point. A

point cloud can therefore be defined as $\mathcal{P} \in \mathbb{R}^{3+k}$ where k are additional features.

An alternative approach to capturing 3D with passive sensors is the use of active sensors. Unlike passive sensors which require external illumination (i.e., light), active sensors generate their own energy. An example of an active sensor is a Light Detection and Ranging (LiDAR) instrument. LiDAR can capture the 3D geometry of a scene by sending pulses of light out from a central emitter. By measuring the time taken for the light to bounce off a surface and arrive back at the sensor the distance d can be determined as $d = \frac{ct}{2}$ where c is the speed of light and t the measured time. By simultaneously measuring the direction of the light path (e.g., θ and ϕ) a 3D point cloud can be obtained. Advantages of active approaches include the ability to capture environments with no light or when visibility is poor (i.e., clouds). Other active sensing instruments include Radio Detection and Ranging (radar) and Sound Navigation and Ranging (sonar) where radio and sound waves are emitted respectively.

1.2 Definition of a Point Cloud

Although in its most general form we can define a point cloud $\mathcal{P}$ as a set of points $\{\mathcal{P} : p \in \mathbb{R}^{3+k}\}$, this is too general for the type of point clouds we collected from passive and active scene reconstruction techniques. This is due to the fact that all observations are on the surface of the 3D objects we measure. Instead, each point p is a sampling from a 2D surface which arises at the boundary of a 3D object. For example, a LiDAR scan of a 3D solid ball Φ would capture samples p of the boundary sphere Θ at the boundary of Φ. Any point p could be defined on Θ using a 2D coordinate system (i.e., Latitude Longitude), therefore $p \in \mathbb{R}^2$. In practice, defining a local 2D coordinate system requires prior knowledge of the surface p is sampled from. Furthermore, we also require knowledge of which surface p belongs to. As such p is defined using a 3D coordinate system containing all surfaces present in the scan.

1.3 3D Scene Understanding

Both passive and active sensors have a plethora of advantages and disadvantages. As such, which type of sensor to use is largely driven by the task at hand. What both have in common is the representation of geometric data capture. Both represent a scene $\mathcal{S}$ and a point cloud $\mathcal{P}^{3+k}$. In the following chapters a point cloud is treated independently from the sensor in which it is captured. Instead, the focus is on approaches for creating a mapping $f : \mathcal{P} \rightarrow X$ where X is some desired semantically meaningful information for which we wish to extract from $\mathcal{P}$, ultimately informing us about S. This problem we define as 3D scene understanding. This is different from the broader field of 3D computer vision which would encompass for example 3D reconstruction (e.g., SfM) or single object classification. Instead, 3D scene understanding is concerned with extracting semantics from a 3D scene representation. We can further sub-define 3D scene understanding into the following tasks: scene classification, per-point classification (also known as semantic segmentation), instance semantic segmentation and object detection. More complex tasks further address the relationship between objects (e.g., car is on the ground), as well as mapping the scene to a coherent text description explaining what the scene contains (Chen et al., 2021) or vice-versa (Chen et al., 2020).

1.4 Challenges in 3D Scene Understanding

Although, arguably not as popular as 2D image understanding, 3D scene understanding has benefited from a significant research effort for as long is its 2D counterpart. There are many reasons why 3D scene understanding is more complex and less attractive to researchers than 2D image understanding for comparable tasks.

Dimensionality As dimensions increase, so does Degree of Freedom (DoF) of the problem and therefore complexity. For example, take a common object detection task; *"Where is the chair in the room?"*. To answer this question in a 2D setting we could predict a common parametrisation (Girshick, 2015; Zhou et al., 2019a): x, y image-space coordinates and height and width of the object in pixels. Our function f can then be defined as $f : \mathcal{I} \in \mathbb{R}^{i,j,c} \rightarrow \mathbb{R}^4$ where c is the image channels.

The 3D equivalent would require an additional z-coordinate and depth length. We would then need to map $f : \mathcal{G}^{x,y,z,c} \rightarrow \mathbb{R}^6$ where $\mathcal{G}$ is some geometric representation of the scene (e.g., a point cloud). A more unfavourable parameterisation could regress directly the bounding box points. This would give us $f : \mathcal{I}^{x,y,c} \rightarrow \mathbb{R}^8$ and $f : \mathcal{G}^{x,y,z,c} \rightarrow \mathbb{R}^{24}$ for 2D and 3D respectively.

Data representation In the above paragraph we write our 3D geometric representation as $\mathcal{G}$, this is because there is no definitive representation for 3D data (unlike a discrete array for 2D images). There are a number of proposed 3D representations, however, the most common can be categorised as: point cloud, volumetric and polygonal mesh. We will briefly discuss each in turn.

As discussed above, at their most basic, a point cloud $\mathcal{P}$ can be defined as a set of points p with an x, y and z coordinate, such that $\mathcal{P} \in \mathbb{R}^{n \times 3}$ where n is the number of points. Whilst point clouds are a very flexible 3D representation, they have several limitations. Firstly, point clouds are unstructured data types. This means that p_i and p_{i+1} can have no assumed spatial relationship. This is fundamentally different from a structured data type such as images where pixel p_{ij} and $p_{i+1,j}$ are spatially relevant (i.e., pixels close together are relevant to one another). Secondly, point clouds are continuous representations. For example, two pixels p_i and p_{i+1} have a known distance $d = ||p_{i,j} - p_{i+1,j}||_1 = (1,0)$ in image space. In contrast, two points p_i and p_{i+1} can have any distance in Euclidean space. This prevents point cloud processing algorithms from exploiting Convolutional Neural Networks (CNN), which have been the fundamental operation behind the success of 2D computer vision. In recent years, this problem has been largely overcome (Qi et al., 2017b; Hermosilla et al., 2018; Thomas et al., 2019). However, point-based convolution operators still require neighbourhood searches at each stage in a neural network significantly hindering their computational efficiency. This is explained in further detail in Chapter 2.

Another key limitation of point clouds is that two points close in world-space have no indication whether they are both samples from the same surface. As such we would refer to point clouds as an unordered data type. An alternative represen-

tation which includes connectivity is the polygonal mesh representation. A mesh $\mathcal{M}$ is made up of the following: vertices $v \in \mathbb{R}^3$, edges $e \in \mathbb{N}^2$ and faces $f \in \mathbb{N}^{\geq 3}$. A vertex v can be thought of as a point p in a point cloud. An edge e contains 2 indexes indicated a connection between two vertices. A face f contains a list of $n \geq 3$ indexes indicated which vertices create a closed loop. This has two significant advantages. Firstly, rendering software can assign face colours and normals to each face to give the appearance of a watertight model. Secondly, mesh-based neural networks have access to the connectivity of vertices, so it is easier to create local features on vertices that are connected and not just close in world-space (Schult et al., 2020). The key drawback in the mesh representation is the ability to acquire the connectivity. Whilst there are many algorithms for automatic mesh generation from point cloud (Edelsbrunner et al., 1983; Bernardini et al., 1999; Kazhdan et al., 2006), the problem is far from trivial. As such even state-of-the-art mesh algorithms contain many incorrect connections. Due to this required post-processing stage, meshes are not considered a raw data type.

Point clouds typically live in a continuous domain. However, many computer vision techniques assume data points lie in a discretised grid. Too address this, discrete volumetric representations are proposed. The most common being a voxel grid. A voxel grid is defined on a regular 3D grid, in the same way an image is defined on a regular 2D grid. As such we can define a voxel grid $\mathcal{V} \in \mathbb{R}^{i,j,k,c}$. The most simple form of voxel grid is a binary occupancy grid where $c \in \{0,1\}$ denotes if the voxel contains a point (1) or not (0). By representing 3D points on a discretised regular grid all operators common in 2D image processing (e.g., image kernels) can be easily extended to 3D. This allows for example a 3D CNN which works with the same fundamental principles as a standard 2D CNN. Whilst this might seem an attractive prospect, a key limitation of voxel-based representation is that the storage and computation cost grows cubically with the grid resolution. As such learning large, deep features (e.g., through a 3D CNN) becomes unfeasible with current computer hardware.

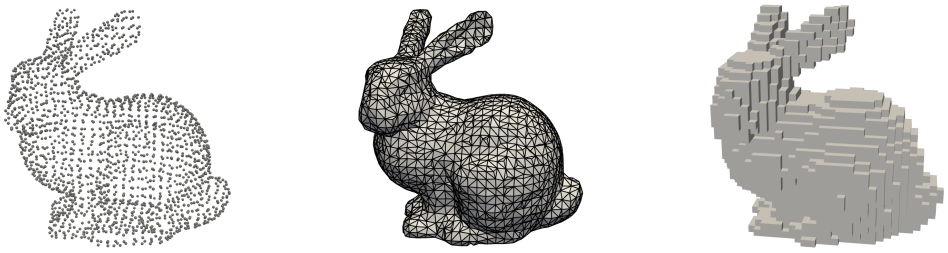


Figure 1.1: Visualisation of the three main 3D representations: point cloud (**left**), mesh (**middle**) and voxel grid (**right**). Depicted is the 'Stanford Bunny', provided by the Stanford Computer Graphics Laboratory.

Data Collection of 3D data has made significant progress in the past two decades, this is due to advancements in both passive and active collection methods. As discussed in Sec. 1.1, modern feature point representations have enabled largely automatic "3D from 2D" pipelines with passive sensors (e.g., off-the-shelf cameras). There are, however, two main caveats when compared to 2D data capture. Firstly, collection of 2D images can be processed on-board (i.e., on the camera), due to the increased computational demands, this is less common for 3D. Large block adjustments of 2D images required for photogrammetry are both computationally and time consuming. A typical workflow would require uploading the captured 2D images onto a powerful desktop computer and processing the data in specialist software. Secondly, capturing 2D images suitable for a photogrammetric pipeline is far from trivial. To obtain a good 3D reconstruction the user must collect high quality images, with a sufficient overlap for the entire scene. Such acquisition can be relatively easy when modelling, for example, linear topologies for aerial topographic mapping. This is path flight paths can be pre-computed and using on-board GPS, flown autonomously. However, for the majority of other scenarios, such assumptions cannot be made. Examples include non-linear topologies, or environments where GPS is either unreliable or not available such as in inner-cities and indoors respectively. As a result of this, even highly experienced professional operators experience difficulties.

Active sensors have reduced in cost and increased in resolution and incredi-

ble rates. For example, LiDAR sensors such as the Livox Mid-40 series can be purchased for only $600 and have the ability to collect 100,000 points per second. Whilst this is undeniably cheaper than the entry price point only a decade ago at around $20,000, this is still considerably more expensive than an entry-level 2D camera (≈$50). Despite this, the addition of a LiDAR sensor on higher end models of the Apple iPhone and iPad, suggest casual 3D capture at the scale of 2D images will be likely in the future.

Labelling One of the key contributors to the success of deep learning for computer vision is the ImageNet dataset (Deng et al., 2009). A characteristic of deep learning-based models is the requirement for very large datasets for training. ImageNet provides over 14 million manually labelled images for common object categories (e.g., cat, dog, house etc.) to train image classification networks. However, labelling complexity scales exponentially with dimensions. An image classification model is defined as $f : \mathcal{I}^{h,w,c} \rightarrow \mathbb{R}^k$ where k is the number of classes. Labelling therefore requires simply indicating for each image $\mathcal{I}$ which class k it belongs to. A simple user-interface can enable very fast, easy labelling for even inexperienced users. A natural next step in labelling complexity is 2D object detection. Here the user must position n boxes over objects of interest, a 2D object detection is therefore $f : \mathcal{I}^{h,w,c} \rightarrow \mathbb{R}^{n,4}$ using the i, j, h, w parameterisation (Girshick, 2015; Zhou et al., 2019a). This allows for the possibility of missing objects in the scene (i.e. only labelling 3 out of 4 cars), as well as introducing a per-object error $||b - \hat{b}||_2^2$ where b is the ideal box position and $\hat{b}$ is the labelled position for each DoF. When labelling transitions to 3D we incur two additional complexities. Firstly, we have a higher DoF and therefore more potential error sources. However, perhaps more significantly is the additional burden of moving from a 2D input to 3D input. Whereas labelling in 2D can be achieved from a single view on a computer screen, labelling in 3D requires the user to explore the 3D space through 3D visualisation software. 3D visualisation software requires an experienced operator and adds a considerable amount of time consumption. While a 2D bounding box labelling may take in the order of 10's of seconds per image, 3D bounding box labelling takes in the order

of minutes for even a simple scene. As a result, collecting 3D labelled data is considerably more expensive, and often lower quality. This has a major impact in the ability to train very large, robust neural networks with available data.

In this thesis we argue that this labelling constraint is a primary bottleneck in the development of 3D scene understanding. Although 3D data is more difficult and costly (both in terms of money and time) to collect than 2D data, 3D data can still be collected at several orders of magnitude higher than what is financially feasible to label. As such we argue finding alternative approaches to model training where per-object labelling is not required is of paramount importance to seeing 3D computer vision models as commonplace as their 2D counterparts.

1.5 Thesis Outline and Contribution

In this thesis we aim to address the bottleneck of requiring labels for training 3D scene understanding deep learning-based models [1]. Through a series of research studies, we look at the following alternative approaches:

Exploiting Existing Data - Chapter 3

For many mapping related tasks, huge manual labelling efforts have already been undertaken for centuries. This is usually the result of national mapping agencies such as the Ordnance Survey (UK), U.S. Geological Survey (United States) or Institut Géographique National (France). In this chapter we utilise such data for automatic building detection from aerial LiDAR data. A key issue with mapping data that was not designed for model training is that it is often very course. Considering this, we propose a method for improving course publicly available mapping data to provide accurate per-pixel segmentation masks for building detection and segmentation. In doing this, we suggest existing mapping data, albeit sometimes of course quality, can be used to automate the national mapping process.

Synthetic Data - Chapter 4

[1] For brevity we refer deep learning-based models simply as models for the remaining chapters as all of our efforts are focused on such models.

Another approach which avoids reformulating the training scheme of a network, is to use synthetic data as training data. In this chapter we outline a simple yet effect approach for generating synthetic point clouds from a virtual Mobile Laser Scanner (MLS) in a world we dub 'SynthCity'. A key advantage to create a point cloud in a synthetic world is that object and instances labels come for free. Our pipeline is simple and easily implemented with standard open-source software.

Weakly-supervised - Chapter 5

An alternative axis of research is to reduce the requirement of labels. In this chapter we propose a novel training procedure which performs 3D object detection with up to 95% less data. We achieve this by first learning a loss function which can be used to train a larger network on raw, unlabelled data. Furthermore, once our lightweight loss function is learnt our main network can continue to be trained on new unlabelled scenes from a similar domain. Such a reduction offers a significant relief on the burden of labelling, and experimentally we show with little sacrifice on performance.

Self-supervised (Analysis-by-Synthesis) - Chapter 6

In many scenarios, geometric object representations are available of 3D objects we wish to find in 2D image, however, 3D labels (e.g., position, orientation, colour etc.) are not. This problem is known as monocular 3D object detection and defined as $\mathcal{I}^{h,w,c} \rightarrow \mathbb{R}^{n,7}$ where n is the number of objects in the scene which are parameterised by x, y, z, h, w, d, θ. We build a simple network to predict an explicit scene parameterisation (e.g., position, orientation colour etc.) directly from a 2D image. We train this through analysis-by-synthesis, by rendering the geometric objects with the predicted scene parameterisation. We identify why a simple L2-like loss always fails and solve this with a novel training strategy using a critic network, akin to those found in a Generative Adversarial Network (GAN).

Finally, in Chapter 7 an in-depth discussion is undertaken comparing and con-

trasting each of the proposed approaches in Chapters 3 - 6. Recommendations for future research is outlined with empirical justifications witnessed in the preceding chapters.

The key contributions of this thesis are therefore as follows:

1. An approach for refining publicly available mapping data for building detection in aerial LiDAR and RGB data.

2. Demonstration of a simple and effective pipeline for generating high-quality synthetic MLS data, with per-object and per-instance ground truth labels.

3. A novel training procedure utilising a learnt loss function which can reduces the requirement of labels by up to 95% for 3D object detection in point clouds.

4. A novel training procedure utilising a GAN-like critic to achieve self-supervised monocular 3D object detection from only a per-object geometric representation as supervision.

Chapter 2

Literature Review

3D scene understanding is a very broad field encompassing many tasks and applications. The relevant literature therefore depends on what scene understanding task is being performed. Popular tasks would include: room layout estimation, object detection, per-point classification [1]. However, this is far from a comprehensive list. Considering this, in this chapter we primarily focus on the tasks of per-point classification and object detection of point clouds (which we refer to as classification and detection for brevity). These are chosen as they are relevant to the work presented in this thesis.

2.1 Classical Machine Learning

Whilst it would of course be attractive for classification and detection to be performed using deterministic functions, deriving a clear and robust set of rules and logic is generally not possible for complex environments. A solution to this problem is to design a framework where an algorithm can learn the complex functions from data. We goal of the machine learning algorithm is to learn a mapping function $f : \mathcal{G}^k \rightarrow Y$ where $\mathcal{G}$ is some geometric representation and Y is the desired output (e.g., positions and extents of cars in a road scene). Learning this mapping generally (in the supervised setting) requires having examples of both $\mathcal{G}$ and the corresponding outputs Y. To obtain Y manual labelling of scenes is performed. In common

[1] In this thesis we refer to the task of assigning an object class label to each point of a point cloud as *per-point classification*. In many works this is also referred to as *semantic segmentation*. Instead, we reserve the term of *semantic segmentation* for the task of per-pixel classification in images.

terminology we refer to $\mathcal{G}$ as the input and Y as the label.

2.1.1 Per-point classification

The problem of classification is to assign for every point $p \in \mathcal{P}$ a class c, indicating the class of object who's surface the point was sampled from. A key issue with this problem is that a single point itself is unlikely to contain enough information to establish which class it belongs to. To account for this problem, it is standard procedure to classify each point based on its neighbourhood U where $p \in U \subseteq \mathcal{P}$. The most common solution to obtaining U is to perform a k-nearest based on Euclidean distance (Lalonde et al., 2006; Weinmann et al., 2014). For example, when $k = 1$:

$$U = \{\min_{p_i \in \mathcal{P}} \sum_{i=1}^{n} ||p - p_i||_2^2\}. \tag{2.1}$$

Another alternative is to simply collect all points within a radius r of p (Lee, Schenk, 2002).

$$U = \{p_i \in \mathcal{P} : ||p - p_i||_2^2 < r\} \tag{2.2}$$

Both methods have advantages and disadvantages for obtaining neighbourhoods in sensed point clouds [2]. Sensed point clouds often contain varying densities of points, this is due to issues such as occlusion as well as being an artefact from angular scanners. To address this Lalonde et al. (2006); Weinmann et al. (2014) propose to estimate an optimal value of k based on a radius search. This avoids assuming a priori knowledge of the scene. It has also been shown to be effective to collect multiple neighbourhoods at varying scales to allow for multi-scale feature representations (Weinmann et al., 2015; Hackel et al., 2016).

Computing a feature descriptor d from neighbourhood U is then performed by computing a local 3D covariance matrix $\mathbf{S} \in \mathbb{R}^{3\times3}$:

[2] A sensed point cloud is one that has been captured using either an active or passive sensor in a real-world environment, as opposed to sampling from a known surface.

$$\mathbf{S}_U = \frac{1}{k}\sum_{i=1}^{k}(p_i - \hat{p})(p_i - \hat{p})^T \tag{2.3}$$

where $\hat{p} = \underset{p}{\operatorname{argmin}} \sum_{i=1}^{k} ||p_i - p||$ (Becker et al., 2017).

From $\mathbf{S}$ the three eigenvalues $\lambda_1, \lambda_2, \lambda_3$ which correspond to an orthogonal system of eigenvectors are determined. It is shown that many useful features can be computed directly from $\lambda_{1,2,3}$. For example: linearity $\frac{\lambda_1 - \lambda_2}{\lambda_1}$, planarity $\frac{\lambda_2 - \lambda_3}{\lambda_1}$, surface variance λ_3, scatter $\frac{\lambda_3}{\lambda_1}$ and omnivariance $(\lambda_1 \dot{\lambda}_2 \dot{\lambda}_3)^{\frac{1}{3}}$ (Weinmann et al., 2015; Hackel et al., 2016). Other useful features include point positions, point colours, neighbourhood density and vertical difference (Becker et al., 2017). For each point a feature descriptor d is constructed by concatenating each feature on the point channel dimension. The final mapping is therefore $f : \mathbb{R}^{n \times d} \rightarrow \mathbb{N}^{n \times 1}$, or more simply, for each point $p_i \in \mathcal{P}$ we must learn the mapping $f : d_i \rightarrow \{c \in \mathbb{N} | 0 < c \leq K\}$, where K is the total number of classes.

Once the per-point features descriptors d have been established, a wide-range of existing machine learning models can be used to learn the mapping. These include (although are not limited to) Maximum Likelihood classifiers (Lalonde et al., 2005), Support Vector Machines (Secord, Zakhor, 2007), AdaBoost (Lodha et al., 2007), Gradient Boosted Trees (Becker et al., 2017), Random Forests (Chehata et al., 2009; Weinmann et al., 2015; Becker et al., 2017) and Bayesian Discriminant Classifiers (Khoshelham, Oude Elberink, 2012). Spatial correlations between classes can also be modelled using discriminative probabilistic approaches such as Conditional Random Fields Niemeyer et al. (2012), which in turn can improve label smoothness in local areas. The performance of classical models is directly correlated to the richness of the feature descriptor. As such, the predominant line of research when using classical machine learning models is to find richer and more expressive features.

2.1.2 Object Detection

3D Object detection aims to extract from a scene $\mathcal{S}$ represented by some geometry $\mathcal{G}$, all objects O of interest. The problem is generally defined as finding the

mapping $f : \mathcal{G} \rightarrow O \in \mathbb{R}^{M \times d}$ where M is the number of objects in $\mathcal{S}$ and d is some parameterisation of each object.

Early work focused on the task of *surface matching*, where the aim is to match known surfaces in a database and sensed surfaces in a scene. This involved finding representations of local areas of point clouds which could be matched against representations of known surfaces. Representations include measuring surface curvature (Chua, Jarvis, 1996) and surface patches (Faugeras, Hebert, 1986). Johnson, Hebert (1999) present a method where spin images are collected within the scenes. To improve the efficiency and the performance principal component analysis is used to compress the spin images to make more compact and robust representations.

Another popular approach involves fitting parametric models to the point cloud. This has shown to be very effective in detecting roofs (Elberink, Vosselman, 2009; Kluckner, Bischof, 2010; Lafarge et al., 2008). In these works, object detection was used as an initial stage for the downstream task of building reconstruction. Region growing algorithms are also shown to be effective for object detection of planar surfaces (Rutzinger et al., 2009) and has subsequently been extended to detection of individual trees Rutzinger et al. (2010).

Golovinskiy et al. (2009) propose a method for multi-class object detection in 3D point clouds of urban environments. The system has four key stages: segmenting and subtracting background, generation of object proposals, object feature description and classification. As an extension to Golovinskiy et al. (2009), Velizhev et al. (2012) propose a method replacing the need for supervision by using implicit shape models. This has the advantage of requiring significantly fewer labelled data. Another advantage of this approach is that it operates on part-based representations which is more appropriate for active sensed 3D point cloud which suffers from occlusion, noise, and varying point density. Random key-points are extracted from connected components and matched to a geometric word in a dictionary. Each key-point then casts a vote for which object it belongs too. Similar approaches were taken by Knopp et al. (2010, 2011) which built on the generalised Hough transform proposed in Leibe, Schiele (2006) for 3D data. The idea of hough transform

based voting still achieves state-of-the-art results (Qi et al., 2019) in modern deep learning-based systems.

2.2 Deep Learning on Unordered Sets

Deep learning methods have been developed for many years (Rumelhart et al., 1985; LeCun et al., 1998), however, it wasn't until the 2012 ImageNet competition (Deng et al., 2009) that they started to become the mainstream direction for computer vision research, with the promising results from the AlexNet architecture (Krizhevsky et al., 2012). Vinyals et al. (2015) is an early example of applying deep learning to point sets. However, the focus of the paper did not consider point clouds as an example of a point set. Instead, the focus of these papers was founded in generic sets and not 3D scene understanding.

2.2.1 Per-point Classification

The seminal work of Qi et al. (2017a) was the first significant deep learning-based network architecture to directly consume unordered point sets for 3D scene understanding tasks. The network PointNet demonstrated state-of-the-art performance at the tasks of both object-level classification[3] and per-point classification. For object-level classification PointNet learns a spatial encoding for each point which are aggregated into a global point cloud signature. Features are generated using a multi-layer perceptron (MLP) and aggregated using a single symmetric function, max pooling, which ensures the network is permutation invariant. The network learns a set of functions that select interesting and informative key points from a subset of points, encoding this information in each layers feature vector. To extend this to per-point classification features are passed into a sub-network which concatenates aggregated global features and per-point local features.

A key limitation to PointNet is that it does not take into account spatial relations between points and therefore does not capture the local structures induced by the metric space in which the points occupy. Qi et al. (2017b) address this prob-

[3]The original authors refer to whole object classification as classification, and per-point classification as semantic segmentation. Instead, here we refer to them as *object-level classification* and *classification*

lem with PointNet++. Taking inspiration from 2D-based CNN's where inputs capture features at progressively larger scales along a multi resolution hierarchy. Point sets are partitioned into overlapping local regions by a distance metric. Features are then extracted from a progressively increasing neighbourhood size. Whereas small neighbourhoods capture fine grain local features (i.e., surface texture), large neighbourhoods capture increasingly global geometric features. The original PointNet architecture is used for feature extraction. To generate overlapping partitions a neighbourhood ball is defined where a given point is used as the centroid location and radius dependant on the hierarchical scale parameter. The *farthest point sampling* algorithm is used to select the center points, ensuring an approximately even coverage of the whole point set. The authors structure the process as three layers: the *sampling* layer defines a point set from the global point set, a *grouping* layer constructs the local region of points within the defined sample, finally the *PointNet* layer uses a mini-PointNet to extract local region features as feature vectors. When selecting points by neighbourhood size, point density plays a significant role. Unlike images, where an image is a uniform grid, point cloud density can vary across a scene, meaning uniform density cannot be assumed. PointNet++ demonstrates that unlike in 2D CNNs where small kernels are preferred, when point density is sparse, larger point samples are required for robust pattern extraction.

Since PointNet(++) there has been an influx in research operating directly on unordered point clouds. In particular there has been a significant effort to incorporate a spatial convolution operator within the network. A key example of this is SplatNet (Su et al., 2018). SPLATNet (SParse LAT-tice Network) takes inspiration from the permutohedral lattice (Adams et al., 2010) where convolutions are performed on sparse data in high dimensions. This is done efficiently by using sparse bilateral convolutional layers, which use indexing structures to apply convolutions only on occupied parts of the lattice. A key difference to PointNet++ is that max pooling layers are not used to aggregate information through multiple scales. Instead, flexible specifications of the lattice structure are designed to enable hierarchical and spatially aware feature learning. In essence, points are mapped onto

bilateral convolutional layers using a barycentric interpolation. Once convolutions are performed in this higher dimensional lattice, the results then undergo barycentric interpolation to be mapped back to the original points. These three processes are referred to as; Splat, Convolve and Slice respectively. Also unique is the ability to combine 2D features from corresponding images to strengthen shared features. This is particularly useful for applications such as photogrammetry and MLS where 3D point clouds and corresponding registered 2D images are available. Features for a corresponding 2D image is first past through a CNN feature extractor prior to a BCL layer. The BCL then maps 2D features onto a 3D lattice where after 2D and 3D features can be concatenated. The joint features are further passed through two 1×1 convolutional layers where finally a softmax layer enables point wise class probabilities.

Another key challenge in raw point cloud processing is the natural non-uniform distribution of real-world data. This can occur from occlusions, distance from sensor and sensor noise to name just a few. These characteristics mean applying a spatial convolution is very challenging. (Hermosilla et al., 2018) address this by proposing a novel method that first represents the convolution kernel as a MLP. Next the convolution is phrased as a Monte Carlo integration problem. Lastly, Poisson disk sampling is incorporated as a scalable means of hierarchical feature learning. The authors demonstrate by estimating the convolutional integral with Monte Carlo computation, with proper handling of the variance in underlying point sampling density, state-of-the-art performance can be achieved for model and point-wise classification. By implementing this approach, the network gains a level of sampling invariance, whereby the convolution becomes invariant to both the order of the points and the variable number of neighbours for each sampled point. Furthermore, the use of Poisson disk sampling for point hierarchy construction (as opposed to the more commonly used farthest point sampling using in PointNet++) demonstrates a higher level of scalability and allows to bound the maximal number of samples in a receptive field.

PointCNN (Li et al., 2018b) also uses convolutions directly on the point cloud.

Here, a k nearest neighbours is used to find spatially local points to introduce point order invariance. The network uses an MLP on the derived local point neighbourhood and learns a transformation $\mathcal{X}$ of size $k \times k$ on points $\{p_i | 0 < i \leq k\}$, which is used to weight and permute the input point features. The latter has a similar effect of PointNet's *T-Net* which attempts to rotate the point cloud into canonical order. Convolutions are subsequently applied on the $\mathcal{X}$-transformed features. PointCNN further expresses the importance for hierarchical feature representations for effective point cloud classification. Although conceptually simple, PointCNN achieves 85.1% on the ScanNet uniform benchmark dataset. In a similar attempt to address the lack of spatial convolution Thomas et al. (2019) present KPConv offering a deformable convolution operator. Each local neighbourhood is convolved by applying the weights of the nearest distance kernel point in the neighbourhood. Originally these are set uniformally, which in essence is a pseudo local voxelisation. However, the position of the kernel points are also learned in the network allowing the points to learn the topology of the local neighbourhoods and deform the voxel grid to suit the such a topology. Similar to Hermosilla et al. (2018), poission disk sampling is used to sub sample the points and a radius search is performed to gain the local neighbourhoods for pooling.

2.2.2 Object Detection

Early attempts to apply deep learning-based networks to point cloud data involved projecting the point cloud into a perspective view and applying 2D image processing techniques (e.g. CNN's) (Premebida et al., 2014; González et al., 2015; Li et al., 2016). Whilst this enables the use of mature 2D-based object detectors (Girshick, 2015; Ren et al., 2015; Redmon et al., 2016), the 2D-3D projection invariably looses information on the projection plane dimension. Most common, this is in the *up* (gravity-aligned) direction (Beltrán et al., 2018) or forward (camera-view) direction (Wu et al., 2018, 2019). Other methods first convert continuous, unordered point clouds to ordered and discreet voxel grids (Fig. 1.1) (Song, Xiao, 2014; Wang, Posner, 2015; Engelcke et al., 2017; Lahoud et al., 2019). This enables the use of the 3D counterparts of 2D CNN's, however, is memory inefficient and does not

scale to large scenes with modern computer hardware (Qi et al., 2019).

A step closer to direct 3D point cloud object detection is the use of RGB-D data (Qi et al., 2018). RGB-D data is the concatenation of RGB and a depth channel where each pixel $\mathcal{I}^{i,j}$ represents distance D along the camera forward (e.g. z) direction of point p_i in camera-space. A point cloud $\mathcal{P}$ can then be created using Eq. 2.4. Using a 2D-based detector, 3D proposals are generated as the points in the camera view frustum. PointNet Qi et al. (2017a) is then applied on the subset $S \in \mathcal{P}$ to perform a binary classification to remove foreground and background clutter. PointNet is therefore defined as $f : S \in \mathbb{R}^{3+k} \rightarrow S \in \mathbb{N} = \{0,1\}$ where $k \in \mathbb{R}^3$ corresponding to RGB value of the respective pixel. (Hou et al., 2019) also operate on a RGB-D input for instance-level object detection. The fusion of RGB and 3D geometry features are shown to be complementary to generating final bounding box proposals.

$$\mathcal{P} = (\frac{x}{f_x} \times z, \frac{y}{f_y} \times z, z) \tag{2.4}$$

where f_x and f_y is the focal length in pixels for x and y respectively.

In this section, we instead look at approaches where object detection is performed directly on point clouds. VoxelNet (Zhou, Tuzel, 2018) proposed one of the first end-to-end networks to map directly from point cloud to 3D bounding boxes. VoxelNet, defines a neighbourhood of points by dividing the input point cloud into equally spaced 3D voxels. A feature representation is then learnt for each neighbourhood, similar to the *grouping* module in Qi et al. (2017b). To handle for varying point density, and to improve model efficiency, a random sampling of points is taken for each neighbourhood (voxel). Direct 3D bounding box regression is proposed by Yang et al. (2019) which adapts a similar anchor-free regression to CenterNet (Duan et al., 2019). Using standard MLP's branches after a shared point-based encoder is shown to be very effective and is the primary motivation for the bounding box prediction architecture presented in Chapter 5.

Qi et al. (2019) build on the idea of Hough voting (Knopp et al., 2010, 2011; Leibe, Schiele, 2006), where the Hough transform is learnt through a neural network

in what they call *deep Hough voting*. A PointNet++ (Qi et al., 2017b) encoder-decoder backbone is utilised before each point votes for its own objects center. Through a clustering module, seed points that cluster together are grouped and passed through a shared object proposal and classification network to obtain 3D bounding box proposals. As with most 2D and 3D object detectors (Hou et al., 2019; Qi et al., 2019) the total number of proposals K is set such that $K \geq N$ where N is the number of ground truth objects in the scene. To obtain the final proposals Non-maximum Suppression (NMS) is applied to overlapping boxes (Girshick, 2015).

2.3 Transfer learning

Training a neural network with learnable weights requires initialisation of the weights. Typically, the weights are initialised from some random distribution (e.g., normal, uniform etc.). More advanced strategies such as Glorot (Xavier uniform) (Glorot, Bengio, 2010) and He (Kaiming uniform) (He et al., 2015) derive the initialisations a function of the dimensionality. However, in either case, no initial knowledge is present in the weights, rather, they just present a useful starting point to start the gradient descent learning process. However, another strategy is where the weights contain knowledge, derived from optimising the same model on a separate dataset. This is defined as "*transfer learning*". In this section we will briefly discuss four main techniques in transfer learning, namely, pre-training, fine-tuning, domain adaptation and knowledge distillation [4].

Pre-training is the process of training a model on a dataset which is different to the dataset used for the final model training. The previous training is generally carried out on a dataset larger than the final dataset. The most common dataset in the computer vision community is ImageNet (Deng et al., 2009), due to its large size. Pre-training can have many positive benefits such as reducing training time, increasing overall model performance and allowing for training on smaller datasets (Studer

[4]These terms are often ambiguous in literature, especially across fields e.g., statistics, computer vision, natural language processing etc.. We adopt the definition of the terms commonly (but not strictly) used in the computer vision community.

et al., 2019). However, recent experiments to statistically verify the benefits of ImageNet pre-training have resulted inconclusive results Kornblith et al. (2019); Huh et al. (2016). Furthermore, He et al. (2019) show that for semantic segmentation on the COCO (Lin et al., 2014) dataset, ImageNet pre-training does not provide any performance benefits and that learning rate scheduling is the most important factor. Whilst He et al. (2019) achieve equal performance with and without pre-training, it is shown that pre-training increases convergence speed.

Fine-tuning Fine-tuning exploits the fact that early layers of a CNN learn more generalisable features than later layers. This is because CNNs generally involve hierarchical pooling and small convolution windows, therefore, early layers typically have very small receptive fields (Simonyan, Zisserman, 2014). This restricts early layers to find local features such as edges and blobs, which are often useful for a very wide range of visual perception tasks. There are two typical approaches to fine-tuning. The first freezes the early layers of the network to preserve the early, more generalisable features. This is common when the final dataset is smaller in size, therefore, preventing the model from over-fitting the features to the new smaller dataset. The second allows all network weights to be retrained. This is common when the new dataset is itself large, or the pre-training and final datasets are likely to contain a large domain-gap.

Domain Adaptation is the most simple form of transfer learning, where no further training is carried out on a final dataset. Instead, the initialised weights are directly applied to the final datasets test data. The performance of domain adaptation is therefore strongly correlated with the domain gap between the old and new datasets.

Knowledge distillation is a method for distilling knowledge from one model to another. This does not involve weight initialising. As such, the two models can have completely different architectures. A typical example of this is the student-teacher framework. A teacher model is first trained on a task. The student network then learns to predict the teacher networks output, effectively transferring knowledge between the two models (Wang, Yoon, 2021; Cho, Hariharan, 2019). This approach is used in Chapter 5 where the loss network distils knowledge into the scene network.

2.4 Label Efficient Approaches

2.4.1 Existing Labels

For many applications, deep learning aims to automate a task which has been historically carried out by humans. A prime example of this is land classification and national scale mapping. Worldwide, mapping agencies create detailed Geographical Information System (GIS) datasets which contains georeferenced vectorised data of features ranging from major land-types (city, water, forest etc.) to local points-of-interest (statues, buildings, roads). When combined with georeferenced sensed data such as: high-resolution satellite imagery, aerial imagery, and aerial LiDAR these vectors can be used to train machine learning systems. Dollar et al. (2006) present an example where Google Maps was used to train a Boosted Edge Learning algorithm for binary road detection from satellite imagery. Despite having access to a very large, labelled dataset, this was not fully exploited. Mnih, Hinton (2010) address this by using very large vector-based road data to train a neural network to solve the same task. A key issue of using vector lines for road segmentation is that they only denote the centerline of the road, and not all the valid road pixels. To address this issues a label map $\mathcal{L}^{i,j} = \{0,1\}$ is computed as $\mathcal{L} = e - \frac{d(i,j)^2}{\sigma^2}$ where $d(i,j)$ is the Euclidean distance of a pixel and the nearest road vector position and σ is a smoothing component which is roughly corresponds to the width of a two-lane road. $\mathcal{L}$ can therefore been seen as a probability map that the corresponding pixel belongs to a vector road centerline.

Du et al. (2015) convert multi-class GIS vector maps directly into raster data which are predicted from high-resolution satellite data. Li et al. (2019) utilised multi-source GIS data (OpenStreet Map, Google Maps and Map world) to train a modern U-Net (Ronneberger et al., 2015) deep learning-based semantic segmentation network.

Audebert et al. (2017) demonstrate that OpenStreetMap data can also be used as an input for improving performance on existing datasets (i.e. (Mou, Zhu, 2016; Debes et al., 2014)). Rasterised OpenStreetMap data can act as a strong prior for pixels. The network can learn consistently correctly labelled pixels (i.e., centers of

rooftops) very quickly and focus on refining incorrectly labelled pixels (i.e. building footprint edge pixels).

Recently, Generative Adversarial Network (GAN) networks have also been used to translate from aerial images to rasterised vector maps (Isola et al., 2017). However, the goal of this work is to make perceptually coherent image-to-image translations and were not evaluated on classification image metrics. The results therefore whilst looking plausible have no guarantee of being accurate.

2.4.2 Synthetic Data

Researchers have a long history of exploiting synthetic data for generating training data for training machine learning models. In this section we will look specifically at the use of synthetic data for training 3D scene understanding models.

Creating realistic synthetic data typically requires two components: a virtual world (geometry and material) and a sensor simulator. Wu et al. (2018) achieve this by first exploiting the large world inside the video game Grand Theft Auto V. Unlike previous examples of using Grand Theft Auto V for 2D semantic segmentation (Richter et al., 2016; Johnson-Roberson et al., 2017), a virtual LiDAR scanner was built inside the game. The LiDAR was set atop an in-game vehicle which was set to drive around autonomously. Ray casting is used to simulate each ray of light, whereby the position of the first hit surface is recorded along with object class, center and bounding box. The system simulates the popular Velodyne HDL-64E LiDAR. Distributions of noise were analysed from the KITTI dataset (Geiger et al., 2013) and applied to the point clouds to further increase realism and reduce the sim2real domain gap. Despite this, their model SqueezeSeg generalised very poorly when switching to a real-world domain. This was accredited to dropout noise, defined as missing points from the sensed point cloud caused by limited sensing range, mirror diffusion of the sensing laser, or jitter in the incident angles. SqueezeSegV2 (Wu et al., 2019) proposed a domain adaption pipeline whereby dropout noise was mitigated by a Context Aggregation Module, increasing real world test accuracy.

Virtual autonomous driving environments also serve as useful resources for generating synthetic data for scene understanding. The Synthia dataset (Ros et al.,

2016) is a vehicle drive through a virtual world. The dataset contains 2D imagery but also a LiDAR inspired 2.5D registered depth map. As the primary purpose of the research is to aid semantic segmentation for a moving vehicle, a full 3D point cloud is not released with the dataset. The authors do demonstrate the added benefit of pre-training on synthetic data. The more recent CARLA simulator (Dosovitskiy et al., 2017) builds on Synthia but is released as a full simulator software with a fully integrated API. Using CARLA Deschaud (2021) created a KITTI-like dataset. As such, within the context of autonomous driving, many multiple systems have been proposed for generating data (Hanke et al., 2017; Fang et al., 2020). Furthermore, Fang et al. (2020) propose a hybrid system whereby real point clouds are augmented with synthetic objects.

To aid the generation of virtual scanners, a popular approach is to combine the scanner with standard 3D modelling software. For example, Gschwandtner et al. (2011); Reitmann et al. (2021); Wang et al. (2019a) built multi-purpose and multi-sensor LiDAR scanners inside the Blender (Community, 2018) open-source application. Both plug-ins allow the user to virtually scan any arbitrary mesh-based geometry built within blender. Reitmann et al. (2021) go further to also incorporate weather simulation noise. Furthermore, material properties are read allowing for RGB information and intensity to also be stored, along with any other useful properties such as object id, instance id and normals. A key issue with these systems is that they are typically slow for larger scenes with complex geometry. Gusmão et al. (2020) propose an approach parallelising the ray casting algorithm to improve performance. Other simulator systems outside of blender include Bechtold, Höfle (2016); Winiwarter et al. (2021).

2.4.3 Weak Supervision

The concept of label-efficient machine learning has attracted growing attention in previous years. Whereas typical dense supervised learning requires a high-quality supervision label for each predicted attribute (e.g. in image classification every image would have a label corresponding to class of the image), this is not a requirement for weakly supervised methods. Instead, noisy, limited, or imprecise supervision is

used. Here we focus on the scenario where limited labels are used (e.g., image classification where n images are used for training where $m < n$ labels are available). This has a number of advantages. Firstly, as labels are typically manually generated at the cost of human labour, the ability to train machine learning models with less labels makes the model training process more cost effective. Secondly, in many cases collecting a large quantity of labelled data is not possible. An example of this is in medical data where some phenomena only occur at rare intervals. Furthermore, this data is often burdened with strict privacy constraints. Lastly, some tasks require highly trained professionals to acquire the labels (e.g., medical and engineering applications). As there are often a shortage of such professionals, who themselves have only finite time available, scaling data annotation can be in some instances infeasible.

Specific to 3D data, there has been significantly less research in weakly supervised learning approaches, when compared to 2D images. As 2D labels are often either available, or easier to obtain, a key form of weak supervision is to adapt 2D labels for 3D tasks. (Tang, Lee, 2019) combine 2D bounding boxes of new class labels, along with transferring information from 3D bounding boxes of existing classes to train 3D object detectors on new classes. (Wilson et al., 2020) adapt off-the-shelf 2D semantic instance segmentation networks to generate 3D detection labels. This is achieved using high-definition maps and strong object size priors. Their work also concludes that whilst their labels are noisy, the deep learning-based architecture is robust to such inaccurate labels. (Wang et al., 2019b) show that 2D semantic segmentation labels can be projected onto a 3D point cloud, again demonstrating the networks robustness to noisy labels resulting from projection errors. (Sanchez Castillo et al., 2021) take this idea further and show that off-the-shelf 2D semantic segmentation predictions can be projected directly onto 3D point clouds to achieve per-point classification. Whilst the results achieved were still subject to noise, further refinement would be possible, for example with the use of Conditional Random Fields (Wu et al., 2018). A key limitation of the above methods is that they do not explicitly learn to model and account for noisy labels. (Genova et al., 2021)

propose a method in which *trusted* labels are separated from *not trusted* labels. The loss is only applied on trusted labels. Similar to (Sanchez Castillo et al., 2021) the labels also come from an off-the-shelf 2D CNN network.

The above approaches all deal with exploiting lower-dimensional labels from the 2D domain and adapting them to the 3D. In Chapter 5 we instead take a different approach. Specifically, we look to exploit sparse manual 3D labels. Similar to our approach, Meng et al. (2020) split their detection pipeline into two stages. By using only, a single horizontal (x, y plane) manually annotated coordinate the first network learns to generate cylindrical object proposals. The second stage learns to refine the proposals into cuboids and confidence scores, using only a few labelled instances. This allows for only a small fraction of scenes to be labelled, and furthermore, within those scenes only a small fraction of instances to be fully annotated.

Concurrently to our work Hou et al. (2021) show that with contrastive pre-training only 0.1% of available points are required to be labelled to perform per-point and instance classification on the ScanNet dataset. Key to the paper was addressing limitation of PointContrast (Xie et al., 2020) whereby only point-level correspondence matching is performed as a pretext task. The authors argue this disregards spatial configurations and contexts in a scene. Instead, the scene is partitioned into multiple regions and contrastive learning is applied to each region separately. Through improved pre-training state-of-the-art results can be achieved with only 20 labelled points per a scene.

To reduce the manual labelling burden even further, Ren et al. (2021) introduce a the WyPR pipeline which only requires a list of scene level tags (i.e., the objects present in the scene). This is the first approach to remove the requirement of any spatial labels for 3D point cloud detection. This is achieved with an elaborate series of self and cross-task consistency losses and mulitple-instance learning objectives.

2.4.4 Analysis-by-Synthesis

An alternative approach to removing the requirement of labels is to adopt an "Analysis-by-Synthesis" workflow. Broadly speaking analysis-by-synthesis aims to analyse a signal by reproducing it using a model. That is given some function

f or an input signal x and a model g, a successful analysis-by-synthesis requires $x \approx g(f(x))$. Specifically, in 2D / 3D computer vision this can have a number of applications. Firstly, if f is a dimension reducing encoder and g a decoder a simple cost function L can be defined as $L = ||g(f(x)) - x||_2^2$. This setup is known as an Auto-Encoder and can be used for learning useful representations without labels or as a lossy compression algorithm (Wang et al., 2016).

When dealing with images representing a world environment the decoder g can be replaced with the use of a physically based renderer R. However, to enable gradient-based learning, it is required that the renderer R be a Differentiable Renderer Differentiable Renderer (DR) [5]. In recent years there have been a number of DR's proposed (Liu et al., 2019; Loper, Black, 2014; Li et al., 2018a; Chen et al., 2019). An in-depth review on the progress in this field is presented by Kato et al. (2020). Assuming g is a DR, the problem is now constrained for the encoder f to derive the configuration required for g such that it accurately reproduces the input signal x. This enables a number of key applications such as predicting 3d objects from a single image (Chen et al., 2019), estimating exterior and interior camera parameters (Li et al., 2018a), 6-DoF object detection and scene level object detection (Beker et al., 2020).

In Chapter 5 we look at the scenario where f learns to predict the parameters of objects in the scenes, along with their corresponding attributes. (Kundu et al., 2018) exploit 2D object detectors to first locate objects in the 2D plane. Learnt shape-priors from CAD databases are subsequently fitted to the regions minimising a reconstruction loss. This procedure is carried out for each detected object in the scene. The result of the optimisation is object orientation and shape. Zakharov et al. (2020) follow a similar approach using continuous Signed Distance Function (SDF) (Park et al., 2019a) priors along with normalised object coordinates (Wang et al., 2019c) to fit the shape priors. To enable analysis-by-synthesis the SDF need to be rendered. Subsequently, a novel SDF renderer is proposed. Unlike Kundu et al. (2018), the authors also exploit sparse LiDAR. The SDF is then aligned to

[5] This is sometimes referred to as "render-and-compare" in some literature.

the sparse LiDAR using a RANSAC algorithm. Beker et al. (2020) relax the sparse LiDAR requirement using an off-the-shelf monocular depth estimation algorithm (Guizilini et al., 2020). The method still relies on 2D instance segmentation masks to locate 3D objects within the scene, similar to Zakharov et al. (2020). All these methods demonstrate that with 2D instance labels (either detection or segmentation) and a DR, the collection of 3D labels can be avoided.

An alternative use of analysis-by-synthesis is to in synthetic data generation. By learning a generative model which can match a target distribution (i.e., the world domain), it is possible to generate labelled data which can be used to train a detector. Mustikovela et al. (2021) achieve this with the use of a controllable GAN. The generative model given specific input parameters such as the position and orientation of the target objects, synthesise realistic scenes respecting the input parameters. If CAD data is available, Devaranjan et al. (2020) successfully employ a procedural model which given a specific scene structure generates high quality labelled synthetic data. Whilst these approaches are promising directions, their performance depends heavily on the domain gap between the synthetic and real data. As such, in scenarios where the target domain is very complex, these approaches may be infeasible.

2.4.5 Self-supervision

Self-supervised pre-training of neural networks has experienced a lot of recent success for 2D image tasks. State-of-the-art representation learning methods (Grill et al., 2020; Goyal et al., 2021; Caron et al., 2020) have demonstrated an ability to learn representations that are en-par with those learnt from fully-supervised methods. Despite such advances, only recently has 3D data achieved similar attention with regards to self-supervised pre-training for deep neural networks. A key benefit of self-supervised pre-training is that rich feature representations can be learnt without any labels, allowing for task/domain specific training to be performed with significantly less labelled data. Achieving similar performance in the 3D scene understanding domain would therefore make a significant contribution to label efficient learning.

Perhaps the first paper to fully demonstrate the power of self-supervised pre-training for 3D scene understanding is PointContrast (Xie et al., 2020). Whereas previous self-supervised tasks in the 3D domain were previously restricted to single object (Achlioptas et al., 2018; Yang et al., 2018) PointContrast operates on complex scenes (ScanNet, S3DIS, SUN RGB-D etc.). A Fully Convolutional Geometric Features network backbone G_θ (Choy et al., 2019) is used to learn both global and local descriptors. As a pre-text task, a point cloud $\mathcal{P}$ is augmented into 2 views $\mathcal{P}^1$ and $\mathcal{P}^2$. The correspondence mapping M is then computed. Next, two tranformations T^1 and T^2 and sampled. Features f^1 and f^2 are then computed for $\mathcal{P}^1$ and $\mathcal{P}^2$ respectively. Finally, the loss is

$$L(G_\theta(T^1(\mathcal{P}^1)), G_\theta(T^2(\mathcal{P}^2))). \tag{2.5}$$

Similar conclusions were are reached by Zhang et al. (2021c) who extended 3D pre-trained with pretext tasks to single-view depth scans without 3D registrations and point correspondences. Their DepthContrast pipeline, again demonstrates the benefits of not training 3D scene understanding networks from scratch. The significance of PointContrast and DepthContrast imply that 3D pre-training will become increasingly used within the 3D scene understanding community.

2.5 Datasets

As with other fields of machine learning, the availability to high quality datasets is essential for algorithm development and comparison. We outline the most commonly used indoor and outdoor datasets in Tbl. 2.1 and Tbl. 2.2 respectively. In general, most datasets focusing on 3D scene understanding can be categorised as indoor hand-held scanners (e.g., RGB-D sensors such as the Microsoft Kinect or Apple iPhone), static outdoor scans (e.g., with a TLS), mobile outdoor scans (e.g. with a MLS) or autonomous driving datasets. Autonomous driving datasets are unique in that they are always focused on road scenes and are generally captured with a Velodyne scanner which is typically much more sparse and low quality than a typical TLS or MLS. Whilst we do not present by any means an exhaustive list of

available datasets, the datasets presented in Tbl. 2.1 and Tbl. 2.2 are the datasets that are most commonly used as benchmarks in publications. More specifically, ScanNet, Semantic3D and KITTI receive the largest popularity for indoor, outdoor and autonomous driving datasets respectively.

Table 2.1: Commonly used **indoor** datasets for 3D scene understanding model training and evaluation. We assign the following keys for tasks: D = Object detection, C = Per-point classification and P = Object pose. Whereas each dataset do contain additional tasks (e.g. trajectory estimation, 2D segmentation), we do not record them here unless they are relevant to 3D scene understanding.

Dataset	Sensor	Task	No. Scenes
ScanNet (Dai et al., 2017)	RGB, Depth	D, C	1500
S3DIS (Armeni et al., 2016)	RGB-D	D, C	6 (floors)
Apple ARKit (Baruch et al., 2021)	iPhone, TLS	D, C	5k
NYUDv2 (Silberman et al., 2012)	RGB-D	C	464
SUN3D (Xiao et al., 2013)	RGB-D	D, C, P	254
SUN RGB-D (Song et al., 2015)	RGB-D	D, C	63
Matterport3D (Chang et al., 2017)	RGB-D	D, C	90 (buildings)

Table 2.2: Commonly used **outdoor** datasets for 3D scene understanding model training and evaluation. We assign the following keys for tasks: D = Object detection, C = Per-point classification and P = Object pose. Whereas each dataset does contain additional tasks (e.g., trajectory estimation, 2D segmentation), we do not record them here unless they are relevant to 3D scene understanding. Where No. Points is preceded with $\approx$, the value is computed as the average number of points per frame multiplied by the number of frames.

Dataset	Sensor	Task	No. Points
Semantic3D (Hackel et al., 2017)	TLS	C	4B
Oakland (Xiong et al., 2011)	MLS	C	1.6M
Sydney Urban Objects (Quadros et al., 2012)	Velodyne	C	2.3M
Paris-rue-Madame (Serna et al., 2014)	MLS	C	20M
iQmulus/TerraMobilita (Vallet et al., 2015)	MLS	C	300M
TUM City Campus (Zhu et al., 2020)	Velodyne	C	1.7B
Paris-Lille-3D (Roynard et al., 2018b)	Velodyne	C	143M
KITTI (Geiger et al., 2013)	RGB, Velodyne	D, C, P	$\approx$ 2.6M
Waymo (Sun et al., 2020)	RGB, Velodyne	D, C, P	$\approx$ 203M
NuScenes (Caesar et al., 2020)	RGB, Velodyne	D, C, P	$\approx$34M
A2D2 (Geyer et al., 2020)	RGB, Velodyne	D, C, P	$\approx$1.2B
DublinCity (Zolanvari et al., 2019)	ALS	C	$\approx$260M
H3D (Kölle et al., 2021)	ALS + RGB[1]	C	73.9M

[1]Captured using Unmanned Aerial Vehicle and delivered as a photogrammetric point cloud and mesh. The dataset is also provided in 3 epochs to accommodate change detection challenges.

Chapter 3

Reapplication of Existing Data as Labels

3.1 Introduction

Reliable automatic building segmentation and mapping from LiDAR and has long been sought for a range of applications. These include urban planning, disaster management, city modelling, national mapping, and population management. Despite rapid technological advances, highly accurate solutions that function over large areas (i.e., entire countries) and land types (i.e., urban, rural etc.) remain unseen. There are many reasons for this including the heterogeneous nature of both the geometry and spectral properties of buildings, unpredictable scene complexity and the loss of relevant sensor data (i.e. occlusion (Awrangjeb et al., 2010)). Furthermore, with increasingly diverse architectural designs, deriving a general solution is arguably becoming more complex. As a result of this, the research problem of detecting buildings has been extensively studied over a number of years using a range of sensing technologies such as; satellite imagery (Saeedi, Zwick, 2008), aerial imagery (Sirmacek, Unsalan, 2008) and Airborne Lidar Scanning (ALS) [1] (Vosselman, 2000; Kraus, Pfeifer, 2001; Akel et al., 2004).

A key ingredient for any supervised deep learning approach is training data. Publicly available remote-sensing and GIS data exists in large quantities in many

[1] We refer to aerial LiDAR data as Airborne Lidar Scanning in this chapter.

countries, with the data usually being made available by national mapping agencies. The fine-grained quality of open national datasets is not usually high with many errors often being present within the dataset. These errors tend to be caused by missed features, over-generalisation, and in-accurately placed boundaries. Furthermore, whilst best efforts can be made to match datasets, there will likely be temporal discrepancies between aerial imagery, ALS and GIS vector data. Despite this, data collection can be undertaken automatically and scales arbitrarily. The largest limitation regarding open access GIS data is the in-ability to provide high quality segmentation ground-truth data. However, such vector data does provide a high-quality object detection dataset, as object detection is only concerned with bounding box coordinates.

In this chapter we develop a pipeline for refining course public GIS for per-point classification of ALS data. The result is per-point classification of ALS data without the requirement for any additional labels, and instead using already collected publicly available data. We achieve this through first proposing a novel methodology for refining course GIS data with active contour snakes. Next 3D point cloud data is projected to 2D raster maps where it is fused with aerial RGB data. Finally, we exploit off-the-shelf semantic segmentation and object detection networks to train our system.

3.2 Problem Statement

Given a point cloud $\mathcal{P} \in \mathbb{R}^3$ we want to assign a binary class $k = \mathbb{N} \in \{0,1\}$ denoting whether point $p_i \in \mathcal{P}$ represents a point on the surface of a building. We achieve this by first projecting $\mathcal{P} \rightarrow \mathcal{I} \in \mathbb{R}^{i,j}$ on the ground-plane such that i and j represent north and east coordinates in a Cartesian coordinate system respectively. We therefore look to find compute the mapping function $f : \mathcal{I} \rightarrow \mathbb{N} \in \{0,1\}$.

3.3 Methodology

3.3.1 Data Collection

Data collection was performed by an automated script which allowed for large data volumes to be collected and processed in a reasonably short amount of time ($\approx$ 4 hours on a standard desktop computer). All data acquired was aerial information over England, UK. RGB imagery with a ground sampling distance of 25cm is first obtained from the Ordnance Survey (OS). Next, pre-computed Digital Surface Model (DSM) maps from 50cm airborne lidar are obtained from the UK Environment Agency and re-sampled to 25cm. Finally, OS OpenMap local data is downloaded over the area. Building footprint shapefiles are extracted and converted to raster format at 25cm resolution. This information is downloaded by 10km^2 OS Grid-Reference tiles. Any areas missing lidar data are removed. The data is then merged into RGB-D images and label data respectively and cropped to 250m^2 tiles. As the OS tiles cover varying land topographies it was also necessary to normalise the depth channel between 0-255 for each image tile. This ensured the model would learn the contextual relationship of the building geometry in respect to its local area, and the absolute height (above a datum) is not considered. As a result, a more consistent channel input is realised, with building roofs typically having pixel values between 150-255 depending on surrounding buildings and presence of high vegetation.

Whilst saving each label tile the bounding box coordinates for each building instance were extracted and saved to `.xml` file in the Pascal VOC (Everingham et al., 2010) dataset format. This was computed by running a binary image border extraction algorithm (Suzuki, be, 1985) to obtain border locations and computing the bounding area. Nine tiles were collected covering a wide diversity of land cover. Tiles covering rural areas were susceptible to containing large areas where no buildings were present. This can cause an extreme foreground-background class imbalance during training as large areas in rural tiles will contain no buildings, effectively saturating the CNN with easy to detect negative examples. This has been shown to cause issues in training and be responsible for decreasing accuracy in object detec-

Table 3.1: UK data collected for model training and testing using a automated script. Tile names correspond to OS area codes for which the images are geographically located.

OS Tile	No. Images	No. Building Instances
TL50	1695	6110
SZ10	4060	38898
TF03	678	2604
TQ24	3450	19296
TQ39	7420	53422
TQ58	4987	42512
TQ67	2266	6993
Total	**24556**	**169835**

tors (Lin et al., 2017). To account for this, any tiles with no buildings (w.r.t OS OpenMapLocal labels) is consequently deleted and not included in the model training/evaluation. The final dataset consisted of 24,556 images containing 169,835 instances of buildings (Tbl. 3.1).

We evaluate our model on the ISPRS Potsdam [2] semantic labelling benchmark datasets. The Potsdam dataset contains similar RGB and lidar data to the data collected via the methods above, however, are at a higher resolution (5cm and 9cm respectively). The Potsdam dataset is not included within the main model training dataset. Evaluation therefore measures the model's ability to generalise. To quantify this, we also fine-tune the OS data derived trained model with a sub-sample of the Potsdam data separately and re-evaluate the model performance.

Once the data is stored in RGB-D images we further refine our image to RGB, RG-D_E and $_N$ channels. Where the depth channel 'D' is the edge magnitude and normalised DSM layers respectively.

3.3.2 Morphological Geodesic Active contours

The frequent use of Active Contour Models (ACM) has been consistent in computer vision applications (i.e. shape recognition, object tracking and segmentation) since its inception (Kass et al., 1988). The Geodesic Active Contours (GAC) (Caselles et al., 1997) advanced on prior work by proposing a robust technique in which ACM's evolve in time according to intrinsic geometric measures of an image. In

[2] http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html

general, a ACM can be defined as an energy-minimising two-dimensional spline curve of points $p_i = (x_i, y_i)$ for $i = 1, 2, \ldots, n$ where x_i and y_i are the x and y coordinates respectively and n is the number of points. To describe a classical energy base snake (Kass et al., 1988), let $C(q)$:$[0,1] \rightarrow \mathbf{R}^2$ be parameterised planar curve and let $I : [0,a]x[0,b] \rightarrow \mathbf{R}^+$ be a given area with edges which we want to segment, the energy of curve C is given by:

$$E(C) = \alpha \int_0^1 |C'(q)^2 dq + \beta \int_0^1 |C''(q)|^2 dq - \lambda \int_0^1 |\nabla I(C(q))| dq \qquad (3.1)$$

where α, β and λ are real positive constants. Here the first two terms are responsible for controlling the smoothness of the contours, and the third for attracting the contour towards the edges in the image (external energy). Therefore, solving the ACM problem amounts to finding, for a given set of constants α, β, and λ and the curve C that minimises E.

GACs differ to classic parametric ACMs in their ability to naturally handle changes in the topology of the curve, as well as not relying on the parameterisation of the contour. In GACs the engery function is a geodesic in a Riemannian manifold with a metric induced by image features (i.e., target borders). The GAC further incorporates methods learned from euclidean curve shortening and level sets. The energy minimisation is achieved by solving Partial Differential Equation (PDE) on an embedding function that has the contour as its zero-level set. The PDE for GACs is defined as:

$$\frac{\delta u}{\delta t} = g(I)|\nabla u| \mathrm{div}(\frac{\nabla u}{|\nabla u|}) + g(I)|\nabla u|\nu + \nabla g(I)\nabla u \qquad (3.2)$$

where $g : [0, +\infty[\leftarrow \mathbf{R}^+]]$ is a strictly decreasing function such that $g(r) \leftarrow 0$ as $r \leftarrow \infty$ and u and ν are signed distance functions. The first term is the smoothing term, the second is the balloon term and the third is the image attachment term.

The GAC also employs the concept of a balloon force as first proposed by Cohen

(1991). The balloon factor makes the curve behave more like a balloon being inflated by an additional force. This ensures the contour is not stopped on weak edges as well as allowing the contour to hold a degree of integrity to its current shape. This is particularly beneficial in our case as the lidar data used is not very high quality (50cm). The coarseness of the data causes apparent gaps in strong edges caused by partial hits of sharp geometric edges of buildings. The result is a strong edge with intermittent low intensity pixels. The balloon factor therefore makes the contour mostly invariant to these small gaps (Figure 3.1).

The most recent advancement in ACM's is the proposal of a morphological approach (Márquez-Neila et al., 2014) which can be used to enhance GAC's as well as Active contours with edges (Chan, Vese, 2001). Morphological ACMs work similarly to classical ACM's, however, instead of solving PDEs and level-sets over a floating-point array, morphological operators (i.e., dilation and erosion) are used over a binary array. The morphological ACM is therefore approximating the PDEs solutions making the model faster and numerically more stable. In this paper we use an implementation of a morphological approach on a GAC (MorphGAC). The approach utilises the ability to express some morphological operator as PDEs. For example, a dilation D_h and erosion E_h with radius h of function u can be defined as:

$$D_h u(x) = \sup_{y \in hB(0,1)} u(x+y) \tag{3.3}$$

$$E_h u(x) = \inf_{y \in hB(0,1)} u(x+y) \tag{3.4}$$

where $B(0,1)$ is a ball of radius 1 centered at 0 and hB is the set B scaled by h so that $hB = \{hx : x \in B\}$.

The dilation D_h can be used to show:

$$\lim_{h \to 0^+} \frac{D_h u - u}{h} = |\nabla u| \tag{3.5}$$

Therefore, the successive application of D_h with very small radius, $\lim_{m\to\infty}(D_{t/m})^m u_0$, is equivalent to:

$$\frac{\delta u}{\delta t} = |\nabla u| \tag{3.6}$$

with initial value $u(0,x) = u_0(x)$. Thus, we can say the dilation has *infinitesimal behaviour* equivalent to the PDE.

This can also be demonstrated for the erosion (E_h) function where:

$$\lim_{h \to 0^+} \frac{E_h u - u}{h} = |\nabla u| = \frac{\delta u}{\delta t} \tag{3.7}$$

Whereas morphological active contours without edges works well without defined boundaries, MorphGAC require a strong edge. This method is chosen here as the sharp geometric shape of buildings on a DSM produce a pronounced edge of gradient change. To further exemplify the edge, we take an approximation of the differential of the DSM channel on the tile to be processed. This is achieved by using the common sobel operator (Kanopoulos et al., 1988) in both horizontal (x) and vertical (y) directions. This, in essence, gives us the gradient magnitude, therefore, sharp changes in elevation have strong responses and vice versa. This offers the favourable conditions for our MorphGAC to grow given an optimum starting (seed) point. To compute the seed point, the moments are computed for each Open-

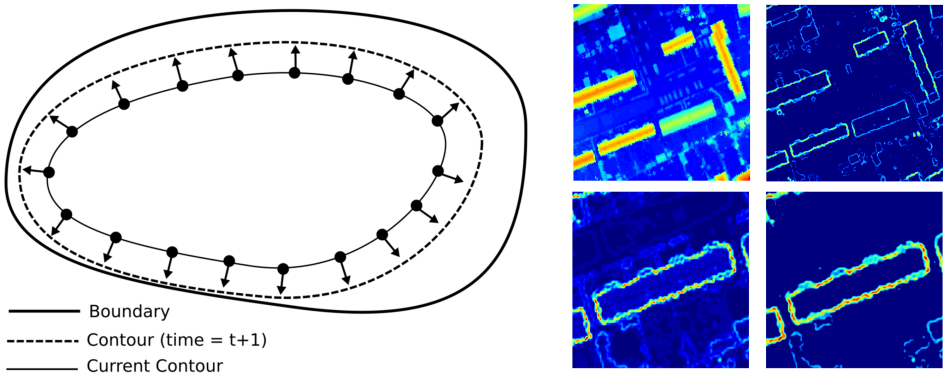


Figure 3.1: A) Typical example of an ACM expanding towards a boundary. B) i) DSM raster channel in image after normalisation between 0-255, ii) Edge magnitude of DSM after data cleaning and pre-processing, iii) Individual building prior to pre-processing, iv) Individual building post pre-processing.

MapLocal building polygon and the centre point which lies within the polygon is extracted. To clean the data, first a simple binary mask is applied with a threshold value of 20. This removes any noise on top of the building roof. Finally, a Gaussian blur is applied to the tile and a single closing morphological operator is run over the image to limit the number of gaps within the building boundaries.

However, one of the largest issues with large-scale manually labelled building footprints is over-generalisation over smaller buildings. To correct for this, before the seed point is determined for the building footprint, a multiple building check is performed. This is achieved by first masking the DSM with the footprint label. Next a k-means where $k = 5$ is computed on the image histogram. A threshold value is then located where $k = 3$ and any point with a pixel value x where $x > (k = 3)$ is defined as above ground. Individual buildings are then detected with the Douglas-Peucker algorithm. The seed points are then computed for each individual building using the contour moments.

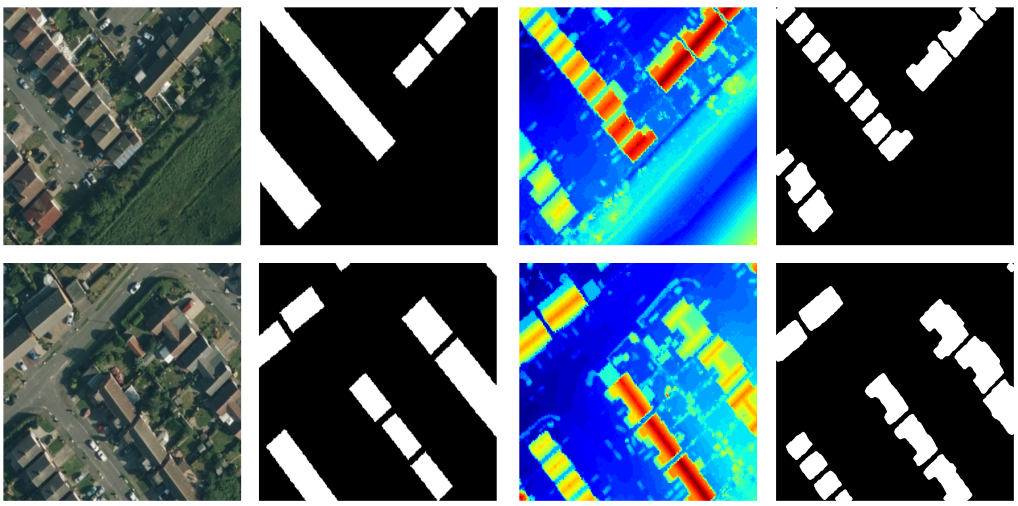


Figure 3.2: Delineation of multiple buildings from over-generalised labels. An otsu binarisation threshold is utilised over each label to determine if the label has been over-generalised. If this is the case multiple geomorphological snake seeds points are extracted for the center of each individual building.

3.3.3 Models

Broadly speaking, CNN-based object detectors fall into one of two categories: one-stage and two-stage. The two-stage approach was popularised by (Girshick et al., 2014) and, work by firstly generating a regional proposal for potential bounding box locations, and secondly, classifying each region proposal candidate using a classification CNN (e.g. (He et al., 2016; Simonyan, Zisserman, 2014)). The one-stage approach was popularised by Sermanet et al. (2013); Liu et al. (2016); Redmon et al. (2016) and motivated by the potential to speed up the two-stage process which has many speed limitations such as their inability to parallelise well. Instead, the one-stage detector applies a dense sample of classifications over the image at various scales and aspect ratios. The classifications with the highest probabilities of containing a given object are considered the objects bounding box. This computationally cheaper method has allowed for one-stage object detectors to be deployed on consumer hardware (i.e., mobile phones) for real-time detection, however, usually achieve poorer accuracy than their two-stage counterparts. We consider two model architectures, RetinaNet (Lin et al., 2017) (one-stage) and Faster R-CNN (two-stage) for our object detection backbone.

For all processing scenarios we use pre-trained weights to initialise our net-

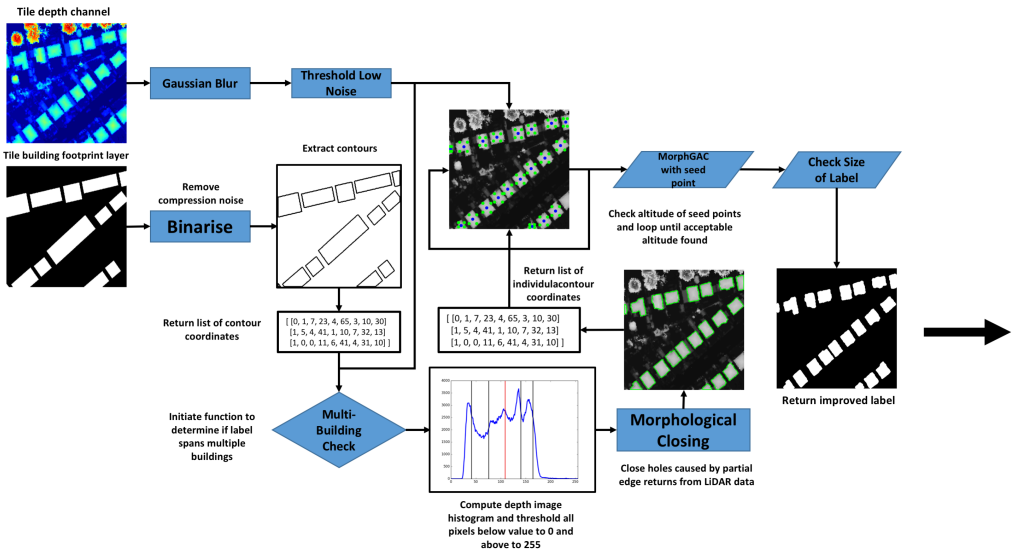


Figure 3.3: Workflow for improving building footprint labels.

works. Firstly, ImageNet is used to pre-train the classification backbones of each detection system. Next, the entire object detection system is pre-trained on the Dataset for Object detection in Aerial images dataset (Xia et al., 2018), which consisted of 2,806 aerial images with 188,282 manually labelled instances.

Initially, adaptations were made to both RetinaNet and Faster-RCNN to allow for 4 channel inputs (RGB-Depth), however in both cases this substantially reduced the networks performance. We therefore opted to solely train on 3 channel models. This also enables existing architectures to be used much more easily without the need for network alteration. The blue channel was removed as it strongly correlated to both the red and green channels and therefore contains a large amount of redundant information. This is further justified as over both urban and rural environments in the UK blue is likely to be the least dominant and informative channel. Although, the third (blue) channel weights had no relevance to depth, we found that initialising the model with RGB pre-trained weights improved accuracy for RG-D datasets. This is likely due to low-level features being generally concerned with local maximas/minimas, blobs and edges which is relevant for the detection of sharp geometric boundaries in the depth channel. We therefore find that the pre-trained weights generalise to the depth channel for the higher layers of the network, which are often accredited to requiring the largest amount of training data. Internal comparisons were made against models initialised with random weights for all channels. Random weights were sampled around a truncated (He) normal distribution centred on 0 with $\sigma = \sqrt{\frac{2}{\eta}}$ where η is the number of input units in the weights tensor (4 in this instance) (He et al., 2015).

To achieve semantic segmentation of the RG-D image we evaluate two approaches. The first is an end-to-end pipeline Mask-RCNN which builds directly on Faster R-CNN by adding a third output branch for each candidate object which predicts a binary mask $\mathcal{M}^{i,j} \in \mathbb{R}^{i,j} = [0,1]$ indicating which pixels inside the bounding box belong to the object.

As no such extension was readily available during our experiments, semantic segmentation is performed using GAC as described in Sec. 3.3.2. The centers of

the bounding boxes predicted by RetinaNet are used as seed points for the GAC to produce the final mask $\mathcal{M}$. Since, our experiments several semantic segmentation pipelines using the focal loss from RetinaNet have been proposed which could enable an end-to-end semantic segmentation pipeline with RetinaNet (Yu et al., 2020; Jaeger et al., 2020; Alon et al., 2019).

3.3.4 Model Evaluation

For segmentation scenarios the model performance is evaluated by computing the precision, recall and F_1 accuracy for each processing scenario. We define each metric as:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} \tag{3.8}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{3.9}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{3.10}$$

$$\text{F}_1 = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \tag{3.11}$$

To evaluate the object detection accuracy of the models during training, we calculate the mean Average Precision (mAP), where a positive is defined with having an Intersection over Union (IoU) with the ground truth box > 0.5. The mAP is computed as the average of the maximum precision at 11 recall vales ($r \in 0.0, 0.1, \ldots, 1.0$). The mAP can therefore be defined as:

$$mAP = \frac{1}{11} \sum_{r \in 0.0, 0.1, \ldots, 1.0} AP_r \tag{3.12}$$

3.4 Results

3.4.1 Mask R-CNN

Mask R-CNN networks are first trained using course unrefined public labels. The process was repeated for RGB, RG-D_E and RG-D_N channel images (Tbl. 3.2). The most prominent disparity exists between models that have been trained with a depth channel. All scenarios perform well ($> 90\%$) under the segmentation accuracy metric, however, the F_1 is a more representative metric for model performance. One reason for this is that the test dataset contained an average of 15% building pixels. The incorporation of depth data resulted in a 25% and 31% increase in F_1 value for RG-D_E and RG-D_N respectively. Precision values demonstrated little variability relative to recall values. This suggests that the incorporation of depth data was most prominent in resolving false negatives. Suggesting under-segmentation was a key issue with inference when the depth data was not present. Improvements were also seen in Faster R-CNN mAP scores, this demonstrates the Faster R-CNN object detection branch also benefited from depth data inclusion.

An overall improvement is noticed with the replacement of public labels for improved labels (Sec. 3.3.2) (Tbl. 3.2). In contrast to the inclusion of depth data, improved labels had a greater impact on the precision value. This therefore suggests the greatest progress was in the resolution of false positives. As one of the key issues with public building footprints is over-generalisation and thus, over-segmentation, this is unsurprising. The improvement of recall resulted in a 21%, 14% and 9% increase against their public label counterparts for RGB, RG-D_E and RG-D_N respectively.

The difference between RG-D_E RG-D_N is minimal, with mAP and F_1 scores being almost identical. However, we note training time was typically less for RG-D_N. Suggesting the network is capable of fitting to both depth channels, likely due to the most significant information being located at the building edge where a sharp change in altitude is represented by a strong edge.

The trained models then applied to the Potsdam benchmark dataset (Table 3.3). The results observed on the Potsdam dataset are not generally consistent with the

Table 3.2: Mask R-CNN segmentation results. mAP is calculated for recall > 0.5 with true positives defined as IoU> 0.5.

Model	mAP	Precision	Recall	Accuracy (%)	F_1 Value
RGB OS labels	0.45	0.73	0.61	91.33	0.61
RG-D_E OS labels	0.54	0.76	0.79	94.24	0.76
RG-D_N OS labels	0.51	0.76	0.87	95.27	0.80
RGB labels+	0.68	0.82	0.72	93.97	0.74
RG-D_E labels+	0.80	**0.89**	0.88	97.09	**0.87**
RG-D_N labels+	**0.81**	0.88	**0.90**	**97.11**	**0.87**

prior results on the main training dataset (Table 3.2). In public label scenarios the most prominent statistic is the reduction of recall value. This is a result from an increase in false negatives. As the reduction in recall is generally caused by under-segmentation of a feature, this explains the steep increase precision values. Buildings in the Potsdam dataset on average covered 22% of the image. This explains that whilst the accuracy values are respectfully high, the performance based on the F_1 metric is very poor. The RG-D_N did improve the performance somewhat, however, poor recall values suggest the results are unreliable and unrepresentative of the overall performance.

Models trained with the improved labels demonstrated a noticeable improvement for training sets contained depth data. This most represented by the increase in recall values by 130% and 25% for RG-D_N and RG-D_E respectively. The absence of improvement in the RGB image sets suggests that the improved labels had the largest impact on the depth channel in the network training. Despite Potsdam being noticeably different to the training dataset in terms of building architecture and radiometric properties, the general model with the inclusion of depth performs as well on the Potsdam dataset as the RGB model trained with improved labels on it's standard test set. The significance of this is two-fold. Not only does this show that the model has learned what a building is on a general level, but also, exemplifies the significance of incorporating geometric data into the network.

Lastly, the weights from their respective models trained using the improved labels are used as pre-trained weights for fine-tuning on the Potsdam dataset. This causes significant improvements to all performance measures, with F_1 scores rang-

Table 3.3: Potsdam Mask R-CNN segmentation results.

Model	mAP	Precision	Recall	Accuracy (%)	F_1 Value
RGB OSLabel Generic	0.30	0.80	0.44	81.16	0.53
RG-D_E OSLabel Generic	0.34	0.88	0.30	77.42	0.41
RG-D_N OSLabel Generic	0.36	0.84	0.55	89.98	0.66
RGB Label+ Generic	0.41	0.96	0.32	83.57	0.55
RG-D_E Label+ Generic	0.48	0.79	0.69	86.98	0.73
RG-D_N Label+ Generic	0.48	0.87	0.69	88.01	0.75
RGB Fine-Tune	0.71	0.91	0.91	95.30	0.91
RG-D_E Fine-Tune	0.78	0.92	0.92	96.03	0.92
RG-D_N Fine-Tune	**0.79**	**0.93**	**0.93**	**96.09**	**0.93**

ing from 0.91 to 0.93. In all instances precision and recall were equal suggesting a stable model. Here the test dataset comprised of 20% of the total data. This contrasts with the generic model tests as these could be tested against 100% of the dataset. Due to the size and homogeneous nature of the Potsdam dataset it is not possible to confidently confirm whether the model has simply over-fit to the data or indeed the solution is more general.

3.4.2 RetinaNet

Experiments undertaken with the RetinaNet architecture followed similar patterns to those observed with the Faster R-CNN architecture. However, the models trained using RetinaNet appear to have performed to a higher standard w.r.t the evaluation metrics on the main aerial dataset (Table 3.4). Models trained using public datasets obtain substantially higher mAP scores, with RGB, RG-D_E and RG-D_N achieving 17%, 24% and 31% increases respectively over their Faster R-CNN counterparts.

Whilst the final per-pixel segmentation process differs between the two network pipelines many similarities are observed. In line with Mask R-CNN, the presence of a depth channel offered little improvement to precision, however, strongly influenced the recall values. Here, the recall values observed are 0.78, 0.89 and 0.92 for RGB, RG-D_E and RG-D_N respectively. This has resulted in a general overall performance increase in F_1 score. The improvement does however come at a cost of computation time. Despite Morphological GACs performing substantially faster than non-morphological GACs they are still comparatively slower than a typical CNN based inference. Whereas a single inference takes 0.5 seconds, a single

image containing 10 buildings takes 30 seconds [3]. By re-implementing the Morphological GAC to process on the GPU computation times could likely be largely decreased.

The use of the improved labels for model training, as with Mask R-CNN, had a significant impact on performance during network training. This resulted in mAP values to reach 0.83, 0.9 and 0.92 for RGB, RG-D_E and RG-D_N respectively. Suggesting the model robustly detects buildings, with almost no buildings not being identified on the test set inference. This resulted in improved recall values for all scenarios. Such improvements suggest mostly false negatives have been resolved indicating less building have been missed, as opposed to false building detection's being resolved. In contrast to Faster R-CNN, RetinaNet learns well when depth information isn't present as well as when it is. This is justified by the final RGB F_1 value exceeding RG-D_E and equalling RG-D_N.

Table 3.4: RetinaNet segmentation results. mAP is calculated for recall > 0.5 with true positives defined as IoU> 0.5.

Model	mAP	Precision	Recall	Accuracy (%)	F_1 Value
RGB OS labels	0.53	0.84	0.78	95.12	0.81
RG-D_E OS labels	0.67	0.84	0.89	96.35	0.86
RG-D_N OS labels	0.67	0.86	0.92	96.82	0.89
RGB labels+	0.83	0.96	0.94	98.42	**0.94**
RG-D_E labels+	0.90	0.90	0.91	97.41	0.88
RG-D_N labels+	**0.92**	**0.95**	**0.95**	**98.62**	**0.94**

The increase performance in mAP over Faster R-CNN in the main building dataset was also present with the Potsdam dataset. This strengthens the evidence that the RetinaNet as an object detector had learned the characteristics of buildings more effectively (Table 3.5). This further indicates the potential benefits of the focal loss algorithm, and its relevance in single-class detection systems. Despite the advantages seen during object detection, over the Potsdam dataset the use of Morphological GAC's as an online segmentation method was not as effective as Mask R-CNN. The most noticeable performance caveat is the recall scores for all scenarios. Whilst the high precision values (0.91-0.95) can be strongly accredited to the

[3] On a single GPU for CNN inference and a standard Intel 8 core CPU with multi-thread parrelisation for Morphological GAC inference.

accurate initial seed points derived from RetinaNet, the poor recall would be accredited to the Morphological GAC segmentation. More specifically, it is observed that the buildings are under-segmented. The Potsdam dataset contains a largely different building architecture and layout design to the majority of buildings in the main dataset. For example, much of the scene contains large continuously terraced buildings (Figure 3.4). As this classifies as a single building it is therefore segmented in a single Morphological GAC minimisation. However, the terrace buildings contain large depth variance across individual dwellings amongst the whole terrace. This therefore terminates the segmentation leaving the remainder of the building classed as background.

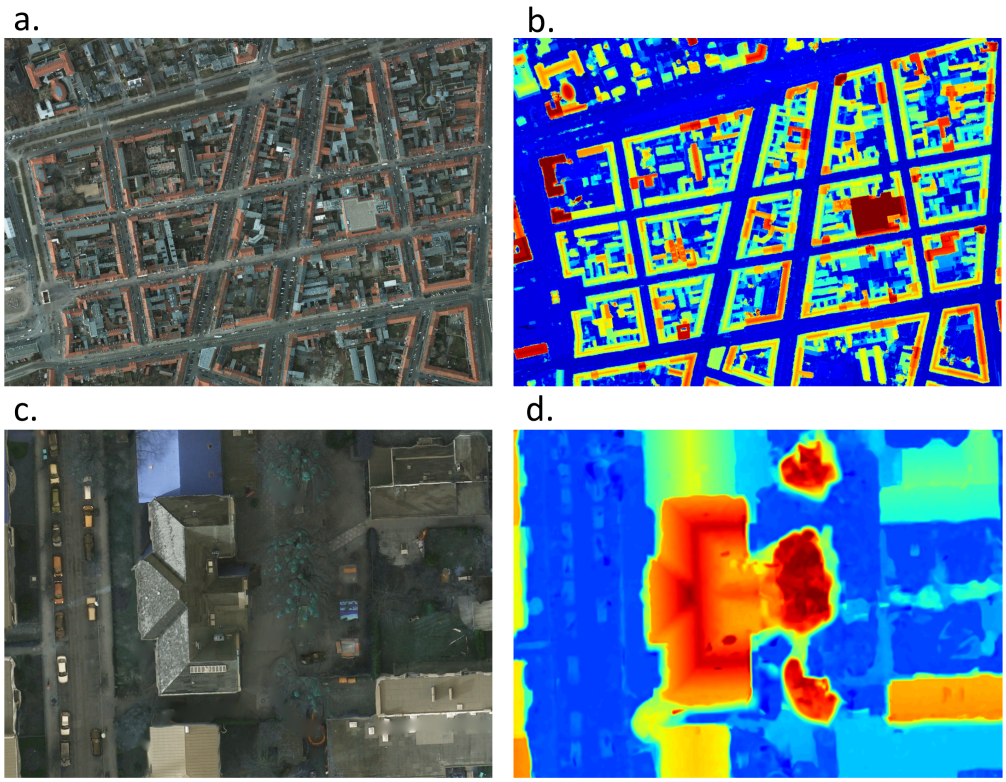


Figure 3.4: **a)** Aerial and DSM of Potsdam town design. Many of the buildings are continuous terrace houses, which can cause issues for a Morphological GAC segmentation. **b)** Example of apparent DSM smearing. In the aerial image the tree appears to be separated from the house, however, the DSM shows a definitive overlap.

The online use of the Morphological GAC segmentation is also the reason for little variance in F_1 values across all the scenarios. Generally, the performance

Table 3.5: Potsdam RetinaNet segmentation results. mAP is calculated for recall > 0.5 with true positives defined as IoU> 0.5.

Model	mAP	Precision	Recall	Accuracy (%)	F_1 Value
RGB OSLabel Generic	0.46	0.91	0.72	93.94	0.80
RG-D_E OSLabel Generic	0.46	0.93	0.78	95.32	0.85
RG-D_N OSLabel Generic	0.48	0.94	0.79	95.54	0.86
RGB Label+ Generic	0.50	0.93	0.75	92.91	0.83
RG-D_E Label+ Generic	0.40	0.92	0.79	93.21	0.85
RG-D_N Label+ Generic	0.48	0.93	0.78	94.87	0.85
RGB Fine-Tune	**0.81**	0.94	0.78	94.89	0.85
RG-D_E Fine-Tune	0.80	**0.95**	0.81	95.67	0.87
RG-D_N Fine-Tune	**0.81**	**0.95**	**0.82**	**96.12**	**0.88**

of the building detector was high enough for all scenarios for the segmentation to be the main caveat. However, there was still a substantial performance increase in mAP for the fine-tuned scenarios. The inclusion of depth data showed no significant improvements. Unfortunately, as the dataset is significantly smaller than what is recommended for training deep CNN's, it is a possibility the network has strongly over-fitted to the training data, and the lack of variance between the training and test produces seemingly very good results.

3.5 Discussion

The results presented in this chapter demonstrate the advance of improved labels for both object detection and segmentation. This was most evident for object detection, where the was an average of 47% increase in mAP score against 11.36% F_1 segmentation score. It was observed that improving labels had the largest impact on the precision values for segmentation. Despite the fact over-generalisation would affect the mean depth channel value the greatest, the largest increases for both object detection and segmentation were observed in the RGB scenarios. Furthermore, this suggests that poor labelling overlapping incorrect textures (i.e., roof vs road/garden) is potentially more detrimental to the networks performance than a constant mean difference. This is not surprising as CNN's are known to learn textures in many of the intermediate layers. These results offer quantitative analysis of the hypothesis that large-scale GIS data can be used to train networks, with sheer quantity counteracting poor quality. We demonstrate here that valuable results can be obtained

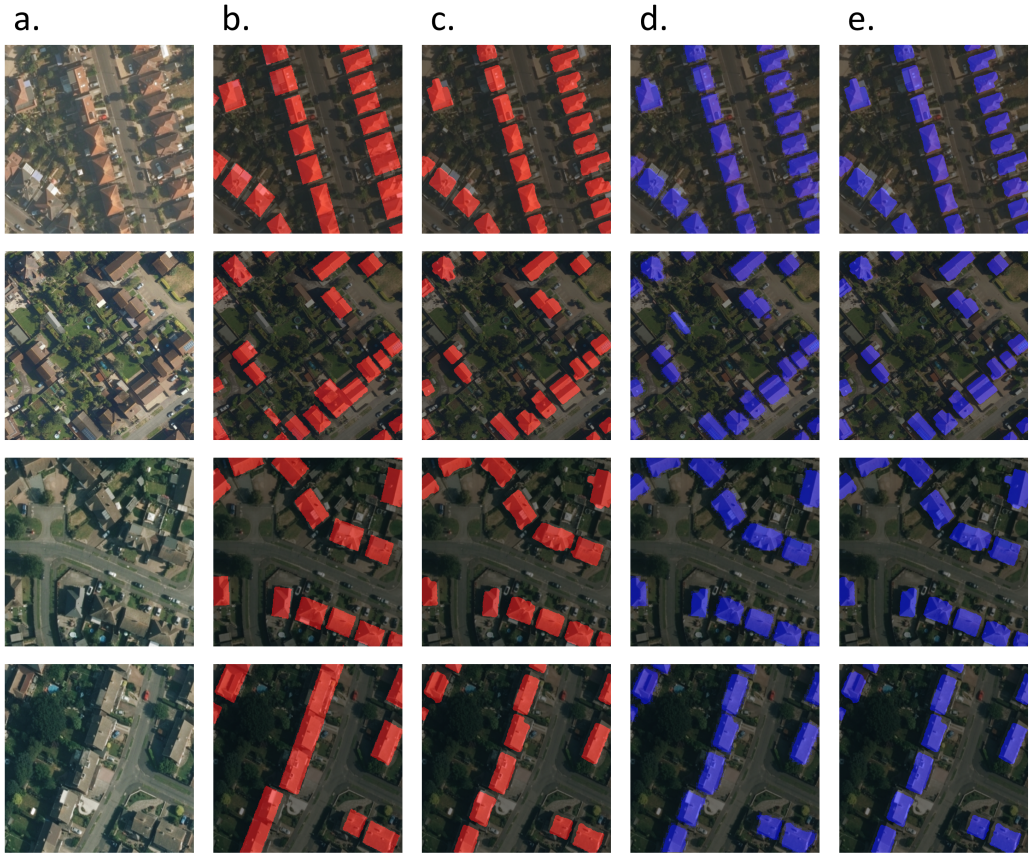


Figure 3.5: **a)** Aerial images of UK buildings to be segmented. **b)** Public GIS data labels **c)** Improved labels using the active contours **d)** Mask R-CNN segmentation results **e)** RetinaNet object detection + active contour segmentation results.

with low quality training data, however, it is essential to have high quality labels for state-of-the-art results. Poor quality can be a consequence of a number of causes such as: over-generalisation, data acquisition temporal variance, missing data etc. however, when using depth data, the quality of aerial image orthorectification is prominent. This was witnessed extensively across both datasets, where errors in othorectification led to a small misalignment in the aerial image and depth data.

The Morphological GACs proved to be an effective tool for pixel-wise segmentation of buildings from initial building footprints. However, the tool was not as effective when continuous terrace buildings as seen in the Potsdam dataset, dominate the local architecture. Despite this, we demonstrate here that rule-based algorithms can still act as a valuable tool to aid the training of machine learning models. Furthermore, Morphological GAC derived segmentation on the UK dataset

was showing to achieve higher accuracy than the state-of-the-art Mask R-CNN for segmentation, provided seed points were given. This was demonstrated by the success of the RetinaNet segmentation compared to Mask R-CNN (0.82 and 0.92 F_1 scores respectively). The results here demonstrate that whilst a fully end-to-end deep learning approach would be more desirable due to increased robustness and computation speed, in many specific applications a combination of bespoke rule-based and deep learning approaches can yield the greatest performance.

The potential benefits of the inclusion of depth data in aerial object detection and segmentation scenarios is also exemplified from our experiments. We demonstrate that existing CNN architectures (RetinaNet and Faster/Mask R-CNN) can be used without alteration, and therefore, bespoke networks to handle depth information are not essential. This was most prominent in the large UK building datasets where the the inclusion of depth data resulted an overall increased mAP for object detection and improved recall values for Mask R-CNN segmentation. Despite this the implications of the depth channel were not as clear when evaluating on the Potsdam benchmark dataset. In all scenarios network inferences after fine-tuning on the Potsdam dataset very high-performance results were achieved. It could be assumed that the larger dataset is more indicative of the general model performance and therefore this suggests the high results observed on the fine-tuned datasets are caused by a form of over-fitting. The Potsdam benchmark, when tiled to a similar geographic size ($250m^2$) to the UK dataset only consisted of 133 images of size (600x600px). All network architectures had $>$ 50 million parameters, and therefore, it is unreasonable to conclude exactly what the model has learned, and how applicable the models would be for applications in different datasets. This highlights an issue where relatively small benchmark datasets are inferred with increasingly deep neural network architectures. Furthermore, the network weights pre-trained on the UK dataset also did not perform well on the Potsdam benchmark. This indicates that even though a large dataset was used for training, the general features of UK buildings were not enough to allow for reliable inference on substantially different building/street designs. This is important as this demonstrates that features learned

from the depth channel alone is not robust enough for a functional general model. Instead, higher layer features which are associated with larger patches of the image and therefore object context are required within the training dataset. Despite this, the data acquisition methodology presented here is scalable and therefore incorporation of more diverse training data would be easily achieved and likely highly beneficial.

The comparison between the two-stage Faster R-CNN and one-stage RetinaNet, concluded in favour of RetinaNet. This is quantified by an average increase in mAP of 1.25 on the main UK building dataset and 0.078 overall. This equates to a percentage increase of 20.13% and 17.79% respectively. An increase was observed for all but one processing scenario when using RetinaNet. Moreover, this performance increase comes at a computation speed gain with the average inference speed amounting to 73ms and 89ms for RetinaNet and Faster R-CNN respectively. Although this contradicts the classic speed/accuracy trade-off (Huang et al., 2017), this instead demonstrates the potential benefits of the focal loss algorithm introduced in RetinaNet. In the UK building dataset, the absolute pixel class imbalance observed was 1-6.66 and 1-4.5 for the Potsdam dataset, for foreground (building) and background respectively. This however becomes exemplified by classification at multiple scales. Therefore, this suggests that for single-class or multi-class where large class imbalances are still present the focal loss is of fundamental use for high performing models. Such conclusions were also made in a previous study (Griffiths, Boehm, 2018). Moreover, Lin et al. (2017) demonstrate that even in the COCO benchmark dataset where class imbalances are not as prominent, the model offers performance improvements. The results indicate that the Mask R-CNN could benefit from the incorporation of a focal loss method into the object detection branch. This would combine the high accuracy object detection of the RetinaNet with power of semantic segmentation.

Chapter 4

Domain Adaptation from Synthetic Data

4.1 Introduction

A fundamental requirement for supervised deep learning is large, labelled datasets. For this reason, progress in 2D image processing is often largely accredited to the wealth of very large, high quality datasets (Deng et al., 2009; Lin et al., 2014; Everingham et al., 2010). It is now common practice to pre-train 2D CNN's on large datasets before fine-tuning on smaller domain specific datasets. Despite the large success of deep learning for 2D image processing, it is evident that automatic understanding of 3D point cloud data is not as mature. In this chapter we argue one of the reasons for this is the lack of training data at a comparable scale of that available for 2D data.

A key reason for the lack of 3D training data is that naturally the amount of prepared labelled data decreases as the complexity of labelling increases (as discussed in Sec. 1.4). For example, in 2D, single image classification is generally trivial and can therefore be carried out by large communities of minimally trained workers. Object detection requires more skill and has an added level of subjectivity. Segmentation again requires further precision, delicacy and involves more subjectivity. Per-point 3D segmentation requires highly skilled users and generating perfect labels for even the most advanced users is non-trivial (Liu, Boehm, 2014). A potential

solution to account for this is to synthetically generate training data (Chang et al., 2015; Gschwandtner et al., 2011; Reitmann et al., 2021; Wang et al., 2019a). Despite wide-spread success observed in the 2D image domain, training with synthetic data and fine-tuning on real-world data, there has been significantly less research on this topic with respect to point cloud classification.

3D data benefits from a wealth of synthetic data in the form of virtual 3D environments generated for the purpose of gaming, virtual reality and scenario training simulators. However, the ability for deep learning networks to generalise from synthetic point clouds to real-world data is infrequently studied, and as such the community risks missing out on a massive resource of data. To help address this, in this chapter we introduce *"SynthCity"*, an open, large scale synthetic point cloud of a typical urban/suburban environment. SynthCity is generated using a simulated Mobile Laser Scanner (MLS). MLS point cloud data capturing is being increasingly used due to its ability to easily cover large areas when compared to a Terrestrial Laser Scanner (TLS) and at a higher resolution than that of Airborne Lidar Scanning (ALS). However, whilst capturing large quantities of data is becoming more accessible, the value of such large datasets is limited without the means to extract useful structured information from otherwise unstructured data. As such, progress in this field offers huge potential for a range of disciplines from city planning to autonomous driving.

In this chapter we develop and evaluate a pipeline for generating 3D MLS data using the open-source 3D modelling software Blender (Community, 2018) and open-source LiDAR simulator Blensor (Gschwandtner et al., 2011). We show that creating such a dataset requires few processing steps providing an existing 3D model is available. We evaluate the use of our dataset for pre-training the popular PointNet++ network (Qi et al., 2017b). Our experiments have two primary axes. First, we evaluate the benefits of pre-training our network to train a small (Paris-Lille-3D $\sim$143.1M points (Roynard et al., 2018a)) dataset. Secondly, we evaluate the effect of adding varying scales of Gaussian noise to the synthetic point cloud to reduce the Synthetic to Real (sim2real) domain gap.

4.2 Problem Statement

We aim to generate a globally registered point cloud $\mathcal{P} \in \mathbb{R}^{n\times 3+k}$ where n is the number of points such that $n = 367.9M$ and k is red, green, blue, time, end of line, and class label c where $c \in \{1,2,\ldots,9\}$.

4.3 Methodology

4.3.1 Synthetic Point Cloud Generation

The SynthCity data was modelled inside the open-source Blender 3D graphics software Community (2018). The initial model was downloaded from an online model database (Fig. 4.1). To increase the overall model size, the model was duplicated with the objects undergoing shuffling to ensure the two areas were not identical to one another. Road segments were also duplicated to connect the two urban environments leaving large areas of unoccupied space. To populate these areas additional typical suburban building models were downloaded and placed along the road. The final model contains: 130 buildings, 196 cars, 21 natural ground planes, 12 ground planes, 272 pole-like objects, 172 road objects, 1095 street furniture objects and 217 trees (table 4.3). The total disk size of the model was 16.9GB. The primary restriction for the size of the dataset was availability of Random Access Memory (RAM) required on the workstation used for creating the model. This was limited to 32GB in our case, however, with a larger RAM the model size could have easily been extended using the described technique for many more iterations.

The open-source Blender Sensor Simulation plugin Blensor (Gschwandtner et al., 2011) was used for simulation of the MLS and thus point cloud generation. We use the following configuration for scanning:

A typical scan took $\sim$330s to render and a total of 75,000 key frames were rendered from a pre-defined trajectory. To increase realism and generate more variability in point density the trajectory spline was randomly perturbed at random intervals in all x, y, z directions. The final rendering required $(330 \times 75000)/86400 = 286.46$ days CPU compute time. This was processed using AWS cloud computing service.

Figure 4.1: Rendered image from the initial downloaded model. Image source Squid (2022).

Table 4.1: Blensor scanner configuration.

Attribute	Value
Scan type	Generic lidar
Max distance	100m
Angle resolution	0.05m
Start angle	-180 °
End angle	180 °
Frame time	$1/24s$

We launched 22 type r4.2xlarge Ubuntu 18.04 EC2 spot instances, each containing 8 virtual CPUs and 61GB RAM. These were selected as rendering typically required ∼50GB RAM. All data was read and written to a EFS file storage system to allow for joint access of a single model instance. The total rendering time took ∼13 days on 22 EC2 instances.

Each render node produces an individual file s_t for the 2D scan at time frame t. To create the global 3D point cloud each point must undergo a transformation **T** with respect to the scanner location $S_{x,y,z}$ and rotation $S_{\omega,\phi,\kappa}$. Blensor can export both $S_{x,y,z}$ and $S_{\omega,\phi,\kappa}$ at time t as a motion file. Each scan is passed through a global registration script where the transformation **T** is computed as the rotation matrix where:

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix} \tag{4.1}$$

$$\mathbf{R_y} = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \tag{4.2}$$

$$\mathbf{R_z} = \begin{bmatrix} \cos\kappa & -sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$\mathbf{R} = \mathbf{R_z}\mathbf{R_y}\mathbf{R_x} \tag{4.4}$$

$$\mathbf{T} = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & S_x \\ R_{2,1} & R_{2,2} & R_{2,3} & S_y \\ R_{3,1} & R_{3,2} & R_{3,3} & S_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

Finally, each transformed point $\hat{p}_t$ is computed as:

$$\hat{p}_t = p_t \cdot \mathbf{T} \tag{4.6}$$

In a separate post-processing stage, we generate the features $\mathcal{F} = n_x, n_y, n_z, time, eol$. To create $\mathcal{F} = n_x$, n_y, n_z we simply apply a 0.005m Gaussian noise to each dimension (x, y, z) of point $\hat{p}_t$ independently such that $\hat{p}_t = \hat{p}_t^x + \sigma_x, \hat{p}_t^y + \sigma_y, \hat{p}_t^z + \sigma_z$ where $-0.005 < \sigma < 0.005$. We choose 0.005m as this is in line with an expected modern scanner noise in a similar scenario. $\mathcal{F} = time$ is calculated by adding the key frame time available in the motion file with the scanner point time available in the individual scan files (i.e., 1Hz scan frequency). This is effectively a simulated Global Navigation Satellite System (i.e., GPS) time available with MLS and ALS point clouds. Finally, the end of line (eol) is calculated as a binary indicator where the

$p_i^{eol} = 1$ if it is the final point acquired by the individual scan s_t or 0 otherwise.

Table 4.2: Data fields and stored types.

Feature	Type
x	float
y	float
z	float
x_n	float
y_n	float
z_n	float
R	short
G	short
B	short
time	double
eol	boolean [0, 1]
label	short [0-8]

We choose to store our data in the parquet data format (Apache, 2022). The parquet format is very efficient with respect to memory storage but is also very suitable for out-of-memory processing. The parquet format is designed to integrate with the Apache Hadoop ecosystem. It can be directly read into python Pandas dataframes but also python Dask data frames which allow for easy out-of-memory processing directly in the python ecosystem.

The dataset is modelled from a completely fictional typical urban environment. The environment would be most similar to that of downtown and suburban New York City, USA. This was due to the initial base model, and not any design choices made by ourselves. Other buildings and street infrastructure are typical of mainland Europe. We classify each point into one category from: road, pavement, ground, natural ground, tree, building, pole-like, street furniture or car. To address the class imbalance issue, during construction of the model we aimed to bias the placement of small less dominant features in an attempt to reduce this as much as possible. As point cloud Neural Network (NN)'s' typically consume spatially small (local) subsets of the dataset we argue that this approach should not introduce any unfavourable bias, but instead help physically reduce the class imbalance.

The final feature list with their respective storage type is shown in table 4.2.

Table 4.3: Label categories and number of points per category.

Label	No. Models	No. Points
Road	172	215,870,472
Pavement	172	21,986,017
Ground	12	6,206,312
Natural ground	21	4,788,775
Tree	217	12,088,077
Building	130	97,973,820
Pole-like	272	1,636,443
Street furniture	1095	1,469,766
Car	196	5,907,347
Total	2287	367,927,029

The total number of points generated is shown in table 4.3. The disk space of the complete parquet file is 27.5GB, as a typical workstation would not be able to load this model into memory, we split this scan into 9 sub areas. Each sub area is split solely on horizontal coordinates and can therefore contain points from any scan at any key frame. The purpose of this is twofold; firstly, users of typical workstations can load an area directly into memory, and secondly, we can nominate a fixed test area. We propose that areas 1-2 and 4-9 be used for training and area 3 be reserved for model testing. This enables consistency if models trained on our dataset are to be compared from one another. We choose area 3 as it contains a good representation of all classes. As SynthCity is not intended as a benchmark dataset we provide the ground truth labels for area 3 in the same manner as all other areas. We show an example output of the point cloud generation in Fig. 4.3.

4.3.2 Experiments

To evaluate the potential benefit of the SynthCity dataset, a series of experiments are described and conducted. The experiments cover domain adaptation, fine-tuning and a fully supervised baseline.

Domain Adaptation is the process of training a model in one domain (*source* domain) and deploying it on data from a different domain (*target* domain). Here, the *source* domain is the SynthCity dataset (synthetic) and the *target* domain is the iQmulus Paris-Lille dataset (real). A common additional technique to reduce the

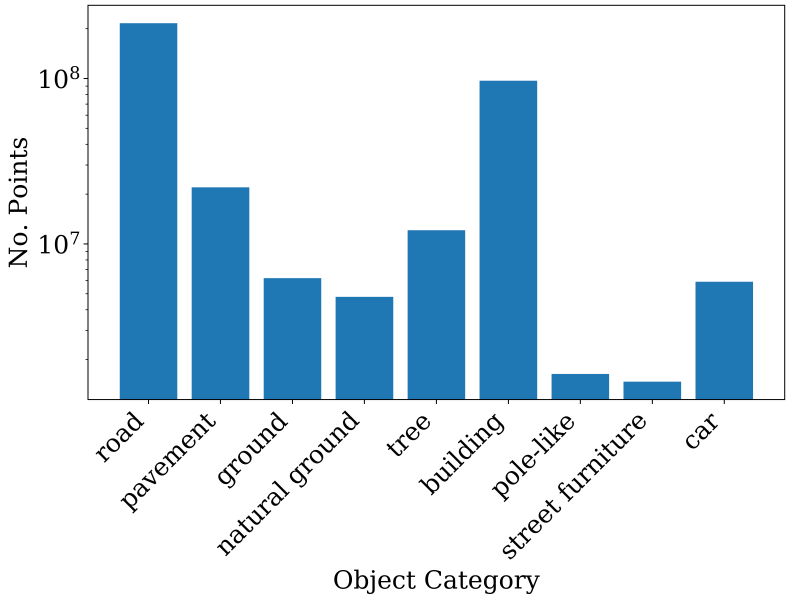
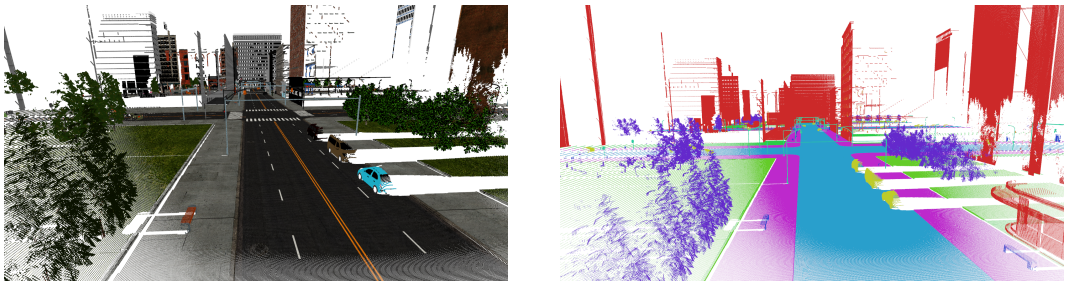


Figure 4.2: Total point counts for each label category. Note the log y-axis scale.



(a) RGB coloured point cloud.

(b) Semantically labelled point cloud

Figure 4.3: Example scene from the SynthCity dataset.

domain-gap from the source to target domain is to apply some form of augmentation to make the source domain data "appear" more like the target domain data. In this work we achieve these through two methods: synthetic sensor geometry and noise.

One simple way to create a point cloud from a mesh model would be to simply sample points randomly on the mesh surface. This point cloud would not contain many of the artefacts expected in a real world MLS point cloud. Artefacts include areas with no points due to occlusion, variable point density and local point patterns. As such, a model trained on a sampled mesh point cloud would unlikely be robust to any of these sensor specific artefacts. To account for this, we use a programme (Blensor) which physically simulates MLS sensor geometry. Points are generated

by shooting rays from a virtual sensor center in a circular sequence. Each stores the x, y, z and object attributes of the first surface it hits. As such, occlusion is correctly modelled. Shooting ray at 2D angles (θ, ϕ) from a virtual scanner center further results in varying point densities and scanner point patterns on surfaces.

Additionally, we also add Gaussian noise to the synthetic scans. We choose noise as this is the most noticeable artefact present in real-world data that is not present in the synthetic scenes. For each point $p \in \mathcal{P}$ we add noise by sampling an offset vector $\hat{x} \sim \mathcal{N}(\mu, \sigma)$ where $\hat{x} \in \mathbb{R}^3$. Our final point cloud is therefor $p_i + \hat{p} \sim \mathcal{N}(\mu, \sigma) \forall p \in \mathcal{P}$. In our experiments we study the effects of incrementally increasing the σ of the normal distribution $\mathcal{N}(\mu, \sigma)$ where $\mu = 0$ and $\sigma = \{0.01, 0.05, 0.1\}$. Scanner noise has two common sources. Firstly, angular measurement errors result in incorrect point positions on the 2D scanner plane. The magnitude of position errors from angular errors are therefore also a function of distance. Secondly, timing errors occur from the scanner computing the time it takes for a ray to return incorrectly. The position error is therefore along the ray of light direction (1D). Although both errors could be simulated independently, the timing error is expected to be at least an order of magnitude less than the angular error (Vosselman, Maas, 2010). As such, we find a simple Gaussian in the order of magnitude expected from the angular error to be sufficient.

Fine-tuning is a process of initialising a model state with the weights obtained by training the same model on a different dataset. It is commonly observed that a model can be fine-tuned with significantly less data when compared to training a model from a random initialisation, provided the original dataset and task are not too dissimilar from the new dataset and task. In these experiments we "pre-train" the network on the SynthCity dataset. The model is then "fine-tuned" on the iQmulus dataset. Note, this ability to retrain the weights on the iQmulus dataset is what differentiates fine-tuning and domain adaptation here. Furthermore, as we do not expect the synthetic domain to be close enough to the target domain of the real-world data, we do not freeze early layers of the network as is often common practise in fine-tuning scenarios where the domains are sufficiently similar, or networks suf-

ficiently large. Our network is pre-trained on the SynthCity dataset (367M points) and fine-tune the same network on a smaller iQmulus dataset (118M points).

Fully-supervised Lastly, to allow for relative comparison we also train the iQmulus data using a standard fully-supervised learning approach. This acts as a baseline for the domain adaptation and fine-tuning experiments.

Network setup We choose PointNet++ (Qi et al., 2017b) as the neural network. PointNet++ was chosen as it has become a standard in 3D point cloud deep learning and consistently performed at a high standard across a range of tasks, including per-point classification. The training configuration as outlined in Tbl. 4.4

Table 4.4: Hyperparameter values for PointNet++ experiments.

Hyperparameter	**Value**
Optimiser	adam
Learning rate	1e-4
Number of down layers	4
Layer feature radius	[0.2, 0.4, 0.8, 1.6]
Dropout rate	0.5

Pre-processing We pre-process all scenes with the same set of pre-processing steps. Each scene is first cropped into 5m $\times$ 5m crops in the x, y plane (horizontal). To ensure permutation invaririance the ordering of the points is randomly shuffled before being input to the network. Lastly, Gaussian noise is added to the points in accordance with the **Domain adaptation** paragraph. As the classes between the SynthCity and iQmulus datasets do not map one-to-one the iQmulus dataset classes are remapped to best match the SynthCity classes. The class remapping is shown in Fig. 4.4.

4.4 Results

All experiments are run according to the methodology discussed in Section 4.3.2. The results are presented in Tbl 4.5 and Tbl 4.6.

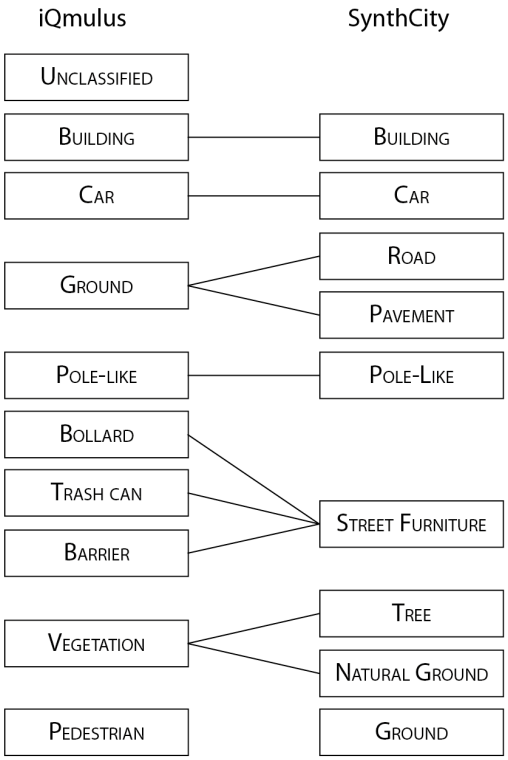


Figure 4.4: Class remapping between iQmulus and SynthCity datasets.

4.4.1 SynthCity

Experimentally we find there is a negative correlation between the magnitude of Gaussian noise and overall performance with respect to the metrics. This is not unsurprising for a synthetic dataset. When no sensor noise is present the network can take advantage of strong consistencies and regularities in the data. For example, all ground points lay on a perfectly planar surface at a fixed height in world coordinates. Such regularities could be more easily learnt by the network. As they are also present in the test dataset, over-fitting to these regularities on the training set would also result in high test set performance.

4.4.2 iQmulus

In Tbl. 4.6 we present the baseline fully-supervised experiment. We apply a small amount of Gaussian noise ($\sigma = 0.01$) as an augmentation to avoid over-fitting, as is typical for per-point classification on point cloud data (Qi et al., 2017b, 2019; Hermosilla et al., 2019; Thomas et al., 2019).

Table 4.5: Experimental results for SynthCity dataset pre-training with varying magnitudes of Gaussian noise.

Dataset	σ	Acc.	Pr.	Re.	F1	mIoU
SynthCity	0.00	**0.938**	**0.736**	**0.701**	**0.703**	**0.663**
SynthCity	0.01	0.894	0.673	0.627	0.630	0.564
SynthCity	0.05	0.841	0.583	0.514	0.522	0.512
SynthCity	0.10	0.818	0.469	0.406	0.413	0.403

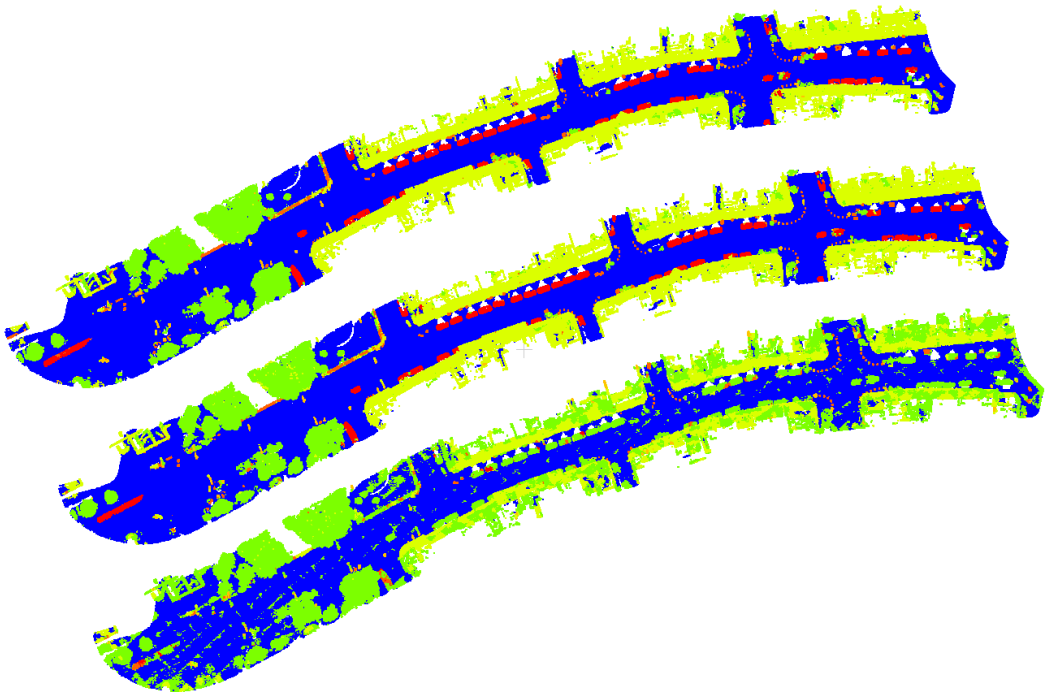


Figure 4.5: Birds eye view qualitative results on the iQmulus dataset for **top**) fully-supervised, **middle**) fine-tuning and **bottom**) domain adaptation.

In Tbl. 4.7 we present the results from the domain adaptation experiments. The inclusion of Gaussian noise, whilst reducing test performance on synthetic data, had a positive correlation with performance on the iQmulus dataset up until 0.05cm. After this, the performance degrades (with the exception of precision). Despite the performance increasing, it is still significantly lower performing when compared to the synthetic test set. This is a measurement of the domain-gap between the synthetic (source domain) and real (target domain) data. Therefore, whilst Gaussian noise can reduce the domain-gap somewhat, there is certainly a lot of room for improvement in eliminating the domain gap.

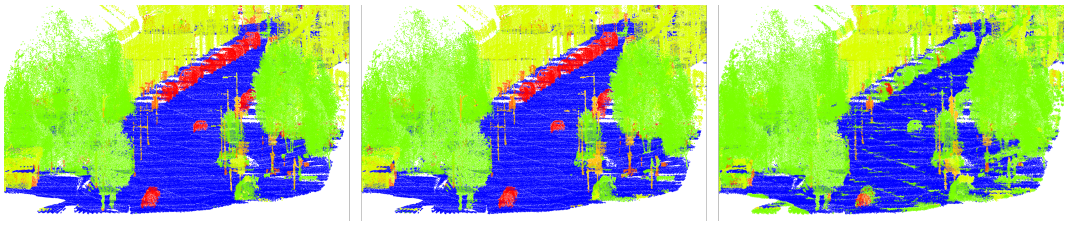


Figure 4.6: Perspective view qualitative results on the iQmulus dataset for **left**) fully-supervised, **middle**) fine-tuning and **right**) domain adaptation.

Table 4.6: Experimental results for iQmulus Paris-Lille using standard fully-supervised learning.

Dataset	Pre-train	Fine-tune	σ	Acc.	Pr.	Re.	F1.	mIoU.
iQmulus	✗	✗	0.01	0.955	0.702	0.667	0.670	0.507

Fine-tuning (Tbl. 4.8) on top of the SynthCity pre-trained model ($\sigma = 0.00$) had an improvement over the random weight initialisation (Tbl. 4.6). The addition of Gaussian noise in the SynthCity model training, whilst improving domain adaptation performance (Tbl. 4.7), had the opposite effect for fine-tuning. One explanation for this is that the network is learning weights specific to modelling the Gaussian noise, which does not directly apply to the real data. It is therefore unlikely the Gaussian noise reduced over-fitting on the synthetic data.

When looking at per-class IoU for synthetic data (Tbl. 4.9) the decrease of performance is seen across all classes. The highest performing classes were `car` and `road`. Interestingly, `building` was not one of the top classes. As discussed above, `road` and `building` are generally have the highest point count and are easily classified due to their planar nature. However, due to the coarse geometry of the SythCity dataset and many gaps in the buildings, there was a lot of noise for building points (see Fig. 4.7). Further evidence is given when looking at per-class IoU for real data (Tbl. 4.10). The `building` class achieves the highest IoU score by a strong margin over all other classes.

When looking at the overall per-class performance of the real data (Tbl. 4.10) there does not appear to be any trends with respect to fine-tuning and domain adaptation at a class level. The addition of Guassian noise has a varying effect of each

Table 4.7: Experimental results for direct domain adaptation (sim2real) from SynthCity to the iQmulus Paris-Lille dataset.

Dataset	Pre-train	Fine-tune	σ	Acc.	Pr.	Re.	F1.	mIoU.
iQmulus	✓	✗	0.00	0.643	0.422	0.323	0.332	0.200
iQmulus	✓	✗	0.01	0.690	0.451	0.358	0.366	0.229
iQmulus	✓	✗	0.05	**0.744**	0.451	**0.370**	**0.380**	**0.244**
iQmulus	✓	✗	0.10	0.675	**0.437**	0.341	0.350	0.215

Table 4.8: Experimental results for iQmulus Paris-Lille dataset through Fine-Tuning (FT) with with different SynthCity Pre-Trained (PT) model initialisation.

Dataset	Pre-train	Fine-tune	σ	Acc.	Pr.	Re.	F1.	mIoU.
iQmulus	✓	✓	0.00	**0.957**	**0.723**	**0.693**	**0.695**	0.501
iQmulus	✓	✓	0.01	0.953	0.700	0.674	0.676	**0.504**
iQmulus	✓	✓	0.05	0.943	0.701	0.668	0.672	0.477
iQmulus	✓	✓	0.10	0.920	0.642	0.597	0.602	0.410

class with, for example, `ground` decreasing in IoU and `vegetation` increasing. However, relatively across classes performance does not change significantly.

4.5 Discussion

The results presented in Sec. 4.4 show the highest performing model on real data with respect to accuracy, precision and recall was achieved through fine-tuning with no added Gaussian noise to the SynthCity dataset. The highest mIoU score was achieved with the fully supervised approach. The overall performance improvement from the fully supervised baseline is very low across all metrics, and this could easily be within the expected variance of the model training as a result of random initialisations and the stochastic training procedure. As such, the improvement is not significant enough to draw any solid conclusions. Furthermore, the addition of Gaussian noise, whilst improving domain adaptation performance, still left a large domain gap and had a negative impact on fine-tuning. There are several problematic areas which might explain why this is the case.

Table 4.9: IoU per class for SynthCity pre-training for classes; Building (B), Car (C), Natural Ground (N), Pole-like (Po), Road (R), Street Furniture (S), Tree (T) and Pavement (Pa).

σ	B	C	N	G	Po	R	S	T	Pa
$\sigma_{0.00}$	0.645	**0.941**	**0.163**	**0.543**	**0.610**	**0.882**	**0.560**	**0.683**	**0.597**
$\sigma_{0.01}$	**0.684**	0.863	0.124	0.491	0.533	0.774	0.476	0.368	0.546
$\sigma_{0.05}$	0.630	0.839	0.103	0.382	0.538	0.635	0.463	0.454	0.350
$\sigma_{0.10}$	0.550	0.733	0.055	0.338	0.331	0.582	0.388	0.305	0.258

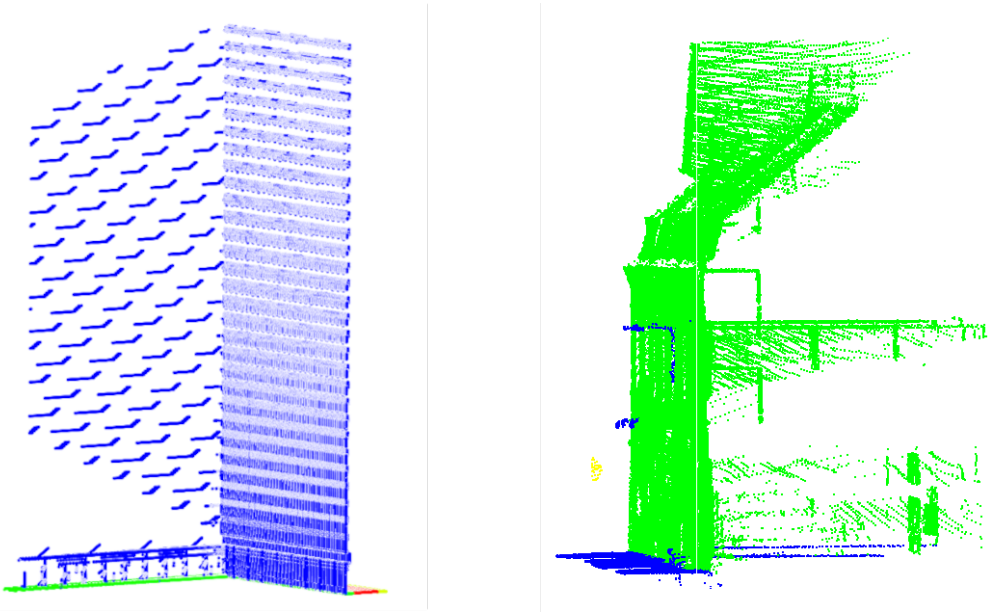


Figure 4.7: Building class noise for SynthCity synthetic (**left**) and iQmulus real (**right**) data. Note the difference in noise patterns.

Size Firstly, the SynthCity dataset may not be large enough. A key advantage of synthetic data is the ability to scale up huge datasets which would otherwise not be possible, and labelling is too expensive. Despite this, due to limited computational resources, the size of SynthCity falls in-line with many other real datasets (see Tbl. 2.2). Increasing the number of points by orders of magnitude may contradict the results found in this chapter. The key computational bottleneck for SynthCity was having to load the entire scene in Blender which requires large amount for RAM. A more dynamic RAM allocation environment such as (Dosovitskiy et al., 2017) could avoid this issue and result in significantly large datasets with the same computational resources.

Table 4.10: IoU per class for fully-Supervised (S) (Tbl. 4.6) and Fine-tuning (FT) (Tbl. 4.8) and Domain Adaptation (DA) (Tbl. 4.7).

Exp.	Building	Car	Ground	Pole-like	Street F.	Veg.
S	0.838	**0.329**	0.646	0.302	0.266	0.660
FT: $\sigma_{0.00}$	0.843	0.297	**0.663**	0.264	**0.304**	0.635
FT: $\sigma_{0.01}$	**0.852**	0.319	0.652	**0.310**	0.271	0.617
FT: $\sigma_{0.05}$	0.833	0.314	0.597	0.217	0.233	**0.670**
FT: $\sigma_{0.10}$	0.806	0.206	0.597	0.188	0.203	0.462
DT: $\sigma_{0.00}$	0.625	0.141	0.265	0.091	0.075	0.004
DT: $\sigma_{0.01}$	0.669	0.146	0.333	0.098	0.122	0.004
DT: $\sigma_{0.05}$	0.688	0.155	0.387	0.108	0.114	0.014
DT: $\sigma_{0.10}$	0.636	0.145	0.310	0.102	0.088	0.008

Realism Although SynthCity looks visually realistic, geometrically it is a simple dataset. This is because it was not originally created for the task of pre-training 3D scene understanding networks. In computer graphics, simple geometry can be made to looks highly realistic with appropriate material and texture properties and a well configured ray-sampling engine. A key drawback to increasing geometric realism would be the additional memory requirement and more vertices and faces would be required to represent the geometry. This would also increase the time it takes to generate the dataset. Furthermore, the fundamental difference of the geometry between synthetic and real leads to very different noise in the data. A network should learn to be robust to specific types of noise present in the scene (e.g., reflection and refraction from windows). Without also have realistic material properties in the blender scene it is not possible for the pre-trained network to learn such noise sources.

Real test set The test set of iQmulus is potentially too similar to its respective train set (Fig. 4.8). The key benefit for synthetic pre-training is to increase generalisation. However, if the train and test set are too similar, the test set performance will not be measuring generalisation. Instead, it will be measuring the networks ability to over-fit to the specific geographical area, sensor noise etc. More conclusive results could therefore be possible on a dataset where generalisation is a clear problem with train-test performance.

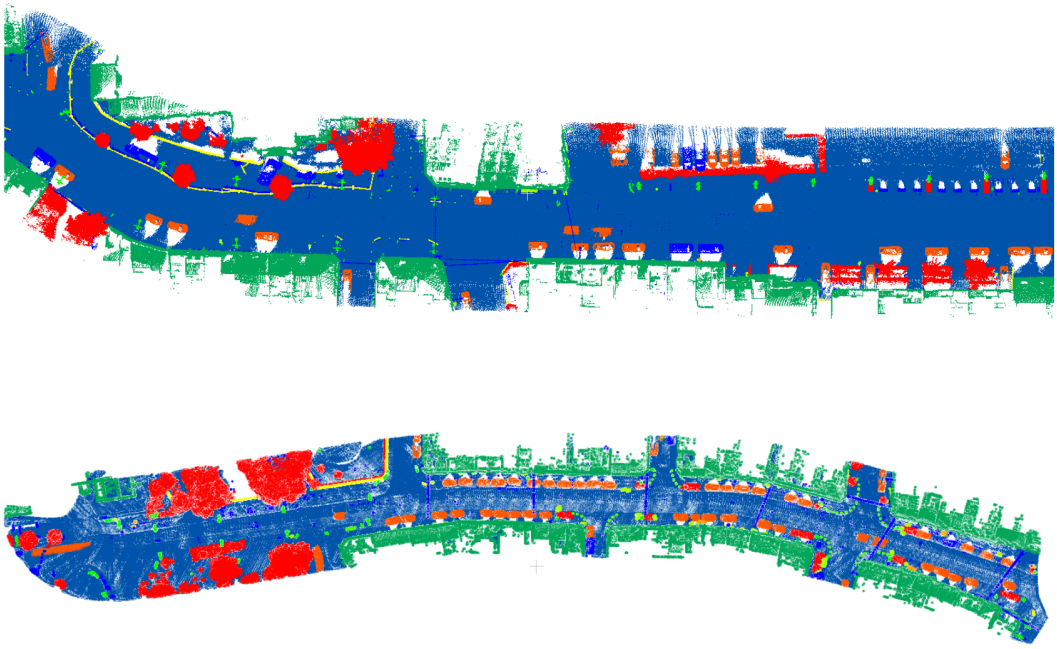


Figure 4.8: Visualisation of a section of the iQmulus train dataset (**top**) and the validation dataset (**bottom**).

Model In this chapter we use the PointNet++ (Qi et al., 2017b) network architecture. Results from a similar experiment by Spiegel et al. (2022), instead with the more modern KPConv model (Thomas et al., 2019), arrive at a contrasting set conclusions. Whilst, the study agrees with our domain adaptation experiment, concluding adding Gaussian noise with $\sigma > 0.05$ degrades results, the authors also note an improvement on performance in the fine-tuning setting. Unfortunately, the authors do not provide baseline experiments allowing conclusion if fine-tuning is more beneficial than fully supervised training as experienced in our experiments.

Class-imbalance Despite efforts being made to reduce the class-imbalance, a very strong class imbalance still remained (see Fig. 4.2). This is a very common problem with MLS where the sensor is typically located on the road in an urban environment. As such $> 90\%$ of points are typically either ground or façade. This is further caused by MLS data density being negatively correlated with distance from the scanner. As such, the highest point density is always on the road. Although techniques such as voxel down-sampling can be used as well as class aware augmentation techniques (Griffiths, Boehm, 2019), this problem can persist. An alternative strategy is to

employ a weighted focal loss (Lin et al., 2017), which has shown to be successful for 2D image object detection (see. Sec. 3). It is evident from the results in Tbl. 4.10 that class over-fitting is occurring. Interestingly, for domain adaptation there is almost no reduction in performance on the `building` class, however, `car` drops by upto 80%. A potential reason for this is that the network learns heavily to rely on absolute z (height) values of the points for classification. This is a reliable feature as SynthCity is completely flat. A small change of scale between the scenes could lead the network to make errors, especially on smaller classes like `car`. However, for larger classes like `building`, which are the largest features in the scene, this may not be such an issue.

4.6 Conclusion

In this chapter we presented SynthCity an open, large-scale synthetic point cloud. The dataset has been made available for public release to help aid research in the potential use for pre-training of segmentation/classification models on synthetic datasets. SynthCity has been downloaded $\sim$ 5500 times[1]. Our experiments show that the addition of Gaussian noise can increase domain adaptation performance when evaluating a network pre-trained on a synthetic dataset and tested on a real dataset. However, the incorporation of Gaussian noise has a negative correlation when fine-tuning a real dataset from a synthetic dataset. This could be due to a number of factors such as synthetic dataset size and realism or the real test set train-test variability. We find the experiments carried out in this chapter to be a useful starting point for future experiments which will be more conclusive in their findings.

[1] Since its release on 11.09.2019. Download count checked on 15.05.2022.

Chapter 5

Learning the Loss Function for Weak Supervision

5.1 Introduction

Deep learning-based 3D object detection systems require very large manually labelled datasets to achieve competitive performance. Labelling in 3D is, however, a challenging task. As such, most works benchmark their algorithms on a small handful of datasets (e.g., KITTI (Geiger et al., 2013), ScanNet (Dai et al., 2017), S3DIS (Armeni et al., 2016)). Whilst these datasets are a valuable resource for comparing different approaches, they are still relatively small when compared to available datasets in the 2D domain. Furthermore, they are generally domain specific to the sensor and environment in which they are captured. Despite this, unlabelled data can be captured with increasing ease. Considering modern laser scanners can capture millions of points every second, and the incorporation of low-cost active sensors on smart phones, capturing very high quantities of 3D data is extremely accessible. The prospect of being able to learn on unlabelled 3D data, therefore, has become very attractive.

In this chapter we propose a novel weakly supervised learning approach which can be trained on unlabelled data from a similar domain. Our key idea is to first teach a *loss network* to solve a much simpler local problem using a small labelled dataset. Specifically, given a small snippet of a labelled object with a random offset,

learn to point towards the center of the object. Learning this simple task has a powerful use case. The vector from a given position in world-space to an objects center is the same as the gradient for a L2-based supervised cost function (e.g., Mean Squared Error / Chamfer Distance) for a 3D object detector prediction. This network can therefore be used as a cost function for a *scene network* which predicts objects positions from global scenes. The advantage to our approach is that the scene network does not require any labels as all the supervision is restricted to the loss network. The has two primary implications. Firstly, we can learn on unlimited amount of unlabelled 3D data, providing it is within a similar domain to the data used to train the loss network. Secondly, as the loss network is a very local task, it can be trained on significantly less training data as it is not required to learn context or global scene attributes. We show that using only 5% of the available labelled data to train the loss network can achieve competitive performance on challenging benchmarks. Finally, we show that the loss network has good generalisation, and can be used to effectively guide a scene network on novel datasets for which no labels were ever accessed.

5.2 Problem Statement

We aim to solve the task of 3D object detection from a geometric point cloud $\mathcal{P} \in \mathbb{R}^3$. A scene is parameterised as a set on n-objects $\mathbf{o}$ which are themselves parameterised as $\mathbf{o} \in \mathbb{R}^f$ where f is the x, y, z object center, height, width, depth object extent and i, j rotation around the up-axis (z) and confidence $c = (0,1)$. Our scene network mapping is therefore $\mathcal{P} \rightarrow \mathbf{o} \in \mathbb{R}^{n \times 9}$.

5.3 Methodology

We learn two networks: a *scene network* and a *loss network* (Fig. 5.1). The first (Fig. 5.1, bottom) is deployed, while the second (Fig. 5.1, top) is only used in training.

The *scene network* maps 3D scenes to sets of 3D object centers. The input data is a 3D point cloud. The output is a fixed sized list of 3D object centers. We assume a feed-forward approach, that does not consider any proposals Hou et al. (2019);

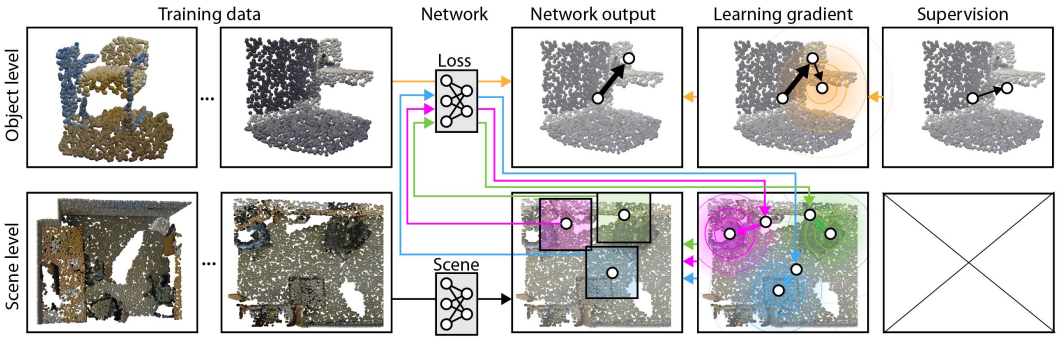


Figure 5.1: Our approach proceeds in two steps of training **(row)** with different training data (column one and two), networks (column three), outputs (column four), gradients (column five) and supervision (column six). Object level training **(first row)** data comprises of 3D scene patches with known objects that are not centered. The loss network maps off-center scenes to their center (big black arrow). Its learning follows the gradient of a quadratic potential (orange field) that has the minimum at the offset that would center the object. This offset is the object-level supervision, as seen in the last column. The scene network **(second row)** is trained to map a scene to all object centers, here for three chairs. The gradient to train the scene network is computed by running the loss network from the previous step once for each object (here three times: blue, pink, green). Note, that there is no scene-level supervision (cross).

Newell et al. (2016); Song, Xiao (2014, 2016) or voting Qi et al. (2019, 2020), but directly regresses centers from input data Zhou et al. (2019a); Yang et al. (2019).

The *loss network* emulates the loss used to train the scene network. The input data is again a 3D point cloud, but this time of a single object, displaced by a random amount and subject to some other distortions. Output is not the scalar loss, but the gradient of a Mean Squared Error loss function.

In the following, we will first describe the training (Sec. 5.3.1) before looking into the details of both the scene and loss network implementation (Sec. 5.3.2).

5.3.1 Training

The key contribution of our approach is a new way of training. We will first look into a classic baseline with scene-level supervision, then introduce a hypothetical oracle that solves almost the same problem and finally show how this problem can be solved without scene-level supervision by our approach.

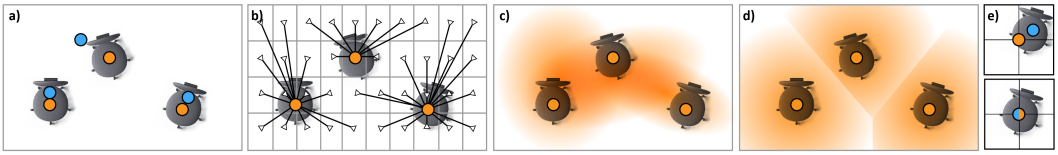


Figure 5.2: a) A 2D scene with three chair objects, supervised by centers (orange) and their predictions (blue). **b)** The same scene, with the vector field of the oracle ∇ shown as arrows. **c)** A 2D Slice through a 6D cost function. **d)** A 2D Slice through an alternative cost function, truncated at the Voronoi cell edges. The oracle is the gradient of this. **e)** The simple task of the loss network: given a chair not in the center (top), regress an offset such that it becomes centered.

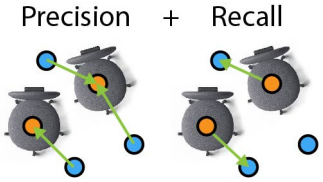


Figure 5.3: Chamfer loss.

5.3.1.1 Supervised

Consider learning the parameters θ of a scene network $\mathcal{S}_\theta$ which regresses object centers $\mathcal{S}_\theta(\mathbf{x}_i) = \hat{\mathbf{c}}$ from a scene $\mathbf{x}_i$. The scene is labelled by a set of 3D object centers $\mathbf{c}_i$ (Fig. 5.2, a). This is achieved by minimising the expectation

$$\arg\min_\theta \mathbb{E}_i[H(\mathcal{S}_\theta(\mathbf{x}_i) - \mathbf{c}_i)], \tag{5.1}$$

using a two-sided Chamfer loss between the label point set $\mathbf{c}_i$ and a prediction $\hat{\mathbf{c}}_\mathbf{i}$

$$H(\hat{\mathbf{c}}, \mathbf{c}) = \mathbb{E}_i[\min_j ||\hat{\mathbf{c}}_i - \mathbf{c}_j||_2^2] + \mathbb{E}_i[\min_j ||\mathbf{c}_i - \hat{\mathbf{c}}_j||_2^2]. \tag{5.2}$$

Under H, the network is free to report centers in any order, and ensures all network predictions are close to a supervised center (precision) and all supervised centers are close to at least one network prediction (recall) (Fig. 5.3).

In this work, we assume the box center supervision $\mathbf{c}_i$ to not be accessible. Tackling this, we will first introduce an oracle solving a similar problem.

5.3.1.2 Oracle

Consider, instead of supervision, an *oracle* function $\nabla(\mathbf{x})$ which returns for a 3D scene $\mathbf{p}$ *the smallest offset by which we need to move the scene so that the world center falls onto an object center* (Fig. 5.2, b). Then, learning means to

$$\arg\min_{\theta} \mathbb{E}_{i,j}[||\nabla(\underbrace{\mathbf{x}_i \ominus \mathcal{S}_\theta(\mathbf{x}_i)_j}_{\mathbf{y}_{\theta,i,j}})||_2^2], \tag{5.3}$$

where $\mathbf{x} \ominus \mathbf{d}$ denotes shifting a scene $\mathbf{x}$ by an offset $\mathbf{d}$. The relation between Eq. 5.1 and Eq. 5.3 is intuitive: knowing the centers is very similar to pointing to the nearest center from every location. It is, however, not quite the same. It assures every network prediction would map to a center, but does not assure, that there is a prediction for every center. We will need to deal with this concern later, by assuring space is well covered, so that there are enough predictions such that at least one maps to every center. We will denote a scene i shifted to be centered around object j by a *scene network* with parameters θ as $\mathbf{y}_{\theta,i,j}$.

Every location that maps to itself, i.e., a *fixed point* Weisstein (2020a) of ∇, is an object center. Hence, we try to get a scene network that returns the roots of the gradient field of the distance function around each object center (Fig. 5.2, c):

$$\arg\min_{\theta} \mathbb{E}_{i,j}[||\nabla(\mathbf{y}_{\theta,i,j})||_2^2]. \tag{5.4}$$

5.3.1.3 Learned loss

The key idea is to emulate this oracle with a *loss* network $\mathcal{L}_\phi$ having parameters ϕ as in

$$\arg\min_{\theta} \mathbb{E}_{i,j}[||\mathcal{L}_\phi(\mathbf{y}_{\theta,i,j})||_2^2]. \tag{5.5}$$

The loss network does not need to understand any global scene structure, it only locally needs to *center* the scene around the nearest object (Fig. 5.2, d). This task can be learned by working on local 3D object *patches*, without scene-level

supervision. So we can train the loss network on any set of objects $\mathbf{o}_k$, translated by a known offset $\mathbf{d}_k$ using

$$\arg\min_{\phi} \mathbb{E}_k[||\mathbf{d}_k - \mathcal{L}_\phi(\mathbf{o}_k \ominus \mathbf{d}_k)||_2]. \tag{5.6}$$

As the loss network is local, it is also only ever trained on 3D patches. These can be produced in several different ways: sampling of CAD models, CAD models with simulated noise, by pasting simulated results on random scene pieces, etc. In our experiments, we use a small, labelled scene subset to extract objects as follows: we pick a random object center and a 3D box of 1 meter size such that at least point representing an object surface is present in the box. Hence the center of the object is offset by a random but known $\mathbf{d}_k$ we regress and subject to natural clutter. Note, that the box does not, and does not have to, strictly cover the entire object – which are of different sizes – but must be just large enough to guess the center. Alg. **??** demonstrates how the *loss network* output can be used to provide *scene network* supervision.

5.3.1.4 Varying object count

The above was assuming the number of objects n_c to be known. It did so when assuming a vector of a known dimension as supervision in Eq. 5.1 and did so, when assuming the oracle Eq. 5.3 and its derivations were returning gradient vectors of a fixed size. In our setting this number is unknown. We address this by bounding the number of objects and handling occupancy i.e., a weight indicting if an object is present or not, at two levels.

First, we train an occupancy branch $\mathcal{O}_\phi$ of the loss network that classifies occupancy of a single patch, much like the loss network regresses the center. We define space to be *occupied*, if the 3D patch contains any points belonging to the given objects surface. This branch is trained on the same patches as the loss network plus an equal number of additional 3D patches that do not contain any objects i.e., occupancy is zero.

Algorithm 1: $\mathcal{L}$: loss network, $\mathcal{S}$: scene network, k: proposal count, n 3D patch point count, m scene point count.

$\mathcal{L}_\phi : \mathbb{R}^{n\times 3} \to \mathbb{R}^3$;
$\mathcal{S}_\theta : \mathbb{R}^{m\times 3} \to \mathbb{R}^{k\times 3}$;
$\texttt{crop} : \mathbb{R}^{m\times 3} \to \mathbb{R}^{n\times 3}$;
while *loss training* **do**
  $x = \texttt{sampleScene}()$;
  $o = \texttt{randObjectCenter}()$;
  $d = \texttt{randOffset}()$;
  $p = \texttt{crop}(x \ominus (o + d))$;
  $\nabla = \frac{\partial}{\partial_\phi} ||\mathcal{L}_\phi(p) - d||_2^2$;
  $\phi = \texttt{optimizer}(\phi, \nabla)$;
end
while *scene training* **do**
  $x = \texttt{sampleScene}()$;
  $c = \mathcal{S}_\theta(x)$;
  for $i = 1 \ldots k$ **do**
    $p = \texttt{crop}(x \ominus c_i)$;
    $\nabla_i = \mathcal{L}_\phi(p)$;
  end
  $\theta = \texttt{optimizer}(\theta, \nabla)$;
end

Second, the occupancy branch is used to support the training of the scene network which must deal with the fact that the number of actual centers is lower than the maximal number of centers. This is achieved by ignoring the gradients to the scene networks parameters θ if the occupancy network reports the 3D patch about a center to not contain an object of interest. So instead of Eq. 5.5, we learn

$$\arg\min_\theta \mathbb{E}_{i,j}[\mathcal{O}_\phi(\mathbf{y}_{\theta,i,j})\mathcal{L}_\phi(\mathbf{y}_{\theta,i,j})]. \tag{5.7}$$

The product in the sum is zero for centers of 3D patches that the loss network thinks, are not occupied and hence should not affect the learning.

Overlap When neither object centers nor their count is known, there is nothing to prevent two network outputs to map to the same center. While such duplicates can to some level be addressed by non-maximum suppression as a (non-differentiable) post-process to testing, we have found it essential to already prevent them (differ-

entiable) from occurring when training the scene network. Without doing so, our training degenerates to a single proposal.

To this end, we avoid *overlap*. Let $v(q_1, q_2)$ be a function that is zero if the bounding boxes of the object in the scene centers do not overlap, one if they are identical and otherwise be the ratio of intersection. We then optimize

$$\arg\min_{\theta} c_1(\theta) = \mathbb{E}_{i,j,k}\left[\mathcal{O}_\phi(\mathbf{y}_{\theta,i,j})\mathcal{L}_\phi(\mathbf{y}_{\theta,i,j}) + v(\mathbf{y}_{\theta,i,j}, \mathbf{y}_{\theta,i,k})\right]. \tag{5.8}$$

We found that in case of a collision instead of mutually repelling all colliding objects, it can be more effective if out of multiple colliding objects, the collision acts on all but one winner object (winner-takes-all). To decide the winner, we again use the gradient magnitude: if multiple objects collide, the one that is already closest to the target i.e., the one with the smallest gradient, remains unaffected ($v = 0$) and takes possession of the target, while all others adapt.

5.3.1.5 Additional features

For other object properties such as size, orientation, class of object, etc. we can proceed in two similar steps. First, we know the object-level property vector $\mathbf{q}$, so we can train a *property branch* denoted $\mathcal{P}_\theta$ that shares parameters θ with the loss network to regresses the property vector from the same displaced 3D patches as in Eq. 5.6

$$\arg\min_{\phi} \mathbb{E}_k[||\mathbf{q}_k - \mathcal{P}_\phi(\mathbf{o}_k \ominus \mathbf{d}_k)||_1]. \tag{5.9}$$

For scene-level learning we extend the scene network by a branch $\mathcal{T}_\theta$ to emulate what the property network had said about the 3D patch at each center, but now with global context and on a scene-level

$$\arg\min_{\theta} c_1(\theta) + \alpha \cdot \mathbb{E}_{i,j}[|\mathcal{T}_\theta(\mathbf{y}_{\theta,i,j}) - \mathcal{P}\phi(\mathbf{y}_{\theta,i,j})|_1]. \tag{5.10}$$

For simplicity, we will denote occupancy just as any other object property and

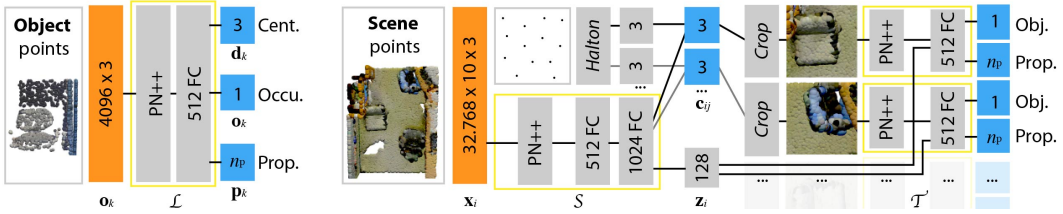


Figure 5.4: The object **(left)** and scene **(right)** network. Input denoted orange, output blue, trainable parts yellow, hard-coded parts in italics. Please see Sec. 5.3.2 for a details.

assume it to be produced by $\mathcal{T}$ just, that it has a special meaning in training as defined in Eq. 5.7. We will next detail the architecture of all networks.

5.3.2 Network

Both networks are implemented using PointNet++ Qi et al. (2017b) optimized using ADAM. We choose particularly simple designs and rather focus on the analysis of changes from the levels of supervision we enable.

Loss and occupancy network The loss network branches $\mathcal{L}$ and $\mathcal{O}$ share parameters ϕ and both map 4,096 3D points to a 3D displacement vector, occupancy and other scalar features (left in Fig. 5.4).

Scene network The scene network branches $\mathcal{S}$ and $\mathcal{T}$ jointly map a point cloud to a vector of 3D object centers and property vectors (including occupancy), sharing parameters θ. The box branch $\mathcal{S}$ first generates positions, next the scene is cropped around these positions and each 3D patch respectively fed into a small PointNet++ encoder $\mathcal{M}$ to produce crop specific local feature encodings. Finally, we concatenate the global scene latent code $\mathcal{S}_z$ with the respective local latent code $\mathcal{M}_z$ and pass it through the scene property branch $\mathcal{T}$ MLP.

The scene property branch is trained sharing all weights across all instances for all objects. This is intuitive, as deciding that e.g., a chair's orientation is the same for all chairs (the back rest is facing backwards), can at the same time be related to global scene properties (alignment towards a table).

Instead of learning the centers, we learn the residual relative to a uniform coverage of 3D space such that no object is missed during training. The Hammersley pattern (Weisstein, 2020b) assures that, no part of 3D space is left uncovered.

We assume a fixed number of 32,768 input points for one scene. Note, that we do not use colour as input, a trivial extension. Each MLP sub-layer is an MLP consisting of 3 fully connected layers where layer 1 has 512 hidden states and the final layer contains the branch specific output nodes.

Post-process Our scene network returns a set of oriented bounding boxes with occupancy. To reduce this soft answer to a set of detected objects, e.g., to compute mAP metrics, we remove all bounding boxes with occupancy below a threshold τ_o, which we set to 0.9 in all our results.

In the evaluation, the same will be done for our ablations SLIDING and SUPERVISED, just that these also require additional Non-maximum Suppression (NMS) as they frequently propose boxes that overlap. To construct a final list of detections, we pick the proposal with maximal occupancy and remove any overlapping proposal with Intersection over Union (IoU) $> .25$ and repeat until no proposals remain.

5.4 Evaluation

We compare to different variants of our approach under different metrics and with different forms of supervision as well as to other methods.

5.4.1 Protocol

Protocol

5.4.1.1 Data sets

We consider two large-scale sets of 3D scanned scenes: Stanford 2D-3D-S dataset (S3D) (Armeni et al., 2017) and ScanNet (Dai et al., 2017). From both we extract, for each scene, the list of object centers and object features for all objects of one class.

We split the dataset in three parts (Fig. 5.5): First, the test dataset is the official test dataset (pink in Fig. 5.5). The remaining training data is split into two parts: a labelled, and and unlabelled part. The labelled part (orange in Fig. 5.5) has all 3D scenes with complete annotations on them. The unlabelled part (blue in Fig. 5.5)

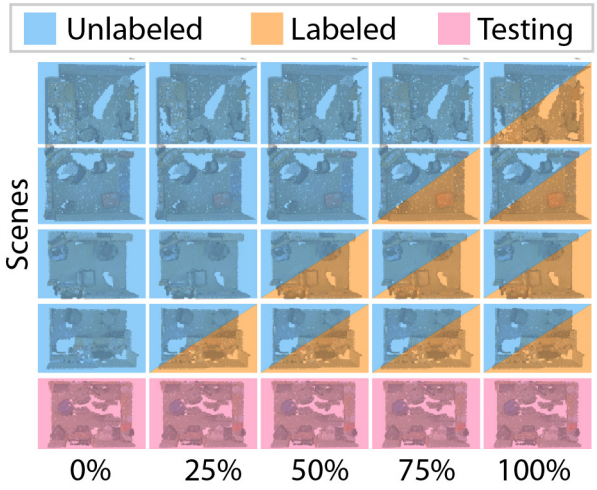


Figure 5.5: Label ratio.

contains only raw 3D point cloud without annotation. Note, that the labelled data is a subset of the unlabelled data, not a different set.

We call the ratio of labelled over unlabelled data the *label ratio*. To more strictly evaluate transfer across datasets, we consider ScanNet completely unlabelled. All single-class results are reported for the class `chair`.

5.4.1.2 Metrics

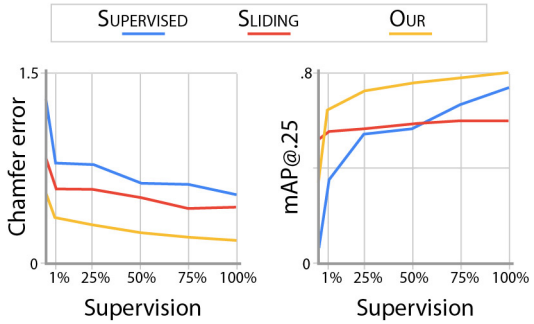
Effectiveness is measured using the Chamfer distance (less is better) also used as a loss in Eq. 5.1 and the established mean Average Precision mAP@.25, (more is better) of a x % bounding box overlap test. X is chosen at 25 %.

5.4.1.3 Methods

We consider the following *three* methods: SUPERVISED is the supervised approach define by Eq. 5.1. This method can be trained only on the labelled part of the training set. SLIDING window is an approach that applies our loss network, trained on the labelled data, to a dense regular grid of 3D location in every 3D scene to produce a heat map from which final results are generated by NMS. OURS is our method. The loss network is trained on the labelled data (orange in Fig. 5.5). The scene network is trained on the unlabelled data (blue in Fig. 5.5), which includes

Table 5.1: Chamfer error (less is better) and mAP@.25 (more is better) (**columns**), as a function of supervision (**rows**) in units of label ratio on the S3D class `chair`. Right, the supervision-quality-relation plotted as a graph for every method (colour).

	Chamfer error			mAP		
Ratio	SUP	SLI	OUR	SUP	SLI	OUR
1 %	1.265	.850	**.554**	.159	**.473**	.366
5 %	.789	.577	**.346**	.352	.562	**.642**
25 %	.772	.579	**.274**	.568	.573	**.735**
50 %	.644	.538	**.232**	.577	.589	**.773**
75 %	.616	.437	**.203**	.656	.592	**.785**
100 %	.557	.434	**.178**	.756	.598	**.803**



the labelled data (but without accessing the labels) as a subset.

5.4.2 Results

Results

5.4.2.1 Effect of supervision

The main effect to study is the change of 3D detection quality in respect to the level of supervision. In Tbl. 5.1, different rows show different label ratios. The columns show Chamfer error and mAP@.25 for the class `chair` trained and tested on S3D.

We notice that across all levels of supervision, OUR approach performs better in Chamfer error and mAP than SLIDING window using the same object training or SUPERVISED training of the same network. It can further be seen, how all methods improve with more labels. Looking at a condition with only 5 % supervision, OUR method can perform similar to a SUPERVISED method that had 20× the labeling effort invested. At this condition, our detection is an acceptable .642,

which SUPERVISED will only beat when at least 75 % of the dataset is labelled. It could be conjectured, that the scene network does no more than emulating to slide a neural-network object detector across the scene. If this was true, SLIDING would be expected to perform similar or better than OURS, which is not the case. This indicates, that the scene network has indeed learned something not known at object level, something about the relation of the global scene without ever having labels on this level.

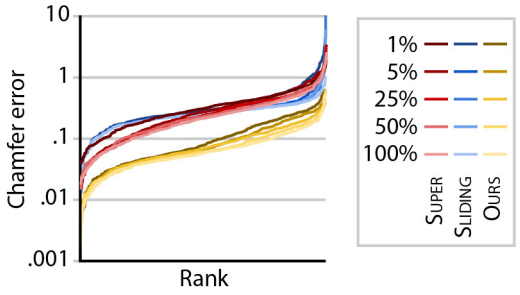


Figure 5.6: Error distributions.

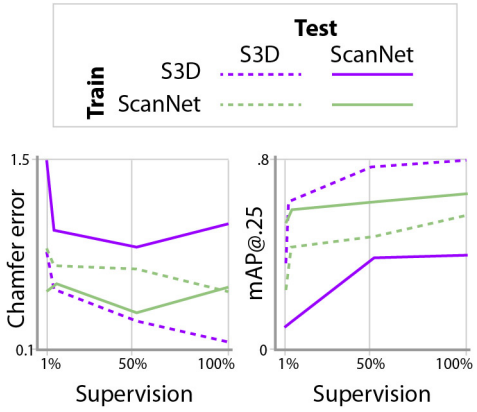
Fig. 5.6 plots the rank distribution (horizontal axis) of Chamfer distances (vertical axis) for different methods (colours) at different levels of supervision (lightness). We see that OUR method performs well across the board, SUPERVISED has a more steep distribution compared to SLIDING, indicating it produces good as well as bad results, while the former is more uniform. In terms of supervision scalability, additional labeling invested into our method (brighter shades of yellow) result in more improvements to the right side of the curve, indicating, additional supervision is reducing high error-responses while already-good answers remain.

5.4.2.2 Transfer across datasets

So far, we have only considered training and testing on S3D. In Tbl. 5.2, we look into how much supervision scaling would transfer to another data set, ScanNet. Remember, that we treat ScanNet as unlabelled, and hence, the loss network will be strictly only trained on objects from S3D. The three first rows in Tbl. 5.2 define the conditions compared here: a loss network always trained on S3D, a scene network trained on either S3D or ScanNet and testing all combinations on both datasets.

Table 5.2: Transfer across datasets: Different rows show different levels of supervision, different columns indicate different methods and metrics. The plot on the right visualizes all methods in all conditions quantified by two metrics. Training either on S3D or on ScanNet. The metrics again are Chamfer error (also the loss) and mAP@.25. colours in the plot correspond to different training, dotted/solid to different test data.

Loss:	S3D				S3D			
Scene:	S3D				SCANNET			
Test:	S3D		ScanNet		S3D		ScanNet	
Ratio	Err.	mAP	Err.	mAP	Err.	mAP	Err.	mAP
1%	0.554	.366	1.753	.112	0.579	.296	0.337	.548
5%	0.346	.642	0.727	.138	0.466	.463	0.703	.599
50%	0.232	.773	0.588	.380	0.447	.497	0.258	.645
100%	0.178	.803	0.789	.384	0.336	.555	0.356	.661

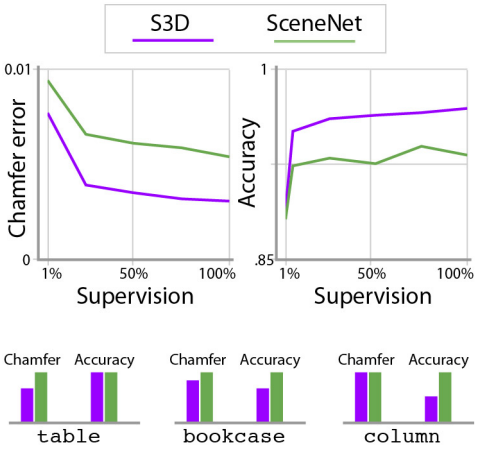


Column two and three in Tbl. 5.2 and the dotted violet line in the plot, iterate the scaling of available label data we already see in Tbl. 5.1 when training and testing on S3D. Columns four and five, show a method trained on S3D but tested on ScanNet. We find performance to be reduced, probably, as the domain of ScanNet is different from the one of S3D. If we include the unlabelled scenes of ScanNet in the training, as seen in columns six to nine, the quality increases again, to competitive levels, using only S3D labels and 0 % of the labels available for ScanNet.

Tbl. 5.3 further illustrate the loss network: how good are we at finding vectors that point to an object center? We see that the gradient error and the confidence error, both go down moderately with more labels when training and testing on S3D (violet). The fact that not much is decreasing in the loss network, while the scene network keeps improving, indicates the object task can be learned from little data,

Table 5.3: Performance of the loss network for different label ratio **(rows)** on different test data and according to different metrics **(columns)**. [1]*Class not present in ScanNet.*

				S3D		ScanNet	
Ratio	Class	#Sce	#Obj	Err.	Acc.	Err.	Acc.
1 %	chair	11	2400	.0085	.853	.0099	.843
5 %	chair	54	16,000	.0052	.936	.0075	.899
25 %	chair	271	47,200	.0049	.949	.0071	.907
50 %	chair	542	121,191	.0046	.953	.0069	.902
75 %	chair	813	162,000	.0045	.955	.0065	.920
100 %	chair	1084	223,980	.0043	.960	.0068	.911
5 %	table	54	5060	.0078	.921	—[1]	—[1]
5 %	bcase	54	4780	.0093	.819	—[1]	—[1]
5 %	column	54	2780	.0100	.855	—[1]	—[1]



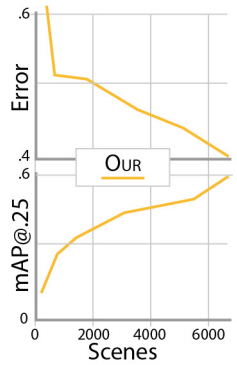
and less object-level supervision is required than what can be learned on a scene level, still. We further see, that the loss network generalizes between datasets from the fact that it is trained on S3D (violet curve) but when tested on ScanNet (green curve) goes down, too.

Besides seeing how quality scales with the amount of labelled supervision for training the loss network, it is also relevant to ask what happens when the amount of unlabelled training data for the scene network is increased while holding the labelled data fixed. This is analysed in Tbl. 5.4. Here we took our loss network and trained it at 5 % label ratio on S3D and tested on ScanNet. Next, the scene network was trained, but on various number of scenes from ScanNet, which, as we said, is considered unlabelled. The number of scenes changes over columns, resp. along the

r

Table 5.4: Chamfer error and mAP@.25 reported for varying the number of scenes.

	OUR	
#Sce	Err.	mAP
66	.643	.079
330	.509	.242
1648	.506	.360
3295	.457	.412
4943	.435	.479
6590	.407	.599



horizontal axis in the plot. We see that without investing any additional labelling effort, the scene network keeps increasing substantially, indicting what was learned on a few labelled S3D objects can enable understanding the structure of ScanNet.

5.4.2.3 Different classes

Tbl. 5.1 was analyzing the main axis of contribution: different levels of supervision but for a single class. This has shown that at around a label ratio of 5 % OUR method performs similar to a SUPERVISED one. Holding the label ration of 5 % fixed and repeating the experiment for other classes, is summarized in Tbl. 5.5. We see, that the relation between SUPERVISED, SLIDING and OURS is retained across classes.

5.4.2.4 Comparison to other work

In Tbl. 5.6 we compare our approach to other methods. Here, we use 20 % of ScanNet V2 for testing and the rest for training. Out of the training data, we train our approach once with 100 % labelled and once with only 5 % labelled. Other

Table 5.5: Chamfer error (less is better) and mAP@.25 precision (more is better) **(columns)**, per class **(rows)** at a supervision of 5 % labeling ratio.

	Chamfer error			mAP		
Class	SUP	SLI	OUR	SUP	SLI	OUR
chair	0.789	0.577	.346	.352	.562	.642
table	1.144	1.304	.740	.282	.528	.615
bookcase	1.121	1.427	.979	.370	.298	.640
column	0.900	2.640	.838	.490	.353	.654

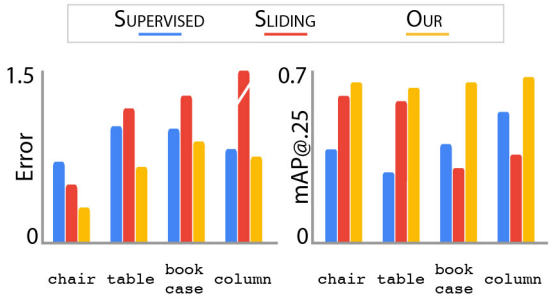


Table 5.6: Performance (mAP(%) with IoU threshold .25) of different methods **(rows)** on all classes **(columns)** of ScanNet V2. [1]*5 images*. [2]*Only xyz*. [3]*Their ablation; similar to our backbone*.

Method		cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	fridge	curtain	toilet	sink	bathtub	other	mAP
3DSIS[1]	Hou et al. (2019)	19.8	69.7	66.2	71.8	36.1	30.6	10.9	27.3	0.0	10.0	46.9	14.1	53.8	36.0	87.6	43.0	84.3	16.2	40.2
3DSIS[2]	Hou et al. (2019)	12.8	63.1	66.0	46.3	26.9	8.0	2.8	2.3	0.0	6.9	33.3	2.5	10.4	12.2	74.5	22.9	58.7	7.1	25.4
MTML	Lahoud et al. (2019)	32.7	80.7	64.7	68.8	57.1	41.8	39.6	58.8	18.2	0.4	18.0	81.5	44.5	100.0	100.0	44.2	100.0	36.4	54.9
VoteNet	Qi et al. (2019)	36.3	87.9	88.7	89.6	58.8	47.3	38.1	44.6	7.8	56.1	71.7	47.2	45.4	57.1	94.9	54.7	92.1	37.2	58.7
BoxNet[3]	Lahoud et al. (2019)	No per-class information available																		45.4
3D-BoNet	Yang et al. (2019)	58.7	88.7	64.3	80.7	66.1	52.2	61.2	83.6	24.3	55.0	72.4	62.0	51.2	100.0	90.9	75.1	100.0	50.1	68.7
Ours 100 %		43.0	70.8	58.3	16.0	44.6	28.0	13.4	58.2	4.9	69.9	74.0	75.0	36.0	58.9	79.0	47.0	77.9	48.2	50.2
Ours 5 %		38.1	68.9	58.9	88.8	42.5	21.1	9.0	53.2	6.8	53.9	68.0	62.3	26.5	45.6	69.9	40.4	66.9	48.0	48.3

methods were trained at 100 % label ratio.

We see that our approach provides competitive performance, both at 100 % of the labels, as well as there is only a small drop when reducing supervision by factor 20×. Our mAP at 100 % of the labels is better than both variants (with and without colour) of 3DSIS Hou et al. (2019) from 2018 and similar to MTML Lahoud et al. (2019) from 2019. VoteNet Qi et al. (2019) and 3D-BoNet Yang et al. (2019) are highly specialised architectures from 2019 that have a higher mAP. We have included BoxNet from Qi et al. (2019), an ablation they include as a vanilla 3D detection approach that is similar to what we work with. We achieve similar even

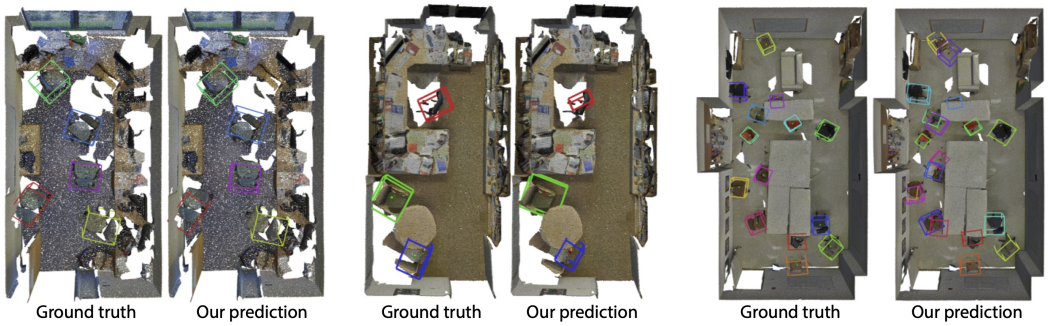


Figure 5.7: Qualitative results of our approach and the ground truth for `chair` on S3D.

slightly better performance, yet at 5 % of the supervision. In some categories, our approach wins over all approaches. We conclude that a simple backbone architecture we use is no contribution and cannot win over specialised ones, but that it also is competitive to the state-of-the-art. We should note here, as we do not carry out Semantic instance segmentation in our network, we did not test on the official test ScanNet benchmark test set. Instead, we reserve 20% of the labelled training scenes for testing.

5.4.2.5 Qualitative results

Fig. 5.7 shows qualitative example results of our approach.

5.4.2.6 Computational efficiency

Despite the additional complexity in training, at deployment, out network is a direct and fast forward architecture, mapping a point cloud to bounding boxes. Finding 20 proposals in 32,768 points takes 189 ms, while the supervised takes the same amount of time, with the small overhead of a NMS (190 ms) on a Nvidia RTX 2080Ti. Our CPU implementation of sliding window requires 4.7 s for the same task on a i7-6850K CPU @ 3.60GHz. All results are computed with those settings.

5.5 Discussion

How can OURS be better than the SUPERVISED? It is not obvious why at 100 % label ratio in Tbl. 5.1, the SUPERVISED architecture performs at an mAP of .756

while OURS remains slightly higher at an mAP of .803. This is not just variance of the mAP estimation (computed across many identical objects and scenes).

A possible explanation for this difference is, that our training is no drop-in replacement for supervised training. Instead, it optimizes a different loss (truncation to the nearest object and collision avoidance) that might turn out to be better suited for 3D detection than what it was emulating in the beginning. We, for example, do not require NMS. As our training does not converge without those changes to the architecture, some of the effects observed might be due to differences in architecture and not due to the training. We conjecture future work might consider exploring different losses, involving truncation and collision, even when labels are present.

Why Hammersley? Other work has reasoned about what intermediate points to use when processing point clouds. When voting Qi et al. (2019), the argument is, that the centers of bounding boxes are not part of the point set, and hence using a point set that is any subset of the input is not a good solution. While we do not vote, we also have chosen not to use points of the scene as the initial points. We also refrain from using any improved sampling of the surface, such as Poisson disk (Hermosilla et al., 2018) sampling as we do not seek to cover any particular instance but space in general, covered by scenes uniformly.

How can the scene network be "better" than the loss network? As the loss network is only an approximation to the true loss, one might ask, how a scene network, trained with this loss network, can perform better than the loss network alone, e.g., how can it, consistently (Tbl. 5.1, 5.2, 5.4 and 5.5), outperform SLIDINGWINDOW?

Let us assume, that a scene network trained by a clean supervision signal can use global scene structure to solve the task. If now the supervision signal would start to be corrupted by noise, recent work has shown for images Lehtinen et al. (2018) or point clouds Hermosilla et al. (2019), that a neural network trained under noise will converge to a result that is very similar to the clean result: under $\mathcal{L}_2$ it will converge to the mean of the noise, under $\mathcal{L}_1$ to its median, etc. The amount of variance of that noise does not influence the result, what matters is that the noise is unbiased. In our case, this means if we were to have supervision by noisy bounding boxes, that

would not change anything, except that the scene network training would converge slower but still to the mean or median of that noise distribution, which is, the correct result. So what was done in our training, by using a network to approximate the loss, means to just introduce another form of noise into the training.

5.6 Conclusion

In this chapter, we have proposed a novel training procedure to reduce the 3D labelling effort required to solve a 3D detection task. The key is to first learn a loss function on a small labelled local view of the data (objects), which is then used to drive a second learning procedure to capture global relations (scenes). The way to enlightenment here is to "find your center": the simple task of taking any piece of 3D scene and shifting it so it becomes centered around the closest object. Our analysis indicates that the scene network understands global scene structure not accessible to a sliding window. Our network achieves state of the art results, executes in a fraction of a second on large point clouds with typically only 5 % of the labelling effort. We have deduced what it means exactly to learn the loss function, the new challenges associated with this problem and proposed several solutions to overcome these challenges.

Chapter 6

Object Detection with Single Geometric Examples

6.1 Introduction

In Chapter 5 we observed that deep learning systems can be trained on significantly less labelled supervision, when compared to standard dense supervised learning. In this chapter we move further in this direction and develop a methodology to infer 3D scene properties using the input data as the only supervision signal. Furthermore, we restrict the input data to 2D images of the 3D scene. This task is commonly known as 3D monocular object detection and despite being a long-standing problem, is still a very activate research area. (Park et al., 2021; Zhang et al., 2021b,a; Liu et al., 2021).

The above approaches all come at the cost of manual labelling of 3D supervision, preventing application to the "heavy tail" of important tasks where no 3D scene supervision (position, orientation, colour etc.) is available, and users have to make do with no more than a set of 2D images. For this reason, many works attempt to infer 3D properties with 2D-only self-supervision (Kato et al., 2018; Chen et al., 2019; Henzler et al., 2019; Tulsiani et al., 2017; Henderson, Ferrari, 2019; Han et al., 2020). In this paper, we address the common situation where the geometry of the object is known, however, scene labels are not available. Such scenarios are common in industrial environments where components are designed

using Computer Aided Design software, as well as scenarios where object geometry is consistent (i.e., cars, street furniture etc.). Armed with just a 3D geometric representation and unlabelled 2D images, we aim to recover the position, orientation, colour, and illumination for every relevant object in the scene.

Our approach uses a common Convolutional Neural Network (CNN) and multi-layer perceptron (MLP) network to map an image to an explicit and interpretable 3D scene code that controls a Differentiable Renderer (DR). This scene code, like a scene graph, has direct meaning and can be used in other downstream tasks such as loading it into a 3D graphics application for manipulation or re-synthesis with many applications in augmented and virtual reality. Alternatively, the scene code itself can be used directly for 3D scene understanding tasks such a 3D object detection.

This problem is difficult, as unfortunately, direct application of a DR for self-supervised learning will not result in a reliable optimisation. Consider (Fig. 6.1, left), a typical 3D scene made of the 3D position of a quad and its RGB colour, an apparently simple 6D problem, and further consider this is to be learned from 2D images of the 3D quad in front of a solid background. Now, if gradient descent starts to minimise a L_2-like error (L_1, L_2, Huber norm, etc.), in almost every case, the initial guess is far off the right values ("Iter 1" to "Iter n" in Fig. 6.1). To satisfy a L_2-like loss, the best thing to do is to make the quad small in 2D, e.g., by moving it away from the viewpoint, and have the colour become the one of the background at that image position, even if it is not at all the position of the quad in the image and neither its colour. This is just one of the many ambiguities that exist in the mapping between scene parameters and images. Essentially, in almost every position of the optimisation space, following the gradient leads to an unusable solution. Furthermore, even if for one optimisation step there is an overlap and the gradient was meaningful, it will be followed by too many bad ones to be useful.

We found this problem can be overcome by enforcing samples generated by the DR to match the distribution of the input dataset. We implement this by introducing a second network, akin to a critic (or discriminator) in adversarial train-

ing Goodfellow et al. (2014). The by-the-book example for adversarial training is super-resolution Ledig et al. (2017), a low-resolution patch can be explained by many high-resolution patches, but their average, which a network without a critic will find, is not a valid high-resolution patch. In our case, however, the degenerate solutions (e.g., small "mini chameleon" quads hiding in the back) are valid samples from the data distribution. Nothing is wrong with an individual sample. They are also not an average of many solutions, they are not blurry, they are just small quads as they occur in the data distribution. The key is, that the overall distribution of such solutions, rendered back to an image, is far from the original data distribution. Hence, a critic will push the optimisation to not rely on the same (and incorrect) answer all the time and be "curious" instead. It forces the optimiser to continue looking for a better solution within the data distribution. Eventually, the optimisation discovers a solution that is not always the same, and ultimately, even more correct in the L_2-like sense. The critic avoids the "easy reward" from following the gradient of the L_2-like loss. Our instrumentation assures this cannot be achieved by just weighting L_2 differently, by using higher learning rate, more randomisation, longer learning, or other hyper-parameters.

Whilst our scene parameterization can be used for many downstream tasks, we evaluate our method on the most direct application of the parameterization itself, 3D object detection. Specifically, we tackle the problem of 3D object detection from monocular 2D images learned from 2D-only self-supervision with known geometry. We evaluate our approach on increasingly complex synthetic scenes as well as real scenes where the background is constant. Finally, we demonstrate the effectiveness of our approach where the background is also unknown by randomly generating training data from the full distribution of possible parameters using a simple rendering pipeline on cluttered backgrounds. We find such an approach achieves state-of-the-art results for unsupervised methods on the LineMOD dataset Hinterstoisser et al. (2011).

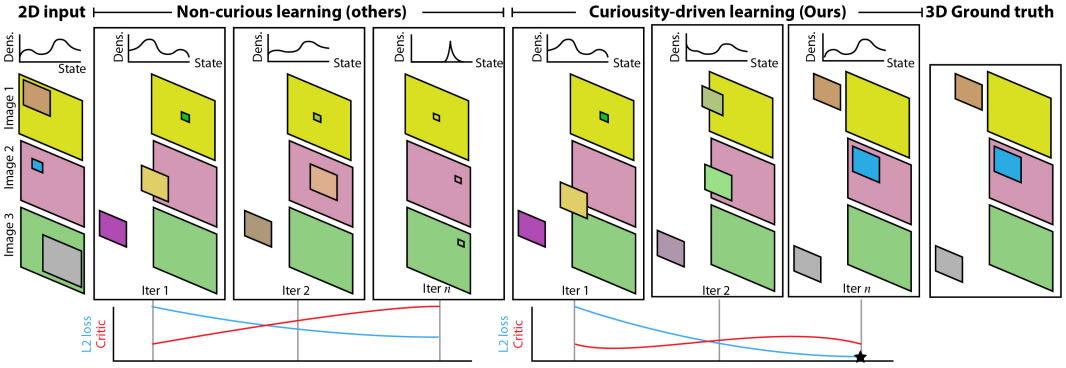


Figure 6.1: Curiosity-driven and direct learning: This simple world comprises of quads with different positions and depth as well as varying colour. Starting from a set of 2D images (three shown), we could train a mapping to directly produce 3D scene parameters (position and colour of the quad). Unfortunately, L_2 rendering gradients will make the optimisation converge (blue plot) to the wrong solution of very small quads with the colour of the background in almost all cases. Our curiosity-driven approach adds a critic to look at the re-rendering of the 3D scene parameters, forcing the optimisation to keep trying unless it finds a solution that matches the 2D image data distribution (red plot). Such learned parameters start to have meaningful gradients, and following them, ultimately, leads to a better solution (star).

6.2 Problem Statement

We aim to solve the task of 3D monocular object detection from a coloured RGB image $\mathcal{I} \in \mathbb{R}^{h,w,3}$. A scene is parameterised as a set on n-objects $\mathbf{o}$ which are themselves parameterised as $\mathbf{o} \in \mathbb{R}^f$ where f is the x, y, z object center and θ rotation. Our solution requires only a geometric representation $\mathcal{G}$ which in our case is triangulated mesh. Supervised training is replaced with an analysis-by-synthesis render-and-compare approach using a differentiable renderer $\mathcal{R}$. Our optimisation is formally

$$\arg\min_{\theta} = \mathbb{E}[||\mathcal{I}_i - \mathcal{R}(E(\mathcal{I}_i), \mathcal{G})||_2^2]$$

where E is a CNN encoder with learnable parameters θ.

6.3 Differentiable Rendering with Curiosity

Input to our network G is a 2D image I and output is a scene code $\mathbf{s}$. The network is a composition $G(I) = \mathcal{R}(E(I))$ of an encoder E and a DR $\mathcal{R}$. The encoder is a

CNN that has learnable parameters θ to first map to a latent code $\mathbf{z}$ and an MLP to map this code to an explicit scene code $\mathbf{s}$. The DR has no learnable parameters, and additionally, requires only a geometric representation of desired objects. At test time the rendering (and therefore geometric representation) is not necessary for tasks only requiring the scene code $\mathbf{s}$ (e.g., 3D object detection). However, for re-synthesis tasks, we can directly modify individual nodes $\mathbf{s}_i$ of the scene code $\mathbf{s}$ to manipulate specific elements on the input image, such as shape, colour, and orientation.

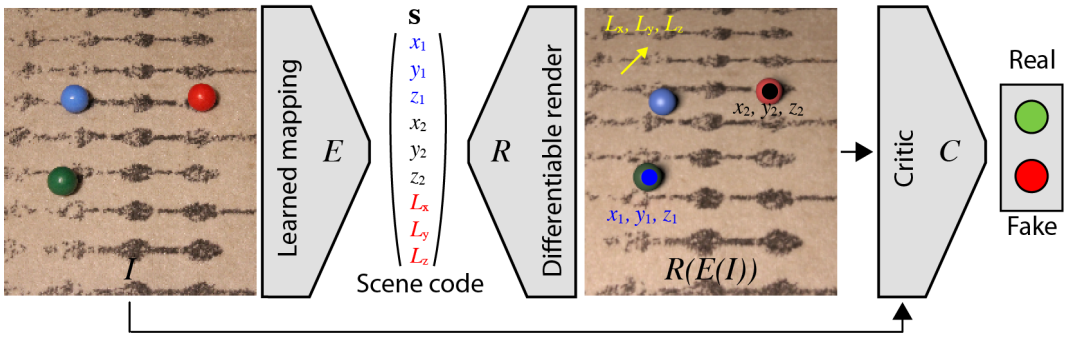


Figure 6.2: Our architecture is simple. First, a learned mapping encodes an RGB image to scene parameters and second, a fixed differentiable renderer maps the predicted parameters to an image. Loss is between input and output images, and at deployment we are interested in the scene code for downstream tasks. Training this analysis-by-synthesis loss without a critic influencing the generated data distribution will almost never converge due to ambiguity (see Fig. 6.1).

Common analysis-by-synthesis would minimise

$$\mathcal{L}(\theta) = ||G_\theta(I) - I||_2.$$

We do not seek to minimise another criterion and would argue there is no need to do so.

The problem is that for almost every possible parameter θ, the gradients $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta}$ will make the result worse due to the countless ambiguities in image formation. Consider for example, a badly placed circle resulting from a choice of parameters θ_1 in a 2D problem of position θ_x and radius θ_r. No small change to its parameters will improve the L2 error with respect to the input image. Instead, gradients will point towards a solution where the sphere just gets smaller or hides in the back-

ground colour (if this is part of the scene vector). However, there are positions θ_2, where gradients indeed lead to improvement, i.e, when the scene code places the re-rendering of the sphere "closer" (this is not limited to space but happens in the high-dimensional scene code space) to the correct solution. How can we distinguish those sub-spaces of the solution space from others and gear the optimisation to follow these?

The key observation is that the re-rendered solutions from invalid local minima —in a reverse Anna Karenina principle fashion, according to which all good solutions would be similar and all bad ones unique— will result in a distribution of re-rendered scenes that are all very similar, e.g., all spheres would come out small. And this distribution is different from the input distribution. So, all that is needed to push the distributions to become similar, is a second network to compare the data and generated distribution (a critic, in a Generative Adversarial Network (GAN)):

$$\mathcal{L}(\theta) = ||G_\theta(I) - I||_2 + \lambda \cdot C(G_\theta(I)).$$

Our curiosity term is versatile and can influence our scene parameterization network in various ways, depending on the task. For example, if $\mathbf{s} \in \mathbb{R}^6$ our critic will enforce the distribution to match both position (x, y, z) and colour (r, g, b) of the input distribution. On the other hand, if $\mathbf{s} \in \mathbb{R}^6$, but all objects in the input images are the same colour, the critic would constrain the parameterization network to always produce predictions with a single fixed object colour.

In practice our critic is a simple fully-convolutional CNN encoder $C_\theta : I \in \mathbb{R}^{h \times w \times 3} \rightarrow \mathbb{R}^{1 \in [0,1]}$.

6.3.1 Differentiable Rendering with Confidence

In the above formulation, analysis-by-synthesis with curiosity works well if we have a fixed-size parameter vector, however, would struggle when the number of objects in the scene is unknown. In the 2D or 3D object detection literature, this is routinely resolved by working with a number of proposals which all carry a confidence (Qi et al., 2019). This is then used to suppress objects with a low confidence (visible to

the loss or not). However, without scene-level labels, we not only lack information of object parameters, but also the object count for a given scene.

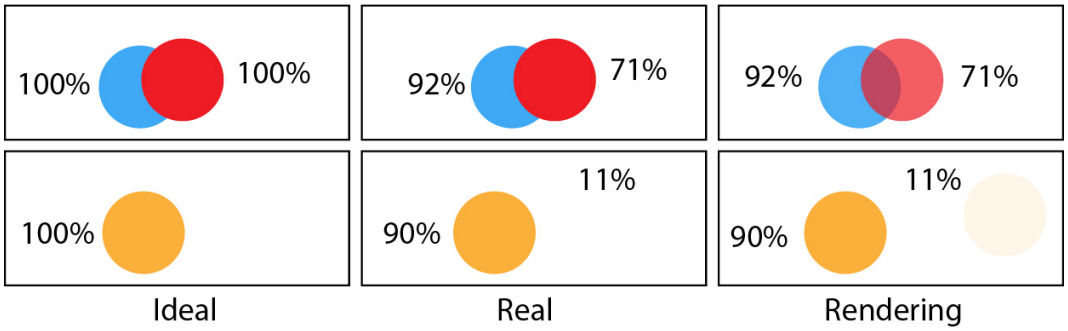


Figure 6.3: Confidence in the ideal and real case, and how it is learned. Top row shows a case of two spheres in a scene, bottom row shows a case of one sphere in scene.

Consider a network which outputs a confidence for every object as part of the scene code $\mathbf{s}$. Ideally, it would be 0 for proposals where no object is present, and 1 for correct proposals, as seen in Fig. 6.3, left, but in practice is a fraction as seen in Fig. 6.3, middle. This confidence has to affect how the scene is rendered if we want it to be learned. Simply turning an object on and off based on any threshold is not differentiable. Instead, we suggest a soft rendering ($\mathcal{R}_c$) which enables the network to learn to hide redundant predictions.

The easiest way to implement a soft rendering would be to modify the objects transparency in the DR. However, as the DR we use in our experiments (Li et al., 2018a) does not support transparency, we emulate the behavior as follows: we first render the scene without any objects and then with every object in isolation, all with an alpha channel. These images are then composed back-to-front using confidence as alpha, which is a differentiable operation itself. This is correct for first-hit rays, but incorrect for the compositing of higher-order shading (a confident object casting no shadow at one pixel, might override the correct shadow of another confident object). In the future we expect DRs to support transparency, allowing us to benefit from guidance by shadow and global illumination for multiple objects. Note that all results where the object count is known, do have correct unbiased image synthesis and benefit from shadow, reflection, etc. informing the loss.

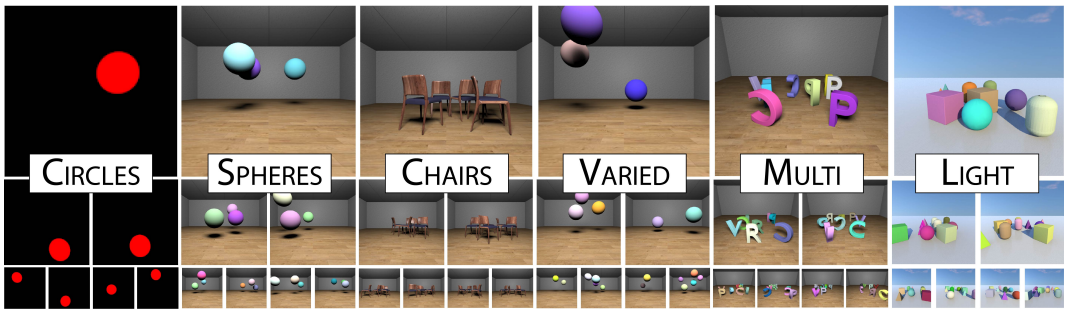


Figure 6.4: Samples from the different datasets we study.

6.4 Implementation

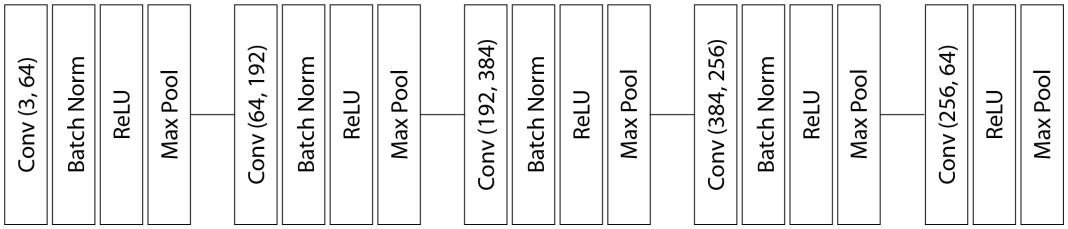
6.4.1 Network architecture

In this section we give further details of our neural network architectures. For simplicity we ignore the batch dimension.

Our encoder is an approximation of the AlexNet architecture. Input is a 3-channel colored image with size 128×128. At each layer we reduce the image dimensions using strided convolutions, avoiding the need for max pooling operations. We use a Rectified Linear Unit (ReLU) for all non-linear activations. Whilst a deeper network, or one with more modern operations (e.g., Residual connections), would likely have helped overall performance, we aimed to keep are architecture as simple as possible to explore the relative benefits of our proposed learning approach.

Table 6.1: AlexNet encoder architecture details.

	Layer	Out Size	Batch Norm	Activation
C1	Conv 2D	[64, 30, 30]	True	ReLU
C2	Conv 2D	[192, 14, 14]	True	ReLU
C3	Conv 2D	[384, 7, 7]	True	ReLU
C4	Conv 2D	[256, 4, 4]	True	ReLU
C5	Conv 2D	[64, 1, 1]	False	ReLU
			Parameters:	2,029,056

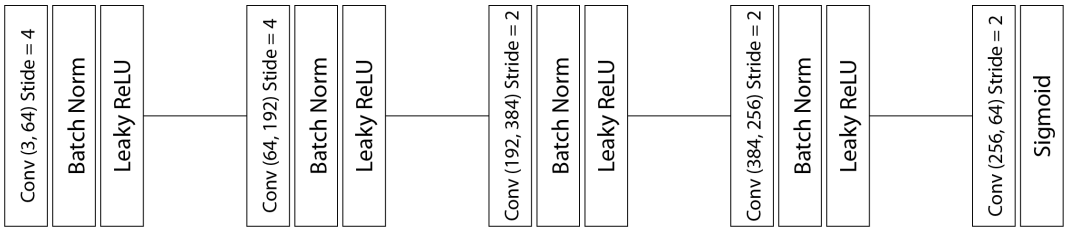


Our critic architecture is fully convolutional, and therefore contains no fully

connected layers. Unlike the image encoder we find Leaky ReLU activation functions to be more effective. The critic also contains no max pooling layers, using convolutions with either stride=2 or stride=4 to reduce the image size. Input is a 3-channel coloured image with size 128×128. In particular with the critic, we found normalisation to be important. Whilst both Instance Normalisation and Batch Normalisation proved to be effective, all results in the paper were obtained using Batch Normalisation.

Table 6.2: Critic architecture details.

	Layer	Out Size	Batch Norm	Activation
C1	Conv 2D	[64, 30, 30]	True	Leaky ReLU
C2	Conv 2D	[192, 7, 7]	True	Leaky ReLU
C3	Conv 2D	[384, 4, 4]	True	Leaky ReLU
C4	Conv 2D	[256, 2, 2]	True	Leaky ReLU
C5	Conv 2D	[64, 1, 1]	False	Sigmoid
			Parameters:	1,883,713



The parameter prediction MLP contains a single shared trunk layer followed by respective parameter branches. No weights are shared on any branch layers. After the shared layers (trunk) we resize the vector to size $n \times d$ where n is the number of object predictions and d number of feature dimensions. We find this effective as it enables each prediction to have its own independent feature transformation. `Center` predicts the per-object translation offset in camera space. `RGB` predicts an per-object RGB colour. `Confidence` predicts a per-object confidence which is subsequently used as an opacity value for differentiable rendering with varying objects. `Light` predicts a per-scene light direction along a hemisphere above the scene.

For experiments where a canonical mesh orientation, and ground plane are assumed, `Rotation` predicts the per-object i, j rotation vector direction. The final rotation r around the y-axis is calculated as $r = \arctan2(j, i)$. For experiments where this is not the case (BUNNY, LINEMOD) we instead using a 6D rotation represen-

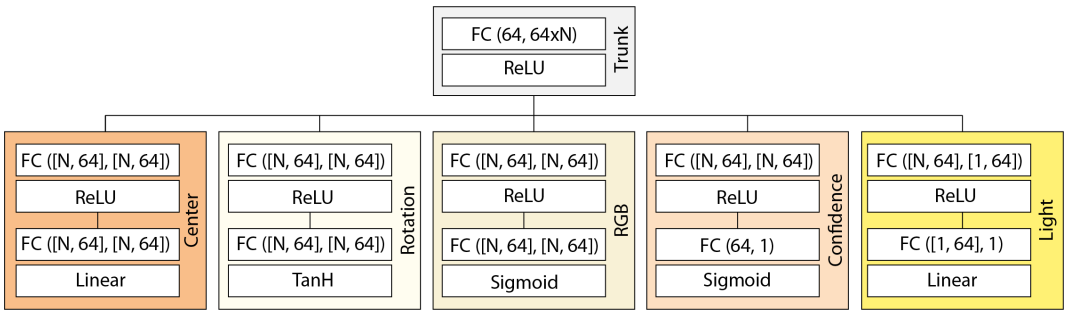
tation as proposed by Zhou et al. (2019b). We choose this representation as it is continuous unlike other representations (Euler angles, rotation matrix, Quaternions etc.), which is more suitable for gradient decent optimisations.

We do not use any normalisation in between MLP layers.

Table 6.3: Parameter MLP network architecture.

	Layer	Out Size	Activation
Trunk	Fully Connected	[64, 1]	ReLU
`Center`	Fully Connected	[n, 64]	ReLU
	Fully Connected	[n, 3]	Linear
`Rotation`	Fully Connected	[n, 64]	ReLU
	Fully Connected	[n, 2]	TanH
`RGB`	Fully Connected	[n, 64]	ReLU
	Fully Connected	[n, 3]	Sigmoid
`Confidence`	Fully Connected	[n, 64]	ReLU
	Fully Connected	[n, 1]	Sigmoid
`Light`	Fully Connected	[1, 64]	ReLU
	Fully Connected	[1]	Linear

Parameter range: 4,387 - 48,266



6.4.2 Training details

All networks were implemented in the PyTorch v1.7 framework. A list of hyperparameters are shown in Tab. 6.4. We trained all models until a convergence threshold based on the image loss is reached. Due to the nature of generator-critic min-max games, we found the critic loss to be an unreliable indicator of convergence. Network training times took between 12-36 hours on a single Nvidia 2080ti depending on task. To allow for a high batch size we utilise virtual batch sizes to avoid running into memory constraints. We achieve this by running multiple forward passes with a smaller batch size and accumulating the gradients before running a backwards pass through our network.

Table 6.4: List of hyperparameters used to train networks.

Hyperparameter	Value
Batch size	128
Optimizer	Adam (β_1: 0.9, β_2: 0.999)
Generator Learning rate	1e-4
Critic learning rate	1e-6
Image Loss λ	0.01
Critic loss λ	10
Gradient L2 clipping	0.5

6.5 Results

We will first analyse our approach on synthetic scenes (Sec. 6.5.1) before applying it to real photos (Sec. 6.5.2). Finally, we evaluate our approach on the popular LineMOD dataset for 6-DOF pose estimation of single objects.

6.5.1 Analysis

Tasks We consider a synthetic dataset of renderings of 3D scenes with different parameters (Fig. 6.4 and Tbl. 6.5).

CIRCLES (●) is a single red 2D circle of constant radius in front of a black background. SPHERE (◆) is a 3D world with three spheres of varying colour. Objects in this task and all following are placed via rejection sampling such as to not intersect. CHAIRS (✷) is a 3D world with five chairs placed on the ground plane at different positions and orientations. VARIED (✷) comprises of a varying number of between 2 and 5 spheres of random position and colour. This is the first task where confidence-based rendering (Sec. 6.3.1) is used. MULTI (▼) has eight objects: four sets of two letters (`WXYZ`), with random colours, placed on the ground plane at random positions and orientations. LIGHT (★) is a 3D world with a varying number of different objects in it (capsules, boxes, cones), all at random positions and orientations on a ground plane and illuminated by a single changing "sunlight" illumination, that is part of the scene description.

For every task we consider a set of 2000 images, labelled with the scene parameters (hidden to our training) with 500 images for validation and 500 images for testing. We will later consider fractions of this supervision.

Table 6.5: Latent scene code structure.

Dataset	DoF	From n	To n	Real	3D	Position	Colour	Rotate	Shape	Light
CIRCLES ●	2	1	1	✗	✗	✓	✗	✗	✗	✗
SPHERE ◆	6	3	3	✗	✓	✓	✓	✗	✗	✗
CHAIRS ✹	5	5	5	✗	✓	✓	✗	✓	✗	✗
VARIED ✹	7	2	5	✗	✓	✓	✓	✗	✗	✗
MULTI ▼	9	8	8	✗	✓	✓	✓	✓	✓	✗
LIGHT ★	11	2	10	✗	✓	✓	✓	✓	✓	✓
SPHERE REAL	6	3	3	✓	✓	✓	✓	✗	✗	✗
BUNNY	6	1	1	✓	✓	✓	✓	✓	✗	✗
SHAPES	6	5	12	✓	✓	✓	✓	✓	✓	✗

Methods We consider three methods: The first is a hypothetical baseline that has access to the ground truth scene parameters learned with a supervised loss. We study three variants of this method which use 5% (`Super5`), 10% (`Super10`) and 100% of the supervision. While our method is trained from images alone, we have to define a supervised loss for this baseline. Direct L_2 between scene parameters cannot work, as it implies object order. Direct application of a Chamfer loss based on position is not practical as it would ignore all non-positional attributes, and it would not be clear how to handle global attributes like light direction. Hence, we first compute the optimal assignment P between n_o objects according to only position $\mathbf{x}$, and then a $\mathbf{w}$-weighted norm between the scene object attributes $\mathbf{y}$, including position, and the global attributes $\mathbf{z}$ of scene A and scene B, paired by P, as in

$$\mathcal{L}(A,B) = \sum_{i=1}^{n_\text{o}} \sum_{j=1}^{n_\text{p}} w_j ||\mathbf{y}^A_{i,j} - \mathbf{y}^B_{P_i,j}|| + ||\mathbf{z}^A - \mathbf{z}^B||,$$
$$\text{with} \qquad P = \arg\min_Q \sum_{i=1}^{n_\text{o}} ||\mathbf{x}^A_i - \mathbf{x}^B_{Q_i}||. \tag{6.1}$$

`NonCur` is a basic self-supervised method, without our curiosity term (L2 loss only) while `Our` is our full method (L2 and critic).

All methods use the same architecture, and only differ by their supervision. Input is a 3-channel coloured image with size 128×128 image, that is reduced to a

single latent code **z** of 64 dimensions in 7 steps (with Batch norm and ReLU). An MLP with three non-linear projection layers maps the latent code **z** into a scene code **s** of a size that depends on the task (between 2 and 11 values, see Tbl. 6.5) to control the DR (Fig. 6.2). The first layer of the MLP is a shared trunk, while the second and third layers produce the respective scene parameters used in independent branches for each group: one for 3D position, 2D rotation, 3D colour, 1D confidence and 2D light.

Metrics We apply two different metrics: Eq. 6.1 on the resulting scene parameters and image error, where we render the scene from a novel viewpoint (to avoid single image scale ambiguity) with the estimated and the ground truth scene parameters and compare the images using Structural Dissimilarity (DSSIM). We report numbers as a ratio relative to the fully supervised method on that task (not shown; it always is 1.00), as this is the upper bound of our relatively simple detection network.

Findings We computed the result of all methods on all synthetic datasets for all metrics (Tbl. 6.6). The main finding is, that `Our` method with curiosity performs better than `NonCur` which has no curiosity. This is compared with methods that have supervision (and do not need curiosity) at 5 % (`Super5`) and 10 % (`Super10`) of the data. Unsurprisingly, more supervision leads to lower errors. The performance of full supervision is shown as a thick line in the plot of Tbl. 6.6 and matches 1.00. We see that in most tasks and for several metrics our method can perform similar to the 100 %-supervised baseline, while it had no access to scene labels. In general, the difference in scene parameter error is larger than the one in image error. In particular, the scene parameter error is too high for `NonCur` to be plotted. More surprisingly, and affirmative, differences according to both metrics seem to be diminishing when the task gets more complex, e.g., comparing the progression of ● Circles / ◆ Spheres / ✹ Chairs vs. ● Circles / ◆ Spheres / ✹ Chairs, we see that the gap between supervision and no supervision closes, a progression found for both metrics.

Fig. 6.10 shows qualitative results for selected tasks. The parameter vector

Table 6.6: Different methods and supervisions *(columns)* according to different metrics for different data sets *(rows)*. The plot shows the 100 % supervision-reference as a thick line.

	Super5		Super10		NonCur		Our	
	Img	Para	Img	Para	Img	Para	Img	Para
CIRCLES●	2.31	8.69	1.61	3.51	5.08	3621.83	0.94	1.11
SPHERES◆	1.67	2.33	1.59	2.29	1.68	8.97	1.19	1.87
CHAIRS✹	1.06	3.48	1.04	1.54	2.23	51.78	1.03	1.40
VARIED✸	1.48	10.61	1.27	5.84	2.28	22.47	1.25	5.04
MULTI▼	1.73	2.59	1.28	1.78	2.11	18.98	1.14	1.51
LIGHT★	2.55	5.44	1.24	2.53	3.09	18.65	1.12	2.05

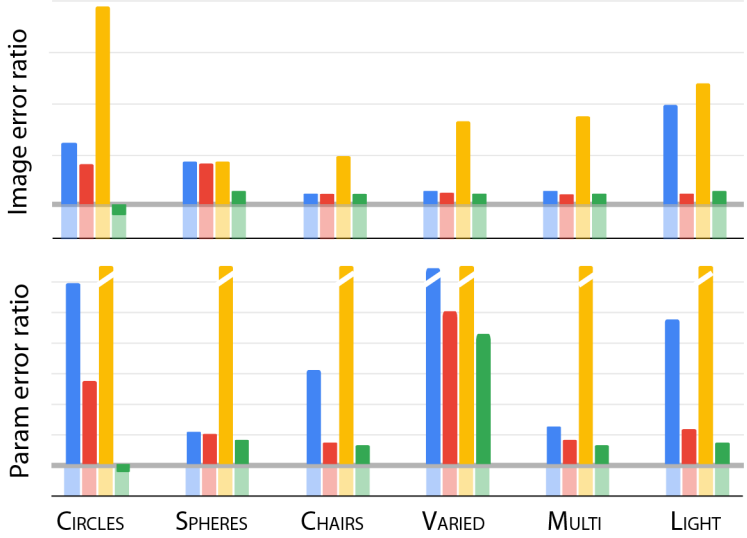


Table 6.7: Details of the per-parameter error of different methods in ratios with respect to a supervised reference. The ratio indicates, by what factor a method is worse, compared to a 100% supervision-baseline.

	Position (m)				Colour (MSE)				Rotation (Deg)				Confidence (MSE)				Direction (deg)			
	5%	10%	NC	0	5%	10%	NC	0	5%	10%	NC	0	5%	10%	NC	0	5%	10%	NC	0
CIRCLE	8.69	3.51	362	1.11	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
SPHERE	5.14	3.79	18.77	2.88	2.36	1.67	4.94	1.46	—	—	—	—	—	—	—	—	—	—	—	—
CHAIR	2.89	2.22	206	1.19	—	—	—	—	3.64	1.37	12.1	1.45	—	—	—	—	—	—	—	—
VARIED	5.29	2.65	4.91	1.25	2.50	1.72	4.51	1.74	—	—	—	—	20.9	11.3	48.3	10.1	—	—	—	—
MULTI	2.76	2.20	28.43	1.41	2.37	1.55	4.43	1.69	2.52	1.32	16.5	2.95	—	—	—	—	—	—	—	—
LIGHT	4.01	2.18	41.57	1.39	2.06	1.49	2.95	1.55	1.92	1.51	14.0	1.24	14.8	5.33	20.2	4.47	1.62	1.32	1.54	1.14

is visualised as a scene graph in an 3D modelling application. Note, that only the orientation, positions etc. (as defined in Tbl. 6.5) is part of our method's output.

Tbl. 6.7 splits the parameter error of all scenes between all applicable classes of scene parameters (position, colour, orientation, confidence, and light direction)

present in our scenes. We see the largest error ratio to be found with position, in particular for NonCur. Extracting the colour in the simple scenes we use is not a particularly hard task, so we find all methods to not lose much quality from less supervision. The error ratios for rotation are smaller, probably as when the network already has learned to position the object, getting rotation right is an easier optimisation task. The confidence error ratio is higher for (▼ Multi) than for (★ Light), likely as the object geometry between shapes in the latter have a higher variance. It could be hypothesised that varied illumination also helps understanding, but we did not study this difference. Finally, (★ Light) contributes little to the overall error, indicating the network has learned light direction from 2D images alone, almost as it had done with supervision.

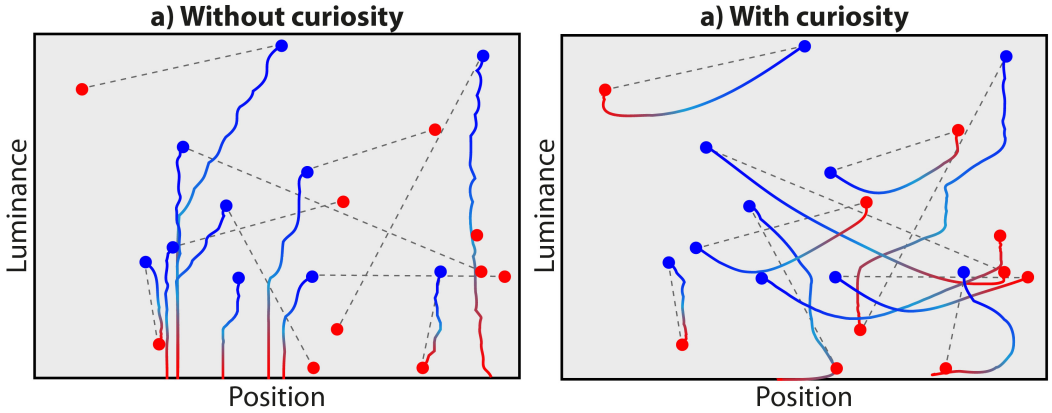


Figure 6.5: Solving an analytic problem with and without curiosity. Blue points are initial and red points ground truth; grey lines show correspondence. Optimisation is the blue-red trajectory.

Validation experiment The effect of curiosity can be verified for a very basic analytic 2D problem, finding positions (x) and luminance (l) of N objects in N images. We visualise the N solution as a red 2D point set in the position-luminance plane in Fig. 6.5. Starting from random initial guesses (blue 2D points in Fig. 6.5, a), optimising those N problems independently, analysis-by-synthesis will in almost all cases converge (trajectories and blue points in Fig. 6.5, a) to a degenerate and incorrect $l = 0$. Consider a world where x and l follow a uniform random distribution. Deviation of the set of resulting x, l pairs from a uniform distribution can be measured by computing discrepancy in closed form. This deviation provides an

idealised measure of curiosity. Adding this term forces the distributions of solutions to be uniform over the x,l plane (Fig. 6.5, b), finds the correct solution for almost all problems jointly.

To generate our training dataset, we sample parameters $p \in \mathbb{R}^{t,l}$ from a uniform distribution $\mathcal{D} : \{t,l\} \sim \mathcal{U}(0,1)$. Our optimisation then becomes

$$\arg\min D(\hat{p} - \mathcal{D}),$$

where D is a distance metric defined below.

We define D such that it allows for gradient-based parameter optimisation. First, we note that a uniform distribution $\mathcal{U}$ would have a constant density across the parameter space. Therefore, we aim to penalise $\hat{p}$ varying from a constant density. This variation we refer to a discrepancy D.

We can measure D by testing for a low discrepancy pattern between all possible subsets of $\hat{p}$ and a random sampling $S \sim \mathcal{D}$. The size N of sampling S here is not strictly important, so long as it represents $\mathcal{D}$ sufficiently. In our experiments $N = 300$.

First, we calculate the density with a Kernel Density Estimate (KDE) E. Whilst it would be tempting to compute E using common methods such as a box or Epechnikov approach, these would not be differentiable. Instead, we use a steep Gaussian kernel

$$K = \exp(-(||\hat{p} - S||)^2).$$

Next, we compute KDE estimates E for each point $e \in K$ as

$$\frac{1}{2}(e_t^i + e_l^i).$$

Finally, we calculate the variance in the KDE response (discrepancy) E as

$$D = \frac{1}{N-1} \sum_{i=0}^{N} (E_i - \bar{E})^2.$$

This is the scalar value we minimise in the optimisation.

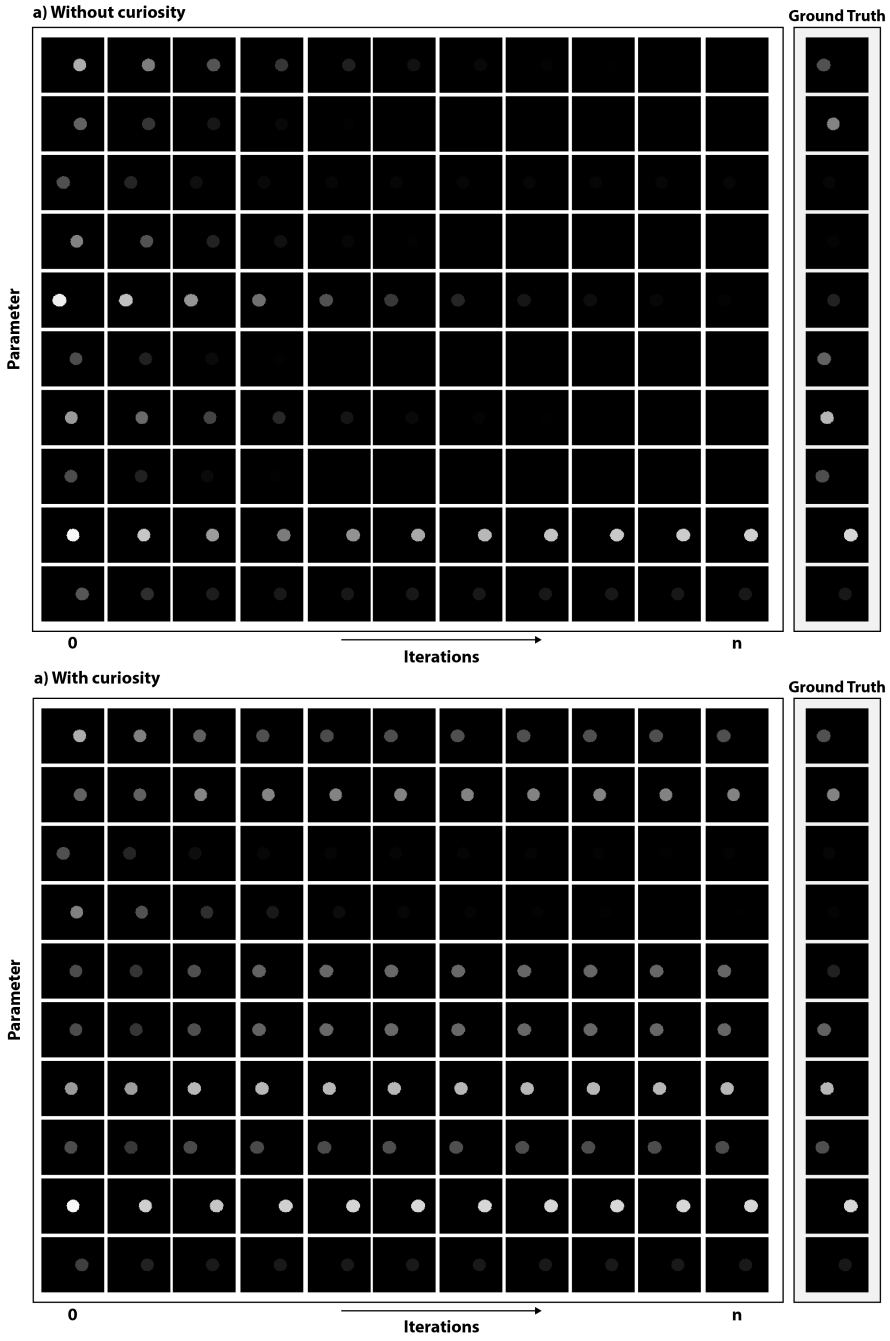


Figure 6.6: Renderings of parameters at regular intervals during optimisation for **a)** without curiosity and **b)** with curiosity. When image loss is used without a curiosity term the parameters degrade into an out-of-distribution solution where luminance l tends towards 0

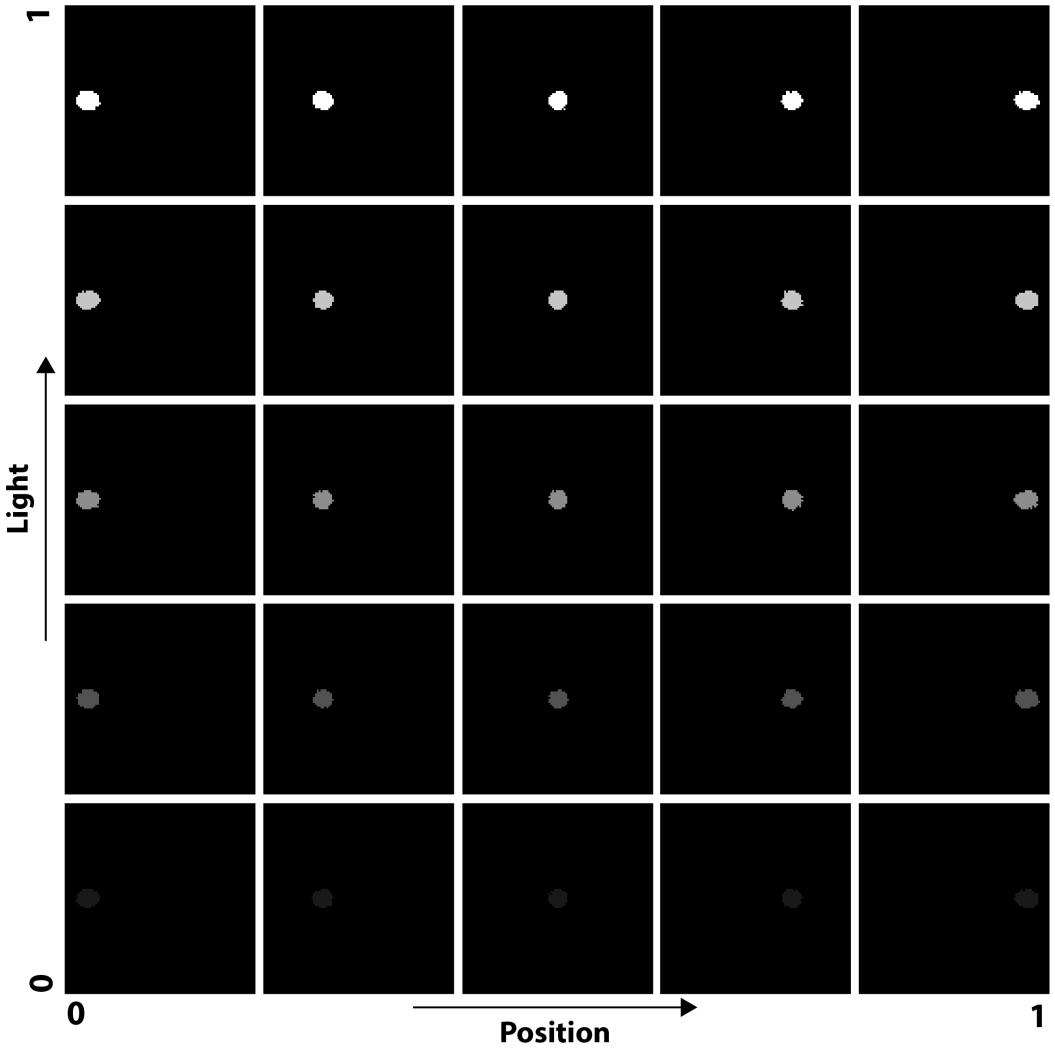


Figure 6.7: Visualisation of the parameter space when rendered.

When a simple L2 loss is used and no distribution is enforced (Fig. 6.6 top) the optimisation almost always falls into the local-minima of turning the circle black ($l = 0$). This is the same phenomenon we experience in the main paper for `NonCur` experiments. We observe this because the majority of gradients for l point towards the black background. As a result, the overall gradient is dominated by this incorrect force. This is visible in the main paper Fig. 6 (left) as parameters falling towards the bottom of the chart. By enforcing $\hat{p} \sim \mathcal{D}$, we see that when the parameters start to fall, they are push back up, where they find more dominant L2 gradients main paper Fig. 6 (right).

6.5.2 Real world data

While we have initially evaluated our approach in a virtual setting, which is suitable for instrumentation as all scene parameters are known, we now take it to a real setting, where parameters are unknown.

Data For training data, we capture sequences of objects in front of a neutral background using standard cameras. For evaluation, we took several photos from multiple directions, allowing us to perform Structure-from-Motion to retrieve camera matrices. We captured the datasets SPHERE REAL, BUNNY and SHAPES (Fig. 6.8). In both, the task is to learn to regress 3D position, orientation and light from an image as summarised in the lower part of Tbl. 6.5.

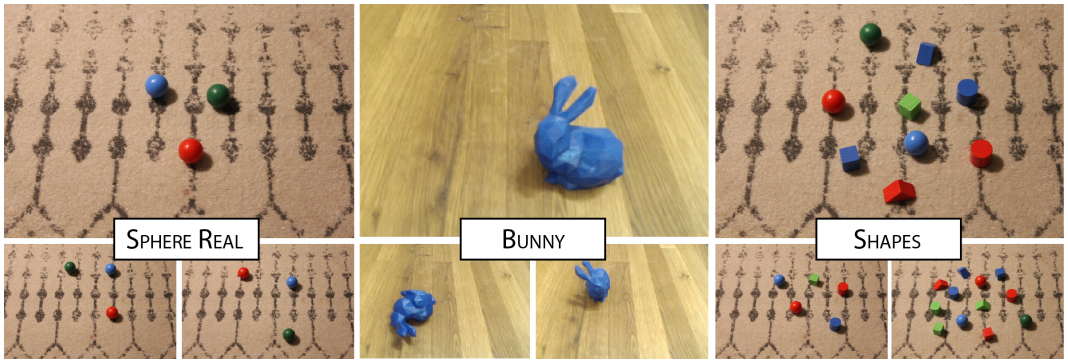


Figure 6.8: Three CUBE and BUNNY samples from the real capture datasets. Features like sensor noise, motion blur, geometric details and depth-of-field are not reproduced by our DR, while still the method can be trained.

To enable real world experimentation, we recreate physical versions of our synthetic world setups. To enable object learning requires 3 steps, a) data acquisition, b) cameras alignment, c) object and scene modelling (Fig. 6.9).

Data acquisition is performed using two standard Canon digital single lens reflex cameras fixed on tripods. We set the validation (novel-view) camera at an $\sim$90°angle from the training camera view. To create the validation dataset objects are then placed at random within the scene and the scene is captured from each camera. A training dataset is then created by randomly placing objects into the scene and only capturing data from the training camera. We finally take an image from each camera with no objects in the scene for scene background modelling.

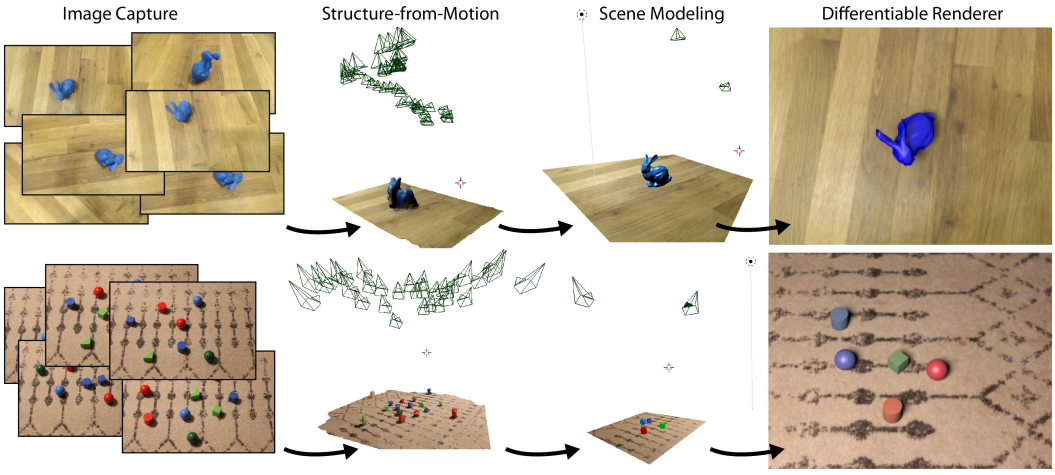


Figure 6.9: Real dataset acquisition pipeline. Images are first captured from either mobile (BUNNY) or fixed (SPHERE REAL, SHAPES) cameras (**a**). Camera alignment is calculated using SfM (**b**). Using reconstructed geometry and camera exterior orientations we can quickly model the scene (**c**). Finally, we pass all the parameters into a DR for scene parameter learning (**d**). Note, camera alignment is only used for validation and not during the training or online phase.

Camera alignment To get relative exterior orientations between the two cameras we utilise a Structure-from-Motion (SfM) pipeline. Each camera is lifted from the tripod and multiple images take from various viewpoints. We ensure we densely cover the space between the two cameras. The images are input into the SfM software which produces the exterior orientations of the cameras in a common (arbitrary) coordinate system. We further generate a dense point cloud using multi-view stereo to aid object modelling. All SfM processing was performed in MetaShapes v1.6.5 using default settings.

Object and scene modeling Each shape is modeled inside Blender software, based on the dense point cloud. This produces mesh objects which we pass to the DR. By doing so we ensure our mesh has the same scale as the scene.

We assume a flat ground plane and that objects are in a canonical position, with the only rotation axis being about the objects up-axis. We further transform the camera exterior orientations such that the reconstructed ground plane lies on the XY plane. This is done manually as we have no ground control points to align the bundle adjustment. If a real co-ordinate system is required, ground control points could be placed into the scene avoiding the need for manual transformation (rotation

and scaling) of the exterior orientations.

Once the camera alignment is complete, we can then create a ground plane mesh which we texture with the image containing no objects for training and validation cameras respectively. To ensure the same view of the ground plane we model the ground plane to the view frustum of the camera. This can be done by importing the intrinsic matrix estimated from the SfM to a virtual camera. Camera exterior and intrinsic properties are exported along with all mesh objects.

In scenarios where 3D CAD objects are available prior to data capture (e.g., industrial applications where parts are designed in CAD software before being manufactured), the object modeling process is not required.

Bunny The BUNNY dataset demonstrates a more casual data capture method. A single 3D-printed Stanford bunny is filmed from a moving mobile camera. Training data is generated by extracting frames every n-frames where $n = 10$ in our case. Camera alignment is then computed for a sequence of frames. The first frame in the sequence is used as the input frame and the last frame is used as the novel view. As the motion is continuous, we can get relative camera positions using SfM. Object is then the same as for the REAL SPHERE and SHAPES datasets. In order to model the scene, we require a background image, which contains no object. We found Adobe Photoshop content-aware delete tool to quickly and effectively remove the foreground object, leaving a clean background image which can be used to texture the ground-plane mesh.

Methods The data available only allows studying `NonCur` in comparison to `Our` as no supervision parameter values are known. To close the domain gap, we blur the rendering and camera image by a 9×9-pixel Gaussian filter to help remove camera noise.

Metric Sets of images of the test scene from multiple views of known relative pose allow to test the reprojection task, as for synthetic data using DSSIM. The object parameters remain unknown and cannot be compared to anything for the real experiment. Consequently, results also cannot be reported as ratio relative to this supervision signal. Further, for the above-mentioned reasons (camera noise, MB, DoF),

there is a domain gap between any of our novel-view images and the reference. To approximately quantify what differences might be if that domain gap would not be there, we additionally report the DSSIM error for the blurred version of both images. While the error can be quantified, the reader might get a better impression from looking at the actual re-synthesized images in Fig. 6.10.

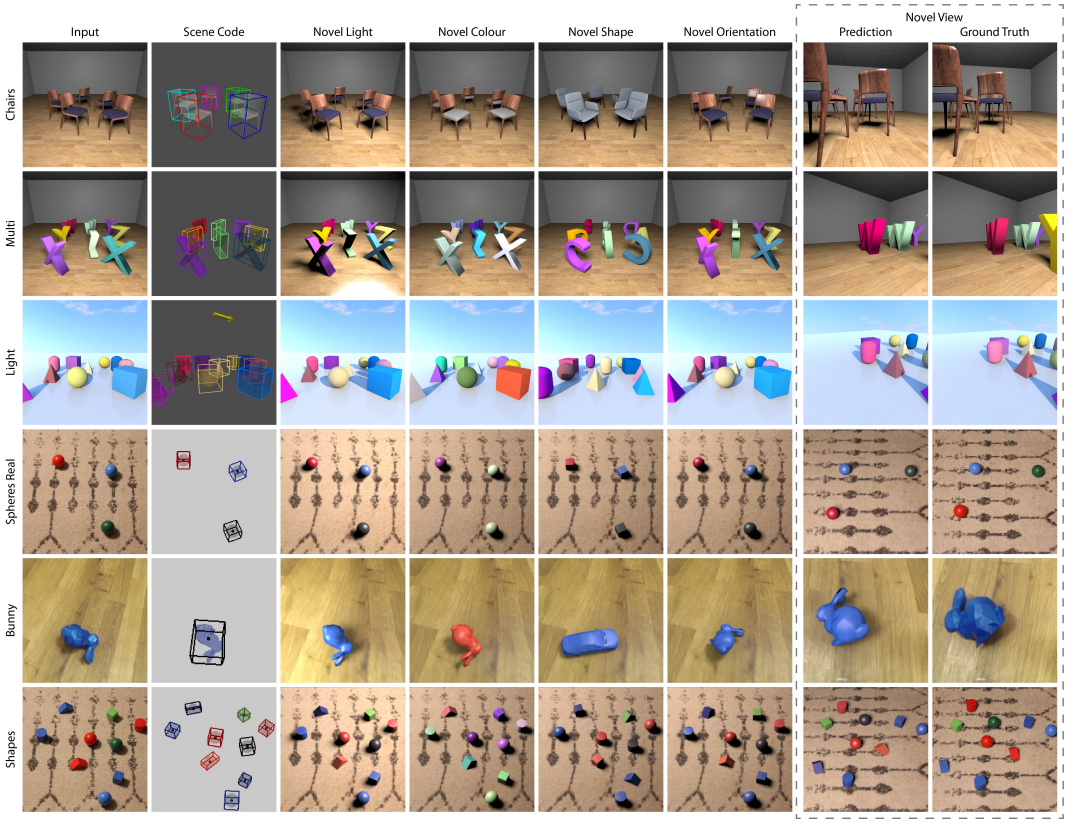


Figure 6.10: Results of our approach. Each row is a different world / task. Note how there was no other supervision than sample images.

Findings We find that our method is able to recover sensible parameters when trained on only real RGB images. Tbl. 6.8 quantifies this. We see that analysis-by-synthesis training without curiosity (`NonCur`) results in a higher image error than `Our`. This corresponds to qualitative results in Fig. 6.10.

6.5.3 LineMOD

This task entails regressing a 6-DOF pose of a known object in images (Hinterstoisser et al., 2012). Unfortunately, we do not have access to scenes without the objects present to generate training data. As a result, to achieve the optimum L2

Table 6.8: Results on real photos for different datasets **(columns)** for different methods **(rows)**. Image error is in DSSIM, not ratio, as no supervised baseline on photos exist.

	SPHERE REAL		BUNNY		SHAPES	
	Sharp	Blur	Sharp	Blur	Sharp	Blur
NonCur	0.48	0.41	0.46	0.43	0.54	0.46
Our	0.33	0.25	0.36	0.31	0.37	0.29

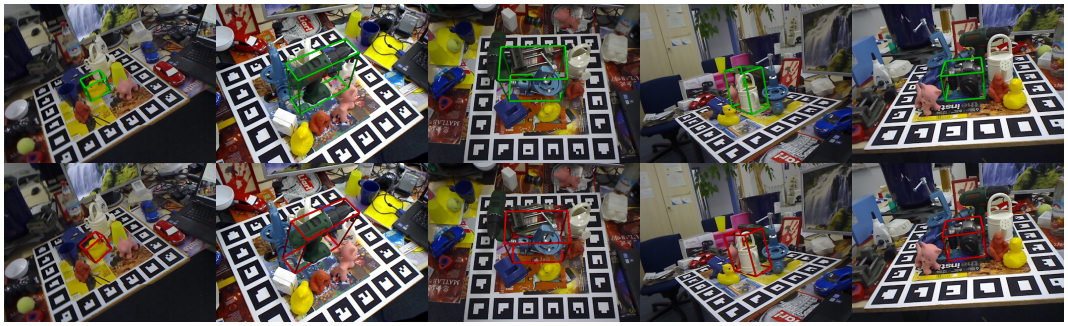


Figure 6.11: Visualization of (**top, green**) ground truth boxes and (**bottom, red**) our network predictions on LineMOD test data.

and critic loss our network could simply render the objects out of the cameras view. To account for this, we take a similar approach to Su et al. (2021) whereby we generate synthetic images by rendering 3D models over 2D background images. We first collect backgrounds by downloading 500 images returned by a Google search

Table 6.9: Results from LineMOD experiment. All results denote the ADD object recall metric proposed by (Hinterstoisser et al., 2012). 'Lbl$_R$' and 'Lbl $_S$' indicates whether real or synthetic pose or mask labels were used respectively. Note that our approach and Sundermeyer et al. (2020) use neither.

Method	Lbl$_R$	Lbl$_S$	Avg.	Ape	B.vise	Cam	Can	Cat	Drill	Duck	Eggb.	Glue	Pun.	Iron	Lamp	Phone
Yolo6d [1]	✓	✗	56.0	21.6	81.8	36.6	68.8	41.8	63.5	27.2	69.6	80.0	42.6	75.0	71.1	47.7
Pix2Pose [2]	✓	✗	72.4	58.1	91.0	60.9	84.4	65.0	76.3	43.8	96.8	79.4	74.8	83.4	82.0	45.0
Kehl et al. [3]	✓	✗	92.9	**98.1**	94.8	93.4	82.6	**98.1**	96.5	97.9	**100.0**	74.1	**97.9**	91.0	98.2	84.9
EfficientPose [4]	✓	✗	**97.4**	87.7	**99.7**	**97.9**	**98.5**	98.0	**99.9**	91.0	**100.0**	**100.0**	95.2	**99.7**	**100.0**	**98.0**
Our (label)	✓	✗	78.2	71.5	93.4	70.2	71.8	76.3	73.8	65.4	90.5	71.5	86.2	81.4	86.2	78.5
Yolo6d [5]	✗	✓	21.4	16.1	33.9	2.9	21.0	27.1	24.7	20.2	2.3	15.0	15.4	57.6	26.8	14.8
Pix2Pose [6]	✗	✓	11.3	3.6	4.0	0.0	16.6	20.2	29.0	0.2	0.0	7.3	2.5	1.8	30.2	31.9
Kehl et al. [7]	✗	✓	2.4	0.0	0.2	0.4	1.4	0.5	2.6	0.0	8.9	0.0	0.3	8.9	8.2	0.2
SynPo-Net [8]	✗	✓	44.1	23.1	**75.2**	6.7	**65.1**	36.2	53.7	19.5	3.9	41.8	21.1	**85.1**	78.7	63.6
Implicit3d [9]	✗	✗	32.7	4.2	22.9	32.9	37.0	18.7	24.8	5.9	**81.0**	46.7	18.2	35.1	61.2	36.3
Our NonCur	✗	✗	1.6	2.6	1.5	0.4	0.9	1.1	1.5	0.0	5.2	1.5	3.2	0.0	0.0	2.3
Our	✗	✗	**60.9**	**50.2**	54.6	**63.5**	58.1	**42.6**	**58.4**	**38.6**	68.0	**56.7**	**68.2**	78.3	**79.4**	**75.3**

[1] (Tekin et al., 2018), [2] (Park et al., 2019b), [3] (Kehl et al., 2017), [4] (Bukschat, Vetter, 2020), [5] (Tekin et al., 2018), [6] (Park et al., 2019b), [7] (Kehl et al., 2017), [8] (Su et al., 2021), [9] (Sundermeyer et al., 2020)

of the term "clutter". For each class we generate 10,000 training images where the respective mesh has a random position and orientation and each scene a random environment light. Unlike Su et al. (2021), we do not access synthetic labels during training.

We use the same network architecture proposed in Sec. 6.3. However, our scene code $\mathbf{s} \in \mathbb{R}^{12}$ learns position $p \in \mathbb{R}^3$, rotation $R \in \mathbb{R}^6$ (as proposed by Zhou et al. (2019b)) and environment light $L \in \mathbb{R}^3$. We use data loading scheduling to train our model. First, we train only on synthetic images to learn to make predictions in the camera view. We then incrementally add real training images (without accessing labels).

Our method on average out-performs all methods in Tbl. 6.9 where either no labels or synthetic labels are used. We perform competitively compared to supervised approaches, with a 20% reduction from ours supervised (the upper bound of our detector). Qualitative results are shown in Fig. 6.11.

6.6 Conclusion and Future Work

We have demonstrated a pipeline combining DR and a curiosity term, that allows to learn an explicit and interpretable 3D scene parameterization of a single image from a set of unlabelled 2D images and a geometric representation of the objects we wish to find. Different from other approaches, we are not limited to simple or single objects and can represent global properties such as light. Direct optimisation under a L2-like loss function leads to many ambiguities, that can be overcome by what looks like a GAN-like critic at first but serves an entirely different purpose: preventing the network from following always the same easy-reward gradients that lead to unusable minima. With this simple addition we show a relatively basic network, without labels, can perform competitively with a fully supervised counterpart.

A limitation to our approach is the requirement of a geometric representation. In future work we propose such a template, if represented as a parameterisable model, such as a deep Signed Distance Function (SDF) (Park et al., 2019a) from a strong prior, can be included in the scene code and learnt during training. Beker

et al. (2020) demonstrate such an approach to be effective.

Chapter 7

Discussion

In this chapter we present a general discussion on the work covered from Chapters 3 - 6. The purpose of this chapter is to reflect on the work presented in this thesis from a holistic point of view. The limitations section (Sec. 7.1) aims to address specific issues with each method presented. The chapter is concluded with a discussion on future works where the author personally believes further research will likely yield significant progress in the field.

7.1 Limitations

7.1.1 Existing data

The utilisation of national mapping data in Chapter 3 was shown to be highly effective for building segmentation. Furthermore, it was shown the effectiveness could be increased by introducing a novel energy-based label refinement algorithm. Despite this, several key limitations are still present in the methodology.

Most prominently is the class restriction imposed by using existing data. Buildings unfortunately represent one of the few accurately mapped national scale classes. Another reasonable class we could have exploited existing labels for would be roads, as well as general land classification. However, outside of these classes this method would not be possible. Fortunately, there is a direct correlation between national-scale features which we want to map, and those which have been mapped. This is the reason they were mapped in the first place. However, this is not an extensive list. For example, street furniture and points of interest are only

sparsely mapped, despite there being a demand for their precise mapped locations nationally.

Whilst in the method 3D LiDAR points were classified, there is a strong assumption that the scene topology is approximately 2D with respect to the camera view-point. This allows the use of 2D mapping labels. The 3D data was utilised by projecting the data onto a regular 2D grid which is aligned with the 2D RGB image grid. These assumptions limit the methodology from being useful in the generic 3D setting. For example, indoor scene understanding would require full 3D labels, which do not exist on a large-scale. So, whilst we make progress for a specific task where 3D data is present, this method does not offer a strong research direction for the more generic tasks set out in Chapter 1.

7.1.2 Synthetic

In Chapter 4 we generate a synthetic point cloud we dub SynthCity. The methodology outlined proved successful in easily adapting an artist generated 3D scene to a 3D scene understanding dataset. Despite looking qualitatively reasonable, regrettably we found relatively little useful application of our dataset for real-world tasks. We hypothesised multiple reasons for this, however, most likely either SynthCity was too small, or the real-world test set was too similar to the real-world train set, ultimately rewarding the network for over-fitting.

Another larger limitation of using synthetic data for training real-world models is the requirement for the digital 3D world models. Similar to the limitations discussed in Sec. 7.1.1, such scenes are only available for a subset of environments. For example, whereas inner cities are frequently modelled, rural areas as well as specialist areas (e.g., railways, tunnels, mines etc.) are not. As a result, the scope of using synthetic data is ultimately limited. Despite this, one potential application could be to use large existing synthetic datasets to pre-train large 3D network backbones which can then be fine-tuned on more domain specific environments. This has shown success in the 2D domain (Huh et al., 2016; He et al., 2019) and more recently Zhang et al. (2021c); Xie et al. (2020) have shown promise in the 3D domain.

Using synthetic data, like existing data, is still a fully-supervised approach to 3D scene understanding. Whilst the labelling effort for training machine learning models is reduced, the labels are still manually created, just not by the machine learning practitioner. Scaling synthetic scenes to perform the wide range of tasks covered by 3D scene understanding would ultimately require digitally modelling the environments first. This offers no immediate benefit to typical manual labelling for supervised learning tasks. Instead, it simply shifts the labelling to another task. Moreover, by shifting the labelling to synthetic, a domain-gap is introduced which is shown in Chapter 4 to be damaging for performance. This is in line with other research findings e.g., Wu et al. (2018).

7.1.3 Weakly-supervised

In Chapter 5 a weakly-supervised approach for 3D scene understanding is proposed through a novel learnt loss function. Unlike Sec. 7.1.1 and 7.1.2, the reduction of labels does not come at the cost of requiring alternative labels. Instead, the network becomes more label efficient. The key limitation here is still the requirement for 3D labels. Whilst we show that with up to a 95% reduction of labels, strong results can be obtained, labelling 5% of the dataset is still time consuming and undesirable. For example, loading and navigating large point cloud data can be very cumbersome even on modern day computers, especially when the size of the point cloud is larger than the available Random Access Memory of the machine.

It is also important that the 5% of labels cover the distribution well of the 95% of missing labels. For example, if only armchairs are labelled, it is not clear how well the method would perform at detecting office chairs. This therefore requires a further form of supervision to ensure the labels cover all strong variations of instances present in the dataset. The trained loss network also defines the domain in which the scene network can operate. Although we still achieve good results across datasets, the datasets are still reasonably similar (both indoor typical U.S. university environments). As the domain of the scene dataset drifts from the domain used to train loss network the performance would start to degrade. This requires the user to be aware of the input stream of scene data and understand its similarity with respect

to the labels used for training the loss network.

7.1.4 Self-supervised

The key advantage of the self-supervised learning approach presented in Chapter 6 is that no scene labels are required at any point in the learning pipeline. This alleviates the problems discussed above in Sec. 7.1.3 whereby a domain gap is introduced between the labelled data and unlabelled data. This is particularly useful for continuous learning where the model is retrained with new data which can potentially drift from the initial data distribution. However, in doing this there was a trade-off with the addition of a new constraints.

The method requires a geometric representation (mesh, in this case) of foreground objects in the scene. This introduces two main constraints. Firstly, a 3D geometric representation needs to be either available or acquired. This can be achieved using classic modelling techniques; however, the difficulty of the task increases as the geometry being modelled becomes more complex. Secondly, we assume that the there is no geometric variation of the objects in the scene. Whilst this is a reasonable assumption for many standardised components (e.g., in an engineering environment), it is not the case for many real-world applications such as city mapping. For example, whilst lampposts are geometrically similar (tall poll-like structures), there is an inevitable variation at a city level. Our method would require an example of each lamppost type as opposed to one example per-class.

Future work to address this could include fitting parametric mesh models as common in human pose estimation (Bogo et al., 2016; Kolotouros et al., 2019) or by learning implicit differentiable representations (e.g., Signed Distance Functions (Park et al., 2019a) or Neural Radiance Fields (Mildenhall et al., 2020)). In these scenarios, the network would not only predict the object class and spatial location, but also the per-class model parameters/weights to ensure a good fit. Further experimentation would be required to validate this approach.

7.2 Summary

Each of the methods presented in this thesis have drawbacks, discussed in Sec. 7.1. Despite this, the arguments put forward tend toward weakly and self-supervised approaches being more favourable. The primary reason for this is that whilst existing and synthetic data can inevitably be useful, they are constricted to the specific domains in which they were originally created. The goal of this thesis was to evaluate the most effective label efficient approaches toward the goal of general 3D scene understanding. Whilst synthetic data has no fundamental limitation, creating the digital models to generate the supervision data is not obviously less effort or more efficient than standard per-object spatial labelling. Furthermore, the introduction of the domain-gap is shown to be problematic and therefore currently unfavourable.

The weakly-supervised approach proposed in Chapter 5 was the most suitable approach for real-world 3D scene understanding tasks. Through experimentation we show that this approach achieves impressive results on established real-world benchmark datasets (ScanNet and S3DIS). The self-supervised approach proposed in Chapter 6, while achieving impressive results on synthetic and simple scenes, has no obvious immediate logical steps of improvements which would allow the method to compete with the weakly-supervised approach on a real-world benchmark. We do, however, show that the introduction of a GAN-like loss function can have application in 3D object detection. The idea, albeit a simple one, could have a wide range of application in future network designs.

7.3 Future works

In this thesis we followed what is arguably the "mainstream" in modern 3D point cloud deep learning architectures. The feature encoder backbones were all built on variants of PointNet++ (Qi et al., 2017b). Such networks achieve hierarchical feature pooling through a Euclidean-based inductive bias. Whilst this approach has undoubtedly been responsible for the impressive performance of 3D vision tasks since their inception, recent trends are beginning to deviate from such architectures. In this section we will look at two architecture designs which present exiting future

research prospects.

7.3.1 Transformers

Introduced in the seminal paper by Vaswani et al. (2017), transformers have been responsible for large advances in the field of natural language processing. Large transformer-based models such as BERT (Devlin et al., 2018), Megatron-LM (Shoeybi et al., 2019) and GPT-3 (Brown et al., 2020) have redefined what is possible for sequence-to-sequence tasks. At the core of the transformer is an attention mechanism (Bahdanau et al., 2014; Parikh et al., 2016; Kim et al., 2017). Crucial to the work of Vaswani et al. (2017) was the idea that a model built entirely of attention modules performs better than traditional recurrent neural network architectures. Intuitively, attention allows the network to learn a per-feature weighting which is used to perform a weighted aggregation of the set's features. The network can learn to increase the weighting of more important features whilst simultaneously down weighting less important features. For example, certain words in a sentence (e.g., happy, elated, angry) are more salient in understanding the sentiment of a sentence than others (e.g., and, the, or).

More recently, transformers have shown to be effective for 2D computer vision tasks. Dosovitskiy et al. (2020) show that by splitting an image into a 16×16 grid a transformer can be used for the image classification. Intuitively, the authors argue that different patches are more salient for determining the main classification of an image than others. This is particularly true when some patches contain only background pixels. Since their inception Vision Transformers (ViT) have been used for a range of tasks including classification (Dosovitskiy et al., 2020), object detection (Carion et al., 2020), semantic segmentation (Strudel et al., 2021) and image synthesis (Liu et al., 2020).

Concurrently to 2D research, 3D researchers have also evaluated the potential for attention modules and transformer models for processing 3D data, specifically, point clouds. One of the early successful integrations of attention modules in a modern 3D neural network was proposed by Hu et al. (2020) in their RandLA-Net model. RandLA-Net showed by the inclusion of attention modules, point sam-

pling for down sampling become much less important. Whereas most modern networks use either grid-based sampling (Thomas et al., 2019), farthest point sampling (Qi et al., 2017b) or poisson disk sampling (Hermosilla et al., 2018), RandLa-Net achieve comparable performance using a simple random sampling. This was made possible as the network could easily learn which salient point features to keep through a 3D attention module.

There are several characteristics to make point clouds and transformers potentially very compatible. Firstly, transformers are by design permutation invariant. The absolute ordering of words in the sentence, like points in a point cloud is not necessarily important. Transformers are also invariant to the cardinality of the set. This is very favourable as point clouds captured from real-world sensors have varying local density and cardinality. Additionally, like words in a sentence, certain points are more salient than others (e.g., corner, skeleton points). Finally, a key component for transformer is a position embedding. This allows the network to learn relative positioning of words. The authors of Vaswani et al. (2017) propose to use sine and cosine functions. However, point clouds already live in a meaningful metric space, as such the positional encoding can simply be the 3D positions of the points (Hu et al., 2020; Zhao et al., 2021). The recent Point Transformer (Zhao et al., 2021) adopts this approach and has achieved state-of-the-art accuracy on model classification and per-point classification tasks.

7.3.2 Mesh-based networks

In this thesis we mostly focus on point cloud as a 3D representation. However, as discussed in Sec. 1.3, this is not the only representation. Another common representation is triangular meshes. Different to point clouds, meshes include connectivity (edges) information between points (vertices). Typical point cloud processing networks pool features using either K-Nearest Neighbour or radius ball searches. That is, given a seed point, the features of the K-Nearest neighbour points, or points less than some defined distance are pooled together. In both approaches the Euclidean distance in metric space is used. This is usually undesirable when two surfaces of different objects are close to each other. For example, points of a chair leg and a

table leg may be very close in terms of Euclidean distance. The key advantage of having a known connectivity is that instead of using a Euclidean neighbourhood, a Geodesic neighbourhood can be used. This ensures all features are pooled from the same surface.

A key challenge in utilising meshes for hierarchical learning methods such as CNNs is a formulation of how to pool features. An example of one approach is proposed by Hanocka et al. (2019), in their MeshCNN network. MeshCNN learns an edge collapsing technique which can collapse less important edges, effectively retaining and expanding important edges. Schult et al. (2020) using graph based convolutions and pooling extend mesh-based methods to achieve competitive performance on a real-world (ScanNet) benchmark. Specific to this method is the use of both Euclidean and Geodesic pooling which is shown to be effective. A disadvantage is the increased memory footprint introduced by storing both vertices and edges [1] and the increased computation footprint for expensive edge collapsing operations. We argue future work in this area could yield performance increases by exploiting this additional information.

[1] On the contrary, meshes are non-uniform representations and in certain scenarios can be more efficient. Simple surfaces can be represented with very few vertices and edges. For example, a ground plane is represented with 4 vertices and 4 edges irrespective of how large the plane is. However, for meshes constructed from real-world sensors this this cannot be exploited without first preprocessing the data was a mesh simplification algorithm, potentially loosing information.

Chapter 8

Conclusions

In this thesis we have addressed four strategies to reduce the labelling requirement of training deep neural networks for 3D related tasks. In Chapter 1 we introduced the general problem of 3D scene understanding and discussed challenges associated with processing 3D point cloud data. We outline 3D specific challenges which has caused progress in 3D computer vision to trail 2D computer vision. We highlight the representation challenges and discuss their trade-offs.

In Chapter 3 we evaluate a method which avoids the requirements of manual labelling by utilising existing mapping data, available from national mapping agencies. We demonstrate why the course nature of the labels cause performance issues when training neural networks. We propose to use energy minimising active contour snakes on co-registered LiDAR data to refine labels to fit buildings. Here, the mapping data only offers a single seed point for the region growing snakes which expand out to fit to the edges of the buildings. Our experimentation shows by improving label quality overall performance is increased leading to state-of-the-art results on the challenging ISPRS Potsdam building segmentation benchmark.

Chapter 4 also utilised existing data, but in the form of artist created synthetic digital world models. We demonstrate the ability to utilise open-source virtual LiDAR software to generate a Mobile Laser Scanner (MLS) dataset we dub SynthCity. We run evaluations to determine the effects of pre-training 3D neural networks with synthetic data to improve test performance on real data. We conclude from our studies that whilst the dataset appeared qualitatively feasible, no performance gains

were made with respect to real-world test metrics. This is likely due to either SynthCity being too small or the train and test sets being too similar on the real-world data, effectively rewarding the network for over-fitting. In future work we propose to increase the size of the SynthCity by utilising larger world models such as those found in video games (Wu et al., 2018, 2019), or driving simulators (Dosovitskiy et al., 2017).

In Chapter 5 we propose a weakly-supervised learning approach. Our method introduces the novel concept of learning the loss function from a smaller subset of labels. Once our loss function is learnt we use it to train a 3D object detector from unlabelled data. The key insight from our method is that the loss function is only required to learn a more simple and local problem which can be trained using significantly less data. We experimentally achieve competitive performance to other works using 95% less training data. Our approach was shown to also work well when the loss function and object detector were trained on different datasets. This approach opens the potential for training on enormous datasets where the domain-shift is not expected to be high.

Chapter 6 proposes a method to further reduce the required labels to only a single instance of each class, represented by a geometric model (polygonal mesh). A 3D monocular object detector is built which achieves impressive performance on synthetic and real scenes without any scene-level labels. We discuss why standard $\mathcal{L}_2$-like losses are not useful when optimising using a differentiable rendering. In light of this we propose to use a Generative Adversarial Network (GAN)-like loss function to avoid local-minima which the $\mathcal{L}_2$-like loss almost certainly falls into. The method shows the potential of using GAN-like losses for 3D object detection, whereas traditionally it is reserved for image synthesis tasks.

We summarise all of our findings and discuss limitations in Chapter 7. We conclude that in specific situations each method can have benefits over the others. However, from the experiments presented in this thesis, our weakly-supervised approach was the most effective at moving towards the goal of highly label-efficient 3D scene understanding. Although for maximum label efficiency self-supervised

approaches would be more favourable, currently they are not as flexible or scalable as the weakly-supervised method proposed.

Bibliography

Achlioptas Panos, Diamanti Olga, Mitliagkas Ioannis, Guibas Leonidas. Learning representations and generative models for 3d point clouds // International conference on machine learning. 2018. 40–49.

Adams Andrew, Baek Jongmin, Davis Myers Abraham. Fast High-Dimensional Filtering Using the Permutohedral Lattice // Computer Graphics Forum. 2010. 29, 2. 753–762.

Akel Nizar Abo, Zilberstein Ofer, Doytsher Yerach. A robust method used with orthogonal polynomials and road network for automatic terrain surface extraction from LIDAR data in urban areas // International Archives of Photogrammetry and Remote Sensing, ISPRS. 2004.

Alon Alvin Sarraga, Festijo Enrique D, Juanico Drandreb Earl O. Tree Detection using Genus-Specific RetinaNet from Orthophoto for Segmentation Access of Airborne LiDAR Data // 2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS). 2019. 1–6.

Andrew Alex M. Multiple view geometry in computer vision // Kybernetes. 2001.

Apache . Apache Parquet. 2022. Accessed: 2019-07-10.

Armeni Iro, Sax Sasha, Zamir Amir R, Savarese Silvio. Joint 2D-3D-semantic data for indoor scene understanding // arXiv:1702.01105. 2017.

Armeni Iro, Sener Ozan, Zamir Amir R, Jiang Helen, Brilakis Ioannis, Fischer Martin, Savarese Silvio. 3d semantic parsing of large-scale indoor spaces // Proceed-

ings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. 1534–1543.

Audebert Nicolas, Le Saux Bertrand, Lefèvre Sébastien. Joint learning from earth observation and openstreetmap data to get faster better semantic maps // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2017. 67–75.

Awrangjeb Mohammad, Ravanbakhsh Mehdi, Fraser Clive S. Automatic Detection of Residential Buildings Using LIDAR Data and Multispectral Imagery // ISPRS Journal of Photogrammetry and Remote Sensing. IX 2010. 65, 5. 457–467.

Bahdanau Dzmitry, Cho Kyunghyun, Bengio Yoshua. Neural machine translation by jointly learning to align and translate // arXiv preprint arXiv:1409.0473. 2014.

Baruch Gilad, Chen Zhuoyuan, Dehghan Afshin, Feigin Yuri, Fu Peter, Gebauer Thomas, Kurz Daniel, Dimry Tal, Joffe Brandon, Schwartz Arik, others . ARKitScenes: A Diverse Real-World Dataset For 3D Indoor Scene Understanding Using Mobile RGB-D Data // Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1). 2021.

Bechtold S, Höfle Bernhard. HELIOS: A Multi-Purpose LiDAR Simulation Framework for Research, Planning and Training of Laser Scanning Operations with Airborne, Ground-based Mobile and Stationary Platforms. // ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences. 2016. 3, 3.

Becker Carlos, Häni Nicolai, Rosinskaya Elena, d'Angelo Emmanuel, Strecha Christoph. Classification of aerial photogrammetric 3D point clouds // arXiv preprint arXiv:1705.08374. 2017.

Beker Deniz, Kato Hiroharu, Morariu Mihai Adrian, Ando Takahiro, Matsuoka Toru, Kehl Wadim, Gaidon Adrien. Monocular Differentiable Rendering for Self-Supervised 3D Object Detection // arXiv:2009.14524. 2020.

Beltrán Jorge, Guindel Carlos, Moreno Francisco Miguel, Cruzado Daniel, Garcia Fernando, De La Escalera Arturo. Birdnet: a 3d object detection framework from lidar information // 2018 21st International Conference on Intelligent Transportation Systems (ITSC). 2018. 3517–3523.

Bernardini Fausto, Mittleman Joshua, Rushmeier Holly, Silva Claudio, Taubin Gabriel. The ball-pivoting algorithm for surface reconstruction // IEEE transactions on visualization and computer graphics. 1999. 5, 4. 349–359.

Bogo Federica, Kanazawa Angjoo, Lassner Christoph, Gehler Peter, Romero Javier, Black Michael J. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image // European conference on computer vision. 2016. 561–578.

Britannica Encyclopedia. Photogrammetry // Britannica, The Editors of Encyclopaedia. 2021.

Brown Tom B, Mann Benjamin, Ryder Nick, Subbiah Melanie, Kaplan Jared, Dhariwal Prafulla, Neelakantan Arvind, Shyam Pranav, Sastry Girish, Askell Amanda, others . Language models are few-shot learners // arXiv preprint arXiv:2005.14165. 2020.

Bukschat Yannick, Vetter Marcus. Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach // arXiv preprint arXiv:2011.04307. 2020.

Caesar Holger, Bankiti Varun, Lang Alex H, Vora Sourabh, Liong Venice Erin, Xu Qiang, Krishnan Anush, Pan Yu, Baldan Giancarlo, Beijbom Oscar. nuscenes: A multimodal dataset for autonomous driving // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020. 11621–11631.

Carion Nicolas, Massa Francisco, Synnaeve Gabriel, Usunier Nicolas, Kirillov Alexander, Zagoruyko Sergey. End-to-end object detection with transformers // European Conference on Computer Vision. 2020. 213–229.

Caron Mathilde, Misra Ishan, Mairal Julien, Goyal Priya, Bojanowski Piotr, Joulin Armand. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments // Advances in Neural Information Processing Systems. 33. 2020. 9912–9924.

Caselles Vicent, Kimmel Ron, Sapiro Guillermo. Geodesic Active Contours // International Journal of Computer Vision. II 1997. 22, 1. 61–79.

Chan T. F., Vese L. A. Active Contours without Edges // IEEE Transactions on Image Processing. II 2001. 10, 2. 266–277.

Chang Angel, Dai Angela, Funkhouser Thomas, Halber Maciej, Niessner Matthias, Savva Manolis, Song Shuran, Zeng Andy, Zhang Yinda. Matterport3D: Learning from RGB-D Data in Indoor Environments // International Conference on 3D Vision (3DV). 2017.

Chang Angel X., Funkhouser Thomas, Guibas Leonidas, Hanrahan Pat, Huang Qixing, Li Zimo, Savarese Silvio, Savva Manolis, Song Shuran, Su Hao, Xiao Jianxiong, Yi Li, Yu Fisher. ShapeNet: An Information-Rich 3D Model Repository. 2015.

Chehata Nesrine, Guo Li, Mallet Clément. Airborne lidar feature selection for urban classification using random forests // Laserscanning. 2009.

Chen Dave Zhenyu, Chang Angel X, Nießner Matthias. Scanrefer: 3d object localization in rgb-d scans using natural language // Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16. 2020. 202–221.

Chen Wenzheng, Ling Huan, Gao Jun, Smith Edward, Lehtinen Jaakko, Jacobson Alec, Fidler Sanja. Learning to predict 3D objects with an interpolation-based differentiable renderer // NeurIPS. 2019. 9609–19.

Chen Zhenyu, Gholami Ali, Nießner Matthias, Chang Angel X. Scan2Cap: Context-aware Dense Captioning in RGB-D Scans // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. 3193–3203.

Cho Jang Hyun, Hariharan Bharath. On the Efficacy of Knowledge Distillation // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). October 2019.

Choy Christopher, Park Jaesik, Koltun Vladlen. Fully convolutional geometric features // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. 8958–8966.

Chua Chin Seng, Jarvis Ray. 3D free-form surface registration and object recognition // International journal of computer vision. 1996. 17, 1. 77–99.

Cohen Laurent D. On active contour models and balloons // CVGIP: Image understanding. 1991. 53, 2. 211–218.

Community Blender Online. Blender - a 3D modelling and rendering package. Stichting Blender Foundation, Amsterdam, 2018.

Dai Angela, Chang Angel X, Savva Manolis, Halber Maciej, Funkhouser Thomas, Nießner Matthias. Scannet: Richly-annotated 3d reconstructions of indoor scenes // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. 5828–5839.

Debes Christian, Merentitis Andreas, Heremans Roel, Hahn Jürgen, Frangiadakis Nikolaos, Kasteren Tim van, Liao Wenzhi, Bellens Rik, Pižurica Aleksandra, Gautama Sidharta, others . Hyperspectral and LiDAR data fusion: Outcome of the 2013 GRSS data fusion contest // IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2014. 7, 6. 2405–2418.

Deng Jia, Dong Wei, Socher Richard, Li Li-Jia, Li Kai, Fei-Fei Li. Imagenet: A large-scale hierarchical image database // 2009 IEEE conference on computer vision and pattern recognition. 2009. 248–255.

Deschaud Jean-Emmanuel. KITTI-CARLA: a KITTI-like dataset generated by CARLA Simulator // arXiv preprint arXiv:2109.00892. 2021.

Devaranjan Jeevan, Kar Amlan, Fidler Sanja. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation // European Conference on Computer Vision. 2020. 715–733.

Devlin Jacob, Chang Ming-Wei, Lee Kenton, Toutanova Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding // arXiv preprint arXiv:1810.04805. 2018.

Dollar Piotr, Tu Zhuowen, Belongie Serge. Supervised learning of edges and object boundaries // 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). 2. 2006. 1964–1971.

Dosovitskiy Alexey, Beyer Lucas, Kolesnikov Alexander, Weissenborn Dirk, Zhai Xiaohua, Unterthiner Thomas, Dehghani Mostafa, Minderer Matthias, Heigold Georg, Gelly Sylvain, others . An image is worth 16x16 words: Transformers for image recognition at scale // arXiv preprint arXiv:2010.11929. 2020.

Dosovitskiy Alexey, Ros German, Codevilla Felipe, Lopez Antonio, Koltun Vladlen. CARLA: An open urban driving simulator // Conference on robot learning. 2017. 1–16.

Du Shihong, Zhang Fangli, Zhang Xiuyuan. Semantic classification of urban buildings combining VHR image and GIS data: An improved random forest approach // ISPRS journal of photogrammetry and remote sensing. 2015. 105. 107–119.

Duan Kaiwen, Bai Song, Xie Lingxi, Qi Honggang, Huang Qingming, Tian Qi. Centernet: Keypoint triplets for object detection // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. 6569–6578.

Edelsbrunner Herbert, Kirkpatrick David, Seidel Raimund. On the shape of a set of points in the plane // IEEE Transactions on information theory. 1983. 29, 4. 551–559.

Elberink Sander Oude, Vosselman George. Building reconstruction by target based graph matching on incomplete laser data: Analysis and limitations // Sensors. 2009. 9, 8. 6101–6118.

Engelcke Martin, Rao Dushyant, Wang Dominic Zeng, Tong Chi Hay, Posner Ingmar. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks // 2017 IEEE International Conference on Robotics and Automation (ICRA). 2017. 1355–1361.

Everingham Mark, Van Gool Luc, Williams Christopher KI, Winn John, Zisserman Andrew. The pascal visual object classes (voc) challenge // International journal of computer vision. 2010. 88, 2. 303–338.

Fang Jin, Zhou Dingfu, Yan Feilong, Zhao Tongtong, Zhang Feihu, Ma Yu, Wang Liang, Yang Ruigang. Augmented LiDAR simulator for autonomous driving // IEEE Robotics and Automation Letters. 2020. 5, 2. 1931–1938.

Faugeras Olivier D, Hebert Martial. The representation, recognition, and locating of 3-D objects // The international journal of robotics research. 1986. 5, 3. 27–52.

Geiger Andreas, Lenz Philip, Stiller Christoph, Urtasun Raquel. Vision meets robotics: The kitti dataset // The International Journal of Robotics Research. 2013. 32, 11. 1231–1237.

Genova Kyle, Yin Xiaoqi, Kundu Abhijit, Pantofaru Caroline, Cole Forrester, Sud Avneesh, Brewington Brian, Shucker Brian, Funkhouser Thomas. Learning 3D Semantic Segmentation with only 2D Image Supervision // arXiv preprint arXiv:2110.11325. 2021.

Geyer Jakob, Kassahun Yohannes, Mahmudi Mentar, Ricou Xavier, Durgesh Rupesh, Chung Andrew S, Hauswald Lorenz, Pham Viet Hoang, Mühlegg Maximilian, Dorn Sebastian, others . A2D2: Audi autonomous driving dataset // arXiv preprint arXiv:2004.06320. 2020.

Girshick Ross. Fast r-cnn // Proceedings of the IEEE international conference on computer vision. 2015. 1440–1448.

Girshick Ross, Donahue Jeff, Darrell Trevor, Malik Jitendra. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation // CVPR. VI 2014. 580–587.

Glorot Xavier, Bengio Yoshua. Understanding the difficulty of training deep feedforward neural networks // Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010. 249–256.

Golovinskiy Aleksey, Kim Vladimir G, Funkhouser Thomas. Shape-based recognition of 3D point clouds in urban environments // 2009 IEEE 12th International Conference on Computer Vision. 2009. 2154–2161.

González Alejandro, Villalonga Gabriel, Xu Jiaolong, Vázquez David, Amores Jaume, López Antonio M. Multiview random forest of local experts combining rgb and lidar data for pedestrian detection // 2015 IEEE Intelligent Vehicles Symposium (IV). 2015. 356–361.

Goodfellow Ian, Pouget-Abadie Jean, Mirza Mehdi, Xu Bing, Warde-Farley David, Ozair Sherjil, Courville Aaron, Bengio Yoshua. Generative Adversarial Nets // NeurIPS. 2014. 2672–2680.

Goyal Priya, Caron Mathilde, Lefaudeux Benjamin, Xu Min, Wang Pengchao, Pai Vivek, Singh Mannat, Liptchinsky Vitaliy, Misra Ishan, Joulin Armand, others . Self-supervised pretraining of visual features in the wild // arXiv preprint arXiv:2103.01988. 2021.

Griffiths D., Boehm J. Rapid Object Detection Systems, Utilising Deep Learning and Unamanned Aerial Systems (UAS) For Civil Engineering Applications // ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. V 2018. XLII-2. 391–398.

Griffiths D., Boehm J. Weighted Point Cloud Augmentation for Neural Network Training Data Class-imbalance // The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2019. XLII-2. 981–987.

Grill Jean-Bastien, Strub Florian, Altché Florent, Tallec Corentin, Richemond Pierre, Buchatskaya Elena, Doersch Carl, Avila Pires Bernardo, Guo Zhaohan, Gheshlaghi Azar Mohammad, Piot Bilal, kavukcuoglu koray, Munos Remi, Valko Michal. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning // Advances in Neural Information Processing Systems. 33. 2020. 21271–21284.

Gschwandtner Michael, Kwitt Roland, Uhl Andreas, Pree Wolfgang. BlenSor: Blender sensor simulation toolbox // International Symposium on Visual Computing. 2011. 199–208.

Guizilini Vitor, Ambrus Rares, Pillai Sudeep, Raventos Allan, Gaidon Adrien. 3d packing for self-supervised monocular depth estimation // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. 2485–2494.

Gusmão Guilherme Ferreira, Barbosa Carlos Roberto Hall, Raposo Alberto Barbosa. Development and validation of LiDAR sensor simulators based on parallel raycasting // Sensors. 2020. 20, 24. 7186.

Hackel Timo, Savinov Nikolay, Ladicky Lubor, Wegner Jan D, Schindler Konrad, Pollefeys Marc. Semantic3d. net: A new large-scale point cloud classification benchmark // arXiv preprint arXiv:1704.03847. 2017.

Hackel Timo, Wegner Jan D, Schindler Konrad. Fast semantic segmentation of 3D point clouds with strongly varying density // ISPRS annals of the photogrammetry, remote sensing and spatial information sciences. 2016. 3. 177–184.

Han Zhizhong, Chen Chao, Liu Yu-Shen, Zwicker Matthias. DRWR: A differentiable renderer without rendering for unsupervised 3D structure learning from silhouette images // arXiv:2007.06127. 2020.

Hanke Timo, Schaermann Alexander, Geiger Matthias, Weiler Konstantin, Hirsenkorn Nils, Rauch Andreas, Schneider Stefan-Alexander, Biebl Erwin. Generation and validation of virtual point cloud data for automated driving systems // 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). 2017. 1–6.

Hanocka Rana, Hertz Amir, Fish Noa, Giryes Raja, Fleishman Shachar, Cohen-Or Daniel. Meshcnn: a network with an edge // ACM Transactions on Graphics (TOG). 2019. 38, 4. 1–12.

He Kaiming, Girshick Ross, Dollár Piotr. Rethinking imagenet pre-training // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. 4918–4927.

He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification // Proceedings of the IEEE International Conference on Computer Vision. 2015. 1026–1034.

He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian. Deep residual learning for image recognition // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. 770–778.

Henderson Paul, Ferrari Vittorio. Learning single-image 3D reconstruction by generative modelling of shape, pose and shading // Int J Comp Vis. 2019.

Henzler Philipp, Mitra Niloy J., Ritschel Tobias. Escaping Plato's Cave: 3D Shape From Adversarial Rendering // ICCV. 2019.

Hermosilla Pedro, Ritschel Tobias, Ropinski Timo. Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning // Proceedings of the IEEE International Conference on Computer Vision. 2019. 52–60.

Hermosilla Pedro, Ritschel Tobias, Vázquez Pere-Pau, Vinacua Àlvar, Ropinski Timo. Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds // ACM Trans. Graph. 2018. 37, 6. 235:1–235:12.

Hinterstoisser Stefan, Holzer Stefan, Cagniart Cedric, Ilic Slobodan, Konolige Kurt, Navab Nassir, Lepetit Vincent. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes // 2011 international conference on computer vision. 2011. 858–865.

Hinterstoisser Stefan, Lepetit Vincent, Ilic Slobodan, Holzer Stefan, Bradski Gary, Konolige Kurt, Navab Nassir. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes // Asian conference on computer vision. 2012. 548–562.

Hou Ji, Dai Angela, Nießner Matthias. 3d-sis: 3d semantic instance segmentation of rgb-d scans // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. 4421–4430.

Hou Ji, Graham Benjamin, Nießner Matthias, Xie Saining. Exploring data-efficient 3d scene understanding with contrastive scene contexts // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. 15587–15597.

Hu Qingyong, Yang Bo, Xie Linhai, Rosa Stefano, Guo Yulan, Wang Zhihua, Trigoni Niki, Markham Andrew. Randla-net: Efficient semantic segmentation of large-scale point clouds // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. 11108–11117.

Huang Jonathan, Rathod Vivek, Sun Chen, Zhu Menglong, Korattikara Anoop, Fathi Alireza, Fischer Ian, Wojna Zbigniew, Song Yang, Guadarrama Sergio,

others . Speed/accuracy trade-offs for modern convolutional object detectors // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. 7310–7311.

Huh Minyoung, Agrawal Pulkit, Efros Alexei A. What makes ImageNet good for transfer learning? // arXiv preprint arXiv:1608.08614. 2016.

Isola Phillip, Zhu Jun-Yan, Zhou Tinghui, Efros Alexei A. Image-to-image translation with conditional adversarial networks // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. 1125–1134.

Jaeger Paul F, Kohl Simon AA, Bickelhaupt Sebastian, Isensee Fabian, Kuder Tristan Anselm, Schlemmer Heinz-Peter, Maier-Hein Klaus H. Retina U-Net: Embarrassingly simple exploitation of segmentation supervision for medical object detection // Machine Learning for Health Workshop. 2020. 171–183.

Johnson Andrew E, Hebert Martial. Using spin images for efficient object recognition in cluttered 3D scenes // IEEE Transactions on pattern analysis and machine intelligence. 1999. 21, 5. 433–449.

Johnson-Roberson Matthew, Barto Charles, Mehta Rounak, Sridhar Sharath Nittur, Rosaen Karl, Vasudevan Ram. Driving in the Matrix: Can virtual worlds replace human-generated annotations for real world tasks? // 2017 IEEE International Conference on Robotics and Automation (ICRA). 2017. 746–753.

Kanopoulos N., Vasanthavada N., Baker R. L. Design of an Image Edge Detection Filter Using the Sobel Operator // IEEE Journal of Solid-State Circuits. IV 1988. 23, 2. 358–367.

Kass Michael, Witkin Andrew, Terzopoulos Demetri. Snakes: Active Contour Models // International Journal of Computer Vision. I 1988. 1, 4. 321–331.

Kato Hiroharu, Beker Deniz, Morariu Mihai, Ando Takahiro, Matsuoka Toru, Kehl Wadim, Gaidon Adrien. Differentiable rendering: A survey // arXiv preprint arXiv:2006.12057. 2020.

Kato Hiroharu, Ushiku Yoshitaka, Harada Tatsuya. Neural 3D mesh renderer // CVPR. 2018. 3907–16.

Kazhdan Michael, Bolitho Matthew, Hoppe Hugues. Poisson surface reconstruction // Proceedings of the fourth Eurographics symposium on Geometry processing. 7. 2006.

Kehl Wadim, Manhardt Fabian, Tombari Federico, Ilic Slobodan, Navab Nassir. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again // Proceedings of the IEEE international conference on computer vision. 2017. 1521–1529.

Khoshelham Kourosh, Oude Elberink SJ. Role of dimensionality reduction in segment-based classification of damaged building roofs in airborne laser scanning data // Proceedings of the International Conference on Geographic Object Based Image Analysis, Rio de Janeiro, Brazil. 2012. 7–9.

Kim Yoon, Denton Carl, Hoang Luong, Rush Alexander M. Structured attention networks // arXiv preprint arXiv:1702.00887. 2017.

Kluckner Stefan, Bischof Horst. Image-based building classification and 3d modeling with super-pixels // ISPRS Technical Commission III Symposium on Photogrammetry Computer Vision and Image Analysis. 2010. 233–238.

Knopp Jan, Prasad Mukta, Van Gool Luc. Orientation invariant 3D object classification using hough transform based methods // Proceedings of the ACM workshop on 3D object retrieval. 2010. 15–20.

Knopp Jan, Prasad Mukta, Van Gool Luc. Scene cut: Class-specific object detection and segmentation in 3d scenes // 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission. 2011. 180–187.

Kolotouros Nikos, Pavlakos Georgios, Black Michael J., Daniilidis Kostas. Learning to Reconstruct 3D Human Pose and Shape via Model-Fitting in the Loop

// Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). October 2019.

Kornblith Simon, Shlens Jonathon, Le Quoc V. Do better imagenet models transfer better? // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. 2661–2671.

Kraus K., Pfeifer N. Advanced DTM Generation from LIDAR Data // Proceedings of the ISPRS Workshop on Land Surface Mapping and Characterization Using Laser Altimetry. 2001.

Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey E. Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. 2012. 25. 1097–1105.

Kundu Abhijit, Li Yin, Rehg James M. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. 3559–3568.

Kölle Michael, Laupheimer Dominik, Schmohl Stefan, Haala Norbert, Rottensteiner Franz, Wegner Jan Dirk, Ledoux Hugo. The Hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and Multi-View-Stereo // ISPRS Open Journal of Photogrammetry and Remote Sensing. 2021. 1. 11.

Lafarge Florent, Descombes Xavier, Zerubia Josiane, Pierrot-Deseilligny Marc. Structural approach for building reconstruction from a single DSM // IEEE Transactions on pattern analysis and machine intelligence. 2008. 32, 1. 135–147.

Lahoud Jean, Ghanem Bernard, Pollefeys Marc, Oswald Martin R. 3d instance segmentation via multi-task metric learning // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. 9256–9266.

Lalonde Jean-François, Unnikrishnan Ranjith, Vandapel Nicolas, Hebert Martial. Scale selection for classification of point-sampled 3D surfaces // Fifth Inter-

national Conference on 3-D Digital Imaging and Modeling (3DIM'05). 2005. 285–292.

Lalonde Jean-François, Vandapel Nicolas, Huber Daniel F, Hebert Martial. Natural terrain classification using three-dimensional ladar data for ground robot mobility // Journal of field robotics. 2006. 23, 10. 839–861.

LeCun Yann, Bottou Léon, Bengio Yoshua, Haffner Patrick. Gradient-based learning applied to document recognition // Proceedings of the IEEE. 1998. 86, 11. 2278–2324.

Ledig Christian, Theis Lucas, Huszár Ferenc, Caballero Jose, Cunningham Andrew, Acosta Alejandro, Aitken Andrew, Tejani Alykhan, Totz Johannes, Wang Zehan, others . Photo-realistic single image super-resolution using a generative adversarial network // CVPR. 2017. 4681–90.

Lee Impyeong, Schenk Toni. Perceptual organization of 3D surface points // International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences. 2002. 34, 3/A. 193–198.

Lehtinen Jaakko, Munkberg Jacob, Hasselgren Jon, Laine Samuli, Karras Tero, Aittala Miika, Aila Timo. Noise2noise: Learning image restoration without clean data // arXiv:1803.04189. 2018.

Leibe Bastian, Schiele Bernt. Interleaving object categorization and segmentation // Cognitive vision systems. 2006. 145–161.

Li Bo, Zhang Tianlei, Xia Tian. Vehicle detection from 3d lidar using fully convolutional network // arXiv preprint arXiv:1608.07916. 2016.

Li Tzu-Mao, Aittala Miika, Durand Frédo, Lehtinen Jaakko. Differentiable monte carlo ray tracing through edge sampling // ACM Transactions on Graphics (TOG). 2018a. 37, 6. 1–11.

Li Weijia, He Conghui, Fang Jiarui, Zheng Juepeng, Fu Haohuan, Yu Le. Semantic segmentation-based building footprint extraction using very high-resolution satellite images and multi-source GIS data // Remote Sensing. 2019. 11, 4. 403.

Li Yangyan, Bu Rui, Sun Mingchao, Wu Wei, Di Xinhan, Chen Baoquan. PointCNN: Convolution On X-Transformed Points // Advances in Neural Information Processing Systems 31. 2018b. 820–830.

Lin Tsung-Yi, Goyal Priya, Girshick Ross, He Kaiming, Dollár Piotr. Focal loss for dense object detection // Proceedings of the IEEE international conference on computer vision. 2017. 2980–2988.

Lin Tsung-Yi, Maire Michael, Belongie Serge, Hays James, Perona Pietro, Ramanan Deva, Dollár Piotr, Zitnick C Lawrence. Microsoft coco: Common objects in context // European conference on computer vision. 2014. 740–755.

Liu Kun, Boehm Jan. A new framework for interactive segmentation of point clouds // International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives. 40, 5. 2014. 357–362.

Liu Shichen, Chen Weikai, Li Tianye, Li Hao. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction // arXiv preprint arXiv:1901.05567. 2019.

Liu Wei, Anguelov Dragomir, Erhan Dumitru, Szegedy Christian, Reed Scott, Fu Cheng-Yang, Berg Alexander C. Ssd: Single Shot Multibox Detector // European Conference on Computer Vision. 2016. 21–37.

Liu Yuxuan, Yixuan Yuan, Liu Ming. Ground-aware monocular 3d object detection for autonomous driving // IEEE Robotics and Automation Letters. 2021. 6, 2. 919–926.

Liu Zhouyong, Luo Shun, Li Wubin, Lu Jingben, Wu Yufan, Sun Shilei, Li Chunguo, Yang Luxi. Convtransformer: A convolutional transformer network for video frame synthesis // arXiv preprint arXiv:2011.10185. 2020.

Lodha Suresh K, Fitzpatrick Darren M, Helmbold David P. Aerial lidar data classification using adaboost // Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007). 2007. 435–442.

Loper Matthew M, Black Michael J. OpenDR: An approximate differentiable renderer // European Conference on Computer Vision. 2014. 154–169.

Lowe David G. Distinctive image features from scale-invariant keypoints // International journal of computer vision. 2004. 60, 2. 91–110.

Márquez-Neila P., Baumela L., Alvarez L. A Morphological Approach to Curvature-Based Evolution of Curves and Surfaces // IEEE Transactions on Pattern Analysis and Machine Intelligence. I 2014. 36, 1. 2–17.

Meng Qinghao, Wang Wenguan, Zhou Tianfei, Shen Jianbing, Van Gool Luc, Dai Dengxin. Weakly supervised 3d object detection from lidar point cloud // European Conference on Computer Vision. 2020. 515–531.

Mildenhall Ben, Srinivasan Pratul P, Tancik Matthew, Barron Jonathan T, Ramamoorthi Ravi, Ng Ren. Nerf: Representing scenes as neural radiance fields for view synthesis // European conference on computer vision. 2020. 405–421.

Mnih Volodymyr, Hinton Geoffrey E. Learning to detect roads in high-resolution aerial images // European Conference on Computer Vision. 2010. 210–223.

Mou Lichao, Zhu Xiao Xiang. Spatiotemporal scene interpretation of space videos via deep neural network and tracklet analysis // 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). 2016. 1823–1826.

Mustikovela Siva Karthik, De Mello Shalini, Prakash Aayush, Iqbal Umar, Liu Sifei, Nguyen-Phuoc Thu, Rother Carsten, Kautz Jan. Self-Supervised Object Detection via Generative Image Synthesis // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). October 2021. 8609–8618.

Newell Alejandro, Yang Kaiyu, Deng Jia. Stacked hourglass networks for human pose estimation // ECCV. 2016.

Niemeyer Joachim, Rottensteiner Franz, Soergel Uwe. Conditional random fields for lidar point cloud classification in complex urban areas // ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. 2012. 1, 3. 263–268.

Parikh Ankur P, Täckström Oscar, Das Dipanjan, Uszkoreit Jakob. A decomposable attention model for natural language inference // arXiv preprint arXiv:1606.01933. 2016.

Park Dennis, Ambrus Rares, Guizilini Vitor, Li Jie, Gaidon Adrien. Is Pseudo-Lidar needed for Monocular 3D Object detection? // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021. 3142–3152.

Park Jeong Joon, Florence Peter, Straub Julian, Newcombe Richard, Lovegrove Steven. Deepsdf: Learning continuous signed distance functions for shape representation // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019a. 165–174.

Park Kiru, Patten Timothy, Vincze Markus. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019b. 7668–7677.

Premebida Cristiano, Carreira Joao, Batista Jorge, Nunes Urbano. Pedestrian detection combining RGB and dense LIDAR data // 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2014. 4112–4117.

Qi Charles R, Chen Xinlei, Litany Or, Guibas Leonidas J. ImVoteNet: Boosting 3D Object Detection in Point Clouds with Image Votes // arXiv preprint arXiv:2001.10692. 2020.

Qi Charles R, Litany Or, He Kaiming, Guibas Leonidas J. Deep hough voting for 3d object detection in point clouds // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. 9277–9286.

Qi Charles R, Liu Wei, Wu Chenxia, Su Hao, Guibas Leonidas J. Frustum pointnets for 3d object detection from rgb-d data // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. 918–927.

Qi Charles R, Su Hao, Mo Kaichun, Guibas Leonidas J. Pointnet: Deep learning on point sets for 3d classification and segmentation // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017a. 652–660.

Qi Charles Ruizhongtai, Yi Li, Su Hao, Guibas Leonidas J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space // Advances in Neural Information Processing Systems. 2017b. 30.

Quadros A, Underwood James Patrick, Douillard Bertrand. An occlusion-aware feature for range images // 2012 IEEE International Conference on Robotics and Automation. 2012. 4428–4435.

Redmon Joseph, Divvala Santosh, Girshick Ross, Farhadi Ali. You only look once: Unified, real-time object detection // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. 779–788.

Reitmann Stefan, Neumann Lorenzo, Jung Bernhard. BLAINDER—a blender AI add-on for generation of semantically labeled depth-sensing data // Sensors. 2021. 21, 6. 2144.

Ren Shaoqing, He Kaiming, Girshick Ross, Sun Jian. Faster r-cnn: Towards real-time object detection with region proposal networks // Advances in neural information processing systems. 2015. 28. 91–99.

Ren Zhongzheng, Misra Ishan, Schwing Alexander G, Girdhar Rohit. 3D Spatial Recognition without Spatially Labeled 3D // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. 13204–13213.

Richter Stephan R, Vineet Vibhav, Roth Stefan, Koltun Vladlen. Playing for data: Ground truth from computer games // European conference on computer vision. 2016. 102–118.

Ronneberger Olaf, Fischer Philipp, Brox Thomas. U-net: Convolutional networks for biomedical image segmentation // International Conference on Medical image computing and computer-assisted intervention. 2015. 234–241.

Ros German, Sellart Laura, Materzynska Joanna, Vazquez David, Lopez Antonio M. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. 3234–3243.

Roynard Xavier, Deschaud Jean-Emmanuel, Goulette François. Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification // The International Journal of Robotics Research. 2018a. 37, 6. 545–557.

Roynard Xavier, Deschaud Jean-Emmanuel, Goulette François. Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification // The International Journal of Robotics Research. 2018b. 37, 6. 545–557.

Rumelhart David E, Hinton Geoffrey E, Williams Ronald J. Learning internal representations by error propagation. 1985.

Rutzinger M, Elberink S Oude, Pu Shi, Vosselman G. Automatic extraction of vertical walls from mobile and airborne laser scanning data // International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. 2009. 38, W8. 7–11.

Rutzinger Martin, Pratihast Arun Kumar, Oude Elberink S, Vosselman George. Detection and modelling of 3D trees from mobile laser scanning data // Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 2010. 38. 520–525.

Saeedi P., Zwick H. Automatic Building Detection in Aerial and Satellite Images // Robotics and Vision 2008 10th International Conference on Control, Automation. XII 2008. 623–629.

Sanchez Castillo E, Griffiths D, Boehm J. Semantic Segmentation of Terrestrial LIDAR Data Using Co-Registered RGB Data // The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2021. 43. 223–229.

Schonberger Johannes L, Frahm Jan-Michael. Structure-from-motion revisited // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. 4104–4113.

Schult Jonas, Engelmann Francis, Kontogianni Theodora, Leibe Bastian. Dualconvmesh-net: Joint geodesic and euclidean convolutions on 3d meshes // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. 8612–8622.

Secord John, Zakhor Avideh. Tree detection in urban regions using aerial lidar and image data // IEEE Geoscience and Remote Sensing Letters. 2007. 4, 2. 196–200.

Sermanet Pierre, Eigen David, Zhang Xiang, Mathieu Michaël, Fergus Rob, LeCun Yann. Overfeat: Integrated Recognition, Localization and Detection Using Convolutional Networks // arXiv preprint arXiv:1312.6229. 2013.

Serna Andrés, Marcotegui Beatriz, Goulette François, Deschaud Jean-Emmanuel. Paris-rue-Madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods // 4th International Conference on Pattern Recognition, Applications and Methods ICPRAM 2014. 2014.

Shoeybi Mohammad, Patwary Mostofa, Puri Raul, LeGresley Patrick, Casper Jared, Catanzaro Bryan. Megatron-lm: Training multi-billion parameter language models using model parallelism // arXiv preprint arXiv:1909.08053. 2019.

Silberman Nathan, Hoiem Derek, Kohli Pushmeet, Fergus Rob. Indoor segmentation and support inference from rgbd images // European conference on computer vision. 2012. 746–760.

Simonyan Karen, Zisserman Andrew. Very Deep Convolutional Networks for Large-Scale Image Recognition // arXiv:1409.1556 [cs]. IX 2014.

Sirmacek Beril, Unsalan Cem. Building detection from aerial images using invariant color features and shadow information // 2008 23rd international symposium on computer and information sciences. 2008. 1–5.

Song Shuran, Lichtenberg Samuel P, Xiao Jianxiong. Sun rgb-d: A rgb-d scene understanding benchmark suite // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. 567–576.

Song Shuran, Xiao Jianxiong. Sliding shapes for 3d object detection in depth images // European conference on computer vision. 2014. 634–651.

Song Shuran, Xiao Jianxiong. Deep sliding shapes for amodal 3D object detection in RGB-D images // CVPR. 2016.

Spiegel Steven, Shanks Casey, Chen Jorge. Effectiveness of Deep Learning Trained on SynthCity Data for Urban Point-Cloud Classification // Photogrammetric Engineering & Remote Sensing. 2022. 88, 2. 113–120.

Squid Turbo. Metropolis City Experience. 2022. Accessed: 2019-07-10.

Strudel Robin, Garcia Ricardo, Laptev Ivan, Schmid Cordelia. Segmenter: Transformer for Semantic Segmentation // arXiv preprint arXiv:2105.05633. 2021.

Studer Linda, Alberti Michele, Pondenkandath Vinaychandran, Goktepe Pinar, Kolonko Thomas, Fischer Andreas, Liwicki Marcus, Ingold Rolf. A comprehensive study of imagenet pre-training for historical document image analysis // 2019 International Conference on Document Analysis and Recognition (ICDAR). 2019. 720–725.

Su Hang, Jampani Varun, Sun Deqing, Maji Subhransu, Kalogerakis Evangelos, Yang Ming-Hsuan, Kautz Jan. SPLATNet: Sparse Lattice Networks for Point Cloud Processing // The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

Su Yongzhi, Rambach Jason, Pagani Alain, Stricker Didier. SynPo-Net—Accurate and Fast CNN-Based 6DoF Object Pose Estimation Using Synthetic Training // Sensors. 2021. 21, 1. 300.

Sun Pei, Kretzschmar Henrik, Dotiwalla Xerxes, Chouard Aurelien, Patnaik Vijaysai, Tsui Paul, Guo James, Zhou Yin, Chai Yuning, Caine Benjamin, others . Scalability in perception for autonomous driving: Waymo open dataset // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. 2446–2454.

Sundermeyer Martin, Marton Zoltan-Csaba, Durner Maximilian, Triebel Rudolph. Augmented autoencoders: Implicit 3D orientation learning for 6D object detection // International Journal of Computer Vision. 2020. 128, 3. 714–729.

Suzuki Satoshi, be KeiichiA. Topological Structural Analysis of Digitized Binary Images by Border Following // Computer Vision, Graphics, and Image Processing. IV 1985. 30, 1. 32–46.

Tang Yew Siang, Lee Gim Hee. Transferable semi-supervised 3d object detection from rgb-d data // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. 1931–1940.

Tekin Bugra, Sinha Sudipta N, Fua Pascal. Real-time seamless single shot 6d object pose prediction // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. 292–301.

Thomas Hugues, Qi Charles R, Deschaud Jean-Emmanuel, Marcotegui Beatriz, Goulette François, Guibas Leonidas J. Kpconv: Flexible and deformable convolution for point clouds // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. 6411–6420.

Tulsiani Shubham, Zhou Tinghui, Efros Alexei A, Malik Jitendra. Multi-view supervision for single-view reconstruction via differentiable ray consistency // CVPR. 2017. 2626–34.

Ullman Shimon. The interpretation of structure from motion // Proceedings of the Royal Society of London. Series B. Biological Sciences. 1979. 203, 1153. 405–426.

Vallet Bruno, Brédif Mathieu, Serna Andrés, Marcotegui Beatriz, Paparoditis Nicolas. TerraMobilita/iQmulus urban point cloud analysis benchmark // Computers & Graphics. 2015. 49. 126–133.

Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Łukasz, Polosukhin Illia. Attention is all you need // Advances in neural information processing systems. 2017. 5998–6008.

Velizhev Alexander, Shapovalov Roman, Schindler Konrad. Implicit shape models for object detection in 3D point clouds // International Society of Photogrammetry and Remote Sensing Congress. 2. 2012. 2.

Vinyals Oriol, Bengio Samy, Kudlur Manjunath. Order matters: Sequence to sequence for sets // arXiv preprint arXiv:1511.06391. 2015.

Vosselman George. Slope Based Filtering of Laser Altimertry Data // International Archives of Photogrammetry and Remote Sensing. 2000. 935–942.

Vosselman George, Maas Hans-Gerd. Airborne and terrestrial laser scanning. 2010.

Wang Dominic Zeng, Posner Ingmar. Voting for voting in online point cloud object detection. // Robotics: Science and Systems. 1, 3. 2015. 10–15.

Wang Fei, Zhuang Yan, Gu Hong, Hu Huosheng. Automatic generation of synthetic LiDAR point clouds for 3-D data analysis // IEEE Transactions on Instrumentation and Measurement. 2019a. 68, 7. 2671–2673.

Wang Haiyan, Rong Xuejian, Yang Liang, Wang Shuihua, Tian Yingli. Towards Weakly Supervised Semantic Segmentation in 3D Graph-Structured Point Clouds of Wild Scenes. // BMVC. 2019b. 284.

Wang He, Sridhar Srinath, Huang Jingwei, Valentin Julien, Song Shuran, Guibas Leonidas J. Normalized object coordinate space for category-level 6d object pose and size estimation // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019c. 2642–2651.

Wang Lin, Yoon Kuk-Jin. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021.

Wang Yasi, Yao Hongxun, Zhao Sicheng. Auto-encoder based dimensionality reduction // Neurocomputing. 2016. 184. 232–242.

Weinmann Martin, Jutzi Boris, Hinz Stefan, Mallet Clément. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers // ISPRS Journal of Photogrammetry and Remote Sensing. 2015. 105. 286–304.

Weinmann Martin, Jutzi Boris, Mallet Clément. Semantic 3D scene interpretation: A framework combining optimal neighborhood size selection with relevant features // ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2014. 2, 3. 181.

Weisstein Eric. Fixed Point. 2020a.

Weisstein Eric. Hammersley Point Set. 2020b.

Wilson Benjamin, Kira Zsolt, Hays James. 3d for free: Crossmodal transfer learning using hd maps // arXiv preprint arXiv:2008.10592. 2020.

Winiwarter Lukas, Pena Alberto Manuel Esmorís, Weiser Hannah, Anders Katharina, Sanchez Jorge Martínez, Searle Mark, Höfle Bernhard. Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic 3D laser scanning // arXiv preprint arXiv:2101.09154. 2021.

Wu Bichen, Wan Alvin, Yue Xiangyu, Keutzer Kurt. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar

point cloud // 2018 IEEE International Conference on Robotics and Automation (ICRA). 2018. 1887–1893.

Wu Bichen, Zhou Xuanyu, Zhao Sicheng, Yue Xiangyu, Keutzer Kurt. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud // 2019 International Conference on Robotics and Automation (ICRA). 2019. 4376–4382.

Xia Gui-Song, Bai Xiang, Ding Jian, Zhu Zhen, Belongie Serge, Luo Jiebo, Datcu Mihai, Pelillo Marcello, Zhang Liangpei. DOTA: A large-scale dataset for object detection in aerial images // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. 3974–3983.

Xiao Jianxiong, Owens Andrew, Torralba Antonio. Sun3d: A database of big spaces reconstructed using sfm and object labels // Proceedings of the IEEE international conference on computer vision. 2013. 1625–1632.

Xie Saining, Gu Jiatao, Guo Demi, Qi Charles R, Guibas Leonidas, Litany Or. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding // European Conference on Computer Vision. 2020. 574–591.

Xiong Xuehan, Munoz Daniel, Bagnell J Andrew, Hebert Martial. 3-d scene analysis via sequenced predictions over points and regions // 2011 IEEE International Conference on Robotics and Automation. 2011. 2609–2616.

Yang Bo, Wang Jianan, Clark Ronald, Hu Qingyong, Wang Sen, Markham Andrew, Trigoni Niki. Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds // arXiv preprint arXiv:1906.01140. 2019.

Yang Yaoqing, Feng Chen, Shen Yiru, Tian Dong. FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2018.

Yu Jun, Yao Jinghan, Zhang Jian, Yu Zhou, Tao Dacheng. SPRNet: single-pixel reconstruction for one-stage instance segmentation // IEEE transactions on cybernetics. 2020. 51, 4. 1731–1742.

Zakharov Sergey, Kehl Wadim, Bhargava Arjun, Gaidon Adrien. Autolabeling 3d objects with differentiable rendering of sdf shape priors // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. 12224–12233.

Zhang Yinmin, Ma Xinzhu, Yi Shuai, Hou Jun, Wang Zhihui, Ouyang Wanli, Xu Dan. Learning Geometry-Guided Depth via Projective Modeling for Monocular 3D Object Detection // arXiv preprint arXiv:2107.13931. 2021a.

Zhang Yunpeng, Lu Jiwen, Zhou Jie. Objects are Different: Flexible Monocular 3D Object Detection // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021b. 3289–3298.

Zhang Zaiwei, Girdhar Rohit, Joulin Armand, Misra Ishan. Self-Supervised Pretraining of 3D Features on Any Point-Cloud // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). October 2021c. 10252–10263.

Zhao Hengshuang, Jiang Li, Jia Jiaya, Torr Philip HS, Koltun Vladlen. Point transformer // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021. 16259–16268.

Zhou Xingyi, Wang Dequan, Krähenbühl Philipp. Objects as points // arXiv preprint arXiv:1904.07850. 2019a.

Zhou Yi, Barnes Connelly, Lu Jingwan, Yang Jimei, Li Hao. On the continuity of rotation representations in neural networks // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019b. 5745–5753.

Zhou Yin, Tuzel Oncel. Voxelnet: End-to-end learning for point cloud based 3d object detection // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. 4490–4499.

Zhu Jingwei, Gehrung Joachim, Huang Rong, Borgmann Björn, Sun Zhenghao, Hoegner Ludwig, Hebel Marcus, Xu Yusheng, Stilla Uwe. TUM-MLS-2016: An annotated mobile LiDAR dataset of the TUM city campus for semantic point cloud interpretation in urban areas // Remote Sensing. 2020. 12, 11. 1875.

Zolanvari SM, Ruano Susana, Rana Aakanksha, Cummins Alan, Silva Rogerio Eduardo da, Rahbar Morteza, Smolic Aljosa. DublinCity: Annotated LiDAR point cloud and its applications // arXiv preprint arXiv:1909.03613. 2019.