Deep Vision for Indoor Understanding and Localisation



Henry Howard-Jenkins
St John's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Hilary Term, 2022

In loving memory of Jonathan Howard and Jan Sarnowski,
my father and mój dziadzia.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Henry Howard-Jenkins, St John's College

Acknowledgements

I first would like to express my deepest gratitude towards my supervisor Prof. Victor Prisacariu, who first convinced me to undertake this DPhil. He has been a great pleasure to work with over the last five and a half years and he has given me the opportunity to learn and grow with his invaluable guidance, ideas and discussions throughout this time.

I would also like to thank Prof. David Murray for the generous wisdom he has shared with me. Thank you to my examiners, Prof. Andrew Davison and Prof. Andrew Zisserman. I appreciate the time they have given to provide such valuable and thoughtful feedback on my research.

I am very thankful to my friends and collaborators in AVL who have offered advice, feedback, lunch and coffee breaks throughout the years. Particularly to Shuda, Wesley, Michael, and Theo (who also offered tech support). Thank you to my collaborators in the MoveCare Project, it was a pleasure to work with you all towards such a meaningful goal. In addition, thank you to Vasileios Balntas who offered me much needed help when starting my DPhil, and for his guidance during my internship at (F)RL-R Surreal. I would also like to thank all those that I worked with during the internship.

Thank you to my friends. In particular Louis and Alice: your chats and support helped me through the most difficult times. Adam, Anton, Babar, Bruce, Conley, Ed (J and S), Greg, Gustaf, Harry, Haven, Jack, Joe, Karan, Karl, Luke, Maxime, Monty, Nick, Ollie, Rory, Tim, and all the friends I've made in OUGC: Oxford is short, but life is long. You've given me some of my favourite memories in the short part, so I'm looking forward to what's to come in the rest. Thank you to my housemates in 42 London Road, who put up with me during this write-up.

Finally, I'd like to thank my sister, Anastasia, and my parents, Hania and David, for their unconditional love. I'll be forever grateful for the support they've given me throughout my life.

Henry Howard-Jenkins
St John's College

Doctor of Philosophy
Hilary Term, 2022

Deep Vision for Indoor Understanding and Localisation

Abstract

The challenge considered in this thesis is that of *domesticating* deep learning-based vision. Firstly, we advance the use of deep learning for indoor architectural comprehension. Secondly, we explore a taming of deep neural networks for low-cost inference such that they can be used in real-time or on constrained hardware.

Room or building architecture forms the foundational level of geometry that will be visible within an indoor space, and remains static. Therefore, better understanding of architectural structure enables reasoning over a long period of time without re-training or re-mapping. We introduce Planar R-CNN which can perform unconstrained room layout estimation. Planar R-CNN detects instances of architectural planes, as well as regresses the 3D plane equation for each detection. Planar R-CNN moves room layout estimation beyond strict constraints and can infer arbitrary planar geometries. Following this, we explore visual localisation through architectural reasoning. We introduce LaLaLoc, which learns a latent representation of layout shared between rendered structure and panorama images, enabling cross-modal alignment without explicit layout prediction. LaLaLoc is able to localise across furniture configurations and in previously unvisited scenes given only a floor plan structure as a prior. LaLaLoc++ extends this by unifying representations of layout structure between 2D floor plans and images.

Neural network based methods can often be too slow to meet the requirements of a mobile robot, or too large to run on its hardware. We consider low-cost network design, and in particular group convolution as a design tool. Grouped convolution has become increasingly common across the spectrum of speed and accuracy trade-offs architecture design. However, the implications of grouped convolutional layers have remained relatively unexplored. We introduce Mobile R-CNN which, through group convolutional design, performs highly accurate real-time instance segmentation. Finally, we formalise this design of grouped convolutional architectures with the GroSS representation of group convolution blocks, which ultimately enables thorough exploration of the design space where it may not have been feasible before.

Contents

1 Introduction **1**

1.1 Tasks . . . 2

1.1.1 Deep Learning for Architectural Understanding . . . 2

1.1.1.1 Room Layout Estimation . . . 3

1.1.1.2 Localisation with Floor Plan Priors . . . 3

1.1.2 Group Convolution for Low-cost Network Design . . . 4

1.1.2.1 Real-time Instance Segmentation . . . 4

1.1.2.2 Enabling Exploration of Group Topologies in Convolutional Neural Networks . . . 5

1.2 Thesis Structure . . . 6

1.3 Publications . . . 7

2 Literature Review **8**

2.1 Room Layout Estimation . . . 8

2.1.1 Implicit Room Layouts . . . 10

2.2 Visual Localisation . . . 11

2.2.1 Scene-specific Localisation . . . 11

2.2.2 Scene-agnostic Localisation . . . 12

2.2.3 Cross-modal and Cross-view Localisation . . . 14

2.2.4 Floor Plan Localisation . . . 15

2.3 Real-time Instance Segmentation . . . 16

2.4 Group Convolution . . . 17

2.5 Low-rank Factorisation . . . 18

2.6 Neural Architecture Search . . . 18

I Architectural Understanding 20

3 Explicit Capture of Unconstrained Room Layout from RGB Images 21
3.1 Introduction . . . 21
3.1.1 Contributions . . . 22
3.2 Thinking Outside the Box: Generation of Unconstrained 3D Room Layouts . . . 23
3.3 Conclusion . . . 40

4 Implicit Understanding of Architectural Structure for Localisation 42
4.1 Introduction . . . 42
4.1.1 Contributions . . . 42
4.2 LaLaLoc: Latent Layout Localisation in Dynamic, Unvisited Environments . . . 44
4.3 Conclusion . . . 58

5 Structural Comprehension From 2D Floor Plans 59
5.1 Introduction . . . 59
5.1.1 Contributions . . . 59
5.2 LaLaLoc++: Global Floor Plan Comprehension for Layout Localisation in Unvisited Environments . . . 61
5.3 Conclusion . . . 75

II Group Convolution for Low-cost Network Design 76

6 Real-time Instance Segmentation Through Group-Convolutions 77
6.1 Context . . . 77
6.2 Introduction . . . 78
6.3 Empirical Motivation: From Depthwise to Group Convolution . . . 79
6.3.1 Depthwise-Separable Convolutions . . . 79
6.3.2 Group Convolution . . . 80
6.4 Mobile R-CNN . . . 81
6.4.1 Architecture . . . 83
6.4.2 Implementation Details . . . 84
6.4.2.1 Training Routine . . . 85
6.5 Instance Segmentation . . . 85
6.5.1 Dataset . . . 85

6.5.2 Methods for Comparison . . . 86
6.5.3 Results . . . 86
6.5.3.1 COCO 2017 . . . 87
6.5.4 Group Selection . . . 88
6.5.5 Image Resolution . . . 88
6.6 Image Classification . . . 90
6.7 Down-stream Application . . . 91
6.7.1 Aphex Twin Set Art at Coachella 2019 . . . 91
6.8 Conclusion . . . 92

7 Enabling Exploration of Group Topologies in Convolutional Neural Networks 94
7.1 Introduction . . . 94
7.1.1 Contributions . . . 95
7.2 GroSS: Group-Size Series Decomposition for Grouped Architecture Search 96
7.3 Conclusion . . . 116

8 Conclusion 117
8.1 Achievements and Future Work . . . 117

Bibliography 122

A Enlarged Figures 138

Chapter 1

Introduction

The application of deep learning to vision tasks offers significant potential to enable domestic robotic agents to better reason their surroundings. In the decade since the explosion of research interest marked by the introduction of AlexNet [Krizhevsky et al., 2012], deep learning has achieved great advancements for a variety of vision problems, such as object detection [Zou et al., 2019] and semantic segmentation [Lateef and Ruichek, 2019]. However, the potential offered by deep learning methods often comes at a (computational) cost, with state-of-the-art[1] models often ignoring model size and latency all together [Ethayarajh and Jurafsky, 2020].

In this thesis, we aim to *domesticate* deep learning-based vision. The challenges considered are motivated by the desire to apply deep learning to vision tasks within a domestic environment, and to explore its design for use on robots that have to make online decisions, but whose computational resources are limited.

To that end, this thesis is one of two overarching research themes. Their core motivation is found through their application to the Horizon 2020 project: MoveCare[2]. Lack of effective caring solutions has led to many elderly people being moved to nursing homes too early. To combat this, MoveCare set forth to develop a multi-actor platform to provide assistance and transparent monitoring of the elderly at home. In doing so, it highlights some key challenges to address to make such a system effective. The project ultimately aims to enable the elderly to live a healthy and social lifestyle at home, and for longer.

In the following, we describe the research themes explored in this thesis, as well as providing their context and application to MoveCare where our research serves to improve robotic operation in the challenging domestic environment.

[1] With state-of-the-art being defined by task leaderboards.

[2] http://www.movecare-project.eu

The first is the application and advancement of deep learning for understanding-based tasks of indoor environments. Specifically, we aim to advance visual understanding of indoor architectural structure for the goal of localisation with respect to a floor plan. This is explored through localisation with a floor plan prior itself, as well as room layout estimation. In the wide and long-term deployment of an autonomous robot in homes, the means to infer and reason with permanent architectural structure would enable the use of existing building plans to reduce initial technical setup, such as pre-scanning of the home, as well as aid in map maintenance over a long period of time where objects and furniture may move.

The second is the design of fast and low-cost networks with group convolutions. This is approached first by significantly reducing latency of instance segmentation to real-time speeds, before utilising the insights gained to form a generalised search for more formalised design and validation of group convolutional topologies across a network. In MoveCare, the low-cost object understanding provided by instance segmentation is leveraged to locate lost objects that are of personal importance to the user. However, the formalised design of group convolution in networks provides an additional tool for the application of deep learning in performance constrained settings.

In the following, we will cover in more detail the tasks that are considered in this thesis in their academic setting.

1.1 Tasks

The previous section outlined the challenges considered in this thesis and their motivation in relation to MoveCare. Here, we will introduce the specific scientific tasks that are addressed in this thesis. We pay particular attention to three understanding tasks: room layout estimation (Section 1.1.1.1); floor plan localisation (Section 1.1.1.2); real-time instance segmentation (Section 1.1.2.1). Finally, we consider a general contribution for design of convolutional neural networks in the form of efficient search of group topologies (Section 1.1.2.2).

1.1.1 Deep Learning for Architectural Understanding

We explore two tasks related to understanding of architectural structure: room layout estimation and floor plan localisation. Both require that the high-level concept of layout (defined below) be inferred from images. For room layout estimation, the layout must be explicitly expressed, whereas for the floor plan localisation it is utilised

to reason location within the building. In the following, we will introduce these tasks more thoroughly.

1.1.1.1 Room Layout Estimation

Layout estimation aims to infer the structure of a room beneath the clutter, leaving behind only the walls, floor and ceiling that enclose the scene. This task is inherently paired with floor plans, as the layout estimated in an image describes the structure that is represented in a floor plan.

Despite the varied rooms found in the buildings around us, work on layout estimation has largely only been able to recover geometries that obey heavy constraints. The box approximation [Hedau et al., 2009] and the Manhattan World assumption [Coughlan and Yuille, 1999] are the most widely used. Although these assumptions lead to good layouts for some indoor environments, it is easy to produce examples of room geometries which cannot be accurately captured due to the constraints. In the context of floor plan localisation, for example, it is important to avoid preventable ambiguity caused in the fitting of an inaccurate approximation to a room.

In Chapter 3, we address unconstrained layout estimation through the introduction of a neural network that is able to detect room planes in monocular images, as well as directly regressing a representative 3D plane. As we show in Chapter 3 this relaxation of the constraints on room shape allows representation of a wider variety of scene geometries.

1.1.1.2 Localisation with Floor Plan Priors

Visual localisation estimates the location of a new image using prior knowledge about the surrounding environment. Practical applications are numerous, for example relocalising within pre-built maps for augmented reality [Glocker et al., 2013] and loop closure for SLAM systems [Cadena et al., 2016; Engel et al., 2014; Mur-Artal et al., 2015]. A difficulty within a domestic environment is that objects and furniture will not remain static. Localisation that is robust to changing environments, therefore, becomes key to building and maintaining a map of a dynamic indoor environment over a period of time. Notably, localisation in an evolving environment is an area where significant scope for improvement has been identified [Wald et al., 2020].

We propose to address the challenge of domestic localisation by localising with respect to a known floor plan. This form is particularly suited to long-term localisation as, although objects and furniture may have moved, the structure represented in a floor plan, such as walls floors and ceilings, will remain static. Therefore localisation

can be performed over a long period of time without requiring continual re-training or re-mapping. Further, with only a floor plan needed as a prior, visual localisation can be performed without a previous visit of the environment.

Our work formulates localisation as an understanding problem: room layout is captured within images and it is used to reason location within a given floor plan of the environment. We explore this approach in two chapters which propose a generalised implicit representation of room layout that can be accurately inferred from images and the floor plan alike. The methods differ most significantly in how structure is understood from the floor plan prior.

In Chapter 4, we perform position estimation through multi-modal retrieval and refinement enabled by explicit rendering of layouts within a floor plan. Chapter 5 extends this further by demonstrating that structure within a 2D floor plan can be hallucinated implicitly for the cross-modal and cross-view position estimation. Therefore, it not only offers an order of magnitude faster inference, but crucially removes the need for some assumptions about the scene that are required in Chapter 4.

Together, these chapters offer advances in architectural understanding: from egocentric views in Chapters 3 and 4, to the unified understanding of structure across floor plans and images in Chapter 5.

1.1.2 Group Convolution for Low-cost Network Design

Methods based on neural networks have demonstrated compelling advantages in a number of different tasks over classical computer vision techniques. However, these approaches can often be too slow to meet the requirements for decision making, or too large to run on mobile hardware. We, therefore, turn our attention to low-cost network design, with a particular interest in group convolution as a design tool. We explore enabling near real-time instance segmentation, and present a method for group design that can be applied more generally to deep methods.

1.1.2.1 Real-time Instance Segmentation

Object detection is concerned with recognising the presence of each object of interest within an image, inferring its type, as well as predicting a tight box which bounds the visible pixels of each instance. By contrast, semantic segmentation aims to predict the semantic category for each pixel within an image. Instance segmentation bridges these two tasks by keeping the instance-level predictions for individual objects but predicting a more precise extent for each object in the form of a mask. This additional

information proves to be very useful for handling partially occluded objects, where a bounding box alone may cause ambiguity. As such, instance segmentation is a component of many downstream applications, such as robotic object grasping [Danielczuk et al., 2019; Xie et al., 2021], autonomous vehicles [De Brabandere et al., 2017], and medical imaging [Anantharaman et al., 2018].

In this thesis, we are specifically motivated by enabling instance segmentation in real-time (>30 frames per second). The influential Mask R-CNN [He et al., 2017] performs instance segmentation at around 5-7 FPS through the addition of mask prediction to an object detector [Ren et al., 2015]. For a period after its publication, follow up work generally focused on improving accuracy [Huang et al., 2019; Liu et al., 2018c]. However, little progress was made on latency. Those methods which did target real-time instance segmentation [Jetley et al., 2017; Uhrig et al., 2018] fell well below real-time object detection counterparts in terms of accuracy.

In Chapter 6, we present an instance segmentation method which is able to achieve real-time speeds. It keeps the overall two-stage structure of Mask R-CNN, but its fast and accurate performance is achieved by designing a novel backbone, which in itself is an extension of MobileNetV2 [Sandler et al., 2018]. Specifically, the depthwise layers in MobileNetV2 are replaced with larger group ranks, producing highly accurate detection and segmentation with minimal slow-down over the original network. In all, these modifications combine to create a highly effective method that offered a significantly improved speed-accuracy trade-off to other instance segmentation methods available before it. It is worth noting that YOLACT [Bolya et al., 2019b] also targeted real-time instance segmentation concurrently with Mobile R-CNN.

1.1.2.2 Enabling Exploration of Group Topologies in Convolutional Neural Networks

Grouped convolution has become an increasingly common tool in network architecture design. ResNeXt [Xie et al., 2017] used it to produce networks of higher accuracy while maintaining inference cost. Many networks designed for low-cost inference utilise depthwise convolutions—a special case of grouped convolution, where the number of groups is equal to the number of channels—to greatly reduce the number of operations and parameters, while still remaining reasonably accurate [Chollet, 2017; Howard et al., 2017; Sandler et al., 2018]. In Chapter 6, we show how, through careful selection of rank, group convolution can enable very fast and highly accurate instance segmentation. However, we were only able to explore a small number of possible group configurations. Similarly in wider literature, the implications of the rank

selection for individual grouped convolutional layers, and how these interact across a whole network, remain quite unknown.

In Chapter 7, we aim to address this through a weight sharing-based search [Bender et al., 2018; Li and Talwalkar, 2020] for group convolution topologies. To make weight sharing possible in the realm of group convolutions we present a novel formulation where convolutions of increasing rank are represented as a sum series. We show that group selection at a particular layer can be expressed as a design decision block, with weights shared between ranks. This parameter sharing scheme means the entire search space of group topologies across the network can be trained *simultaneously*. With this, the expense of group topology search is shifted away from training, and towards cheaper test-time sampling. In the context of low-rank network factorisation, this allows the exploration and validation of more possible configurations while removing the need for heuristics for the group design hyperparameter selection.

1.2 Thesis Structure

This thesis is an integrated thesis[3]. It is presented as a hybrid between written and typeset paper chapters. As such, the bulk of some chapters (Chapters 3, 4, 5, and 7) are presented in the format in which they were published or submitted for publication. However, in each of these chapters we have also included an additional introduction and conclusion, and pull out contributions from the paper. These sections are intended to aid the reading of the thesis as a cohesive whole. They offer additional context, motivation and discussion, usually specifically in relation to the other work in this thesis. In addition, where a figure caption is marked with the 🔍 symbol, we provide larger size version of the figure in Appendix A.

Chapter 2 explores the context for all the tasks considered in this thesis. In the previous sections, we have drawn attention to the two-sided motivation that underlie the work in this thesis: the advancement of architectural understanding with deep learning based vision, and the design of fast and accurate convolutional networks. The chapters of this thesis each fall neatly into one of these themes. Therefore, we divide the research chapters into two parts, each with a coherent through-line for the progression of work.

Part I is concerned with the application of deep visual methods to understand and reason about the environment. This covers two main tasks. In Chapter 3, we explore

[3] https://www.mpls.ox.ac.uk/graduate-school/information-for-postgraduate-research-students/submitting-your-thesis

unconstrained room layout estimation. In Chapters 4 and 5 we approach the task of localisation through layout reasoning alone.

Part II explores the design of networks for low-cost, but accurate, inference using group convolutions. In Chapter 6, we target real-time instance segmentation, presented as a written chapter. Chapter 7 then extends the insights and presents a tool for group convolution design which can be applied more generally.

Finally, Chapter 8, summarises and discusses the contributions and accomplishments of the work presented in this thesis.

1.3 Publications

The following publications are discussed in this thesis:

- **Howard-Jenkins, H.**, Li, S., and Prisacariu, V. (2018). Thinking Outside the Box: Generation of Unconstrained 3D Room Layouts. In *Asian Conference on Computer Vision (ACCV)*
- **Howard-Jenkins, H.**, Li, Y., and Prisacariu, V. A. (2020). GroSS: Group-Size Series Decomposition for Grouped Architecture Search. In *European Conference on Computer Vision (ECCV)*
- **Howard-Jenkins, H.**, Ruiz-Sarmiento, J.-R., and Prisacariu, V. A. (2021). Lalaloc: Latent layout localisation in dynamic, unvisited environments. In *International Conference on Computer Vision (ICCV)*
- **Howard-Jenkins, H.** and Prisacariu, V. A. (2021). LaLaLoc++: Global Floor Plan Comprehension for Layout Localisation in Unvisited Environments. *Submitted for Publication.*

Chapter 2

Literature Review

This chapter aims to provide an overview of the work that is relevant to that presented in this thesis. We hope to convey the context in which the work was conducted by exploring other approaches that came before it. In doing so, we will discuss the core topics on which our work builds upon, as well as highlighting the key differences and novelties of our presented work.

Mirroring the structure of the thesis as a whole, we will present the wider context of our work in the context of the two overarching motivations for the work covered, namely the advancement of architectural understanding and the design of low cost networks that can be run on mobile hardware. In the first, we explore room layout estimation and its extension to implicit representations, before describing visual localisation methods more generally as the setting for our work on floor plan relocalisation.

In the second part, we explore low-cost neural network design. We begin by outlining the task of real time instance segmentation, which is a key application of our low-cost design. We then describe the *tools* on which our work in this part of the thesis is formed, specifically group convolutions as well as neural architecture search techniques.

2.1 Room Layout Estimation

The task of estimating room layout aims to produce a clutter-free representation of the architectural structure seen in an image. In the literature, existing methods impose strict constraints on the room geometry. Two of the most common constraints applied to room geometry are the box approximation [Hedau et al., 2009], models a room as the interior of a parametric 3D box, and Manhattan world [Coughlan and

Yuille, 1999], where all structure must be aligned to one of three mutually orthogonal axes: walls allowed in two axes, with floors and ceilings aligned to the third.

These constraints enable layout estimation methods to operate in a bottom-up approach, extracting low-level image features such as edges which, when combined with the set of rules lended by the constraint on room shape, are enough to reason overall layout. In this vein, Hedau et al. [2009] leverage the box approximation. First line segments are detected in the image, which are then used to estimate vanishing points. By sampling rays from the vanishing points, a set of candidates for room layout are generated, the most likely candidate is then predicted through a learned SVM model. The heavy constraints provided by the box approximation limit the number of candidates possible, as well as allow layout to be parameterised as a fixed set of five faces for the candidate selection phase. Follow-up work has improved upon this same general estimation paradigm with the inclusion of object reasoning [Gupta et al., 2010] and a more dense parameterisation [Schwing and Urtasun, 2012].

Lee et al. [2009], on the other hand, choose to constrain rooms to a Manhattan world, with a single floor and ceiling. Again, line-segments are detected in the image and used to estimate vanishing points, which in turn allow orientation of the three axes of the building to be inferred. The constraint on having only three axes, means that images can be divided into three regions that are separated by the two vertical vanishing lines. Each of these three regions can only contain four types of corner and thus there is a heavy restriction on layout hypothesis generation. Layout hypotheses are then evaluated by the support that they receive from the detected line segments. However, these bottom-up methods with heavy constraints are still highly dependent on their low-level feature extraction, which are very susceptible to noise.

More recently, neural networks have been used to recover spatial layout from perspective images. Mallya and Lazebnik [2015] train fully convolutional networks to replace the earlier hand-crafted features with informative edge maps. The box approximation, in particular, lends itself for easy extension to neural network approaches. The constraint allows for only eleven possible layout types in a 2D image, each defined by Zhang et al. [2015]. This has led to the problem being formulated as a segmentation task [Dasgupta et al., 2016], with classes each representing separate instances of the bounding surfaces: left wall, middle wall, right wall, floor and ceiling. Furthermore, the eleven representations have been leveraged to tackle layout estimation as an ordered keypoint detection and classification problem [Lee et al., 2017], where the room type and the type's respective labelled corners are inferred directly

from the image. These 2D representations have been used to produce 3D geometry by informing and ranking room hypotheses [Izadinia et al., 2017].

Due to the limited field-of-view, layout estimation from a single perspective image is limited to prediction of partial layouts. Panoramas, on the other hand, offer a means to perform layout recovery for the whole room. Zhang et al. [2014] leverage the box approximation to predict whole room layout by detecting vanishing points, using them to generate hypotheses, and ranking them according to image evidence in the form of surface normal estimation and object cues. Xu et al. [2017] leverage similar cues to predict a Manhattan world room shape, and Zou et al. [2018] enhance this same general paradigm by taking the whole panorama as input. Lukierski et al. [2017] form a box estimate of a room by fitting a box room model to a dense map, which is built from only a short tour of the scene, through differentiable raytracing. Whole room prediction has also been conducted with perspective camera images, leveraging multi-view constraints to recover a constrained layout [Cabral and Furukawa, 2014; Flint et al., 2011].

In Chapter 3, we remove the need to apply these strict constraints on room geometry in order to predict layout from a single view. Instead, we formulate single-view layout estimation as the piece-wise planar surface reconstruction. In doing so, we are able to infer the architectural geometry of a room with free-form 3D plane detection and regression, therefore without any constraint on overall geometry. Liu et al. [2018a] perform similar direct plane regression from a single RGB image. Although their method is designed to predict visible planes, rather than potentially occluded structural ones, they are able to predict layout through a heuristic that their network predicts architectural planes (walls, floors and ceilings) in a particular order of dominance. However, layouts cannot be estimated where a wall, floor or ceiling is completely occluded, unlike our own work. Later, and concurrent with our own work, [Liu et al., 2019] also leverage an R-CNN structure for plane detection and regression, but again the focus is on visible planes and they do not attempt room layout estimation.

2.1.1 Implicit Room Layouts

The work discussed so far in this section has focused on being able to produce an explicit geometry that is visible in single or multiple views. To an extent, it was this explicit representation which particularly lent itself to the introduction of constraints on layout as they made modelling relationships between layout features easier. However, some have explored whether a layout representation could instead be learned.

Kim et al. [2019] first proposed the learning of a metric space to represent room layout. The latent space was designed such that if the layouts in two images are similar then so should their projections onto a learned manifold. This was validated on the task of image retrieval, where the goal was to return other images with similar layouts. Zheng et al. [2020b] later showed that this metric space for room layout can aid in the prediction of room layouts. This was achieved by creating an end-to-end differentiable autoencoder for a keypoint parameterisation of room layout, as well as an image encoder which maps images to the encoded layout space.

The learning of a metric space for room layout is largely dependant on being able to compute the inherent similarity between two layouts. To achieve this, both of these approaches applied the box approximation to the images that they handled. This enables each to use a pixel-wise label similarity to compute layout similarity, but it leads to much reduced variability in the space of possible layouts: there are only eleven types of room layout which can be seen in an image, and Kim et al. [2019] only considered layouts belonging to a single of those types. In contrast, our work in Chapters 4 and 5 explore how this data-driven representation can be used to again remove the need for constraints on layout.

2.2 Visual Localisation

A wide variety of methods have been produced to tackle the task of visual localisation. Although a slightly unconventional grouping, for the purposes of this thesis it perhaps best to categorise these methods by how they leverage prior information. This categorisation leads to two main classes of relocalisation methods: there are scene-specific methods, which require fine-tuning to each individual scene; and there are scene-agnostic methods, which learn general priors that can be applied to new scenes without any adaptation. In this section, we will provide an introduction to the main types of method which form these two classes of localisation method. We then introduce our task of floor plan localisation, describing its key differences to the bulk of work on visual localisation.

2.2.1 Scene-specific Localisation

Absolute pose regression methods [Brahmbhatt et al., 2018; Kendall et al., 2017, 2015] train a deep network for each scene to directly predict the camera pose from an input. The reference database of images captured within a scene is used to learn

a mapping between the visual appearance and the camera pose, thus forming an implicit representation of the scene in the network's learnable parameters. A particular advantage of these methods is their efficiency, with inference requiring only a single forward-pass of the network. However, early iterations of absolute pose regression failed to generalise to viewpoints away from those in the reference database [Sattler et al., 2019]. More recent work [Chen et al., 2021; Ng et al., 2021] has been able to achieve accuracy that is competitive to other localisation philosophies.

Instead of regressing camera pose, scene-coordinate regression methods [Brachmann et al., 2017; Brachmann and Rother, 2018, 2019; Shotton et al., 2013; Valentin et al., 2015] densely predict 2D-3D correspondences between the query and the environment, which then allows for solving the Perspective-n-Point problem (PnP) for pose. Similar to absolute pose regression methods, the reference images are used to learn the location of visual content in the global coordinate frame and thus the environment is also captured implicitly in the model. Early methods leveraged regression forests to predict the scene coordinates [Shotton et al., 2013; Valentin et al., 2015]. More recently, the regression forests have been replaced by neural networks [Brachmann et al., 2017; Brachmann and Rother, 2018, 2019], with Brachmann et al. [2017] introducing a differntiable approximation of RANSAC [Fischler and Bolles, 1981] for end-to-end learning. This style of localisation is able to achieve high accuracy, and generalise well to new viewpoints. Since the environment must be learned implicitly, they do not scale well due to the limited network capacity [Brachmann and Rother, 2019], and like absolute pose regression, they require re-training or adaptation [Cavallari et al., 2017] to localise in new environments. In the following section, we will discuss methods which can be applied to new scenes without the need for any additional training or fine-tuning, which we will refer to as *scene-agnostic*.

2.2.2 Scene-agnostic Localisation

Early pieces of work for visual localisation leverage the retrieval of similar images to infer location [Schindler et al., 2007]. These methods represent the world as a database of images and their respective poses. To localise a query image, the most similar image from the database is retrieved through comparison of image descriptors, such as NetVLAD [Arandjelovic et al., 2016]. These methods have a number of advantages: they scale well to large environments and the image descriptor can be trained to be robust to changes in environmental conditions [Sattler et al., 2018], and a pretrained embedding network can be applied in new scenes without adaptation. However, their

accuracy is limited by the range and density of coverage by the reference images in the database.

Some retrieval-based localisation work [Balntas et al., 2018; Ding et al., 2019] has sought to address the limitation of retrieval alone by incorporating a relative pose regression stage after the initial retrieval. Balntas et al. [2018] train a neural network to predict the relative pose between query image and the image retrieved image from the reference database. However, while these initial relative pose regression methods had low accuracy [Sattler et al., 2019], similar to their absolute pose regression counterparts, more recent work by Ding et al. [2019] achieve accuracy comparable to structure-based methods that we will discuss in the following.

Structure-based methods [Sattler et al., 2017] regress the camera pose by establishing direct 2D-3D correspondences between the query image and a sparse 3D point-cloud of the scene. The point-cloud is built through structure-from-motion (SfM), with local descriptors such as SIFT [Lowe, 2004] used for both the model building and correspondence prediction at inference time. Once correspondences between the query and the point-cloud, the pose can be regressed through a RANSAC-based PnP solver. These structure-based methods achieve very accurate localisation, however, they are heavily dependant on the predicting stable keypoints to ensure accuracy and robustness of the correspondence prediction. This leads to degradation of localisation performance in some challenging scenarios, such as under changing conditions [Sattler et al., 2018], motion blur [Kendall et al., 2015] and in texture-less scenes [Shotton et al., 2013].

Hierarchical localisation methods [Irschara et al., 2009; Sarlin et al., 2019, 2018] aid local feature matching by leveraging the robustness of global image descriptors to the challenging scenarios just mentioned. These methods first perform an image retrieval as an coarse localisation step and therefore provide a strong prior for the prediction of 2D-3D correspondences such that only a plausible subset of the overall point-cloud needs to be considered. This combination of image retrieval and structure-based localisation leads to high localisation accuracy with efficient computation, exhibiting particularly strong performance in changing scenes [Sarlin et al., 2020].

Although these scene-agnostic methods do not need to be tuned for localisation in a new environment, they do have a commonality with the scene-specific methods in that they require a prior tour of the environment. We define a prior tour as the reference set of images which are leveraged by the scene-specific methods as a training set, and used by the scene-agnostic methods for map or database building. By contrast, our work on floor plan localisation in Chapters 4 and 5 is able to operate

without this tour of the target environment, instead it requires only a floor plan as a prior. This, therefore, means that there is a significant domain gap between the visual information we wish to localise and the reference we have of the world. Next, we explore existing approaches that are able to overcome this divide.

2.2.3 Cross-modal and Cross-view Localisation

The floor plan localisation problem we explore in Chapters 4 and 5 is closely related to two avenues of research, namely cross-modal and cross-view localisation techniques. These methods are attractive because they enable localisation with priors that are easier to acquire or ones with greater coverage than if they had to be captured like-for-like with the query data. Examples of such priors include street maps, satellite imagery and terrain height maps.

Cross-modal localisation, localises images with respect to priors of a different data modality, in particular localisation of ground images to maps or 3D textureless extrusions, such as those provided by OpenStreetMap (OSM), has garnered significant research interest. Hentschel and Wagner [2010] localise against the OSM map through the combination of GPS and building planes extracted from laser scans fed into a Monte Carlo Localisation (MCL) [Thrun, 2002] framework. Similarly, Vojir et al. [2020] learn to match a building-edge-based descriptor that can be extracted through instance segmentation on the image and from the map. Arth et al. [2015] propose to localise through the alignment of horizontal and vertical image lines to edges of the extruded 3D structural map. This general idea was later extended to align a semantic segmentation of the query image into facades and edges [Armagan et al., 2017], or unique building instance IDs [Budvytis et al., 2019a,b]. Our work in Chapter 4, is cross-modal in nature. However, unlike the other cross-modal methods discussed here, we do not look to extract common explicit structure between the query and the prior that can be compared. Instead, our method learns a shared data-driven representation between layout renderings and ground imagery.

Cross-view visual localisation, on-the-other-hand, aims to localise images where reference and query are captured from starkly different view-points. Perhaps the largest area of active research for the cross-view task is to localise ground images against overhead imagery. Early approaches performed this localisation through matching of handcrafted classical features [Leung et al., 2008; Noda et al., 2010; Pink, 2008], therefore requiring invariance across extreme view-point changes, or by conversion of one view type into the another, such as through backprojection of stereo ground images [Senlet and Elgammal, 2011]. More recently, learning-based methods

have approached the task of localisation by learning to match images between the viewpoints [Hu et al., 2018; Hu and Lee, 2020; Lin et al., 2015]. These approaches are similar in principal to our own in Chapter 5. However, in our work we must contend with the difference in data modality too and we show that a relatively simple network architecture is able to reason across both these domain gaps.

2.2.4 Floor Plan Localisation

Due to the ubiquity of floor plans in the built environment and their relative permanence, it is not surprising that floor plan localisation has caught the attention of researchers, with particular interest in the robotics community.

Methods that aim to achieve this infer layout features from the query imagery, and in turn estimate the likelihood of this layout being observed given the structure presented in the plan. A wide source of variety in these methods comes from the manner in which layout is inferred and represented from the incoming visual data. It has been performed through the detection of layout corners [Hile and Borriello, 2008], edges [Boniardi et al., 2019; Unicomb et al., 2018; Wang et al., 2015], or aggregation of depth [Winterhalter et al., 2015].

Scan-matching techniques [Pomerleau et al., 2015] aim to find a rigid alignment between the scene and ranged query imagery through methods such as Iterative Closest Point (ICP) [Besl and McKay, 1992]. Through careful design of the scan-matching system, this methodology have also been leveraged to perform localisation within a floor plan [Blum et al., 2020; Watanabe et al., 2020].

However, these methods generally rely on information over multiple measurements through the use of an MCL [Thrun, 2002] framework in order to make predictions. This localisation scheme means that uncertainty or ambiguity in measurements can be mitigated over a sequence. Therefore, the expressiveness of the layout inference from the incoming imagery itself can be slightly overlooked, as evidenced by many of the methods above only considering non-occluded architectural structure. Instead, we approach the task from an instantaneous observation, with no motion cues or depth information given, and thus aim to maximise the information that can be inferred between the RGB query and the floor plan. In fact, we remove the need for any explicit prediction of layout geometry at query time entirely, instead solely leveraging latent representations of layout.

2.3 Real-time Instance Segmentation

Instance segmentation is the task of detecting individual occurrences of objects or semantic classes in an image, as well as drawing their outline. Mask R-CNN [He et al., 2017] has become the representative two-stage instance segmentation method. Its first stage operates by generating regions-of-interest (ROIs), which are areas of an image likely to contain an object. The second stage, then classifies, refines, and segments those ROIs to produce the final segmentation. Follow up work on this two-stage paradigm generally focuses on improving the accuracy of these two-stage methods, such as by the inclusion of a mask prediction confidence [Huang et al., 2019] or by improving the feature extraction stage [Liu et al., 2018c]. But, work on reducing the latency of these networks was sparse.

Real-time object detection [Liu et al., 2016; Redmon et al., 2016; Redmon and Farhadi, 2017, 2018] is an area of significant research effort. Methods aimed at performing detection in real-time generally combined classification into the region proposal stage, thus forming detection method with just a single stage. This reduced time spent with the feature pooling before the second stage in an R-CNN.

However, real-time instance segmentation methods remained relatively rare. In the same manner that Mask R-CNN performs accurate two-stage instance segmentation through the addition of a mask head to Faster R-CNN [Ren et al., 2015], real-time instance segmentation has been attempted through the addition of mask prediction to an already real-time one-stage object detector. Key to these methods is to reformulate the mask prediction to be lightweight. Box2Pix [Uhrig et al., 2018] and Straight to Shapes [Jetley et al., 2017] both follow this design principle and demonstrated ability to perform in real time. Box2Pix [Uhrig et al., 2018] predicting masks as a pixel-wise offset from the respective object centre, and Straight to Shapes [Jetley et al., 2017] learning a general shape latent code that can be predicted directly at each anchor. However, their respective accuracies are well short of representative objection detection methods.

A concurrent work with that in Chapter 6 is YOLACT [Bolya et al., 2019b], which, to the best of our knowledge, became the first peer-reviewed method for real-time instance segmentation. YOLACT also operates through the addition of a lightweight mask head to a single-stage detector, but their chosen mask prediction proves to be able to capture a high degree of accuracy while still remaining very fast. Specifically, their mask head predicts a set of whole-image mask prototypes. Along with each detection, a set of attentional coefficients are predicted which predict the specific

object's representation in each prototype. Mask prediction is then conducted through the linear combination of the prototypes weighted by the predicted coefficients, which is shown to be very fast. With this paradigm, YOLACT reduces time spent with the pooling operation between stages and therefore can afford to use heavy backbones such as a ResNet101 FPN [He et al., 2016; Lin et al., 2017] to achieve high accuracy while still maintaining real-time speeds.

Our work on Mobile R-CNN in Chapter 6, however, takes a different approach. Rather than a significant reformulation of instance segmentation so that one-stage networks can predict masks more accurately, we explore how the existing two-stage R-CNN design can be made faster. Specifically, this is achieved by replacing the backbone network with one designed to be lightweight while maintaining high accuracy which is achieved through group convolution.

2.4 Group Convolution

The introduction of grouped convolution to deep learning was first made for a practical reason. Krizhevsky et al. [2012] used two groups in AlexNet so that they could train the network in parallel across two GPUs. Since then, group convolution has had a wide impact across the speed/accuracy trade off curves of CNNs, but it has largely been their efficiency that has motivated their use [Ioannou et al., 2017]. In ResNeXt [Xie et al., 2017], the complexity saving means that deeper layers could be used in the network's residual blocks [He et al., 2016] without increasing the overall inference cost.

One particular form of group convolution has arguably made the most impact on architecture design: depthwise convolution. Depthwise convolutions are the extreme case of grouped convolution, where each group consists of only a single channel. This style of convolution has gained particular traction in the design of low-cost networks, largely in tandem with pointwise convolution (a convolution with a 1x1 kernel) to form a depthwise separable convolution [Sifre and Mallat, 2014]. Depthwise separable convolutions have become the building blocks for a number of notable network architectures, including Xception [Chollet, 2017], MobileNets [Howard et al., 2017; Sandler et al., 2018], which are all generally designed for extremely lightweight inference.

However, these low-cost designs almost exclusively only considered the most extreme case of groups, *i.e.* depthwise convolution. There is an exception to this in ShuffleNet [Zhang et al., 2018], where they additionally made the pointwise layers

grouped convolutions for additional computational savings. By contrast, in Mobile R-CNN (Chapter 6) and GroSS (Chapter 7), we more thoroughly explore group topologies beyond this extreme.

2.5 Low-rank Factorisation

In contrast to the specific design of low-cost network architectures from the outset, low-rank factorisation aims to reduce the cost of large networks. These methods exploit the redundancy present in direct implementations of convolution to achieve lower cost inference, ideally with a minimal decrease from original accuracy. Rigamonti et al. [2013] take advantage of redundancy in the spatial dimensions by expressing a filter as a linear combination of a smaller basis set of rank-1 filters. Taking this one step further, Jaderberg et al. [2014] targeted cross-channel redundancy by performing low-rank factorisation in the channel dimension in addition to the spatial ones achieving even greater speed-ups.

Network factorisation methods began to leverage standard tools from tensor linear algebra to formalise the decomposition procedure. Lebedev et al. [2014] applied CP-decomposition [De Lathauwer et al., 2000] on the convolution kernel to perform low-rank factorisation across the full tensor, resulting in the layer being approximated as a series of four rank-1 filters. More recently, Block Term Decomposition (BTD) [De Lathauwer, 2008] has been applied to the task of network factorisation [Chen et al., 2018]. Chen et al. [2018] showed that the BTD factorisation of a convolutional weight was equivalent to a grouped convolution within a bottleneck architecture, which is why the decomposition procedure bears particular significance for our work in Chapters 6 and 7.

Although Chen et al. [2018] and Wang et al. [2018a] applied this group convolution equivalency for network acceleration previously, since the decomposition is costly, these methods have relied on heuristics and intuition to set hyperparameters such as the rank of successive layers within the decomposition. With GroSS, we present a method which allows for exploration of the decomposition hyperparameters and all the combinations in the form of architecture search.

2.6 Neural Architecture Search

While deep learning enables the bypassing of manual feature crafting through its data-driven representation learning, the networks themselves are typically hand designed.

Neural architecture search (NAS) [Zoph and Le, 2016] aims to offer an alternative to this manual design by instead automatically identifying which architectures are most suitable for the design task.

Generally the focus of architecture search work is the manner in which search itself is conducted. NAS methods have largely favoured reinforcement learning [Elthakeb et al., 2018; Tan et al., 2019; Zoph and Le, 2016; Zoph et al., 2018]. In broad terms, these methods all set a baseline structure, which is manipulated by a separate controller. The controller optimises the structure through and objective based on network performance. Gradient-based architecture search [Liu et al., 2018b; Wu et al., 2019] and Bayesian optimisation [Jin et al., 2019; Kandasamy et al., 2018] offer alternatives to the controller-based methods. Gradient-based search methods make the network architecture manipulations themselves differentiable so that the architecture can be optimised through the application of gradients. Bayesian optimisation methods define a distance function between networks and evaluate new candidate architectures by finding a balance between exploration—choosing largely different architectures—and exploitation—choosing architectures close to the best that have been evaluated. This is made possible by assuming that the architecture accuracy follows a Gaussian Process and the next candidate architecture should give the maximum expected improvement. However, these search methods bring with them significant complexity, with limited experimentation performed to evaluate which components are necessary to achieve good empirical performance [Li and Talwalkar, 2020].

To address this, a simple random search has been proposed [Bender et al., 2018; Brock et al., 2017]. In these schemes, the cost of training child structures is amortised and search is thus reduced to performing evaluation of a random selection of candidate architectures. SMASH [Brock et al., 2017] train a HyperNetwork [Ha et al., 2016], to predict model weights for variations of model architecture. By contrast, weight sharing methods [Bender et al., 2018; Li and Talwalkar, 2020] train a singular set of weights for all possible architectures. These weight-sharing methods combined with random search not only form a conceptually simple package, but also they achieve highly competitive results [Li and Talwalkar, 2020]. GroSS, in Chapter 7, follows a similar strategy, by formulating BTD factorisation of convolutional layers such that weights can be shared across multiple ranks of decomposition, its search then takes the form of simple greedy algorithms to find the optimal configuration.

Part I

Architectural Understanding

Chapter 3

Explicit Capture of Unconstrained Room Layout from RGB Images

3.1 Introduction

In this chapter, we take our first step towards a more complete architectural understanding by considering the task of room layout estimation. Room layout estimation aims to produce a representation of the enclosing structure of an indoor scene, one consisting of the walls, floor and ceiling which bound the environment, while removing the clutter. For the target of floor plan localisation, explored in Chapters 4 and 5, performing layout estimation accurately is an important stage as it bridges the gap between images and floor plans by predicting the floor plan structure that is visible within an image.

We apply a detection paradigm to the layout estimation. We introduce Planar R-CNN, an R-CNN style network able to detect instances of the architectural planes required to estimate room layout components (floor, wall and ceiling) and regressed a 3D plane equation for each detection present in a single RGB image. In doing so, its prediction of room layout is therefore unconstrained. To the best of our knowledge, this was the first work to use an R-CNN style network to detect and regress 3D planes, later followed by Plane R-CNN [Liu et al., 2019]. We demonstrate two types of room layout estimation to evaluate the effectiveness of our Planar R-CNN: 2D layouts for individual images, and 3D room layouts composed through the aggregation of detections in a video sequence.

We believe that the work covered in this chapter represents a significant step forward for layout estimation with fewer constraints, and therefore more applicable to real-world tasks, such as our targeted floor plan localisation.

3.1.1 Contributions

- A room-plane instance detector giving the type and extent of the enclosing walls, floor and ceiling, as well as direct plane regression for each detection.
- The introduction of unconstrained room layout estimation from single perspective RGB images.
- Combining the instantaneous measurements to obtain a 3D layout from an RGB image sequence which is not limited to the boxy or Manhattan World constraints.

Thinking Outside the Box: Generation of Unconstrained 3D Room Layouts

Henry Howard-Jenkins, Shuda Li, and Victor Prisacariu

Active Vision Lab, University of Oxford, UK
{henryhj,shuda,victor}@robots.ox.ac.uk

Abstract. We propose a method for room layout estimation that does not rely on the typical box approximation or Manhattan world assumption. Instead, we reformulate the geometry inference problem as an instance detection task, which we solve by directly regressing 3D planes using an R-CNN. We then use a variant of probabilistic clustering to combine the 3D planes regressed at each frame in a video sequence, with their respective camera poses, into a single global 3D room layout estimate. Finally, we showcase results which make no assumptions about perpendicular alignment, so can deal effectively with walls in any alignment.

Keywords: room layout estimation · scene understanding · 3D vision

1 Introduction

3D room layout estimation aims to produce a representation of the enclosing structure of an indoor scene, one consisting of the walls, floor and ceiling which bound the environment, while removing the clutter. These supporting surfaces provide essential information for a variety of computer vision tasks, not limited to augmented reality, indoor navigation and scene reconstruction.

Almost all prior work to produce 2D and 3D room layout estimations introduce heavy constraints on scene geometry, such as the box approximation [15, 34] or the Manhattan World assumption [9, 32]. Where the box approximation is used, rooms are constrained to be rectangular in shape. Manhattan World geometry, on the other hand, slightly eases the constraints of a boxy world; any number of walls are instead allowed to lie along either of two perpendicular directions. Although these assumptions can still lead to highly representative layouts for a great number of indoor environments, it is trivial to produce examples of room geometries which cannot be accurately captured due to the constraints.

The aim of this paper is to relax the constraints on room shape further. In doing so, we are able to represent a wider variety of scene layouts. To accomplish this, we introduce a neural network which is able to detect the presence and the extent, as well as directly regress a representative 3D plane, of room bounding surfaces in single monocular images. By taking the per-image detections accumulated over the frames in a video sequence, along with their respective camera poses, we are able to produce a single global map of plane measurements. From

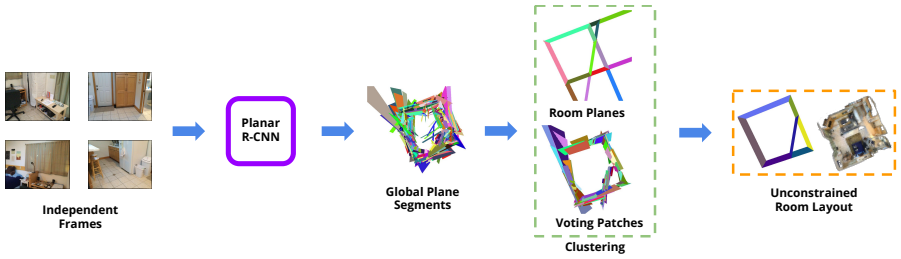


Fig. 1. An overview of our method: instantaneous room plane detections are provided by our Planar R-CNN and then combined with their respective camera pose to form a collection of global measurements. Probabilistic clustering is used to infer the number of room planes. These planes are intersected and each segment is considered a candidate wall. Candidates are then accepted into the global room layout based on the support of the measurements, thus capturing the room geometry.

these measurements, we infer the overall geometry of the room without placing constraints on the shape. The scheme is outlined in Figure 1.

The main contributions of this paper are: (i) a bounding surface instance detector giving the type and extent of the enclosing walls, floor and ceiling, (ii) direct plane regression for each room plane instance in an RGB image, and (iii) combining the instantaneous measurements to obtain a 3D layout from an RGB image sequence which is not limited to the boxy or Manhattan World constraints.

2 Related Work

Many researchers have sought to produce clutter-free representations of the room layout from monocular images. Early efforts largely had taken the bottom-up approach of extracting image features including colour, texture and edge information to inform vanishing point detection. In these pieces of work, a post-processing stage is used to generate, and in turn rank, a large number of room layout hypotheses with either conditional random fields (CRFs) or structured SVMs [11, 14]. It is possible to recover 3D reconstruction from this approach, but in practice the final layout proves to be heavily dependant on the low level features extracted from the image, which are highly susceptible to noise and clutter. These low level features have also been used to infer 3D spatial layout through depth-ordered planes, by first grouping into lines, quadrilaterals and finally planes.

Mura *et al.* [22] recover accurate floor plans from planes detected in point clouds produced by cluttered 3D input range scans. Even when occluded, planes can be extracted and used to form a set of wall candidates which are projected into 2D. Rooms present in the floor plan are found by modelling a diffusion process over the spatial partitions caused by candidate walls. However, this method can only produce 2.5D extruded floor plans and requires a point cloud.

More recently, neural networks have been used to recover spatial layout from RGB images. In [20] fully convolutional networks are trained to replace the earlier hand-crafted features with informative edge maps. The eleven possible representations of the box model room in 2D, as defined in [35], have lead to the problem being formulated as a segmentation problem with classes each representing separate instances of the bounding surfaces: left wall, middle wall, right wall, floor and ceiling as in [7]. Furthermore, the eleven representations have been leveraged to tackle layout estimation as an ordered keypoint detection and classification problem as in [18], where the room type and the type's respective labelled corners are inferred directly from the image. These 2D representations have been used to produce 3D geometry by informing and ranking room hypotheses [15].

Panorama images have been leveraged for full scene recovery with box models in [34], and for a Manhattan World approximation in [32]. For whole room layout from more traditional camera images, incremental approaches have been used, such as that in [2]. Flint *et al.* used a combination of monocular features, with multiple-view and 3D features to infer a Manhattan World representation of the environment [9]. In [30], a RGB-D panorama is split and half of the scene is taken as input with the aim of producing reasonable, cluttered room layout estimation for the unseen portion of the panorama.

Alternatively, maps of the environment can be produced iteratively using a top down approach, such as the mapping and planning network architecture detailed in [12], where an egocentric map forms a top down 2D representation of the environment. Once again the representation of the scenes is constrained to 2D and the mapping component aims to recover the navigable space in a scene, rather than the clutter free representation which we seek.

The closest work to this paper is the direct plane regression from a single RGB image performed by Liu *et al.* in [19]. They are able to produce clutter-free room representations from single images. However, their method relies on visible planar regions and therefore room-layouts cannot be produced where a wall, floor or ceiling is completely occluded.

In this paper, rather than focusing on producing a self-complete 2D layout segmentation, we aim to recover the underlying 3D room planes in each image. Further, wall, floor and ceiling would generally be considered uncountable *stuff*, rather than countable *things* for which object detectors are most commonly used. However, there is work showing that *stuff* classes provide useful context for *thing* detection [1, 21, 24, 27]. We take this further by treating these architectural planes as *things*, which we detect using an instance detector. These planes are then used to compile a singular representation of the scene which does not rely on any assumption about the geometric structure of the world other than that rooms are planar.

3 Planar R-CNN

To address the challenges presented by a non-boxy room layout, we approach the task of finding a room's supporting surfaces as an instance detection problem. In

doing so, our network detects an arbitrary number of surfaces and outputs their extents in the form of a bounding-box. Furthermore, by removing the Manhattan World assumption, the walls, floor and ceiling can have any orientation and are no longer just aligned with one of only two dominant directions or the ground plane. Therefore, our network also outputs arbitrary alignment for each of the detection by directly regressing the 3D plane equation of each surface.

We choose to approach the task by directly targeting abstract room-planes rather than using a method which relies on visible patches of wall, floor and ceiling. The advantage gained is two-fold: (i) we are able to infer room planes that are completely or mostly occluded by furniture; (ii) we are able to disregard large planar surfaces that are irrelevant to room structure. On the first point, furniture can often block the wall behind, thus reducing the ability to capture the plane from visible pixels. Considering the second point, the occluding object itself may be the dominant plane within a region, such as the surface of a table.

In all, we are able to predict an arbitrary number of room-bounding planes, as well as their directions and extents, providing all the information required to construct an unconstrained layout.

3.1 Network Architecture

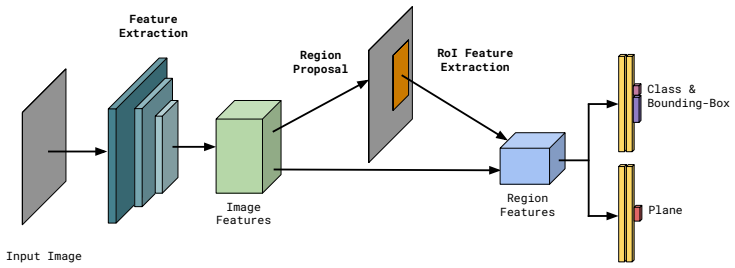


Fig. 2. An overview of the complete network from an RGB input image to the outputted classes, bounding-boxes and plane equations. The single dimensional columns in the network heads represent fully connected layers. Yellow columns have an output size of 4096, the plane has output size 4, and the class and bounding-box columns have output size equal to and four times the number of classes, respectively

We design a network architecture capable of detecting an arbitrary number of walls, floors and ceilings, as well as their extents, positions and alignments. Inspired by Mask R-CNN [13], where Faster R-CNN [25] is augmented by an additional network head which outputs a mask for each detection, we take a similar approach by adding an additional head to regress a 3D plane equation for each region of interest (RoI). From an input of a single RGB image, the network outputs a number of detections. Each detection has an associated class (wall or floor/ceiling), bounding-box and plane equation.

For clarity, we use the same nomenclature as in [13] when referring to: (i) the *backbone* which is a convolutional architecture to extract features over the entire input image, and (ii) a network *head* as an architecture applied separately to each set of RoI features, for example the Fast R-CNN head which predicts class and regresses a bounding-box.

Instance Detection. For detection, assigning of classes and determining bounding-boxes for instances of visible room planes, we use an implementation of the Faster R-CNN detector. Faster R-CNN is composed of two stages: the Region Proposal Network (RPN) and a second stage which is essentially Fast R-CNN [10].

Direct Plane Regression. In a similar fashion to Mask R-CNN, we introduce an additional network head to the second stage of Faster R-CNN. This network head directly regresses the 3D plane equation of the candidate in parallel to the classification and bounding-box regression. The plane regression head takes the same RoI features as the classification and bounding-box regression head. It outputs a 4-vector which represents the plane coefficients where a 3D point, $\mathbf{x}$, lies on plane $\mathbf{p} = [\mathbf{n}\ d]^T$ if $\mathbf{n} \cdot \mathbf{x} + d = 0$. The first 3 components of the 4-vector, which represent the normal of the plane, are normalised to unit length. The 4th plane coefficient, d, takes on a more direct physical meaning as the shortest distance between the plane and the camera origin.

The network head itself is comprised of two fully connected layers with 4096 output channels each, in addition to another fully connected layer to take the dimensions down from 4096 to the required 4 representing the plane equation coefficients, as detailed in Figure 2. The plane head is trained using two losses, L_{norm} and L_d, which represent the loss contributed by the normal of the plane and the depth coefficient, respectively. L_{norm} is defined as the negative of the cosine similarity between predicted and ground truth normals, and L_d is the mean-squared-error of the depth coefficients. Both are computed as follows:

$$L_{norm} = -\frac{1}{m}\sum_{m} \mathbf{n}_p \cdot \mathbf{n}_{gt} \quad and \quad L_d = \frac{1}{m}\sum_{m}(d_p - d_{gt})^2 \tag{1}$$

where m is the number of predicted planes, $\mathbf{p}_p = [\mathbf{n}_p\ d_p]^T$, and their corresponding ground truth planes, $\mathbf{p}_{gt} = [\mathbf{n}_{gt}\ d_{gt}]^T$.

3.2 Implementation

We use an implementation of Faster R-CNN with a VGG-16 [28] as its backbone. The final pooling and all the fully connected layers from the VGG-16 network are removed. In addition to RoIPool from the default Faster R-CNN implementation, we also implement RoIAlign [13]. While both methods achieve the same general goal of gathering the features relevant to a proposed region, RoIAlign better preserves the spatial information of the region's features, which may lead to improved performance for the spatially sensitive task of plane regression.

The loss for each sampled RoI is defined as $L = L_{cls} + L_{box} + L_{norm} + kL_d$ where L_{norm} and L_d are as above. The classification loss, L_{cls}, and the bounding-box loss are as in [10]. We found it was necessary to introduce a weight constant, k, for L_d to keep the network stable while training. Setting $k = 0.05$ provided good stability, while ensuring that the depth coefficient still trains.

4 3D Room Layout Generation

The network described in the previous section outputs a number of plane segments representing the room geometry. However, since we do not assume that all rooms are convex, the geometric information predicted from a single image is not complete in itself. For example, there may be entire walls occluded by others. Therefore, we treat each set of plane segment predictions as local measurements from a global room layout.

We define the global room layout as a set of 3D plane segments, each produced by the intersection of underlying room planes. The first task in determining the room layout is to obtain these room planes. We transform the regressed planes and corresponding bounding box measurements from each frame into the global coordinate system. This represents an amalgamation of noisy plane segment measurements each belonging to one of an unknown number of bounding planes.

We adopt a Bayesian approach for a Gaussian mixture model, and infer the posterior distributions of the room's characteristic planes through EM training. The number of room planes is determined by placing a threshold on the weights in the mixture model.

4.1 Spatial Voting

Once we have obtained the room planes, we intersect them all to form a collection of candidate plane segments to add to the room layout. In the convex case, these intersections would provide an immediate layout, however, since we allow non-convexity we must evaluate whether a candidate segment should be added to the room set. To do this, we make use of the regressed bounding-boxes as extents for the predicted planes. Since each measured plane is spatially constrained, it is used to vote for a candidate segment's existence in the room geometry.

We assign every evidence segment to a mixture component distribution. For each of these distributions, we then define a subset of n voter segments by ranking the likelihood of the segment having come from the cluster distribution. This ensures that the measurements lie on the most similar planes to the candidate segments for which they will vote. Support for a wall segment is determined through a robust inlier weighting scheme, such as in [4]. A voting plane patch is considered an inlier for a candidate wall by thresholding the proportion of the voter that overlaps the candidate, i_{vc}, and vice versa, i_{cv}.

Every candidate is assigned an energy, $\eta = 0$, which is updated on each of the voting segments belonging to the same plane cluster. If a voting patch is considered an inlier, the candidate's energy is increased by $(1 - i_{vc})$, and

the number of inliers is incremented. Once all voting patches have been tested, the ratio of inliers to the total voting on the candidate, r_c is computed and the candidate's energy is divided by r_c^a, where the exponent, a, is a tuneable parameter. To prevent cases where candidate patches are accepted with only a small total number of voter patches, we threshold total energy. The final result of the voting stage is to reject plane segments without sufficient evidence. In all, leaving only the walls that belong to the global room layout.

5 Experimental Evaluation

We train and evaluate our network's performance, both quantitative and qualitatively, capturing room geometry from an image in the SUN RGB-D dataset. We compare the regressed planes to their corresponding ground truth instances. In addition we plot the captured planes and their extents with an overlaid point cloud to demonstrate the network's ability to recover bounding planes.

Further, we evaluate our room layout estimation on single frames by testing on the NYUDv2 303 dataset [33]. We demonstrate the ability of our method to capture useful room layout information by using the output of our network to produce a prediction at the 2D room segmentation for an indoor image.

Finally, we prove the informative nature by amalgamating the independent frame-by-frame predictions on ScanNet image sequences into the global coordinate system and using these measurements to infer a whole room layout. This demonstrates the robustness of the enclosing surface detection and plane regression, since the network was never trained on any ScanNet sequence.

5.1 Training using SUN RGB-D

To train our network, we use the SUN RGB-D dataset [29]. The dataset provides room corner coordinates on the $x - z$ plane, as well as the upper and lower y height of the room, all defined in the world coordinates. This information allows us to build a 2.5D extruded floor plan for the room for each pair of RGB and Depth images in the dataset. We transform the room model from the world coordinate system to one aligned with the camera. From this aligned floor plan, we compute the plane equation of each section of wall, floor and ceiling. The floor plan is then projected into the camera, providing us instance-level segmentation, as well as corresponding plane equations, for every image.

Since we do not test on the benchmarks provided as part of the SUN RGB-D dataset, we use our own image split comprising of 9335 training, 500 validation and 500 testing images. It was found that some of the images in the dataset do not appear to have corresponding room layout information. In addition, for some images, it was found that the picture was taken outside of the annotated room. This tended to occur when pictures were captured in a department store and the room corners were defined as the limits of the display cubicle. We removed these images from the dataset for training, validation and testing. This resulting in the final split containing 7677 training, 441 validation and 439 test images. It is

worth noting that the depth images are not used in either training or prediction: instead they are only used to help visualise results.

Bounding surface instances are divided into two classes, one representing walls and the other representing floors or ceilings. This is a result of trying to keep the plane segments as generic as possible, but still maintaining the differences in their representation in the dataset. Because the rooms in the dataset are 2.5D rather than complete 3D floorplans, walls always have normals in the world $x-z$ plane, whereas floors and ceilings must have normals in the world y direction.

Training The models were trained on the dataset for 25 epochs in total. To initialise the feature extractor and classifier in the class and location head, we use a VGG-16 model pretrained on ImageNet [16]. Other fully connected layers in both network heads are initialised from a zero mean normal distribution with variance of 0.1, 0.01, 0.001 for the final plane regressor layer, the final classifier layer and the two hidden layers in the plane head, and the bouding-box regressor respectively. We use a batch size of 1 with stochastic gradient descent, 0.5 dropout rate, 0.0005 weight decay, 0.9 momentum. The learning rate is initially set to 0.001 and decreased to 0.0001 after 20 epochs. This training schedule takes around 12 hours on a single Nvidia GTX 1080Ti.

5.2 2D Room Layout Estimation with NYUDv2 303

We explore the generation of 2D layouts using our network, and thus demonstrate its ability to capture room information. As discussed in Section 4, our method does not require the convex box approximation. Since we do not impose any such constraints, there is no set number of possible room layouts, or wall types. We do not seek to classify left or right wall, instead we just detect the presence of a wall, floor or ceiling.

To produce predictions, we first back-project each pixel within a detection's bounding box onto its predicted plane, forming a plane patch in 3D. We then form a clutter-free 2D room segmentation by project all of an image's plane patches back into the image. Our reprojection into 2D provides insight into the usefulness of the plane detections for capturing room layout without having to place constraints on the layout, as well as providing comparison against other state-of-the-art solutions to the layout estimation problem.

Since this dataset takes a subsample of the NYUDv2 dataset [23], which is included in the SUN RGB-D dataset [29], we retrained our network ensuring that none of the 101 test images appeared in our training or validation dataset.

5.3 Whole Room Layout Estimation using ScanNet

In order to demonstrate not only the informative nature of the measurements produced by our network, but also its versatility, we produce 3D room layouts on the ScanNet dataset [5] having never trained on it and without the network using intrinsic data for each sequence. For this task we assume that camera poses are known, as in this case they are implemented using BundleFusion [6].



Fig. 3. The first and second rows show the image overlaid with the ground truth and network detections, respectively. The last row shows the predicted bounding-boxes projected onto their matching regressed planes to produce the informative plane segments. For additional 3D context, a point cloud from a depth image matching the input RGB image is displayed with the plane segments. 🔍

Implementation We start by collecting all the plane detections in a ScanNet image sequence. Since bounding-boxes do not provide tight constraint on the extent of a plane, especially if the plane is nearly perpendicular to the image plane. Therefore, when collecting instances from images, we discard those surfaces which are less than 30° from perpendicular to the image plane.

We found that our spatial voting method for determining which candidate wall planes was sensitive to changes in set parameters. We believe that this is due, at least in part, to the lack of any training on the ScanNet dataset. Planar R-CNN does not explicitly infer the camera's intrinsics. Therefore, for spatial information recovery at full accuracy, the network should be fine-tuned on new datasets to allow the relation between image coordinates and features with the surrounding 3D environment to be re-assessed. To account for this, we adjusted parameters on a per-scene basis when constructing room layouts. Generally, the parameters were set in the vicinity of: the minimum weight for an accepted plane posterior as 0.05, the 100 most likely measurements from each wall's posterior distribution allowed to vote for existence of the plane's candidate segments, the energy update exponent set to 2, and finally the mutual overlap threshold as more than 70% of the voter's area and more than 20% of the candidate's area for the voter to be labelled an inlier for the candidate.

Almost all of the ScanNet sequences have only brief appearances of the ceiling. To provide an upper bound on the wall segments, we instead tested the normal of the dominant posterior from planes labelled as the floor or ceiling class. If the direction of the normal was in the positive z direction, we labelled the plane floor and created a ceiling plane 2 m above and with a normal in

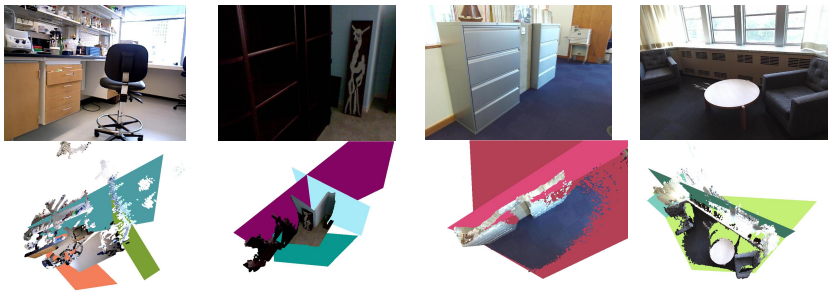


Fig. 4. Examples demonstrating the discussed advantages: The first column shows a plane being accurately regressed in a cluttered environment; The second and third columns show examples walls are regressed behind shelves and cabinets; In the forth column a wall is correctly estimated behind a chair and a non-planar curtain. 🔍

the other direction. If a downward facing dominant plane was detected, it was labelled ceiling and a floor plane was created 2 m below it.

5.4 Results

Since, to our own knowledge, this network is the first to directly regress abstract room-planes, we consider its main tasks when evaluating its performance. We define these tasks as accurate detection and plane regression. The detection task is evaluated using the average precision for each of the two classes, and the mean average precision. For plane regression, the network must be able to discern the alignment of the plane, *i.e.* the normal of the surface, and the depth coefficient in order to fix the plane in 3D space. We propose a two stage evaluation of the plane regression; first, the quality of the normals, secondly, the spatial accuracy of the plane.

Instantaneous Detections on SUN RGB-D We assess the quality of the plane normal predictions from the network in the same manner as surface normal estimation from monocular images, as in [3, 8, 31]. However, we compare the normal of a detected instance to its corresponding ground truth plane, instead of at each pixel. The angular disparity each pair, α, is computed. We then compute the mean, median and root-mean-square of α, as well as listing the percentage where the error falls below the thresholds 11.25°, 22.5° and 30°. These results are shown in Table 1.

To determine the accuracy of the regressed plane in terms of spatial location, we back-project all the pixels within the predicted bounding-box and on to the predicted plane. For each of the back-projected points, we compute a point-to-plane distance between the point and the corresponding ground truth plane. The mean of these distances are averaged for each instance detection in the image, providing a single distance score, δ, for every predicted bounding plane.

Table 1. Evaluation of the normal regression on our test set from the SUN RGB-D dataset.

Normals	Error			Accuracy ($\alpha < \theta$)		
	mean	median	rms	11.25°	22.5°	30°
FCRN + Room	33.5	20.0	44.8	36.4	52.7	59.7
FCRN + Fine	28.6	17.7	38.5	38.7	56.7	64.7
PlaneNet	**8.35**	**6.31**	**13.4**	**82.3**	**96.9**	**97.8**
Ours (RoIAlign)	11.3	7.44	17.0	70.3	89.5	93.7
Ours (RoIPool)	9.74	6.67	15.3	76.4	92.9	94.9

Table 2. Evaluation of the location of the regressed plane compared to its corresponding ground truth plane on our test set from the SUN RGB-D dataset.

Plane	Error		Accuracy		
Location	mean	median	$\delta < 0.2m$	$\delta < 0.5m$	$\delta < 1m$
FCRN + Room	1.83	0.357	33.3	60.7	85.7
FCRN + Fine	1.01	0.242	43.7	74.0	89.2
PlaneNet	1.79	0.275	40.0	67.8	84.3
Ours (RoIAlign)	1.18	0.269	42.0	67.4	84.1
Ours (RoIPool)	**0.503**	**0.217**	**47.1**	**75.5**	**90.7**

In Table 2 we show the mean and median of these instance distances, as well as the percentage which fall within the thresholds 0.2, 0.5 and 1 metres.

We compare our normal and plane location results to a baseline visible pixel-based method. This method produces a pixel-wise depth estimation using the FCRN method of [17]. The depth map is then masked by a ground truth segmentation of the image and back-projected into 3D space. We use RANSAC to fit a plane to the 3D points corresponding to each ground truth room surface. *FCRN + Room* in Tables 1 and 2 uses the ground truth room-layout segmentation to mask the estimated depth map, meaning that all depth points may influence the plane fitting. However, *FCRN + Fine* leverages the full segmentation and only points labelled wall, floor, ceiling or window are included in the plane fitting.

We also compare against PlaneNet, using the method described in [19] for inferring room layout. This takes advantage of the consistent ordering of plane inference of the PlaneNet network, where floors are predicted at a specific index etc. We fine-tuned PlaneNet on SUN RGB-D using the depth-only method mentioned in their paper, this did not outperform their pretrained model.

The results against the baseline in Tables 1 and 2 demonstrate the quality of the planes regressed using our method and further illustrate the advantages of targeting the abstract room surfaces, rather than relying on visible pixel patches. In addition, it is worth noting that the baseline makes use of ground truth segmentation. For a true comparison, segmentation predictions should be used, further reducing the quality of the fitted planes. Even with the advantage of ground truth segmentation, the baseline is out performed by our method.

While PlaneNet slightly outperforms our method for the accuracy of the regressed normals, our method predicts room planes that are closer to the

Table 3. The performance of our Planar R-CNN for predicting 2D room layouts. Evaluated on the NYUDv2-303 dataset [33]. Results for other methods are as stated in [19]

2D Layouts	[33] RGB + D	[33] RGB	[26]	RoomNet	PlaneNet	**Ours**
Pixel Error	*8.04%*	13.94%	13.66%	12.96%	12.64%	**12.19%**



Fig. 5. Example 2D room layouts produced by our method on the NYUDv2 303 dataset. The first row shows the input image. The second and third rows show the ground truth and predicted room layout segmentations, respectively. The fourth row shows the prediction overlaid onto the input image. It is worth noting that in the last two columns the rooms in the input are not box-shaped. We believe that our prediction better reflects the room shape in these cases than the ground truth. 🔍

groundtruth. We would suggest that the high normal accuracy shown by PlaneNet could be explained by planar objects, such as cupboards and closets, which are generally arranged to be parallel to the walls that they occlude. Further, PlaneNet tends to miss planes which are not near the camera. It is worth noting that the architectural planes which are missed by PlaneNet do not negatively impact the results shown in Tables 1 and 2. Whereas, our method performed well as a room-plane detector, with the RoIAlign and RoIPool methods each achieving a mAP of 67.9 and 69.6, respectively.

Example detections from our network, as well as their respective planes from single images, are pictured in Figure 3. Predicted planes are limited in extent in this figure by the predicted bounding-box, in order to produce a number of plane segments. In each case, a colourised point cloud produced from the paired depth image is shown to provide context for the predictions. Also included are examples which break the box and Manhattan world assumptions. Further, in Figure 4 we show example detections demonstrating the advantages of targeting abstract room-planes over plane-fitting to visible wall, floor or ceiling patches. These examples include walls being correctly detected and their plane being accurately regressed when it is entirely or mostly occluded.

2D Room Layout Predictions We demonstrate the ability for our method to capture room geometry further by producing room layout predictions for

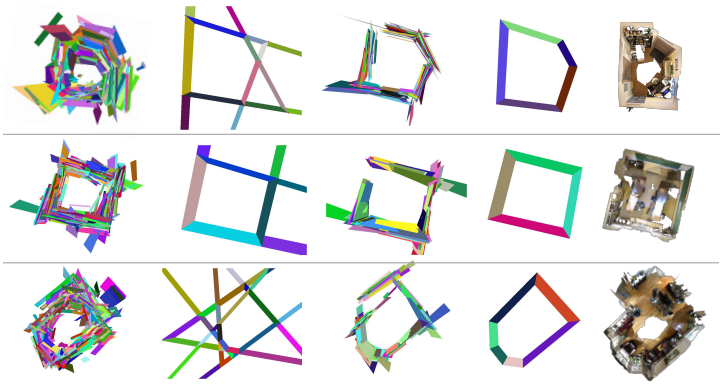


Fig. 6. Each row visualises the steps of our spatial voting method for determining the room layout from Planar R-CNN measurements on a ScanNet sequence. The first column shows a subset of the raw measurements. After inferring the underlying room planes, we intersect to find the set of wall segment candidates, shown in the second column, as well as the top n measurements from each cluster with the highest likelihood to form the group of voter patches, shown in the third column. The fourth and fifth column show the final layout estimation and the ground truth mesh, respectively. 🔍

individual frames on the NYUDv2 303 dataset. Since we do not assume a box layout, we do not classify the type of wall as left, right, or front. Although floor and ceilings are obvious, these wall classes present a level of ambiguity. In an image of a room corner it is often an arbitrary choice about whether the two walls should be labelled left and front, or front and right. Therefore, to evaluate our predicted layouts we must map our wall detections to the five possible plane types in the ground truth, which we do using the Hungarian algorithm.

As can be seen with the results presented in Table 3, we outperform all RGB methods tested, including PlaneNet [19] and RoomNet [18], designed for this task. We also believe our results could be improved by merging architectural plane detections if the plane equations are similar. This merging of detections would better reflect the ground truth box approximations used in the dataset.

Further, we present qualitative results of the 2D layouts that are produced by our method in Figure 5. These examples help to illustrate that while the box representation of rooms is often very well suited to the room pictured, as shown in the first five columns of Figure 5, there are cases where this approximation does not reflect the geometry of the scene particularly well. For example, in the last two columns of Figure 5, we believe that our predictions offer a better representation of the room geometry than the ground truth.

3D Room Layout Estimation on ScanNet. Unfortunately, ScanNet does not provide the room layout annotations required to train our network. This meant we were unable to fine-tune or provide quantitative results on this dataset.

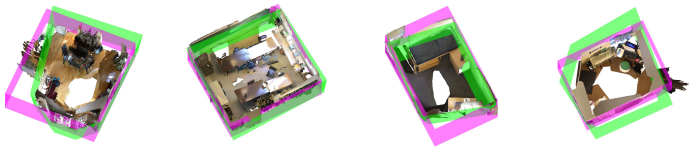


Fig. 7. Manhattan boxes (magenta) fitted to the ground truth point clouds using the method provided in SUN RGB-D shown with our own unconstrained layouts (green).

Therefore, we provide qualitative room-layouts to demonstrate that our method intended to visualise the geometric information captured by our Planar R-CNN.

The qualitative results in Figure 6 show example room layouts obtained on the ScanNet dataset. From the noisy raw plane measurements, the clustered planes encapsulate a broad representation of the room. The fourth row provides an interesting case as the upper segment of room is missed altogether. When inspecting the ground truth mesh, it was found that there are limited vertices present in the mesh. This leads us to believe that this area was not visible for much of the video. It is worth noting that, most likely because of the level of noise in the initial measurements, fine room geometry is missed in the clustering.

Although the produced room-layout estimates lack fine detail, we believe that they provide a better representation of room geometry than a Manhattan box approximation would. To demonstrate this, we produce a qualitative baseline using the Manhattan model fitting provided in SUN RGB-D. The boxes produced in Figure 7 were fitted to the ground truth point clouds, and thus would likely deteriorate if point clouds obtained from a monocular method were used.

We believe that the results shown in Figures 6 and 7 demonstrate our methods ability to capture useful structure from ScanNet sequences without using depth, enforcing a Manhattan assumption, or imposing temporal consistency.

6 Conclusions

We have proposed a method for room layout estimation without a box or Manhattan world assumption. It combines instance detection implemented via an R-CNN with Gaussian Mixture Model-based probabilistic clustering, and allows for the effective reconstruction of walls which do not lie on either of the two principle aligned axes, or the vertical direction.

Possible extensions would include temporal coherency and consistency both in the 2D inference stage and in the clustering. We believe that this would improve the fidelity of the measurements in the global room, further increasing the fidelity of the layout estimation.

Acknowledgements. We gratefully acknowledge the European Commission Project Multiple-actOrs Virtual Empathic CARegiver for the Elder (MoveCare) for financially supporting the authors for this work.

References

1. Brahmbhatt, S., Christensen, H.I., Hays, J.: Stuffnet: Using 'stuff'to improve object detection. In: Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on. pp. 934–943. IEEE (2017)
2. Cabral, R., Furukawa, Y.: Piecewise planar and compact floorplan reconstruction from images. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. pp. 628–635. IEEE (2014)
3. Chen, W., Xiang, D., Deng, J.: Surface normals in the wild. arXiv preprint arXiv:1704.02956 (2017)
4. Chetverikov, D., Stepanov, D., Krsek, P.: Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. Image and Vision Computing **23**(3), 299–309 (2005)
5. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). vol. 1, p. 1 (2017)
6. Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., Theobalt, C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. ACM Transactions on Graphics (TOG) **36**(3), 24 (2017)
7. Dasgupta, S., Fang, K., Chen, K., Savarese, S.: Delay: Robust spatial layout estimation for cluttered indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 616–624 (2016)
8. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2650–2658 (2015)
9. Flint, A., Murray, D., Reid, I.: Manhattan scene understanding using monocular, stereo, and 3d features. In: Computer Vision (ICCV), 2011 IEEE International Conference on. pp. 2228–2235. IEEE (2011)
10. Girshick, R.: Fast r-cnn. arXiv preprint arXiv:1504.08083 (2015)
11. Gupta, A., Hebert, M., Kanade, T., Blei, D.M.: Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In: Advances in neural information processing systems. pp. 1288–1296 (2010)
12. Gupta, S., Davidson, J., Levine, S., Sukthankar, R., Malik, J.: Cognitive mapping and planning for visual navigation. arXiv preprint arXiv:1702.03920 **3** (2017)
13. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Computer Vision (ICCV), 2017 IEEE International Conference on. pp. 2980–2988. IEEE (2017)
14. Hedau, V., Hoiem, D., Forsyth, D.: Recovering the spatial layout of cluttered rooms. In: Computer vision, 2009 IEEE 12th international conference on. pp. 1849–1856. IEEE (2009)
15. Izadinia, H., Shan, Q., Seitz, S.M.: Im2cad. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2422–2431. IEEE (2017)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
17. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 3D Vision (3DV), 2016 Fourth International Conference on. pp. 239–248. IEEE (2016)
18. Lee, C.Y., Badrinarayanan, V., Malisiewicz, T., Rabinovich, A.: Roomnet: End-to-end room layout estimation. arXiv preprint arXiv:1703.06241 (2017)

19. Liu, C., Yang, J., Ceylan, D., Yumer, E., Furukawa, Y.: Planenet: Piece-wise planar reconstruction from a single rgb image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2579–2588 (2018)
20. Mallya, A., Lazebnik, S.: Learning informative edge maps for indoor scene layout prediction. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 936–944 (2015)
21. Mottaghi, R., Chen, X., Liu, X., Cho, N.G., Lee, S.W., Fidler, S., Urtasun, R., Yuille, A.: The role of context for object detection and semantic segmentation in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 891–898 (2014)
22. Mura, C., Mattausch, O., Villanueva, A.J., Gobbetti, E., Pajarola, R.: Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. Computers & Graphics **44**, 20–32 (2014)
23. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
24. Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., Belongie, S.: Objects in context. In: Computer vision, 2007. ICCV 2007. IEEE 11th international conference on. pp. 1–8. IEEE (2007)
25. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
26. Schwing, A.G., Urtasun, R.: Efficient exact inference for 3d indoor scene understanding. In: European Conference on Computer Vision. pp. 299–313. Springer (2012)
27. Shi, M., Caesar, H., Ferrari, V.: Weakly supervised object localization using things and stuff transfer. In: Proc. IEEE Int. Conf. Comput. Vis.(ICCV) (2017)
28. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
29. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 567–576 (2015)
30. Song, S., Zeng, A., Chang, A.X., Savva, M., Savarese, S., Funkhouser, T.: Im2pano3d: Extrapolating 360 structure and semantics beyond the field of view. arXiv preprint arXiv:1712.04569 (2017)
31. Wang, X., Fouhey, D., Gupta, A.: Designing deep networks for surface normal estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 539–547 (2015)
32. Xu, J., Stenger, B., Kerola, T., Tung, T.: Pano2cad: Room layout from a single panorama image. In: Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on. pp. 354–362. IEEE (2017)
33. Zhang, J., Kan, C., Schwing, A.G., Urtasun, R.: Estimating the 3d layout of indoor scenes and its clutter from depth sensors. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1273–1280 (2013)
34. Zhang, Y., Song, S., Tan, P., Xiao, J.: Panocontext: A whole-room 3d context model for panoramic scene understanding. In: European Conference on Computer Vision. pp. 668–686. Springer (2014)
35. Zhang, Y., Yu, F., Song, S., Xu, P., Seff, A., Xiao, J.: Large-scale scene understanding challenge: Room layout estimation. accessed on Sep **15** (2015)

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Thinking Outside the Box: Generation of Unconstrained 3D Room Layouts
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	**Howard-Jenkins, H.,** Li, S., and Prisacariu, V. (2018). Thinking Outside the Box: Generation of Unconstrained 3D Room Layouts. In *Asian Conference on Computer Vision (ACCV).*

Student Confirmation

Student Name:	Henry Howard-Jenkins		
Contribution to the Paper	• Joint conception and sole validation of the central idea • Design of the architecture and framework • Implementation and running of all experiments • Research of prior work • Writing and presentation of the paper		
Signature	HH-J	Date	31/01/2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Prof Victor Adrian Prisacariu		
Supervisor comments	The description is accurate		
Signature	VPA	Date	09/02/2022

This completed form should be included in the thesis, at the end of the relevant chapter.

3.3 Conclusion

In this chapter, we presented a novel formulation of room layout estimation for capture of arbitrary room structure without constraint. Specifically, we consider architectural structure, such as walls, floors and ceilings as countable *things*. With this new perspective, we are able to approach the task of room layout estimation as one of instance detection using an R-CNN structure, which we call Planar R-CNN. To augment the room plane detections, we also directly regress the 3D plane equation. Through a combination of the extents predicted by the bounding boxes, as well as the 3D plane equations, we are able to leverage Planar R-CNN to predict room layouts in single RGB images, as well as aggregated across a posed video sequence.

Planar R-CNN's ability to capture layout was demonstrated with its good performance on the NYU-D-303 layout dataset [Zhang et al., 2013]. Despite the single image layout estimation being evaluated on a dataset which leverages a box-approximation for layouts, Planar R-CNN's layout prediction outperformed other methods tested without imposing any constraints on layout. This strong performance exemplified an ability to build an arbitrary layout from visual cues, rather than simply matching to a predetermined set of candidates. Notably, by detecting room planes directly, Planar R-CNN demonstrated ability to capture room layout beyond the visible pixel space. For example, walls could be detected and location inferred even if they were fully occluded, such as by a cupboard.

The general capability to capture layout is further demonstrated through the generation of 3D layouts from a video sequence. However, while these results do demonstrate a coherency across predictions, the final 3D estimates are coarse, missing fine detail in room geometry. This is due to the noise in the plane-depth predictions, which in large part can be explained by the lack of fine-tuning on the ScanNet dataset [Dai et al., 2017], although to some extent this was also encountered in the NYU-D-303 dataset.

There is an interesting interplay between the method's strength in predicting a coherent set of walls, floors and ceilings to be able to form an accurate instantaneous layout prediction in the form of segmentation, and its prediction of 3D geometry only allowing a coarse estimation of room geometry. Some of this will be explained by the method relying on implicitly learning camera parameters. But it may also suggest that layout estimation from an image can be considered as a combination of two tasks. The first is the comprehension of the room layout in an image, and the second is the articulation of that layout to the chosen representation. In this setting,

it seems that Planar R-CNN is better suited to articulating a layout segmentation than the explicit 3D geometry. However, the results validate that the method is able to accurately comprehend the layout of rooms of arbitrary geometry.

In the following chapters, we turn to the other stated goal of Part I: localisation with respect to layout alone. We explore the insight that comprehension and articulation of layout from an image can be disentangled. We remove the articulation of an explicit layout entirely and instead localise with respect to an implicit representation of layout.

Chapter 4

Implicit Understanding of Architectural Structure for Localisation

4.1 Introduction

Camera localisation from a known floor plan provides some notable benefits over many more traditional deep relocalisation work. Due to the permanence of architectural structure and the availability of floor plans themselves, this form of localisation enables positioning in previously unvisited environments, and in scenes with considerable other dynamics, such as objects and furniture.

In this chapter, we present LaLaLoc, our work to position a panorama image within a known floor plan through the implicit alignment of room layout in the query image with the known geometry from a floor plan. One approach for architecture-based localisation would be explicit layout estimation followed by layout-to-layout matching and alignment. Instead, in this chapter we match and align the RGB panorama to layouts directly. This is fuelled by the intuition that comparison between a layout and an image is an easier task than predicting the layout explicitly. Our work conditions an embedding space that encodes layout information and is shared between images and with layouts rendered within the scene.

4.1.1 Contributions

- We propose LaLaLoc, a localisation method that is robust to scene dynamics such as the configuration of furniture, and able to localise in a new scene without a prior visit.

- We introduce direct pose optimisation in the latent space. This allows for cross-modal position optimisation, without the need for a decoder to traverse data modes for the computation of the matching cost.
- Through experimental evaluation, we demonstrate the accuracy of LaLaLoc as well as validate its formulation. This includes showing that the representation of room layout has significant influence on the efficacy of layout-based localisation and cannot be used interchangeably.

LaLaLoc: Latent Layout Localisation in Dynamic, Unvisited Environments

Henry Howard-Jenkins
Active Vision Laboratory
University of Oxford
henryhj@robots.ox.ac.uk

Jose-Raul Ruiz-Sarmiento
Machine Perception and
Intelligent Robotics Group
University of Málaga
jotaraul@uma.es

Victor Adrian Prisacariu
Active Vision Laboratory
University of Oxford
victor@robots.ox.ac.uk

Abstract

We present LaLaLoc to localise in environments without the need for a prior visit, and in a manner that is robust to large changes in scene appearance, such as a full rearrangement of furniture. Specifically, LaLaLoc performs localisation through latent representations of room layout. LaLaLoc learns a rich embedding space shared between RGB panoramas and layouts inferred from a known floor plan that encodes the structural similarity between locations. Further, LaLaLoc introduces direct, cross-modal pose optimisation in its latent space. Thus, LaLaLoc enables fine-grained pose estimation in a scene without the need for a prior tour, as well as being robust to dynamics, such as a change in furniture configuration. We show that in a domestic environment LaLaLoc is able to accurately localise a single RGB panorama image to within 8.3cm, given only a floor plan as a prior.

1. Introduction

Camera relocalisation is a fundamental problem in computer vision. Image-based relocalisation represents the goal of estimating the camera pose of an unseen image, given some prior knowledge about the surrounding environment. In this paper, we tackle the task of localisation in an environment that has not been previously visited, and one in which there may be considerable scene dynamics – an area where significant scope for improvement has been identified [37].

To address this, we propose to localise with respect to a known floor plan and the layout visible at a location within the scene. Floor plan-based localisation is particularly suited for the long-term localisation setting as, while objects and furniture may have moved, items represented in a structural floor plan, such as walls, floors and ceilings, will remain static. Therefore, it enables localisation over a long period of time without requiring continual re-training or re-mapping. In addition, in this formulation we only need the floor plan as prior, thus removing the need for a prior tour of the target environment, *i.e.* without a training trajectory of images.

We present LaLaLoc, which performs floor plan-based localisation through latent representations of room layout. This layout latent space is cross-modal, shared between layouts inferred from the floor plan and the RGB panoramas queried at inference time. More specifically LaLaLoc performs localisation in two stages, depicted in Figure 1. The first stage provides a coarse estimate of pose through cross-modal retrieval. For the second stage, we propose a cross-modal direct optimisation of pose through differentiable rendering.

Differentiable rendering has been shown to be effective for object pose estimation [23, 12]. But these methods typically rely on like-for-like rendering losses, such as the pixelwise error between the rendered and target images. However, since LaLaLoc operates across multiple modes of data between query and prior, the prediction of a common data mode would be required for the comparison losses. Instead, we propose to optimise for pose directly in the layout latent space. Through this formulation, LaLaLoc is able to accurately align the floor plan to a cluttered RGB panorama without ever explicitly predicting its layout.

The contributions of this paper can be summarised as:

- We propose LaLaLoc, a highly accurate localisation method that is robust to scene dynamics such as the configuration of furniture, and able to localise in a previously unseen environment.
- We introduce direct pose optimisation in the latent space. This allows for cross-modal pose optimisation, without the need for a decoder to traverse data modes for the computation of the matching cost.
- Through experimental evaluation, we demonstrate the accuracy of LaLaLoc as well as validate its formulation. This includes showing that the representation of room layout has significant influence on the efficacy of layout-based localisation and cannot be used interchangeably.

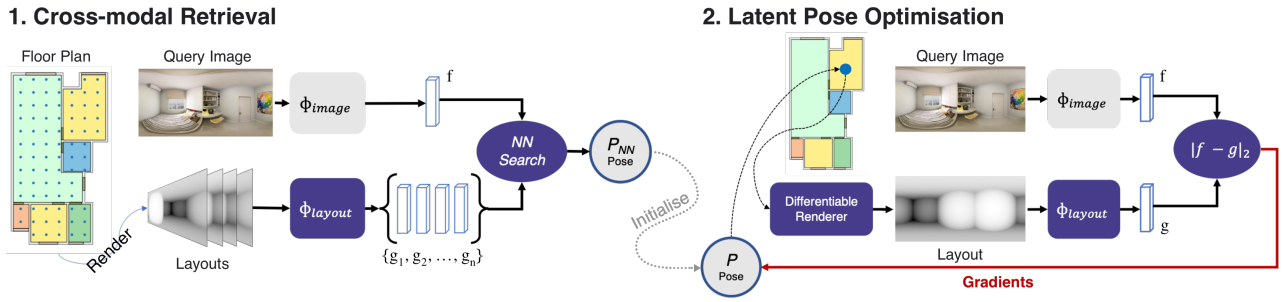


Figure 1. Overview of localisation using LaLaLoc. In the retrieval stage, the query image is mapped to the layout latent space by Φ_{image}. We then sample a grid of poses from a known floor plan, render their layouts, and compute their respective latent representation through Φ_{layout}. An initial pose estimate is found by a nearest neighbour search within this shared latent space. The nearest neighbour pose is then used as an initialisation for direct pose optimisation with our proposed latent pose optimisation. This is a gradient-based optimisation of pose conducted in the shared latent space, thus removing the need to decode into a common data mode. Our Vogel Disc resampling stage is ommitted from this viusalisation for clarity.

2. Related Work

A wide variety of methods have been produced to tackle the task of camera localisation. There are scene-specific methods, which require fine-tuning to each individual scene. Pose-regression methods [18, 17, 10] train a deep network for each scene to directly predict the camera pose from an input, but these methods are limited in accuracy [27]. Instead of regressing camera pose, scene-coordinate regression methods [30, 7, 8, 9, 35] densely predict 2D-3D correspondences between the query and the environment, which then allows for solving for pose via PnP. On the other hand, some groups of methods are able to generalise across scenes without need for retraining. Image retrieval methods [28, 1, 2, 13] estimate pose through by using the pose of the most similar image within an image database. 3D structure-based methods [21, 20, 26] instead establish 2D-3D between the query image and points in a SfM model. However, the scene-specific and generalisable methods discussed do have a commonality in that they require a prior tour in the same data modality of the region in which localisation is performed: scene-specific methods require this for re-training, while the generalisable methods require it for map/database building. By contrast, LaLaLoc performs localisation without a prior tour, instead leveraging only a known floor plan as a prior.

Particularly within the field of robotics, there have been a few proposed approaches to localise with respect to a floor plan. These methods operate by aggregating depth [40], detecting suitable features such as layout corners [14], or extracting layout edges [6, 38, 34] within query RGB and/or depth images to estimate and compare with the inferred layout at a location within the floor plan to estimate the observation likelihood. When depth information is available, scan-matching techniques [24] find a rigid alignment between the scene and the scan through alignment methods such as Iterative Closest Point (ICP) [3], General-ICP (GICP) [29], or Normal Distributions Transform (NDT) [4]. ICP-based scan matching techniques have successfully been used to perform localisation within a floor plan [5, 39]. These methods generally rely on a flow of information over multiple sequential measurements, typically alongside motion priors, such as from wheel or camera odometry, and are fed as hypothesis weightings into a Monte Carlo Localisation (MCL) [33] framework. In this paper, however, we approach the task of localisation from an instantaneous observation with no motion or time-coherency cues, and without assuming a good initialisation for pose. Furthermore, unlike the scan-matching methods, we do not require depth information, and in fact we remove the need for any explicit prediction of layout geometry at query time entirely, instead solely leveraging latent representations of layout.

Kim *et al.* [19] first proposed the learning of a latent space which captures room layout similarity. Specifically, this is applied to the task of image retrieval, where images are embedded to reflect their underlying room layout and for a given query image the goal is to return other images with similar layouts. Zheng *et al.* [42] later employed a layout embedding space to aid in the prediction of room layouts. However, both of these pieces of work focused on room layouts that follow a box approximation, where the layout takes the form of the inside of a convex cuboid. This leads to much reduced variability in the space of possible layouts: there are 11 types of room layout which can be seen in an image, of which [19] only considered 1. We, instead, consider general room layouts, without imposing any assumptions about their structure, leading to more variety in potential layout. However, it is the same concept of a layout latent space on which LaLaLoc's localisation is enabled.

3. Task and Definitions

We perform camera pose localisation within a floor plan provided as prior knowledge. We localise the 2 DoF camera

pose, P, to a point in a 2D floor plan assuming known orientation. For rendering of the reference layouts, we assume that the camera and ceiling height are given, and produce a 3D floor plan $\mathcal{M}$ of the scene through extrusion of the 2D floor plan. $\mathcal{M}$ consists of only walls, floors and ceilings.

Throughout the paper, we deal with three main data types that may be captured or rendered at a position within the floor plan, P_p: I_p is the RGB panorama captured; L_p is a rendered depth image corresponding to $\mathcal{M}$ being projected into the image; C_p is a pointcloud formed by back-projecting L_p, in effect a sub-sampling of $\mathcal{M}$.

4. Latent Layouts

LaLaLoc consists of a network with two parallel branches to form a quasi-Siamese network. One branch, Φ_{image}, computes a feature descriptor for an RGB panorama image, while the other branch, Φ_{layout}, computes a descriptor from layout renderings. The embedding for a RGB panorama image should be identical to the embedding computed for a layout rendering at the same camera pose. This means that LaLaLoc is tasked with computing a singular latent representation of the room layout as visible at a particular location within a floor plan, independent of the sensor data used to compute it.

To learn such an embedding space, we take an approach analogous to knowledge distillation [15]. Knowledge distillation generally first trains a complex model as the teacher, which is then used to improve the convergence of a less complex model. Instead, we train a model on the easier task: Φ_{layout} performing layout-to-layout matching, and use it as a "teacher" model for a "student" performing a more complicated task: Φ_{image} learning to encapsulate the layout information present in an RGB image. This approach allows us to maximise the richness of our layout latent space by only considering learning the relationships between layouts, before training Φ_{image} to also map images to this space. However, this learning formulation in LaLaLoc differs from more general knowledge distillation as we keep Φ_{layout} to use for matching with the outputs of Φ_{image}. Therefore, the task requires that Φ_{image} not only maintains the relative differences between embeddings, and with it their respective layout similarity, but that it is anchored to the corresponding embedding of Φ_{layout}.

In the following, we describe the learning of the latent space, as well the mapping of other modalities to it. An overview of this training procedure is shown at the top of Figure 2 and the architecture used is given in the supplementary materials.

4.1. Φ_{layout} : Layout-similarity Latent Space

We first learn a rich latent representation of room layouts by imposing a metric loss for training the layout branch Φ_{layout} alone. The embedding space is conditioned so that

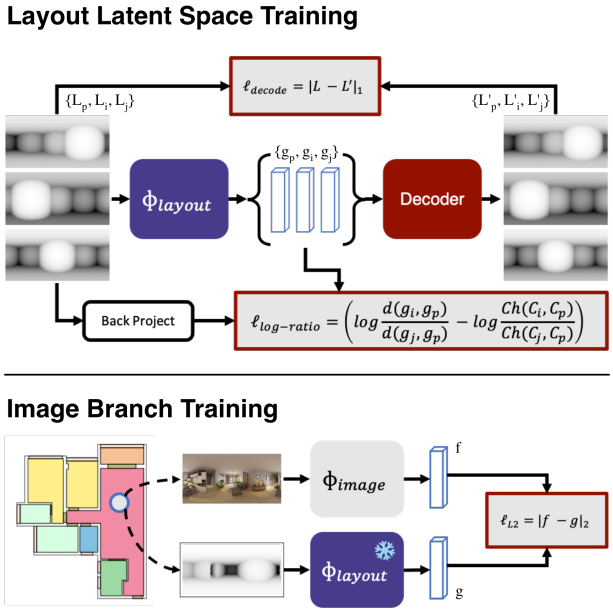


Figure 2. Overview of the LaLaLoc training process. *Top:* The learning of the layout latent space performed solely by Φ_{layout}. *Bottom:* The routine used to train Φ_{image} to map RGB panoramas to the latent space learned by Φ_{layout}.

the distance between the latent representations of layout renderings should reflect the structural difference in their respective layouts. This is achieved through a log-ratio loss formulation, as proposed by Kim *et al.* [19]:

$$\ell_{log_ratio}(p,i,j) = \left(log\frac{D(g_i,g_p)}{D(g_j,g_p)} - log\frac{\text{Ch}(C_i,C_p)}{\text{Ch}(C_j,C_p)}\right)^2, \quad (1)$$

where (p,i,j) represents a triplet of consisting of an anchor p, given by the pose of the RGB panorama, with i and j as two neighbours of p; $g = \Phi_{layout}(L)$ represents the layout embedding computed from layout rendering L; C indicates the pointcloud generated from back-projection of the layout depths; $D(\cdot)$ is the Euclidean distance function and $\text{Ch}(\cdot)$ is the Chamfer distance.

In addition to the layout similarity loss, we also leverage a layout decoder for training. The decoder takes the latent vector $g = \Phi_{layout}(L)$ and decodes it to layout depth image, L' thus forming a layout auto-encoder during training. We apply a decoding loss in the form:

$$\ell_{decode} = |L' - L|_1, \quad (2)$$

with the L1-norm chosen for its favourable performance for depth estimation [11].

4.2. Φ_{image} : Learning to Match Images to Layouts

Once the layout latent space has been learned by the teacher branch, we train our student branch, Φ_{image}, to embed RGB images to the same layout embedding space. For

this training, we freeze Φ_{layout} , therefore the RGB branch is tasked only with mapping images to a fixed layout space and can be aided by the response of Φ_{layout} .

We do this simply by applying a loss on the Euclidean distance between the RGB embedding, f, and the layout embedding, g. This mirrors the localisation strategy where image-layout matching will be predicted through the Euclidean distance between the respective embeddings in the latent space. The loss is given simply by:

$$\ell_{L2}(p) = |f_p - g_p|_2. \tag{3}$$

5. Localisation

While it would be possible to fine-tune the latent representations computed by LaLaLoc when localising in new scenes, we instead propose the approach as a fixed network, which is able to generalise to new scenes without any fine-tuning or RGB training trajectories.

LaLaLoc is a 2-stage localisation approach: a coarse, global retrieval stage, followed by cross-modal pose optimisation through differentiable rendering. An overview of the localisation method is depicted in Figure 1. In the following, we describe each of these stages in more detail.

5.1. Coarse retrieval

The coarse retrieval stage operates by sampling candidate poses in a uniform grid across the known floor plan. At each sampled pose, we render the layout from the known floor plan geometry, which is then used to compute a latent vector. These latent vectors and their associated poses form the reference database for localisation. When performing localisation of a query image, we compute the latent vector of the query image and compute its distance with respect to each entry in the reference database. The coarse localisation estimate is given by the pose belonging to the nearest neighbour latent vector.

5.2. Pose refinement

The first, coarse stage of the LaLaLoc localisation method is limited by the density of the sampled poses within the floor plan. We, therefore, include pose refinement to enable more fine estimate of the camera location. We propose two methods for this: a retrieval-based approach which resamples more densely around the coarse estimate and again localises by retrieval; or direct optimisation of pose by layout similarity in the latent space.

To more densely explore the region around the nearest neighbour from coarse retrieval, we sample poses in a Vogel Disc [36] centred around the nearest neighbour. This produces an (approximately) even sampling in a circular region around the neighbour. The refined pose is then returned as the most similar from these newly sampled poses.

We will describe the formulation of the direct optimisation more thoroughly below.

5.2.1 Latent Optimisation of Pose

We propose a direct pose optimisation through differentiable rendering. While differentiable rendering-based approaches have been shown to be effective for pose estimation [23, 12], these methods rely on homogeneous data to compute losses between the prediction and the target, often employing pixelwise losses based on photo-metric or depth reconstruction errors. However, in our application we again must tackle the challenge of the asymmetry of our query (RGB) and reference (layouts) data types.

Instead, we optimise for pose with latent losses, using distance between embeddings computed by our network to model the matching energy between RGB images and layouts. In doing, so we are able to bridge the gap between data modalities, without the need for explicit prediction. This is achieved by employing a differentiable renderer to compute the layout at a pose estimate, P_r, we can ensure that the chain of operations to compute a layout embedding at P_r are differentiable: $g_r = \Phi_{layout}(\Omega(P_r, \mathcal{M}))$, where $\Omega(P_r, \mathcal{M})$ is the layout rendering. We therefore can refine the pose using a gradient-based optimisation with objective:

$$\min_{P_r} D(\Phi_{layout}(\Omega(P_r, \mathcal{M})), f_p). \tag{4}$$

6. Dataset

Training and evaluation is performed on the Structured3D dataset [41]. The dataset consists of 3,500 synthetic indoor scenes. Each scene comprises of multiple rooms, leading to 21,835 rooms in total. Importantly for our task, a 3D floor plan is provided, as well as a photorealistic panorama image rendered for each room in the scene.

We follow the predefined split of scenes, with 3000 used for training, 250 used for validation and 250 used for testing. We found that some scenes had corrupted data, which were excluded resulting in 2979/246/249 scenes for training/validation/testing. Due to the split being conducted over scenes, it means that all evaluation within this paper is conducted on an unvisited room, where we define unvisited as a scene with no prior image capture *e.g.* a training trajectory captured in the target scene.

For each image, there are three furniture configurations: empty, simple and full. Each of these are rendered in three lighting conditions: warm, cool and raw. Unless otherwise stated, a configuration is selected randomly at each iteration during training and evaluation is performed in the "full" and "warm" furniture and lighting settings, respectively – with full being the most difficult due to the presence of more distractors when inferring layout.

7. Experiments

We detail the localisation performance of LaLaLoc for our main task of localisation in an unseen environment, as

Query Type	Model	Layout R@1	Pose R@1	Localisation Accuracy Median (cm)	<1cm	<5cm	<10cm	<1m
Layout	*Oracle*	*100%*	*91.1%*	*20.4*	*0.3%*	*3.61%*	*13.5%*	*92.9%*
	2D ICP	-	-	0.5	**87.9%**	**99.8%**	**99.8%**	**99.9%**
	LaLaLoc	90.1%	87.4%	**0.2**	69.1%	82.4%	91.4%	94.9%
RGB/Furnished	2D ICP	-	-	21.8	**9.5%**	26.4%	35.6%	68.5%
	HorizonNet [32] + Loc	72.4%	68.0%	9.1	3.3%	29.3%	53.4%	77.4%
	LaLaLoc	72.4%	70.6%	**8.3**	3.6%	**32.0%**	**58.0%**	**87.5%**

Table 1. Layout localisation accuracy for our trained models and baselines on the Structured3D test set.

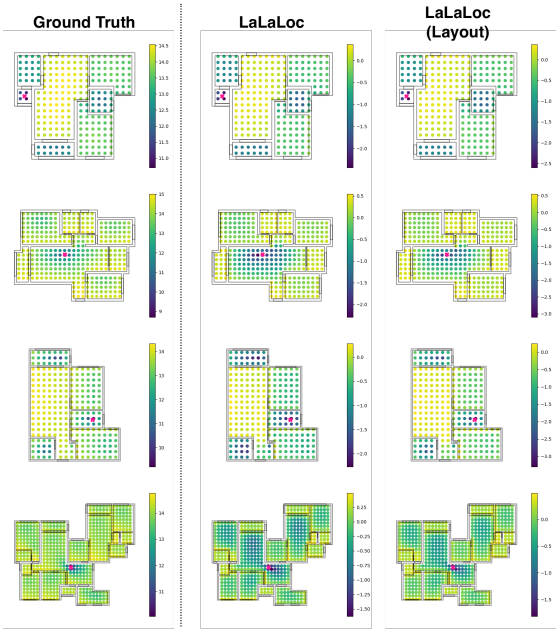


Figure 3. Qualitative depiction of the layout distances between a query pose, marked with a pink "X", and the grid of sampled poses across the scene. For each sampled pose, we colour it by the log of its respective distance from the query. *Left:* we plot the ground-truth layout distances. *Middle:* the layout distance is that predicted by LaLaLoc. *Right:* predicted by LaLaLoc in its layout-to-layout configuration.

well as analyse LaLaLoc's components and their contribution to the final accuracy. Unless otherwise stated, we sample a 2-dimensional grid of locations at a resolution of 0.5m $\times$ 0.5m. We then either take the nearest neighbour pose as our prediction, or use it to initialise the refinement stage. Our Vogel Disc re-sampling is set to a radius of twice the grid resolution. It is worth noting that all evaluations are performed with only the top 1 retrieval, which is the hardest test setting as there is limited scope to recover from a bad retrieval.

We include results for the two main modes of operation for LaLaLoc. The first assumes that the ground truth room layout is known for the query image. In this configuration, the layout branch, Φ_{layout} , is used alone. This setting is ideal for determining the expressiveness of the learned layout latent space as there are no discrepancies caused by translation from image to the layout latent space. The other configuration, however, is the more realistic scenario, where RGB panoramas are used as the query image, with Φ_{image} mapping them to LaLaLoc's embedding space.

For a baseline comparison, we include results for an ICP-based method. Similar methods are widely used in fields like robotics for motion estimation [22], scene reconstruction, map building and localisation [25]. The implementation details of this method are provided in the supplementary materials. In addition, we compare against a baseline formed from HorizonNet [32]. Given a query image, we use HorizonNet to predict its layout explicitly. We then retrieve the nearest neighbour from the sampled grid via L1 distances between depths. We perform VDR initialised at the nearest neighbour. Finally, we perform a gradient-based optimisation between the predicted layout and layouts rendered from the floor plan. It is worth noting that this is a significant extension to HorizonNet to perform localisation.

Various performance metrics are compiled across the following experiments that pertain to retrieval, and the final estimation of pose. The retrieval metrics provide insight into the performance of the first localisation stage: *Layout Recall @1* measures the proportion of the time the predicted nearest neighbour layout is the most similar layout; *Pose Recall @1* measures the fraction of predicted nearest neighbours that are the neighbour with the nearest pose to the query. Localisation accuracy is given by: *Median Pose Error* as the median distance between query and predicted pose; *Accuracy*$< \tau$ lists the fraction of frames localised to within the threshold, τ.

7.1. Localisation with a Floor Plan Prior

Through the full localisation procedure outlined in Section 5, we detail LaLaLoc's performance in Table 1. Worth considering first is the retrieval accuracy. The layout-to-layout retrieval performance deteriorates only slightly from the layout oracle, suggesting that the latent space is able to

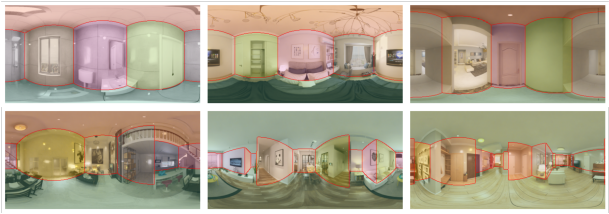


Figure 4. Example failures. *Top:* Cases where the wrong room has been retrieved due to ambiguity of layout, but the alignment is generally coherent with the layout seen in the query image. *Bottom:* The correct room is retrieved, but the alignment is incorrect. In these cases, it is often seen that room corners are incorrectly aligned to the edges of objects such as tables and counter tops. 🔍

capture layout extremely well. For qualitative confirmation of this, we plot ground truth and inferred layout distances in Figure 3. There it can be seen that the inferred difference or similarity between layouts is highly representative of the ground truth. Given the increased difficulty of inferring layout from RGB images, the cross modal retrieval performance of LaLaLoc, as seen qualitatively in Figure 3, is very good. This is emphasised by the retrieval accuracy where 87.5% of retrievals were within 1m of the true pose and thus within the radius of the Vogel Disc refinement.

In the layout-to-layout configuration, LaLaLoc is able to perform competitively against our ICP baseline. This is the ideal case for ICP, since it computes alignment on the point-clouds directly, both of which are known to the exact scale, whereas LaLaLoc introduces a layer of abstraction between the reference and the query. In the panorama-to-layout setting we see that LaLaLoc outperforms the baselines tested, including the method involving explicit layout prediction with HorizonNet [32] and aligning it to the known floorplan geometry.

We visualise LaLaLoc's two main types of failure in Figure 4. The most common is that an incorrect room with similar geometry is retrieved. In this scenario, the pose is still generally refined to a location that produces a plausible alignment. In the other scenario, the correct room is retrieved but the alignment is incorrect. Often in this scenario we see that floor or wall edges are aligned to objects in the room, suggesting that the refinement may have been caught in a local minimum.

In Table 2, we provide further results for localisation performance with a lower resolution grid of poses used for sampling. Specifically we sample at a resolution of 1m $\times$ 1m, as opposed to the original 0.5m $\times$ 0.5m. Despite the retrieval error being more than double that seen with the 0.5m grid, LaLaLoc is still able to recover from this during its refinement stage with final accuracy only seeing a small degradation in accuracy.

	Method	Localisation Error Med.	<1cm	<10cm	<1m
	Retr. Oracle	*42.7*	*0.0%*	*2.9%*	*75.7%*
Lay.	2D ICP	0.7	**76.9%**	**99.4%**	**99.6%**
Lay.	LaLaLoc	**0.5**	56.1%	70.5%	84.3%
Pano.	2D ICP	21.7	**7.3%**	34.7%	69.4%
Pano.	LaLaLoc	**11.5**	2.6%	**46.1%**	**79.5%**

Table 2. Localisation performance when sampling on a lower resolution, 1m $\times$ 1m grid.

Furniture Level	Method	Layout R@1	Pose R@1	Median Error (cm)
Empty	2D ICP	-	-	**2.4**
	LaLaLoc	76.4%	74.0%	5.7
Simple	2D ICP	-	-	10.0
	LaLaLoc	74.8%	73.3%	**6.7**
Full	2D ICP	-	-	23.0
	LaLaLoc	72.4%	70.6%	**8.3**

Table 3. Impact of furniture level on localisation performance. LaLaLoc is run with a full refinement scheme (VDR + LPO) for computation of localisation error.

7.1.1 Robustness to Level of Furniture

With the introduction of furniture in the second test setting, the effectiveness of LaLaLoc becomes readily apparent. LaLaLoc now significantly outperforms the ICP baseline in nearly all the metrics evaluated. In Table 3, we further analyse the impact of the furniture configuration on the final localisation accuracy by evaluating performance across those provided in the Structured3D dataset. Here, it becomes apparent that, although the ICP baseline performs very well when operating on the ground-truth layouts, when the query type becomes increasingly furnished its performance drops remarkably. On the other hand, the results show that LaLaLoc sees very little deterioration as the number of distractors in the scene increase from empty to full. This shows that Φ_{image} within LaLaLoc is able to accurately infer room layout from the RGB images, even when the room is cluttered.

7.1.2 Pose Refinement

To evaluate the performance of LaLaLoc's pose refinement, here we compare various strategies. Specifically, we investigate the combination of the Vogel Disc re-sampling and latent pose optimisation to improve upon the coarse pose as predicted by retrieval. We also include an explicit prediction and alignment strategy,"VDR + Decode". This is implemented by keeping the decoder from training and using it to predict the query layout explicitly. Pose is then pre-

Refine Method	Localisation Error Med.	<1cm	<5cm	<10cm
Retrieval	22.5	0.3%	3.19%	11.8%
VDR-only	11.0	0.8%	15.1%	45.3%
LPO-only	10.5	2.0%	27.4%	48.4%
VDR + Decode	12.2	0.5%	13.2%	39.7%
VDR + LPO	**8.3**	**3.6%**	**32.0%**	**58.0%**

Table 4. Comparison of latent pose optimisation against a render-and-compare method, "Decode", as well as against no optimisation. All refinement methods are initialised at the pose returned from our Vogel disc re-sampling.

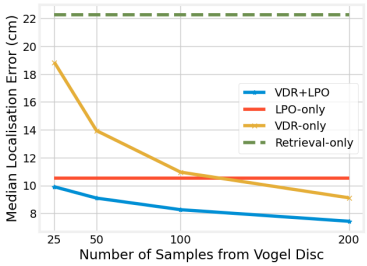


Figure 5. Investigation of the impact of sampling density from the Vogel Disc before performing latent pose optimisation. We include results for performing latent pose optimisation without the Vogel Disc re-sampling beforehand, as well as reference to the initial retrieval accuracy used to initialise these refinement schemes.

dicted via a gradient-based optimisation on the L1 distance between the predicted layout and rendered layouts from the floor plan. Results are listed in Table 4. The VDR and LPO refinement stages of LaLaLoc are both shown to be effective, with each able to improve on the retrieval alone. Our latent pose optimisation applied alone outperforms VDR, but the best localisation performance is given by using the re-sampling to initialise the latent pose optimisation.

In comparison to the decode baseline, it is clear from the results that the optimisation within the latent space leads to superior results. In fact, the refined poses through decoding performed worse than the retrieval poses used to initialise them. We believe the discrepancy in refinement accuracy can be explained by layouts being easier to capture layout in the latent space, than they are to articulate, *i.e.* to explicitly predict. The simple depth decoder may not be sufficiently intricate to predict layouts suitable for this refinement. After all, prediction of general room layouts is an activate area of research in itself [16, 32, 31].

As a further evaluation of the latent pose optimisation, we perform refinement with a varying number of more densely sampled poses in the naive Vogel Disc refinement strategy. As can be seen in Figure 5, the benefit from the re-sampling decreases quickly as the number of samples decreases. This experiment also demonstrates the ability of latent pose optimisation to recover from increasingly poor initialisation provided by the Vogel Disc re-sampling. In all tested configurations, the latent pose optimisation was able to improve upon the re-sampled nearest neighbour, and the decrease in accuracy was far less severe.

7.2. Ablation

In this section, we validate various components of LaLaLoc's design, from the demonstration that layout localisation is feasible and the metric used for layout similarity, to the training objectives for the network. Unless otherwise stated, the results are all validation accuracy and correspond to retrieval only, with no pose refinement performed. In the experiments we mark our chosen design with an *.

Similarity Metric	Recall @1	Pose Error Median (cm)	Correct Room
Pose	*100%*	*19.8*	*100%*
Edges	79.3%	20.3	83.6%
Depth	80.3%	20.1	84.0%
Rel. Depth	77.1%	20.1	85.4%
Chamfer*	**90.8%**	**20.0**	**92.1%**

Table 5. Evaluation of layout similarity metrics for the localisation task on the Structured3D validation split. *Pose* refers to picking the nearest location in the sampled grid to the query.

7.2.1 Layout-similarity Metric

When discussing localisation from a floor plan, it is not unreasonable to think of room shapes identical to one another, thus leading to significant ambiguity and rendering localisation ineffective without other cues. To some degree this ambiguity cannot be completely removed when localising with respect to layout alone. However, we hypothesise that there are significant differences between layout representations and the respective measures of similarity in their susceptibility to this ambiguity. This means that, if the wrong metric is chosen, layout similarity can become a much less expressive signal for localisation. Therefore, we explore which representation of an arbitrary room layout gives the best performance for localisation.

Specifically, we evaluate four representation and layout similarity metric pairs: *Edges*, the 2D Chamfer distance between sets of edge pixels in layouts represented as an edge segmentation, used in [6] for floor plan localisation; *Depth*, defined as the L1 distance between layouts rendered as depth images; *Relative Depth*, the L1 distance between relative depth images (max depth value = 1 in all images); and *Chamfer distance*, where the depth images are back-projected and the Chamfer distance is computed between the resulting pointclouds.

We list the results in Table 5. As can be seen in the table,

Query Type	Training Routine	Layout R@1	Pose R@1	Median Error (cm)
Lay.	End-to-End	81.1%	80.4%	20.9
	2-stage*	**89.3%**	**87.6%**	**19.9**
RGB	End-to-End	24.4%	23.7%	62.7
	2-stage*	**70.7%**	**70.5%**	**21.7**

Table 6. Retrieval comparison with an end-to-end training on the Structured3D validation split.

Losses	Layout R@1	Pose R@1	Median Error (cm)
Oracle	*100%*	*90.8%*	*20.0*
ℓ_{log_ratio}	78.8%	77.3%	20.5
ℓ_{decode}	**90.9%**	84.6%	20.1
$\ell_{log_ratio} + \ell_{decode}$*	89.3%	**87.6%**	**19.9**

Table 7. Layout-to-layout teacher model ablation on the Structured3D validation set. We evaluate the contribution of each of the losses used to train our layout branch.

representing layout as a pointcloud with the Chamfer distance provides the best formulation of those tested, significantly out-performing the image-based similarity metrics tested.

7.2.2 End-to-End Training

We compare our proposed 2-stage training to an end-to-end formulation. We use the same losses as described previously, however we found it best to cut the gradient of g when applied in ℓ_{L2}. Therefore, Φ_{layout} is only optimised to minimise ℓ_{log_ratio} and ℓ_{decode}, mirroring the 2-stage approach. In Table 6, we list the results of this training strategy. Φ_{layout} is still able to learn a representative latent space, albeit significantly less so than when trained alone. However, Φ_{image} performs very poorly when trained in this manner, despite trying a number of different loss scaling factors. We hypothesise that the other losses pollute the gradients to Φ_{image}, similar to the introduction of additional losses for training Φ_{image}, as explored later. With this, it appears that our 2-stage training is the more effective, or at least significantly more forgiving.

7.2.3 Layout Latent Space Learning

Here we evaluate multiple objective functions to learn LaLaLoc's latent layout embedding space, *i.e.* for training Φ_{layout}. Results are listed in Table 7, where we see that the combination of the log-ratio loss and decoder loss provides the most effective latent space for layout-similarity matching and localisation. Notably, it appears that the autoencoder formulation is more representative than training with the relational similarity constraints alone.

Backbone	Predictor	Layout R@1	Pose R@1	Median Error (cm)
ResNet18*	FC*	**89.3%**	**87.6%**	**19.9**
ResNet50	FC	87.5%	85.3%	20.0
ResNet18	MLP	87.2%	86.3%	**19.9**

Table 8. Comparison of network architecture for Φ_{layout} on the Structured3D validation split.

Losses	Layout R@1	Pose R@1	Median Error (cm)
Oracle	*100%*	*90.8%*	*20.0*
ℓ_{L2}*	**70.7%**	**70.5%**	**21.7**
ℓ'_{log_ratio}	59.6%	56.9%	25.1
$\ell_{L2} + \ell'_{log_ratio}$	60.7%	59.9%	24.6
$\ell_{L2} + \ell_{kd_lr}$	58.9%	56.9%	25.0

Table 9. Image-to-layout model ablation on the Structured3D validation set. We evaluate the contribution of each of the losses used to train Φ_{image} from the frozen layout branch.

In Table 8 we evaluate some alternative architectures for Φ_{layout}. Specifically, we test replacing the ResNet18 backbone with ResNet50, and replacing the single fully connected layer with a 2-layer MLP after pooling. Although all options perform similarly, we find that the simplest architecture produced the best retrieval results.

7.2.4 Mapping Images to the Latent Space

When training Φ_{image}, there are many possibilities for how to best learn a mapping of RGB images to the existing latent space. In Table 9, we list results of various training objectives considered in the design of LaLaLoc. In this table: ℓ'_{log_ratio} refers to a cross-modal adaptation of the original loss, where g_p is replaced by f_p; ℓ_{kd_lr} is a knowledge distillation variant of the log-ratio loss, where the ground-truth layout similarity is replaced by the embedding distances as computed by Φ_{layout}. Equations for these losses are provided in the supplementary materials. Interestingly, the most simple objective, reducing the Euclidean distance between the image embedding and its corresponding layout embedding, resulted in the best results.

8. Conclusion

In this paper we have presented LaLaLoc, a localisation method which localises RGB queries to a known floor plan by matching in a latent space that encodes layout similarity. We further leverage this expressive latent space for direct pose optimisation through differentiable rendering. We show that LaLaLoc is able to localise with considerable accuracy in unseen or highly dynamic environments.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016. 2

[2] Vassileios Balntas, Shuda Li, and Victor Prisacariu. Relocnet: Continuous metric learning relocalisation using neural nets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 751–767, 2018. 2

[3] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 2

[4] P. Biber and W. Strasser. The normal distributions transform: a new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 3, pages 2743–2748 vol.3, 2003. 2

[5] Hermann Blum, Julian Stiefel, Cesar Cadena, Roland Siegwart, and Abel Gawel. Precise robot localization in architectural 3d plans. *arXiv preprint arXiv:2006.05137*, 2020. 2

[6] Federico Boniardi, Abhinav Valada, Rohit Mohan, Tim Caselitz, and Wolfram Burgard. Robot localization in floor plans using a room layout edge extraction network. *arXiv preprint arXiv:1903.01804*, 2019. 2, 7

[7] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2017. 2

[8] Eric Brachmann and Carsten Rother. Learning less is more-6d camera localization via 3d surface regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4654–4662, 2018. 2

[9] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7525–7534, 2019. 2

[10] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2018. 2

[11] Marcela Carvalho, Bertrand Le Saux, Pauline Trouvé-Peloux, Andrés Almansa, and Frédéric Champagnat. On regression losses for deep depth estimation. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2915–2919. IEEE, 2018. 3

[12] Wenzheng Chen, Jun Gao, Huan Ling, Edward J Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. *arXiv preprint arXiv:1908.01210*, 2019. 1, 4

[13] Mingyu Ding, Zhe Wang, Jiankai Sun, Jianping Shi, and Ping Luo. Camnet: Coarse-to-fine retrieval for camera relocalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2871–2880, 2019. 2

[14] Harlan Hile and Gaetano Borriello. Positioning and orientation in indoor environments using camera phones. *IEEE Computer Graphics and Applications*, 28(4):32–39, 2008. 2

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[16] Henry Howard-Jenkins, Shuda Li, and Victor Prisacariu. Thinking outside the box: generation of unconstrained 3d room layouts. In *Asian Conference on Computer Vision*, pages 432–448. Springer, 2018. 7

[17] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5974–5983, 2017. 2

[18] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 2

[19] Sungyeon Kim, Minkyo Seo, Ivan Laptev, Minsu Cho, and Suha Kwak. Deep metric learning beyond binary supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2288–2297, 2019. 2, 3

[20] Hyon Lim, Sudipta N Sinha, Michael F Cohen, and Matthew Uyttendaele. Real-time image-based 6-dof localization in large-scale environments. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1043–1050. IEEE, 2012. 2

[21] Liu Liu, Hongdong Li, and Yuchao Dai. Efficient global 2d-3d matching for camera localization in a large-scale 3d map. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2372–2381, 2017. 2

[22] Jorge L Martínez, Javier González, Jesús Morales, Anthony Mandow, and Alfonso J García-Cerezo. Mobile robot motion estimation by 2d scan matching with genetic and iterative closest point algorithms. *Journal of Field Robotics*, 23(1):21–34, 2006. 5

[23] Andrea Palazzi, Luca Bergamini, Simone Calderara, and Rita Cucchiara. End-to-end 6-dof object pose estimation through differentiable rasterization. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 1, 4

[24] François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015. 2

[25] Jose-Raul Ruiz-Sarmiento, Cipriano Galindo, and Javier Gonzalez-Jimenez. Building multiversal semantic maps for mobile robot operation. *Knowledge-Based Systems*, 119:257–272, 2017. 5

[26] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019. 2

[27] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3302–3312, 2019. 2

[28] Grant Schindler, Matthew Brown, and Richard Szeliski. City-scale location recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2007. 2

[29] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009. 2

[30] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013. 2

[31] Sinisa Stekovic, Shreyas Hampali, Mahdi Rad, Sayan Deb Sarkar, Friedrich Fraundorfer, and Vincent Lepetit. General 3d room layout from a single view by render-and-compare. In *European Conference on Computer Vision*, pages 187–203. Springer, 2020. 7

[32] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1047–1056, 2019. 5, 6, 7

[33] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002. 2

[34] James Unicomb, Ravindra Ranasinghe, Lakshitha Dantanarayana, and Gamini Dissanayake. A monocular indoor localiser based on an extended kalman filter and edge images from a convolutional neural network. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018. 2

[35] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip HS Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4400–4408, 2015. 2

[36] Helmut Vogel. A better way to construct the sunflower head. *Mathematical biosciences*, 44(3-4):179–189, 1979. 4

[37] Johanna Wald, Torsten Sattler, Stuart Golodetz, Tommaso Cavallari, and Federico Tombari. Beyond controlled environments: 3d camera re-localization in changing indoor scenes. In *European Conference on Computer Vision*, pages 467–487. Springer, 2020. 1

[38] Shenlong Wang, Sanja Fidler, and Raquel Urtasun. Lost shopping! monocular localization in large indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2695–2703, 2015. 2

[39] Yoshiaki Watanabe, Karinne Ramirez Amaro, Bahriye Ilhan, Taku Kinoshita, Thomas Bock, and Gordon Cheng. Robust localization with architectural floor plans and depth camera. In *2020 IEEE/SICE International Symposium on System Integration (SII)*, pages 133–138. IEEE, 2020. 2

[40] Wera Winterhalter, Freya Fleckenstein, Bastian Steder, Luciano Spinello, and Wolfram Burgard. Accurate indoor localization for rgb-d smartphones and tablets given 2d floor plans. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3138–3143. IEEE, 2015. 2

[41] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. *arXiv preprint arXiv:1908.00222*, 2(7), 2019. 4

[42] Wenzhao Zheng, Jiwen Lu, and Jie Zhou. Structural deep metric learning for room layout estimation. In *European Conference on Computer Vision*, pages 735–751. Springer, 2020. 2

LaLaLoc - Supplementary Material

Henry Howard-Jenkins
Active Vision Laboratory
University of Oxford
henryhj@robots.ox.ac.uk

Jose-Raul Ruiz-Sarmiento
Machine Perception and
Intelligent Robotics Group
University of Málaga
jotaraul@uma.es

Victor Adrian Prisacariu
Active Vision Laboratory
University of Oxford
victor@robots.ox.ac.uk

In this document, we provide details about implementation such as network architecture and losses. In addition, we perform additional evaluation to complement the results in the main paper.

1. Implementation Details

This section contains precise details about the implementation of LaLaLoc and the baselines that it is compared against in the main paper.

1.1. LaLaLoc Architecture

We implement Φ_{layout} as the convolutional layers from a ResNet18 [1] network, with the first layer replaced with an equivalent taking a 1-channel input. Φ_{image}, on the other hand, is implemented as a ResNet50 [1] network. In each, we remove the default classifier and replace it with an average pooling layer and a single fully connected layer which takes the embedding dimension to 128d, from 512d or 2048d for ResNet18 and ResNet50, respectively. The embedding vectors are all L2 normalised before any retrieval or comparison task.

1.2. Decoder Architecture

The architecture of the decoder that is used to aid training of Φ_{layout} is defined as follows. First, the latent representation is fed into a fully connected layer which expands it to 2048d. This is reshaped to (2×4) with 256 channels. The up-scaling stage is formed of multiple repeated blocks. Each block comprises of a $2\times$ bilinear up-sample, a 2D convolution with kernel size 3, a ReLU non-linearity, and a BatchNorm layer. The number of out channels from each block follows the pattern $(128 \rightarrow 64 \rightarrow 32 \rightarrow 32)$. Finally, depth is predicted with a point-wise convolution which predicts a depth image. With four $2\times$ up-samples the final resolution of the decoded depth image is (32×64).

1.3. Training Routine

We train Φ_{layout} for a total of 20 epochs, with a batch size of 4. We optimise using SGD, with an initial learning rate of 0.01, and decay by a factor of 0.1 after 10 and 15 epochs. For the log-ratio loss, we follow the positive:negative sampling ratio of 1:20 as used in [2], therefore a single minibatch contains a total of 84 layouts. We define positives and negatives by their spatial distance from the anchor: positives less than 0.5m from the anchor; negatives greater than 2m from the anchor.

Φ_{image} is trained for a total of 200 epochs, with a batch size of 64. Due to the dearth of RGB images, only a single image-layout pair is sampled per iteration. We again use a SGD optimiser with a learning rate set to 0.1. This is then decayed by a factor of 0.1 after 100 and 150 epochs. Φ_{layout} is frozen for the whole training process. Worth noting is that, when Φ_{image} is trained with a variant of ℓ_{log_ratio}, the training routine follows that of Φ_{layout}.

1.4. Image Branch Losses

Here we provide the full equations for the additional image losses evaluated in the ablation in the main paper. First, we restate ℓ_{log_ratio} [2], as used for the layout branch:

$$\ell_{log_ratio}(p,i,j) = \left(log\frac{D(g_i,g_p)}{D(g_j,g_p)} - log\frac{\text{Ch}(C_i,C_p)}{\text{Ch}(C_j,C_p)} \right)^2, \tag{1}$$

where (p,i,j) is a triplet of locations within a floor plan, at which we infer their layouts (L_p, L_i, L_j), $g = \Phi_{layout}(L)$ represents the respective embedding of a layout, C is the back-projection of layout L, $D(\cdot)$ is the Euclidean distance and $Ch(\cdot)$ is the Chamfer distance.

For the training of Φ_{image}, we adapt the loss so that the anchor of the triplet now also has an RGB panorama image $(\{I_p, L_p\}, L_i, L_j)$:

$$\ell'_{log_ratio}(p,i,j) = \left(log\frac{D(g_i,f_p)}{D(g_j,f_p)} - log\frac{\text{Ch}(C_i,C_p)}{\text{Ch}(C_j,C_p)} \right)^2, \tag{2}$$

where the layout embedding g_p, has now been replaced with the embedding of the panorama captured in the same location, $f_p = \Phi_{image}(I_p)$. Therefore, this loss aims to ensure Φ_{image} captures the relative similarities between layouts.

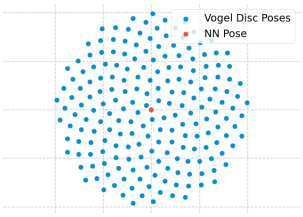


Figure 1. Visualisation of the Vogel Disc sampling pattern with N=200.

We also include a further adaptation, ℓ_{lr_kd}, where we remove ground-truth layout similarities between (p, i, j):

$$\ell_{lr_kd}(p,i,j) = \left(log\frac{D(g_i, f_p)}{D(g_j, f_p)} - log\frac{D(g_i, g_p)}{D(g_j, g_p)} \right)^2, \quad (3)$$

such that Φ_{image} should instead learn to capture the relative similarities between layout embeddings, rather than the layouts themselves. This loss more closely follows typical knowledge discrimination, as the "student" is trained without ground-truth labels.

1.5. Vogel Disc Re-sampling

Our Vogel Disc re-sampling method offers a retrieval-based refinement to the estimated pose from the nearest neighbour in the coarse sampling grid. Specifically, we sample a circular local region centred at the nearest neighbour pose. For N total samples, the ith location is given by:

$$r_i = \sqrt{i/N} \qquad \theta_i = 2\pi(1 - 1/\phi)$$

where ϕ is the golden ratio, r_i and θ_i are the radius and angle in polar coordinates centred at the nearest neighbour pose. The resulting local sampling is visualised in Figure 1.

1.6. Latent Pose Optimisation

For our latent pose optimisation, we render layouts using Redner [4]. We use an Adam optimiser [3], with initial learning rate set to 0.01. The learning rate is scaled by a factor of 0.5 as the loss plateaus with a threshold of 0.05 and a patience of 10 iterations. Convergence is considered reached after 20 steps with reduction in the cost less than a threshold of 0.001, or until 150 steps have elapsed.

1.7. Iterative Closest Point Baseline

In the following, we detail the implementation of our ICP baseline. We emulate a point cloud obtained from a laser scan, C_m, through the back-projection of furnished depth images, and perform matching between those and the 2D slices to the point cloud from the known floor plan, C_f. At each iteration, the points in C_m are assigned to their nearest counterparts in C_f. The rigid transform, $[R, t]$, that aligns

Method	Retr.	Inference Time (s) VDR	LPO	Total
2D-ICP	-	-	-	20.1
LaLaLoc w/o VDR	0.05	-	2.33 (0.85)	2.38
LaLaLoc	0.05	2.87 (2.71)	1.73 (0.63)	4.65

Table 1. Inference time comparison. Times are given as: Total (Render), where "Render" refers to the amount of time spent rendering layouts from the floorplan.

C_m and C_f is determined by minimising a point-to-point cost function given this assignment:

$$E(R,t) = \sum_{i=1}^{|C_f|} \sum_{j=1}^{|C_m|} w_{ij} ||f_i - (Rm_j + t)|| \quad (4)$$

where f_i and m_j stand for points in C_f and C_m respectively, being w_{ij} 1 if f_i was matched with m_j and 0 otherwise, and $||\cdot||$ a distance metric (*e.g.* Euclidean distance). The process of assignment and alignment is repeated until a maximum number of 50 iterations is reached, or the update in translation, t is less than a threshold of 0.01mm for 3 successive iterations.

To localise in a room, we initialise the ICP alignment at each of the sampled grid of poses and take the result as the resulting alignment that has the lowest RSME error between point clouds. A typical concern when dealing with this iterative process is its execution time. To handle this, we performed a grid-average sampling method that merges points in the same grid cell, thus preserving the shape of the point clouds but reducing execution time. The grid cell side was heuristically fixed to 5cm.

2. Additional Experiments

In this section, we provide some additional experimental evaluation to complement the results in the main paper.

2.1. Time Complexity

In Table 1 we list the inference times. We omit the rendering time of the sampled grid in the retrieval stage as this time cost is shared between the methods and can be computed offline. In addition, the rendering during VDR could be optimised (currently 2.7s mean). We see that LaLaLoc offers a significant speed advantage over ICP. Interestingly, VDR appears to reduce the time required for LPO since it provides a better initialisation for LPO, reducing the mean number of iterations to convergence, 41.7 down to 31.4.

2.2. Layout Similarity Low-resolution Sampling

Here, we re-perform our evaluation of differing layout representations and similarity metrics for localisation on a lower resolution sampling grid. The results are listed in Table 2. As can be seen, the lower resolution sampling only

Similarity Metric	Recall @1	Pose Error Median (cm)	Correct Room
Pose	*100%*	*40.1*	*99.7%*
Edges	55.6%	48.7	59.2%
Depth	55.2%	48.3	59.5%
Rel. Depth	54.1%	48.5	61.6%
Chamfer	**71.8%**	**42.7**	**74.4%**

Table 2. Evaluation of layout similarity metrics on the Structured3D validation split, now sampled with a lower-resolution 1m $\times$ 1m grid. *Pose* refers to picking the nearest location in the sampled grid to the query.

serves to exaggerate the performance improvement seen by using the 3D Chamfer distance similarity metric, further emphasising the need for careful selection.

References

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

[2] Sungyeon Kim, Minkyo Seo, Ivan Laptev, Minsu Cho, and Suha Kwak. Deep metric learning beyond binary supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2288–2297, 2019.

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	LaLaLoc: Latent Layout Localisation in Dynamic, Unvisited Environments
Publication Status	☑Published □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	**Howard-Jenkins, H.**, Ruiz-Sarmiento, J.-R., and Prisacariu, V. A. (2021). LaLaLoc: Latent Layout Localisation in Dynamic, Unvisited Environments. In *International Conference on Computer Vision (ICCV)*.

Student Confirmation

Student Name:	Henry Howard-Jenkins		
Contribution to the Paper	• Conception and validation of the idea • Design of the architecture and framework • Implementation and running of all experiments, aside from the ICP baseline experiments. • Research of prior work • Writing and presentation of the paper		
Signature	HHJ	Date	31/01/2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Prof Victor Adrian Prisacariu		
Supervisor comments	The description is accurate		
Signature	VPA	Date	09/02/2022

This completed form should be included in the thesis, at the end of the relevant chapter.

4.3 Conclusion

Within this chapter, we presented LaLaLoc to enable localisation of panorama images with respect to a floor plan prior. Rather than taking an approach of explicit room layout prediction, matching and alignment, LaLaLoc performs each *implicitly*. We demonstrated that LaLaLoc is able to learn a rich and expressive latent space for room layouts, and that this latent space can be shared between panorama images and renderings of specific locations within a floor plan. We further showed how the panorama can be finely aligned to the floor plan with an optimisation of a cost defined solely in the latent space. Through the experimental evaluation performed in the paper, it was shown that the implicit formulation proposed by LaLaLoc outperforms an explicit counterpart where layout is explicitly predicted, using HorizonNet [Sun et al., 2019], and also aligned to renderings of layout from the floor plan prior.

To some degree, ambiguity cannot be completely alleviated when localising with respect to floor plan alone. However, we showed that, at least with known orientation, room layout similarity is expressive enough to be able to localise with a good accuracy. Interestingly, it was found that the choice of metric for layout similarity has a significant impact on the localisation accuracy that can be achieved by such a method.

While LaLaLoc offers inference speed-up over the ICP baseline, inference is still slow in an absolute sense, taking multiple seconds for each query. This is due to the rendering stage of the explicit 3D layout representation being relatively slow and also required multiple times for both retrieval and the fine-grained alignment. In addition, the creation and rendering of this explicit representation of the scene requires the ceiling and camera height to be known. Therefore, the rendering of an explicit scene representation is not only a bottleneck for the inference speed, it imposes a practical limitation on the density at which the layout can be inferred from the floor plan, and forces assumptions to be made about the scene. In the following chapter, we explore how these limitations can be removed.

To summarise the work in this chapter, we presented LaLaLoc, a localisation method which localises RGB queries to a known floor plan by matching in a latent space that encodes layout similarity. We further leverage this expressive latent space for direct pose optimisation through differentiable rendering. We show that LaLaLoc is able to localise with considerable accuracy in unseen or highly dynamic environments.

Chapter 5

Structural Comprehension From 2D Floor Plans

5.1 Introduction

With LaLaLoc, we presented a method for localisation in an indoor environment using architectural structure as the only prior. This task was motivated by a desire to localise in previously unseen environments, as well as demonstrating ability to localise in environments despite scene dynamics, such as a change in furniture configuration. However, in order to process a floor plan prior, LaLaLoc required that the plan first be used to construct an explicit 3D representation of the scene. This step was not only expensive, particularly when the rendering was performed differentiably, limiting the density of locations across the floor plan where the structure is inferred, but it also meant that assumptions about the height of the ceiling and camera had to be made during the extrusion and rendering.

In this chapter, we address the limitations introduced by the requirement of an explicit 3D model produced from the floor plan. We present LaLaLoc++, which learns a general prior over 2D floor plans which allows us to implicitly hallucinate 3D structure across the entire plan in a single pass, and without the need for the intermediate conversion into an explicit 3D representation.

5.1.1 Contributions

- We present a floor plan comprehension module, which learns to implicitly hallucinate 3D visible structure from 2D floor plans. This removes the need for explicit extrusion with an assumed or known camera height.

- We design a panorama embedding module that leverages a transformer encoder to better capture room layout from an RGB panorama.
- We demonstrate that, when combined, these modules are able to significantly advance the state-of-the-art for floor plan localisation both in terms of accuracy (over 37% and 29% improvement on Structured3D [Zheng et al., 2020a] and Zillow Indoor Dataset [Cruz et al., 2021], respectively) and speed (35x speed up).

LaLaLoc++: Global Floor Plan Comprehension for Layout Localisation in Unvisited Environments

Henry Howard-Jenkins and Victor Adrian Prisacariu
Active Vision Laboratory
University of Oxford
{henryhj, victor}@robots.ox.ac.uk

Abstract

We present LaLaLoc++, a method for floor plan localisation in unvisited environments through latent representations of room layout. We perform localisation by aligning room layout inferred from a panorama image with the floor plan of a scene. To process a floor plan prior, previous methods required that the plan first be used to construct an explicit 3D representation of the scene. This process requires that assumptions be made about the scene geometry and can result in expensive steps becoming necessary, such as rendering. LaLaLoc++ instead introduces a global floor plan comprehension module, which is able to efficiently infer structure densely and directly from the 2D plan, removing any need for explicit modelling or rendering. On the Structured3D dataset this module alone improves localisation accuracy by more than 31%, all while increasing throughput by an order of magnitude. In addition, we introduce a transformer-based embedding module which can more accurately capture long-range layout structure across the panorama, improving accuracy by a further 8%. Combined, these improve accuracy on the state-of-the-art method by over 37% with dramatically faster inference.

1. Introduction

Floor plans are ubiquitous in the built, indoor environment. For almost any modern building, these structural plans are stored by local government, builders, real estate agents and even as fire-safety maps and other guides to indoor environments. The prevalence of these documents across domains is not an historical fluke, but instead because they have a number of very useful characteristics. *Permanence*, while furniture and objects present within an environment may move or change, the structure represented in a floor plan remains the same, and therefore these documents are able to provide a description of the environment

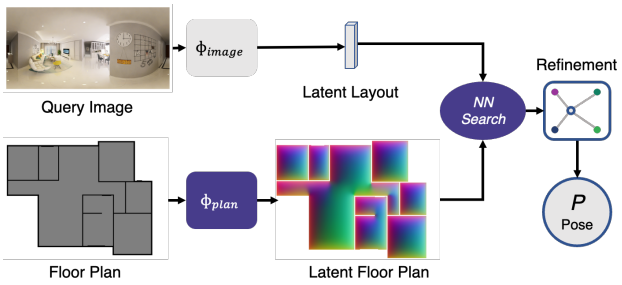


Figure 1. An overview of our proposed method for floor plan localisation. LaLaLoc++ takes as input a 2D floor plan from which it infers 3D structure across the free space of the plan. To localise, an image branch is trained to infer layout structure from the query image. Pose is predicted by aligning the image's latent structure to the globally inferred structure.

over a period of years and decades. *Expressiveness*, the structure depicted in a floor plan gives the reader a good basis for understanding that particular indoor environment: in a fire safety map, one is able to bridge the gap between their egocentric view and the plan to determine an exit route; when viewing a property online, one is able to imagine living in the space illustrated by the top-down plan. *Convenience*, these blueprints are extremely lightweight format for expressing this structure, allowing them to be easily stored and interpreted across physical and digital formats.

It is for these reasons that, in this paper, we turn our attention to direct interpretation of structural blueprints, specifically in the context of floor plan localisation. Visual relocalisation is the task of estimating the pose of a camera given some prior knowledge of an environment. Commonly in localisation methods, the prior takes the form of a built 3D structural point-map, *e.g.* [22], a series of stored images, *e.g.* [1, 2], or in the weights in a neural network, *e.g.* [4]. Floor plan localisation, however, takes the prior as an architectural plan of the indoor environment.

Specifically, we present LaLaLoc++ which is able to perform floor plan localisation directly from 2D plans. Previ-

ous methods for floor plan localisation have relied on translation of the architectural floor plan into an explicit 3D geometry, either relying on the assumption of known [14], or manually-estimated [9] ceiling and camera heights to extrude the plan into the 3D representation. LaLaLoc++, however, learns a general prior over 2D floor plans which allows us to implicitly hallucinate 3D structure across the entire plan in a single pass, and without the need for the intermediate conversion into an explicit 3D representation. This is depicted in Fig. 1.

The contributions of this paper can be summarised as follows:

- We present a floor plan comprehension module, which learns to implicitly hallucinate 3D visible structure from 2D floor plans. This removes the need for explicit extrusion with an assumed or known camera height.
- We design a panorama embedding module that leverages a transformer encoder to better capture room layout from an RGB panorama.
- We demonstrate that, when combined, these modules are able to significantly advance the state-of-the-art for floor plan localisation both in terms of accuracy (over 37% and 29% improvement on Structured3D [33] and Zillow Indoor Dataset [9], respectively) and speed (35x speed up).

2. Related Work

Camera relocalisation has seen a vast collection of research progress, inspiring a diverse set of problem settings and methods to solve them. Generally, visual localisation seeks to make associations between query image and prior knowledge of the environment in order to predict the capture location. However, the type and construction requirements of this prior offers a variety of design choices. A large portion of methods offer prior construction that is generalisable across scenes. Structure-based methods [18, 19, 22] establish correspondences by matching points between the query image and an explicit 3D representation built using a method such as SfM [12, 24]. Retrieval-based methods [1, 2, 10, 23] primarily estimate pose by matching the query image to its most similar counterpart in a database. On the other hand, other methods require that a new prior be learned for each environment in which localisation is performed, usually learning an implicit representation of the scene within neural network weights. Absolute pose regression methods [8, 15, 16] train a deep neural network to directly regress the camera pose. Scene-coordinate regression methods [4–6, 25, 29] densely regress global 3D coordinates across image locations. These generalisable and scene-specific methods share a common attribute, which is that, although the form their prior takes differs, they require a training sequence of images in order to construct or learn it. In this paper, however, we approach the task of floor plan localisation, which takes only a floor plan as its prior. Therefore, the need for a training set of images to enable localisation in a new environment is alleviated.

Like other forms of visual localisation, floor plan-based methods can take many forms. The task has seen particular interest in the context of robotics, where methods operate by forming a room layout observation model between a location in the floor plan and query by extracting layout edges [3, 28, 31], layout corners [13] or aggregating depth [32] from which an observation likelihood can be estimated. Further, if depth information is available, scan-matching techniques [20] estimate an alignment between the observed depth and the floor plan. These methods are generally not standalone, instead relying on multiple sequential measurements used as hypothesis weightings for a Monte Carlo Localisation [27] frameworks. In this paper, however, we approach the task as defined by LaLaLoc [14], which is also the most similar method to this work. This localisation formulation, unlike other methods, aims to localise a panorama image within a floor plan without depth information, motion or time-coherency cues, and without assuming a good initialisation. Notably, Cruz *et al*. [9] construct a similar layout-based localisation task for panorama registration, however, they aim to localise with respect to other panoramas captured in the scene, rather than in an unvisited location.

More specifically, LaLaLoc [14] learns a latent space to capture room layout [17]. This latent space allows LaLaLoc to relate the layout seen in a panorama with that computed by rendering an explicit model of the environment. Localisation is performed by retrieving nearest neighbours between a database consisting of embedded layouts from sparsely sampled poses within the floor plan and the embedding of the panorama image. This coarse estimate of pose from retrieval is then refined via a gradient-based optimisation of pose to minimise the distance between layout and query embeddings, enabled through differentiable rendering. However, the model of the scene used by LaLaLoc had to be created by extruding the floor plan in the z-dimension, assuming a known camera and ceiling height. In addition, LaLaLoc's formulation requiring rendering across the plan for retrieval, and multiple (differentiable) rendering steps for pose refinement is slow. In this work, we remove these limitations of LaLaLoc through the introduction of direct and dense structural comprehension of the 2D plan. In doing so, LaLaLoc++ is able to dramatically improve accuracy and inference speed, all while removing the need for height assumptions in the method.

3. Method

LaLaLoc++ consists of two parallel branches which operate across data modalities. An overview of its architecture is depicted in Fig. 1. The floor plan comprehension module, Φ_{plan} , infers the layout structure that is visible across a 2D floor plan, computing a latent vector at every location in the $x - y$ plane that covers the scene. The image embedding module, Φ_{image} , on the other hand, aims to capture the layout that is visible within a cluttered, RGB panorama image. The latent space to which they both map their respective input data is shared, so that a panorama captured at a location and embedded by Φ_{image} should have the same latent vector as that location sampled from Φ_{plan} 's output. Therefore, LaLaLoc++ performs localisation of a query panorama by finding the location in the floor plan that most accurately matches the visible room layout, with matching cost being computed in the layout latent space shared between branches. In the following, we detail the task, the form that our architecture takes, how it is used to localise, and how it is trained.

3.1. Floor Plan Localisation

We follow the floor plan localisation task as used in LaLaLoc [14]. This is the task of performing localisation of a panorama within a scene which has not been visited previously, given only a floor plan as prior. Specifically, we predict the 2 DoF camera pose to an $x - y$ location in the 2D floor plan assuming a known orientation, an assumption we explore in Sec. 5.6. The floor plan consists of a 2D image which denotes an empty scene consisting of only walls, floors.

3.2. Φ_{plan} : Global Latent Floor Plan

Given a 2D floor plan of an environment, we wish to infer the architectural structure of the scene at any given location within it. In order to achieve a similar objective, LaLaLoc [14] formed an explicit representation of the scene in the form of a mesh using a known ceiling height. This explicit geometry could then be rendered as a depth image at a given location, given a known camera height, from which the implicit layout representation could be computed by the LaLaLoc's Φ_{layout} . While shown to be effective, this method proves to be slow, requiring multiple rendering steps and forward passes in order to form only a sparse map of layout across the floor plan, and also requires that camera and ceiling heights are known.

Instead, with LaLaLoc++ we propose that the implicit representation of layout can be inferred directly from the floor plan itself. This formulation allows us to walk well-trodden ground in architecture design for fast, efficient, and *dense* inference of layout across the floor plan.

Specifically, Φ_{plan} takes the form of a UNet [21] inspired encoder-decoder structure, which takes as input a 2D structural floor plan and outputs a dense feature map. The vector at each location is a latent representation of the layout visible there. Therefore, in just a single forward pass we are able to build a dense expression of layout within a floor plan. This structure is depicted at the top of Fig. 2.

3.3. Φ_{image} : Image Embedding with Long-range Attention

While forming an expressive latent space for layout inferred from a floor plan was shown to be readily achievable to a high degree of effectiveness in LaLaLoc [14], the difficulty of (implicitly) predicting layout from RGB panoramas proved to be a significant source of error. We therefore propose an improved image embedding module based on a transformer encoder [30]. This module allows the image branch to better capture long-range structural patterns present in the query panorama image.

We implement a transformer encoder to refine the feature-map computed by the image branch backbone, which is a ResNet50 network [11]. The features outputted by the backbone are linearly projected from 2048d to 128d. These projected features are then fed into the transformer encoder, with 2 encoder blocks before finally being pooled and projected to form the image embedding for a panorama. The multi-headed self-attention present in the encoder blocks encourages long-range attention. This architecture is visualised in Fig. 3.

3.4. Localisation

At inference time, a dense reference grid of latent layout vectors are computed by passing the 2D floor plan. An initial estimate of position is computed as the location corresponding to the nearest latent layout in this grid. However, since this is still a spatially discretised grid across the scene, we introduce further alignment in the form of pose optimisation allowing for a continuous estimate of position. We describe this more concretely in the following.

We denote the floor plan encoded by Φ_{plan} as $\mathbf{G}$, and the query latent layout computed by Φ_{image} as f. The initial position estimate, p_0, is taken as whole pixel location, *i.e.* $p_0 = (i_0, j_0) \in \mathbb{Z}^2$, with the lowest matching cost $|f - \mathbf{G}[p_0]|_2$, where $[\cdot]$ is a look-up function. This location is used to initialise the refinement.

We then perform the continuous refinement of pose through interpolation of the local neighbourhood of latent layout representations in the encoded floor plan. Position is initialised at the nearest location in the feature grid. During refinement, the position estimate is relaxed to allow sub-pixel locations, $p_r \in \mathbb{R}^2$, and $[\cdot]$ is extended to use sub-pixel interpolation. The refined pose is iteratively updated to minimise the matching cost in a gradient-based optimisation scheme,

$$\min_{p_r} |f - \mathbf{G}[p_r]|_2. \tag{1}$$

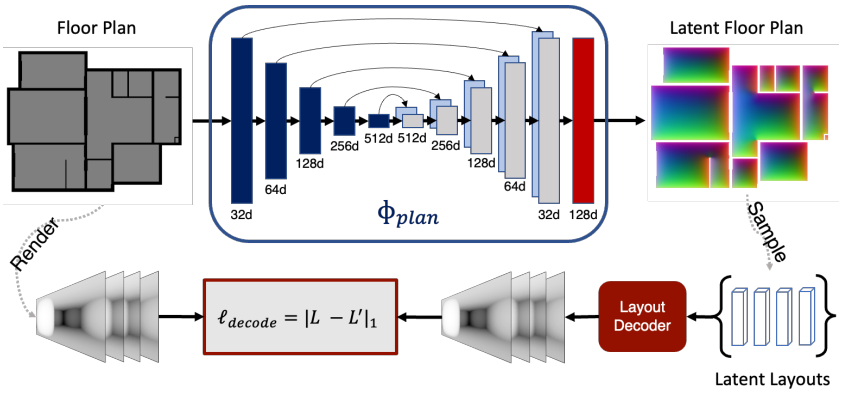


Figure 2. Φ_{plan} , the floor plan comprehension network and its training routine. The network takes as input a 2D plan of the scene and outputs a dense grid of features which capture the 3D structure visible at their respective locations. Feature maps within the UNet structure are depicted with spatial dimensions unrolled for simplicity, and their channel dimensions are listed.

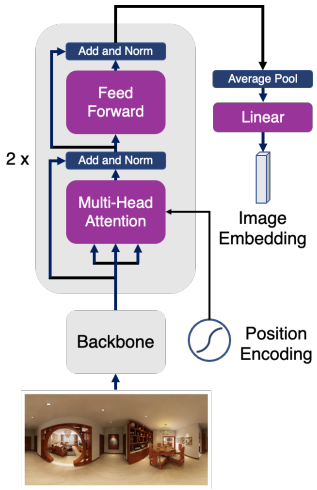


Figure 3. Depiction of the transformer-based image embedding module. The panorama is fed into the ResNet backbone and the resulting feature map is unrolled and fed into a transfomer encoder architecture where long-range structural patterns can be correlated across the image. The resulting features are pooled and projected to get the final embedding.

3.5. Training

LaLaLoc++ follows an analogous training philosophy to that proposed in LaLaLoc [14] where the network is trained in two separate stages. In the first stage, we train Φ_{plan} alone. Since Φ_{plan} takes a form of explicit structure as input, its training is used to establish an expressive latent space for capturing the room layout at any given location in the scene. This latent space is then frozen and, in the second stage, Φ_{image} is trained to map images to this fixed representation space.

3.5.1 Φ_{plan} : Learning the Latent Space

We train Φ_{plan} with a single loss, where the layout embedding at a given location is passed through a simple layout depth prediction decoder and compared with a rendered layout at the same location:

$$\ell_{decode} = |L' - L|_1, \tag{2}$$

where L' is the predicted layout, and L is the ground-truth rendering at a given location. During training, we use known camera and ceiling heights for the rendering of the target layout, however, the floor plan comprehension module is forced to infer the structure without these parameters. Therefore, it must learn a generalised prior for the hallucination of 3D structure from the top-down 2D input. Illustration of how this loss is computed can be found in Fig. 2.

3.5.2 Φ_{image} : Mapping to the Latent Space

The training of Φ_{image} takes the form of a simple strategy, which reflects its role in the localisation method. Given a training batch of panorama images, their respective poses, and a floor plan, the global latent floor plan is computed by Φ_{plan} and sampled at the panorama poses. The panorama images are then embedded by Φ_{image} , and a loss is applied to the difference between the embeddings produced by Φ_{image} , f, and those sampled from Φ_{plan} 's output, g, which is defined as follows:

$$\ell_{L2} = |f - g|_2. \tag{3}$$

4. Implementation Details

In this section we describe the specific nature of the task evaluated in the experiments section of the paper. We start with by detailing the datasets, followed by the configuration of our method. The precise training schedule can be found in Appendix A.1.

4.1. Datasets

The bulk of evaluation experiments are performed on the Structured3D dataset [33], which consists of 3,500 photorealisticly-rendered synthetic indoor scenes. We follow the split of data used in LaLaLoc [14], which itself follows the dataset's predefined split with some scenes excluded due to corrupted data. This leaves 2979/246/249 scenes for training/validation/testing, respectively. Every scene comprises of multiple rooms, each containing a single panorama image which is used as a query. Since there are no reference images captured from within the same scene, we label each scene as *unvisited*, as there is no RGB data that can be leveraged for training or database building and therefore the method must rely on floor plan data alone. There are three furniture configurations, empty, simple and full, as well as three lighting configurations for each scene, cold, warm and raw. During training, we randomly sample a furniture and lighting configuration at each iteration. At test time, we perform evaluation in the full and warm setting. This is the most difficult furniture configuration as there are the most possible distractors from layout present in the image.

We also explore how this method extends to real data by performing evaluation on the recently released Zillow Indoor Dataset [9]. We processes the dataset to match the problem setting on the Structured3D dataset, namely considering localisation across a single floor and alignment of panorama images. This results in 1957/244/252 scenes for training/validation/testing, respectively. However, the dataset only provides a single lighting and furniture configuration for each panorama. The remaining implementation details apply across both Structured3D and the Zillow Indoor Dataset.

4.1.1 Floor plans

We represent the 2D floor plans as image. Each pixel location can take three possible values: 1 represents a wall, 0.5 is used to denote free-space within the floor plan, and 0 is assigned to free space outside of the floor plan. To ensure that absolute scale is preserved in the creation of these plans, we use a consistent scale across all scenes in the dataset. It is set such that one image pixel covers 40mm of space in reality. This value was set heuristically as a balance between keeping the overall size of the floor plan manageable, and ensuring that all walls are still discernible. At the most compact representation in Φ_{plan}, the spatial resolution of the feature map is 1/32th of the original height and width, we zero-pad the floor plans so that both height and width are divisible by 32.

4.2. Network Architecture

As depicted in Fig. 2, Φ_{plan} consists of 5 down-sample and 5 up-sample layers. These layers each consist of a convolutional block which contains a 3x3 convolution, a batch-norm and a ReLU activation repeated twice. In the down-sample block, the convolutional block is followed by a 2x2 max-pooling layer. In the up-sample block it is preceded by a 2x2 bilinear up-sampling of the feature map. This up-sampled feature map is then concatenated with the feature map from the corresponding down-block (taken before pooling), to form skip connections. The output of the final up-sampling layer is passed through a final 3x3 linear convolution to form the final latent floor plan. This latent floor plan keeps the same spatial resolution as the original input floor plan with a 128d descriptor for each location. This descriptor is then normalised such that each one is unit length. The layout decoder used in training takes an identical structure to that used in [14].

Φ_{image} takes the form of a ResNet50 backbone. In its simplest formulation, the feature map outputted by this backbone is pooled to a single vector, then linearly projected to 128d and normalised. However, in our transformer formulation, a transformer encoder forms an intermediate refinement of the image feature map before pooling and projecting. More specifically, a 2D sinusoidal positional encoding [30] is generated for each spatial location in the feature map. The feature map is then flattened, linearly projected to 128d, and fed into a vanilla transformer encoder [30] with two repeated blocks. There are 8 attention heads, and the feed-forward hidden dimension is set to 256 dimensions. We apply the positional encoding at the attention layers, rather than input, following [7]. The encoded features are then pooled and projected once more before being normalised to form the image embedding.

5. Experiments

In this section, we perform experimental evaluation of the method as described in the previous section. Initially, we explore the performance of the method as a whole. We then explore the constituent components and their impact on the final accuracy, before finally performing ablation experiments to validate the design of these components. Unless otherwise stated, experiments are performed on the Structured3D [33] dataset.

5.1. Floor Plan Localisation

We first perform localisation on the test split of the Structured3D [33] dataset. We compare the performance of LaLaLoc++ to three other methods, LaLaLoc itself, as well as the 2D-ICP and HorizonNet [26]-based baseline presented in LaLaLoc [14]. Results are listed in Tab. 1.

As can be seen in Tab. 1, LaLaLoc++ is able to localise

Method	Median (cm)	Localisation Accuracy <1cm	<5cm	<10cm	<1m
ICP	21.8	**9.5%**	26.4%	35.6%	68.5%
HorizonNet [26] + Loc	9.1	3.3%	29.3%	53.4%	77.4%
LaLaLoc [14]	8.3	3.6%	32.0%	58.0%	87.5%
LaLaLoc++	**5.2**	5.4%	**48.8%**	**72.3%**	**92.3%**

Table 1. Localisation performance of LaLaLoc++ compared against three other floor plan localisation methods on Structured3D.

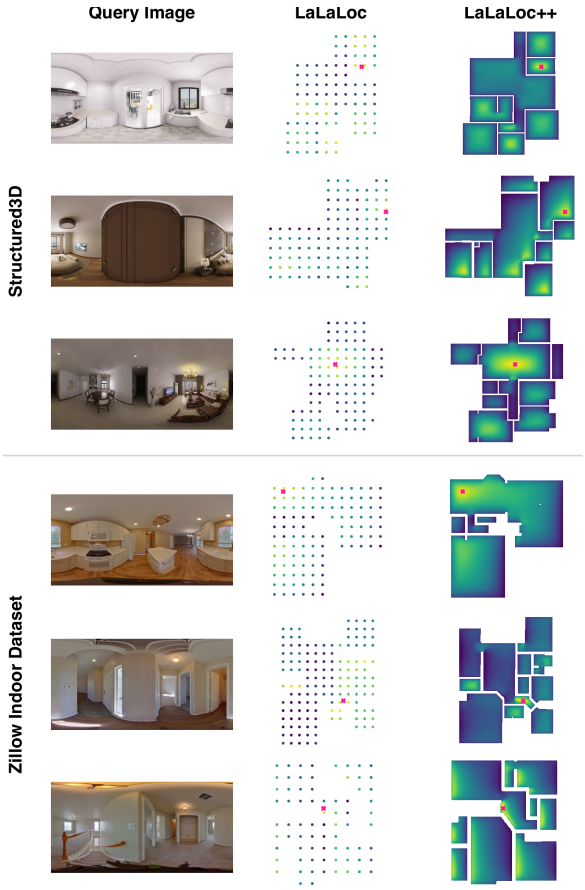


Figure 4. Qualitative comparison of the floor plan comprehension differences between LaLaLoc++ and LaLaLoc. Visualised as a plot of the latent distance between the query image and embeddings across the floor plan in each of the two methods. While LaLaLoc++'s Φ_{plan} is able to infer layout densely within the scene, LaLaLoc instead just has a sparse sampling (in this case every 0.5m in each direction). The difference in the resulting understanding of the environment is stark.

in these unseen environments significantly more accurately than the original LaLaLoc. The median localisation accuracy represents a 37% improvement over LaLaLoc. When considering the accuracy thresholds, it is notable that

Method	Med.	Localisation Accuracy <1cm	<5cm	<10cm	<1m
LaLaLoc	13.1	1.3%	19.9%	40.9%	77.6%
LaLaLoc++	**9.3**	**2.5%**	**32.1%**	**51.7%**	**80.7%**

Table 2. Localisation performance of LaLaLoc++ compared against LaLaLoc with real data on the Zillow Indoor Dataset.

Method	Inference Time (s) Embedding & Retrieval	Refinement (Rendering)	Total
2D-ICP	-	-	20.1
LaLaLoc w/o VDR	0.05	2.33 (0.85)	2.38
LaLaLoc	0.05	4.60 (3.34)	4.65
LaLaLoc++	**0.01**	**0.12**	**0.13**

Table 3. Inference time comparison. Times are displayed as: Total time (Render time). Each of these timings is computed on a single Nvidia Titan RTX GPU.

LaLaLoc++ not only is able to localise more images to the fine-grained accuracy thresholds, but is also able to localise more images to within 1m (87.5% *v.s.* 92.3%). This suggests that LaLaLoc++ reduces the frequency of catastrophic failure to localise. We visualise some failure cases in Appendix A.2.

We further investigate how this performance extends beyond synthetic data into the real world captures on the Zillow Indoor Dataset [9]. We report these results in Tab. 2. Inline with results on Structured3D, LaLaLoc++ offers significantly improved accuracy over LaLaLoc (29% increase). The real world layout distributions appear to pose a harder challenge, with more complex floor plan geometry. However, this result does show that the method can extend to real data.

Figure 4 illustrates the contrast in prior-forming between LaLaLoc and LaLaLoc++. The dense prediction of Φ_{plan} is able to far more expressively capture the layout across the floor plan, whereas the sparse strategy to form reference embeddings in LaLaLoc is comparatively low in fidelity. Although LaLaLoc could sample reference poses

Method	Localisation Accuracy				
	Med.	<1cm	<5cm	<10cm	<1m
LaLaLoc	8.3	3.6%	32.0%	58.0%	87.5%
Mimic	8.4	1.8%	27.3%	56.8%	88.5%
Decoder Loss	**5.7**	**4.9%**	**45.0%**	**69.4%**	**92.0%**

Table 4. Contribution of Latent Floor plan training to the final localisation accuracy. Identical image branch architecture is used in each scenario. *Mimic* is used to refer to a training scheme where the global floor plan network is trained to copy LaLaLoc's layout branch output, and then is substituted in its place with no other fine-tuning to it or the image branch.

more densely, inferring at the same level of density with LaLaLoc quickly become intractable.

In Tab. 3, we compare the time complexities of LaLaLoc++ against LaLaLoc. LaLaLoc++ is able to offer more than a 35x speed up over LaLaLoc. This is predominantly for two reasons. Firstly, LaLaLoc++'s direct inference from 2D plans means that expensive rendering steps can be removed. Notably, this also includes the expensive backwards passes in the differentiable rendering optimisation scheme. Secondly and more subtly, LaLaLoc required a forward pass through a ResNet18 [11] for each rendered layout. Although this could be performed in batches, depending on the density of sampling the total number of passes could reach hundreds. On the other hand, LaLaLoc++ allows dense inference of structure in a single forward pass, which proves to be significantly faster.

5.2. Global Floor Plan Comprehension

To evaluate the contribution of our proposed global floor plan processing, we perform localisation using various configurations of floor plan branches all with equivalent image branch architectures. The image branch architecture is chosen to be identical to the one proposed in LaLaLoc. The first comparison is to LaLaLoc's layout branch and its scheme of sparse floor plan rendering and embedding. We then train our proposed global floor plan module as a direct substitution for the layout embedder in LaLaLoc. Therefore, the floor plan network is trained to mimic the output of LaLaLoc's layout branch, and the original image branch is used without any modification or fine-tuning. We call this "Mimic". Finally, we compare to our proposed method, where the Φ_{plan} is trained with the layout decoding loss, is frozen, and then Φ_{image} is trained to map panoramas to the learned space.

Results for these experiments can be found in Tab. 4. Worth considering first is the results for the "Mimic" configuration. This experimental setup can be considered a test of whether 3D structure can be directly inferred from the 2D layout to a similar degree of expressiveness as the original LaLaLoc's render and embed methodology. The results show that the module is in fact able to nearly match the original performance of the network for which it is substituting. This is particularly notable as the 2D comprehension module is not given height information (neither ceiling nor camera) and instead must learn a generalised 2D to 3D room structure prior which holds for these unseen environments. As such, there is an inherent degree of inaccuracy expected from this formulation as it must replicate a method while dealing with significantly more ambiguity.

When trained to form its own layout latent space in the "Decoder Loss" setting, the performance improvement over LaLaLoc is dramatic. This is likely because the latent space for layouts can be constructed in a way compatible with the structure that can be determined from a 2D floor plan. In Appendix A.3, to aid in the intuition of the learned space, we visualise the output of the decoder. In addition, the dense estimation of layout may provide a better setting for pose refinement than the Latent Pose Optimisation proposed in LaLaLoc, which relies on differentiable rendering. We investigate this in Sec. 5.3.

5.2.1 Floor Plan Saliency

In the explicit rendering formulation used in LaLaLoc, the structure rendered at any given point in the scene has exactly the same level of context as the query image with which you wish to compare it to, *i.e.* only the visible structure from that location. However, when processing the entire 2D floor plan, the receptive field of the network for structural inference at any given location quickly moves beyond what is strictly visible from that location. If this global information is leveraged in forming the layout descriptor from the floor plan, then it is likely that the image branch will not be able to map to the same latent space.

To investigate this, we plot saliency of the global floor plan structure in computation of the descriptor at any given location. The saliency is computed by masking a sliding window across the floor plan such that any walls in this window are instead assigned the value denoting free-space inside the scene. We then compare how a descriptor at a chosen location in the plan changes compared to that computed with the unmasked floor plan via an L2 distance computation. This distance is assigned to the centre point of the sliding window to form a saliency map across the plan.

Some examples of these saliency maps are plotted in Fig. 5. It is apparent that the floor plan module does not unduly exploit global structure, but instead the structure local to the test point is the most salient regions of the plan.

5.3. Sub-pixel Refinement

In this section, we wish to investigate the performance properties of the proposed sub-pixel floor plan optimisation process. We perform refinement initialised with retrieval

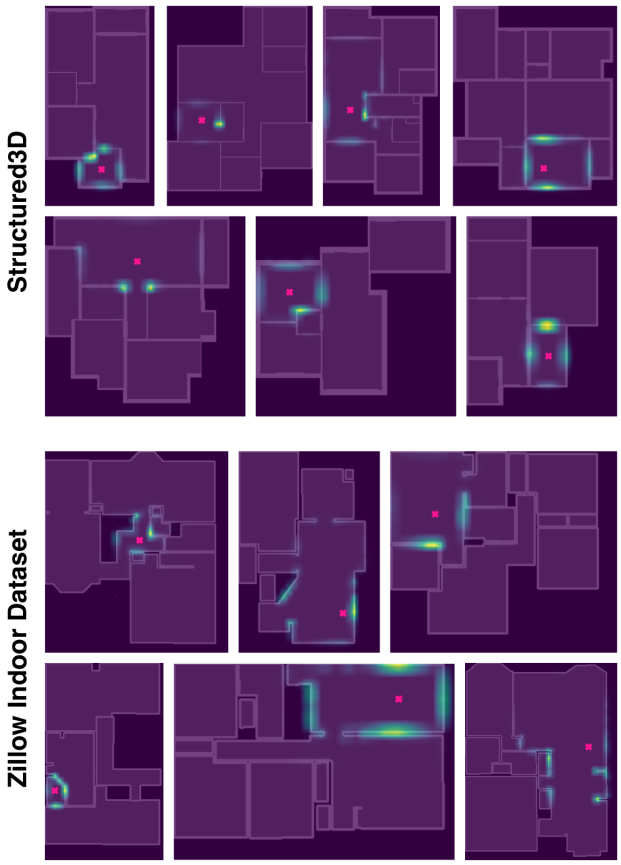


Figure 5. Investigation into floor plan saliency for inferring layout at a given location from a global floor plan. The query location is marked with a pink cross, and computed saliency is overlaid onto the floor plan. The floor plan comprehension module demonstrably learns to differentiate between visible and invisible structure for the query locations when computing their descriptor.

Floor plan Subsampling	Init.	Localisation Accuracy Med.	<1cm	<5cm	<10cm	<1m
LaLaLoc LPO	22.5	10.5	2.0%	27.4%	48.4%	87.5%
LaLaLoc	11.0	8.3	3.6%	32.0%	58.0%	87.5%
8x	28.4	6.5	4.7%	41.3%	64.3%	87.2%
4x	15.1	5.3	5.7%	47.7%	71.4%	92.1%
2x	8.8	5.3	5.3%	48.0%	69.9%	92.1%
Full-size	6.1	5.2	5.4%	48.8%	72.3%	92.3%

Table 5. Exploration of the Latent Floor plan resolution on the localisation performance. *Init.* refers to the median accuracy of the initialisation for refinement, reported in cm.

results from increasingly sparse representations of the floor plan. In each of these experiments the floor plan is embedded at full-resolution, but the resulting feature maps are sub-sampled by factors of 2. This has the effect of testing the refinement performance with wider baseline initialisation. For each level of sub-sampling, we re-train the image branch. We also include results for LaLaLoc in its operation mode where the latent pose optimisation is the only pose refinement after retrieval, thus skipping the local retrieval normally performed, and in the configuration with local re-sampling and continuous refinement.

The results for this are listed in Tab. 5. The median error of the refinement initialisation is included for each method and formulation. It can be seen that LaLaLoc++, and specifically its continuous pose refinement through interpolation, is able to recover extremely well from increasingly poor initialisation. Even subsampling the latent floor plan by 8x, LaLaLoc++ still significantly outperforms LaLaLoc by 1.8cm, despite the initialisation being nearly 3x worse.

Method	Localisation Accuracy Med.	<1cm	<5cm	<10cm	<1m
ResNet50	5.7	4.9%	45.0%	69.4%	92.0%
R50 + Transformer	**5.2**	**5.4%**	**48.8%**	**72.3%**	**92.3%**

Table 6. Comparison of the image embedding architecture that is used. ResNet50 denotes a simple backbone followed by average pooling and a linear projection. R50 + Transformer is the proposed formulation where the image feature map is refined with the self-attention mechanisms present in a transformer encoder.

Φ_{plan} # Layers	Localisation Accuracy Med.	<1cm	<5cm	<10cm	<1m
3	12.9	2.5%	27.4%	44.8%	70.0%
4	6.0	4.0%	44.0%	65.4%	90.5%
5*	**5.2**	**5.4%**	**48.8%**	**72.3%**	**92.3%**
6	5.3	5.1%	48.0%	70.4%	92.1%

Table 7. Localisation with differing numbers of down and up-sampling layers in Φ_{plan} . * denotes the chosen configuration.

5.4. Image Transformer

We show a localisation performance comparison between the image embedding structure consisting of a simple feature backbone, pooling and projection and our proposed transformer formulation. Results are listed in Tab. 6. As can be seen, the transformer leads to a nearly 9% improvement in localisation accuracy over the baseline. This result suggests that the long-range attentional mechanisms introduced by the transformer encoder allow the embedding module to better capture layout from the panorama image.

5.5. Ablation Experiments

We validate LaLaLoc++'s architecture through the ablation experiments discussed here. First, we see how the localisation accuracy is impacted by the number of down and up-sample layers in Φ_{plan} . We then vary the number num-

Φ_{image} # Blocks	Med.	Localisation Accuracy <1cm	<5cm	<10cm	<1m
1	5.9	3.7%	43.5%	69.2%	**92.3%**
2*	**5.2**	5.4%	**48.8%**	**72.3%**	**92.3%**
3	5.6	**5.7%**	45.2%	68.7%	92.0%
4	5.7	4.8%	45.1%	69.3%	92.2%

Table 8. Localisation as the number of transformer encoder blocks in Φ_{image} is varied. * denotes the chosen configuration.

Method	Localisation Success <5cm/2°	<10cm/5°	<20cm/10°
LaLaLoc	19.9%	40.9%	61.6%
LaLaLoc++	32.1%	51.7%	67.5%
LaLaLoc++ (Top 1)	16.4%	28.0%	36.0%
LaLaLoc++ (Top 2)	27.9%	45.6%	58.5%

Table 9. Comparison of successfully localised panoramas with and without an orientation prior. Each column lists the percentage of panoramas that are localised to within the error thresholds stated. Methods which assume a known rotation are *italicised*.

ber of encoder blocks used in Φ_{image}. Results for each of these experiments are listed in Tabs. 7 and 8, respectively. In Tab. 8 it is notable that, although the chosen configuration performs almost identically to the configuration with 6 down-sample and up-sample layers, the chosen configuration has significantly fewer trainable parameters (24.1M *vs.* 63.3M). Therefore, the configuration with 5 down-sample and up-sample layers is the most appropriate.

5.6. Rotational Ambiguity

In this section, we conduct experiments extending LaLaLoc++ to also predict orientation of the query panorama. We start with a brief description of how the method is extended.

To estimate the unknown orientation of the query panorama, we first rectify the image using the prepossessing step as described in [34]. Through vanishing point detection, this estimates a semi-canonical alignment of the panorama. We consider this orientation, as well as 3 other candidates formed by successive 90° rotations. LaLaLoc++'s retrieval stage is performed on each of these candidate rotations. The predicted position and orientation being given by the lowest computed embedding distance across all the candidates. These are then used to initialise the sub-pixel refinement stage to further optimise position.

We list results from the Zillow Indoor Dataset in Tab. 9. It can be seen that rotation prediction significantly increases the difficulty of the task. However, analysis of the rotational errors (Fig. 6) suggests that the vanishing point alignment is generally accurate, and that there is potential for ambiguity

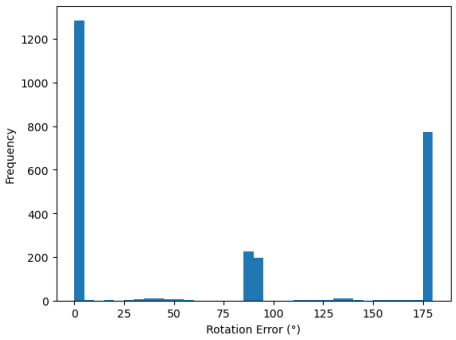


Figure 6. Histogram of rotation error frequency across on the Zillow Indoor Dataset. It can be seen that errors largely fall close to the correct alignment or at 180°, suggesting a bi-modal ambiguity.

when multiple orientations can be considered, with multiple poses near equally plausible. To combat this, we also consider a strategy where the *Top 1* orientation is computed as before, but the orientation with the second lowest predicted distance is also kept, labelled *Top 2* in Tab. 9. Localisation is considered successful if either of these pose predictions are within the tolerated error.

The results show that this generation of multiple pose hypotheses allows LaLaLoc++ to largely recover accuracy. And this setup actually outperforms LaLaLoc [14] for the strict thresholds, despite LaLaLoc using a known orientation. However, disambiguation of these hypotheses would require additional architectural information, such as windows and doors.

6. Discussion

Through experimental evaluation, LaLaLoc++ has been shown to significantly outperform other floor plan localisation methods. However, floor plan localisation itself has some inherent limitations. For example, it is trivial to imagine buildings with many near structurally identical rooms. Even within the correct room, rotation can cause symmetries which leave two locations and orientations plausible, as explored in the previous section. In these scenarios, localisation through layout structure alone will fail due to the level of ambiguity. Despite this, and for the reasons stated in Sec. 1, we believe that floor plan localisation is a promising and worthwhile avenue for research. In practical application, localisation methods are seldom used in isolation, and floor plan localisation is particularly useful as a strong prior on location from which more accurate methods can be initialised or verified, especially as it is designed for indoors where GPS is significantly less reliable.

7. Conclusion

In this paper, we have presented LaLaLoc++, a method for localisation in unseen environments with leveraging a floor plan as its only prior. We have demonstrated that a representative space of 3D layout structure can be inferred directly from 2D floor plans, and that doing so yields significant accuracy and performance improvements over previous methods for this task. We therefore show that localisation can be performed effectively in unseen environments by leveraging data that is near universal in the built world.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016. 1, 2

[2] Vassileios Balntas, Shuda Li, and Victor Prisacariu. Relocnet: Continuous metric learning relocalisation using neural nets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 751–767, 2018. 1, 2

[3] Federico Boniardi, Abhinav Valada, Rohit Mohan, Tim Caselitz, and Wolfram Burgard. Robot localization in floor plans using a room layout edge extraction network. *arXiv preprint arXiv:1903.01804*, 2019. 2

[4] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2017. 1, 2

[5] Eric Brachmann and Carsten Rother. Learning less is more-6d camera localization via 3d surface regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4654–4662, 2018. 2

[6] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7525–7534, 2019. 2

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 5

[8] Shuai Chen, Zirui Wang, and Victor Prisacariu. Direct-posenet: Absolute pose regression with photometric consistency. *arXiv preprint arXiv:2104.04073*, 2021. 2

[9] Steve Cruz, Will Hutchcroft, Yuguang Li, Naji Khosravan, Ivaylo Boyadzhiev, and Sing Bing Kang. Zillow indoor dataset: Annotated floor plans with 360deg panoramas and 3d room layouts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2133–2143, 2021. 2, 5, 6, 12

[10] Mingyu Ding, Zhe Wang, Jiankai Sun, Jianping Shi, and Ping Luo. Camnet: Coarse-to-fine retrieval for camera re-localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2871–2880, 2019. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 7

[12] Jared Heinly, Johannes L Schonberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the world* in six days*(as captured by the yahoo 100 million image dataset). In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3287–3295, 2015. 2

[13] Harlan Hile and Gaetano Borriello. Positioning and orientation in indoor environments using camera phones. *IEEE Computer Graphics and Applications*, 28(4):32–39, 2008. 2

[14] Henry Howard-Jenkins, Jose-Raul Ruiz-Sarmiento, and Victor Adrian Prisacariu. Lalaloc: Latent layout localisation in dynamic, unvisited environments. *arXiv preprint arXiv:2104.09169*, 2021. 2, 3, 4, 5, 6, 9

[15] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5974–5983, 2017. 2

[16] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 2

[17] Sungyeon Kim, Minkyo Seo, Ivan Laptev, Minsu Cho, and Suha Kwak. Deep metric learning beyond binary supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2288–2297, 2019. 2

[18] Hyon Lim, Sudipta N Sinha, Michael F Cohen, and Matthew Uyttendaele. Real-time image-based 6-dof localization in large-scale environments. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1043–1050. IEEE, 2012. 2

[19] Liu Liu, Hongdong Li, and Yuchao Dai. Efficient global 2d-3d matching for camera localization in a large-scale 3d map. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2372–2381, 2017. 2

[20] François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015. 2

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[22] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019. 1, 2

[23] Grant Schindler, Matthew Brown, and Richard Szeliski. City-scale location recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2007. 2

[24] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 2

[25] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013. 2

[26] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1047–1056, 2019. 5, 6

[27] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002. 2

[28] James Unicomb, Ravindra Ranasinghe, Lakshitha Dantanarayana, and Gamini Dissanayake. A monocular indoor localiser based on an extended kalman filter and edge images from a convolutional neural network. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018. 2

[29] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip HS Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4400–4408, 2015. 2

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3, 5

[31] Shenlong Wang, Sanja Fidler, and Raquel Urtasun. Lost shopping! monocular localization in large indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2695–2703, 2015. 2

[32] Wera Winterhalter, Freya Fleckenstein, Bastian Steder, Luciano Spinello, and Wolfram Burgard. Accurate indoor localization for rgb-d smartphones and tablets given 2d floor plans. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3138–3143. IEEE, 2015. 2

[33] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. *arXiv preprint arXiv:1908.00222*, 2(7), 2019. 2, 5

[34] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2059, 2018. 9



Figure 7. Qualitative visualisation of failure cases. *Left:* failures where the wrong room is retrieved entirely. *Right:* misalignment of the layout to the image.

A. Appendix

A.1. Training Schedule

On Structured3D, Φ_{plan} is trained for a total of 100 epochs through SGD. The batch size is set to 16. The initial learning rate is set to 0.05, with momentum at 0.9 and weight decay of 10^{-4}. The learning rate is scaled by a factor of 0.1 after 50 and 75 epochs. Φ_{image} is trained for 200 epochs, again through SGD and with a batch size of 64. 0.1 is taken as the initial learning rate, and again with a momentum of 0.9 and weight decay of 10^{-4}. After 100 and 150 epochs the learning rate is scaled by a factor of 0.1.

On the Zillow Indoor Dataset [9], the training epochs are all scaled by a factor of 1.5x. Other than this they remain identical to the schedules above.

A.2. Failure Cases

We provide qualitative visualisation of localisation failures with LaLaLoc++ in Fig. 7. These instances of failed localisation seem to generally fall into two main modes. In the first mode, the pose is predicted to be in an incorrect room, this is an example of layout ambiguity across the floor plan. However, in these scenarios the alignment does frequently still appear plausible. In the second mode of failure, the pose is predicted to be in the correct room, but the layout is then misaligned. By visualising these cases, we see that room edges and corners are often mistakenly aligned to the edges of room clutter, such as tables and counter-tops.

A.3. Decoder Output

In this section, we provide additional visualisation of the layout decoder output. Figure 8 illustrates the 2D to 3D hallucination that is learned by LaLaLoc++ during the floor plan embedding training stage. It can be seen that the layouts captured by the floor plan encoder generally reflect the structure well. The wall/floor or ceiling boundary seems to be the largest source of error in many cases, as is seen in the top three rows. This is to be expected as LaLaLoc++ has to learn a general prior for heights from 2D plans.

It is also apparent that some finer layout detail is lost, but the prediction is still very representative of the overall structure, as depicted in the bottom three rows. Interestingly, when considered with Fig. 5, the type of structure that is missed in decoding still contributes to the descriptor as evidenced by its saliency. This would suggest that this type of error is due to the limited decoder design. However, we emphasise that this paper is not targeting room layout estimation. This layout decoder offers a simple inductive bias and its output is only computed during training of the embedding space. These results only serve to improve intuition about how LaLaLoc++ infers 3D structure.

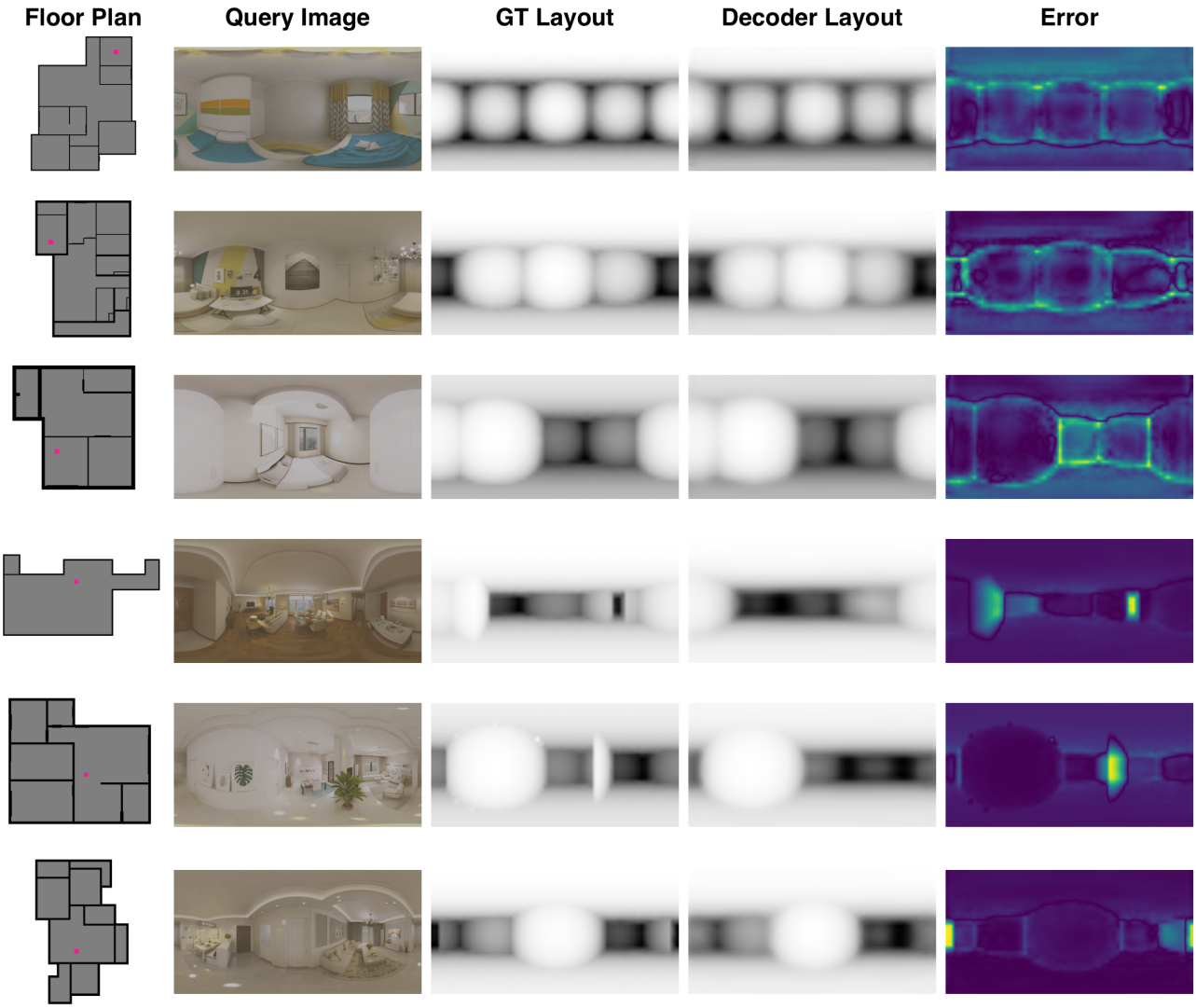


Figure 8. Qualitative layout predictions from the decoder, helping to illustrate the 2D to 3D hallucination of layout. *Left to right:* floor plan with query location marked in pink; query image; ground truth layout; decoded layout; L1 error.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	LaLaLoc++: Global Floor Plan Comprehension for Layout Localisation in Unvisited Environments
Publication Status	☐ Published ☐ Accepted for Publication ☑ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	**Howard-Jenkins, H.** and Prisacariu, V. A. (2021). LaLaLoc++: Global Floor Plan Comprehension for Layout Localisation in Unvisited Environments. *Submitted for Publication.*

Student Confirmation

Student Name:	Henry Howard-Jenkins		
Contribution to the Paper	• Conception and validation of the idea • Design of the architecture and framework • Implementation and running of all experiments. • Research of prior work • Writing and presentation of the paper		
Signature	HHJ	Date	31/01/2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Prof Victor Adrian Prisacariu		
Supervisor comments	The description is accurate		
Signature	VPA	Date	09/02/2022

This completed form should be included in the thesis, at the end of the relevant chapter.

5.3 Conclusion

In this chapter, we presented LaLaLoc++ a method for floor plan localisation that introduces a dense, global plan comprehension module. Unlike previous work, such as that presented in Chapter 4, the plan comprehension module implicitly hallucinates 3D structure directly from 2D floor plans of an environment. This means that an explicit 3D extrusion stage is no longer required, and therefore fewer characteristics need to be known or assumed about the scene.

We have also shown that LaLaLoc++ is able to achieve a substantial improvement in efficiency. This improvement in efficiency is largely due to the holistic processing of the floor plan. Instead of performing an explicit rendering and forward pass of a layout network, inference of layout at a location within the floor plan now takes the form of a single forward pass, which is shared across all locations, followed by a bilinear look-up within the produced feature map. Therefore inference at each new location only requires a cheap look-up.

In these past two chapters, we have explored the task of floor plan localisation. Notably, we show that, in our test setting, layout is expressive enough property of panoramas taken within an indoor environment to be able to localise with only a floor plan prior with a high degree of accuracy. However, investigation of how this methodology could be applied to more generalised indoor visual imagery, such as video sequences, images with narrower fields of view, as well as rotation forms interesting and important future work. While we have shown that floor plan localisation alone can provide accurate localisation, ambiguous structure remains an inherent limitation of the approach. Therefore, in a practical system an additional method is likely required in tandem to disambiguate repeated structure.

In spite of this, the work represented by LaLaLoc and LaLaLoc++ are significant steps forward in validating the possibility of a solely floor plan based localisation system. In particular, with its significantly improved simplicity and efficiency, LaLaLoc++ lays the ground work for expansion beyond the test setting that we explored.

Part II

Group Convolution for Low-cost Network Design

Chapter 6

Real-time Instance Segmentation Through Group-Convolutions

6.1 Context

A core task for the domestic robot proposed in the MoveCare project, was to enable the robot to locate everyday objects of immediate importance to the user, such as their phone. Although the retrieval of lost objects in MoveCare was also handled through the application of RFID tags to the objects of interest, these tags were found to perform poorly for shielding objects, such as mobile phones[1]. Therefore, we were set the task to create a visual system for the localisation of objects within the scene to compliment the RFID-based strategy.

We approached this challenge through dense visual semantic mapping of its environment. The component of the robot's perception systems targeted the building of a semantic layer on top of its own internal geometric map of the home, which itself was created through the dense visual SLAM system InfiniTAM [Prisacariu et al., 2017]. Approaches such as [Runz et al., 2018] demonstrate that this integration of instance-level semantic information from incoming image frames into a dense 3D volume is possible. However, this style of semantic mapping requires instance segmentation on the incoming images and, at the time of the project, accurate instance segmentation was too slow and too large for deployment onto the robot.

To address this limitation, we explored the creation of a real-time instance segmentation method, Mobile R-CNN, that could be leveraged for this semantic mapping task. With the introduction of Mobile R-CNN, onboard instance segmentation of

[1] http://www.movecare-project.eu/wp-content/uploads/2019/07/movecare-04-d4.1_objects_searching_by_radio-frequency_2.3.pdf



Figure 6.1: Example mobile phone detections in a household environment. Mobile R-CNN is able to accurately infer the presence of a phone in the image.

incoming frames became feasible and accurate enough for meaningful fusion of semantic information. Mobile R-CNN is able to run inference at about 5Hz on the NVidia Jetson onboard the robot. And, as a light demonstration of its potential for the localisation of mobile phones specifically, examples of phone detections in domestic environments can be seen in Figure 6.1, where we are able to accurately detect a phone in the image.

In the rest of this chapter, we detail the design of this instance segmentation method for use in the MoveCare project.

6.2 Introduction

For each object in an image, instance segmentation predicts its type, location, and precise visible extent as a mask. This offers a representation that is very useful for down-stream tasks such as object grasping [Danielczuk et al., 2019; Xie et al., 2021] and dense instance-level semantic mapping [Runz et al., 2018]. However, instance segmentation methods had previously been slow, posing a significant challenge for their adoption in online visual perception systems.

In this chapter, we address this limitation through creation of a real-time instance segmentation method, Mobile R-CNN. To achieve real time speeds, previous and concurrent methods [Bolya et al., 2019b; Jetley et al., 2017; Uhrig et al., 2018] propose a substantial reformulation of instance segmentation so that inference can consist of only a single stage. Mobile R-CNN, however, is a surprisingly simple modification of the two-stage Mask R-CNN [He et al., 2017].

There are two key insights which enable Mobile R-CNN. The first is that replacing the ResNet [He et al., 2016] backbone in Mask R-CNN with one designed for

lightweight inference, specifically MobileNetV2 [Sandler et al., 2018], results in an instance segmentation network that runs in real-time. The second, and key contribution for this chapter, is that the accuracy of the low-cost backbone can be significantly improved with minimal slow-down by exploration of non-depthwise group convolution topologies. In tandem, these insights produce an instance segmentation method which achieves high accuracy at real time speeds.

The work presented in this chapter offers the following contributions:

- It introduces Mobile R-CNN which, at the time, offered high accuracy instance segmentation at real-time inference speeds. This represented a significant improvement in the speed-accuracy trade-off compared to other methods available before it.

- The work conducted for this chapter provides interesting insight into group convolutions as a design tool for convolutional network architectures. It demonstrates that intermediary ranks of convolution can offer benefits over the extremes of maximal expressive power (full convolution) or minimal computational complexity (depthwise convolutions).

This chapter presents a method designed which was designed for a performance target, real-time inference, and leveraged for a specific application, instance segmentation. Beyond this, however, it also serves as an important initial exploration and motivation for the more generalised work presented in the Chapter 7, which can be applied across any number of performance targets and applications.

6.3 Empirical Motivation: From Depthwise to Group Convolution

Before introducing Mobile R-CNN and its architecture for instance segmentation, we first discuss the background of group convolution in low-cost network design and present our empirical findings that provide one of the key insights in the design of Mobile R-CNN.

6.3.1 Depthwise-Separable Convolutions

Low-cost network architectures frequently rely on depthwise convolution [Howard et al., 2017; Sandler et al., 2018]. The most commonly used design block for these network architectures are *depthwise-separable* convolutions [Sifre and Mallat, 2014].

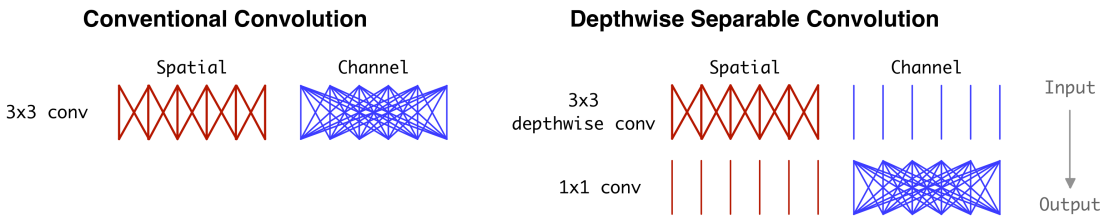


Figure 6.2: An illustration of input/output dependencies in a conventional 3×3 convolution (*left*) and its depthwise-separable equivalent (*right*). The depthwise separable convolution applies an individual 3×3 convolution to each channel of the feature map, then applies a pointwise convolution. Only one spatial dimension is depicted for clarity.

Although there are numerous permutations of their exact formulation, in essence depthwise-separable convolutions approximate a full convolution by the composition of convolutions which each operate in either the spatial dimensions, thereby iterating through each depth dimension, *i.e.* *depthwise*, or the depth dimension, thus iterating through each spatial localisation, *i.e.* *pointwise*.

We illustrate a simple depthwise-separable convolution in Figure 6.2. In this figure we use lines to illustrate dependency of the output element on the input element in both the *spatial* and *channel* dimensions. Given an input tensor of size $h \times w \times m$ and a convolutional layer formed by a set of n filters, each $(k \times k \times m)$[2], the number of multiplications performed in the convolutional layer is given by $n \cdot m \cdot hw \cdot k^2$. Conversely, in the depthwise-separable case, also depicted in Figure 6.2, the overall layer consists of m filters, each $(k \times k \times 1)$, followed by n filters, each $(1 \times 1 \times m)$. This leads to an overall complexity of $m \cdot hw \cdot k^2 + n \cdot m \cdot hw$.

These computations allow us to compute a theoretical speed-up factor given by the ratio of the original convolution complexity by the depthwise-separable complexity, given by:

$$\text{Speed Up} = \frac{n \cdot k^2}{k^2 + n}. \tag{6.1}$$

For reasonable values of $n = 512$ and $k = 3$, a depthwise separable convolution has a theoretical speed-up of $8.8\times$ – a considerable reduction in complexity.

6.3.2 Group Convolution

Thus far, we have only considered the case of a depthwise convolution. However, depthwise convolutions are in fact a special case of the more general group convolu-

[2]For simplicity we will assume a stride of 1 and padding set to floor$(k/2)$ such that the output dimensions are given by $(h \times w \times n)$

tion. A group convolution effectively splits the input in the channel dimension into g groups, performs a convolution on each, and concatenates the resulting tensors to form the final result. Therefore, a depthwise convolution is simply the extreme case of a group convolution where $g = n$ and so each group consists of a single channel. Using the same notation as before, a group convolution has complexity given by $\frac{n}{g} \cdot m \cdot hw \cdot k^2$ and a separable convolution using a generalised group convolution has a theoretical speed-up given by:

$$\text{Group Speed-Up} = \frac{g \cdot k^2}{k^2 + g}. \tag{6.2}$$

By again considering example values for a convolution, $n = 512$, $k = 3$ and $g = 16$, the theoretical speed-up offered this grouped version of a separable filter is 5.8×. This is still considerably faster than the original convolution, but less so than the depthwise variant.

Interestingly, however, we found that the depthwise convolutions that are leveraged for these low cost network architectures are often not well optimised in deep learning frameworks, such as Pytorch [Paszke et al., 2019]. This causes significant deviation from the theoretical speed-up figures when the real-world speed-up is measured, *i.e.* wall-clock speed-up. Ultimately, this can lead to depthwise convolutions having either very little speed-up or even slower inference when compared to counterpart convolutions with fewer groups.

As a small illustration of this phenomenon in the case of filter groups, in Figure 6.3 we plot the speed-up offered by various group configurations of separable convolutions, with both the relative inference time and the theoretical vs actual speed-ups plotted. It is notable that the measured times vary quite significantly from the theoretical speed-up that should be offered by the separation. Even more interestingly, on the GPU, some of the theoretically fastest configurations are actually slower than larger sizes. This discrepancy has very recently been explored thoroughly by Selvam et al. [2021]. And this implementation-aware design has now been considered for EfficientNet [Tan and Le, 2019] on Google's Edge TPU [Gupta and Akin, 2020].

This provides our first indication of an opportunity to target groups for the design low-cost networks. In the following, we will detail our work to do so.

6.4 Mobile R-CNN

With Mobile R-CNN we exploit the observation that the implementation of group convolutions can lead to unintuitive real speed-ups by replacing the depthwise convolutions with convolutions of fewer groups. In particular, we aim to leverage the

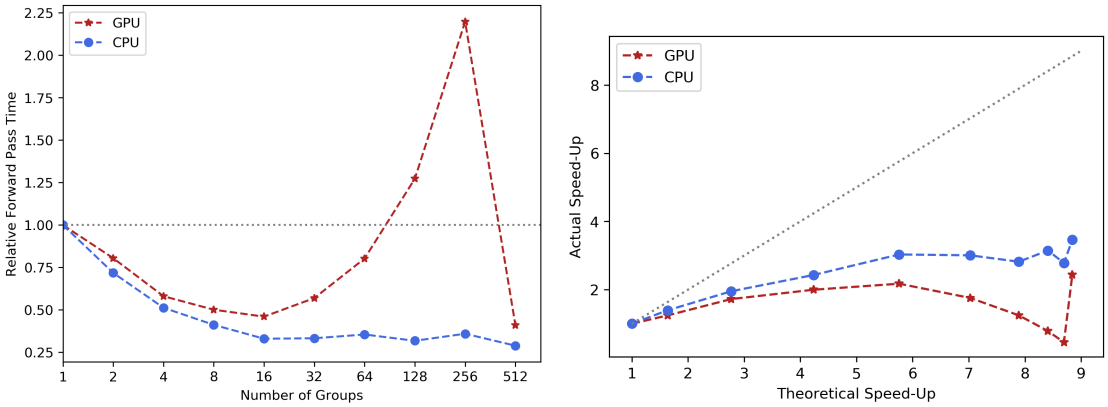


Figure 6.3: Example of the actual speed-up provided by separable convolutions with varying numbers of groups. Latency is evaluated on a single NVidia 1080Ti GPU and on a single CPU. *Left:* The relative forward pass time for each configuration. *Right:* The theoretical *vs.* actual speed-up. Each data point is the mean time of 500 repeated forward passes, $n = m = 512$, $k = 3$. When $g = 1$, a full convolution is used without a pointwise convolution, at other values of g the convolution is split into a grouped convolution followed by a pointwise convolution. This test is performed with Pytorch 1.0.0.

additional expressiveness offered by these convolutions of a higher order while not experiencing the proportional slow-down. To do so, we alter the depthwise-separable convolutions present in MobileNetV2 [Sandler et al., 2018], with convolutions with larger groups. This allows far greater representation power, but with minimal additional latency.

MobileNetV2 uses a different form of depthwise-separable convolutions which form a bottleneck-style structure, often referred to as an *inverted residual*. An example illustration of the structure of these blocks is depicted in Figure 6.4. Figure 6.5 depicts simplified input/output dependencies for these blocks. Our proposed design change uses a group convolution in place of the original depthwise convolutions, which is also included in both of these figures. Table 6.1 details how the typical MobileNetV2 residual block is altered to form the expanded group network.

The typical verbiage for describing group convolution, and indeed the one that has been used thus far in the chapter, is to define the number of groups. However, when dealing with design group convolutions across the network architecture, layers' filters often expect input tensors of a different channel dimension to each other and therefore the order of the convolutions may be different despite being defined by the same number of groups. A notable exception to this are depthwise convolutions, where the layer is not usually defined by the number of groups, but instead the size of

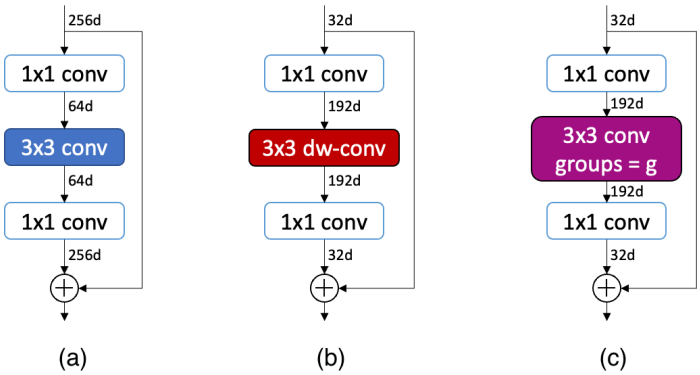


Figure 6.4: Block diagram of residual bottleneck architectures. (a) ResNet bottleneck. (b) MobileNetV2 inverted residual. (c) Mobile R-CNN expanded inverted residiual. BatchNorm layers and ReLU activations are not included.

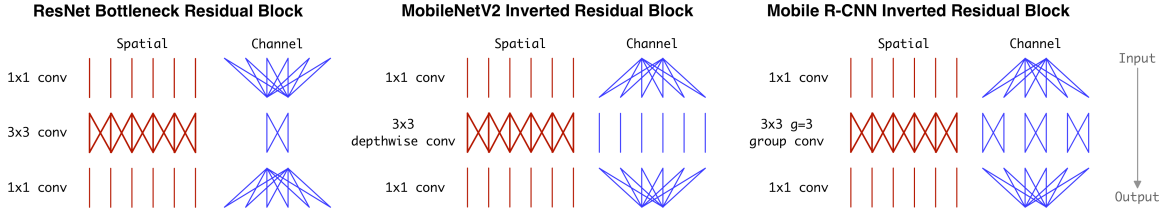


Figure 6.5: Illustration of the residual blocks used in ResNet and MobileNetV2, as well as our proposed expanded group inverted residual that is used in Mobile R-CNN.

each group, *i.e.* a single channel dependency per group. For convenience, we extend this beyond depthwise, and instead also define our convolutions by *group-size*, $\gamma = \frac{n}{g}$.

6.4.1 Architecture

The overall detection architecture structure of Mobile R-CNN follows the standard FPN-based Mask R-CNN [He et al., 2017]. Here we provide a brief overview of the main components.

The ResNet50 [He et al., 2016] that is typically present as the FPN [Lin et al., 2017] backbone is replaced by MobileNetV2 or our filter-group expanded version. In

Table 6.1: Bottleneck block transforming from k to c channels, with group size γ, stride s and expansion factor t. If $\gamma = 1$, we recover the original MobileNetV2 bottleneck.

Input	Operator, Activation	Output
$h \times w \times k$	1x1 Conv, ReLU6	$h \times w \times tk$
$h \times w \times tk$	3x3 gConv (group-size=γ, stride=s), ReLU6	$\frac{h}{s} \times \frac{w}{s} \times tk$
$\frac{h}{s} \times \frac{w}{s} \times tk$	1x1 Conv, None	$\frac{h}{s} \times \frac{w}{s} \times c$

Table 6.2: Mobile R-CNN backbone layer sequences as well as their corresponding FPN stage in our R-CNN. As with the structure definition in Sandler et al. [2018], each row of the table represents an identical layer (excluding stride) repeated n times. Stride, s, represents the stride of the first layer, all others have stride 1. Each of the layers has c output channels and expansion factor of t. The network follows the same overall structure as MobileNetV2, but with a change to the definition of the bottleneck blocks.

Operator	t	c	n	s	γ	**FPN Stage**
conv2d	-	32	1	2	-	stage 1
bottleneck	1	16	1	1	16	
bottleneck	6	24	2	2	32	stage 2
bottleneck	6	32	3	2	32	stage 3
bottleneck	6	64	4	2	32	stage 4
bottleneck	6	96	3	1	32	
bottleneck	6	160	3	2	32	stage 5
bottleneck	6	320	1	1	32	

Table 6.2, we detail the overall structure of the expanded network, as well as how the stages form the FPN backbone. The channel depth of the FPN is set to 128.

The region proposal network (RPN) first applies a 3×3 convolution with 128 out channels to the the feature map produced by the FPN [Lin et al., 2017] backbone. The objectness and box prediction for each of the n_a anchors are then produced by two parallel 3×3 convolutions with n_a and $4 \cdot n_a$ output channels.

The box head applies two fully connected layers each with 1024 output channels and ReLU non-linearities to each region's features. Class logits and the per-class box refinements are then predicted by parallel fully connected layers with n_c and $4 \cdot n_c$ output channels, respectively.

Finally, the mask head applies four 3×3 convolutions, each with 256 output channels and followed by ReLU non-linearities. These are followed by a 2×2 transpose convolution with 256 output channels which produces a feature map that has a spatial resolution of 28×28. The final mask prediction is performed by a pointwise convolution with n_c output channels.

6.4.2 Implementation Details

We found empirically that the depthwise separable convolutions from the standard implementation of MobileNetV2 provided a good initialisation for our grouped bottlenecks. We adapt MobileNetV2 weights pretrained on ImageNet [Krizhevsky et al.,

2012] by repeating the respective depthwise weights by the number of channels in the group within the grouped convolution.

All networks, aside from YOLACT [Bolya et al., 2019b], are implemented in the `maskrcnn-benchmark` repository [Massa and Girshick, 2018] using Pytorch [Paszke et al., 2019].

6.4.2.1 Training Routine

We train each full resolution configuration for a total of 360k iterations and a batch size of 12. Full resolution is considered as images resized so their minimum length is 800 pixels, maintaining aspect ratio. If after this rescaling the maximum length exceeds 1333 pixels, the image is scaled further so that the maximum length is 1333 pixels. A SGD optimiser is used with initial learning rate of 0.015, momentum set to 0.9 and weight decay factor of 10^{-4}. The learning rate is decayed by a factor of 0.1 after 240k and 320k steps, and an initial linear learning rate warm up is used from 1/3 of the initial learning rate over the first 500 iterations.

To train the lower resolution configurations, for example those which take images of minimum length 600 instead of 800 and maximum length scaled accordingly, we run a short fine-tuning schedule initialised from the trained weights of their respective full resolution counterpart. SGD is used again, with initial learning rate of 0.0015. The fine-tuning lasts for a total of 40k iterations, with learning rate decayed after 26,667 and 35,555 steps. All other training settings remain the same as in the full resolution schedule.

6.5 Instance Segmentation

6.5.1 Dataset

We perform training and testing on the COCO [Lin et al., 2014] dataset. This dataset contains a total of 120k images, with full instance annotations for 80 object categories that appear within the images. For most experiments in this chapter, we utilise the 2014 splits for COCO. We train on the union of 80k training images and 35k subset of the validation split (`valminusminival`). Results are then reported on the remaining subset of the validation split (`minival`), which consists of 5k images. However, we do report results on the 2017 `testdev` and `val` splits, where models are trained on the 2017 `train` using an identical training routine.

Table 6.3: A comparison of the inference time and accuracy of lightweight R-CNN methods for object detection on COCO 2014 minival, as well as a ResNet50 Mask R-CNN for reference. Real-time methods are highlighted.

Model	Resolution	Inference Speed		mAP	
		Time (ms)	FPS	Box	Mask
ResNet-50 C4	800	*149*	*6.7*	*35.7*	*31.3*
FBNet (default)	600	31.5	31.8	29.0	26.1
FBNet (xirb16d)	600	35.0	28.6	30.7	26.9
MobileNetV2 FPN	600	33.3	30.0	30.1	27.7
Mobile R-CNN	600	34.4	29.1	31.8	28.9
Mobile R-CNN	550	32.5	30.8	31.4	28.5

6.5.2 Methods for Comparison

The most similar real-time baseline is the one provided by Mask R-CNN with a MobileNetV2 backbone. In this case, the R-CNN structure is identical to that used in Mobile R-CNN, however, the backbone keeps the original depthwise inverted residuals without our proposed group expansion.

We also compare with FPN-based R-CNN structures which utilise FBNets [Wu et al., 2019] as backbones. FBNets are convolutional networks whose macro-structure follows that of MobileNetV2. However, precise design parameters of the inverted residual blocks are found through a low-latency driven architecture search. The precise architectures and how they are leveraged in the R-CNN can be found in the `maskrcnn-benchmark` repository [Massa and Girshick, 2018], although they are broadly similar to the MobileNetV2 baseline.

Finally, we compare against YOLACT [Bolya et al., 2019b], a concurrent work on real-time instance segmentation. It reformulates mask prediction into a linear combination from whole-image mask prototype predictions. We compare against their ResNet50-FPN version, their most accurate configuration for which they state real-time speeds.

6.5.3 Results

In this section, we perform analysis of instance segmentation results for Mobile R-CNN offering comparison to baseline and competitive methods that approach real-time speeds. For reference values for a "full" R-CNN, we include results for Mask R-CNN with a ResNet50 backbone.

The inference speed and detection performance results are included in Table 6.3. Mobile R-CNN demonstrates a significant improvement in accuracy over the other

Table 6.4: A comparison of Mobile R-CNN with YOLACT on COCO2017 testdev. Real-time methods are highlighted.

Model	Resolution	Inference Speed		mAP	
		Time (ms)	FPS	Box	Mask
YOLACT	550^2	34.8	28.8	32.3	29.8
Mobile R-CNN	600	34.6	28.9	31.8	29.0
Mobile R-CNN	550	33.1	30.2	31.2	28.4

Table 6.5: A comparison of Mobile R-CNN with YOLACT on COCO2017 val. Real-time methods are highlighted.

Model	NMS	Resolution	Inference Speed		mAP	
			Time (ms)	FPS	Box	Mask
YOLACT	Standard	550^2	40.2	24.9	32.2	29.6
Mobile R-CNN	Standard	600	34.2	29.2	31.7	29.1
Mobile R-CNN	Standard	550	32.6	30.7	31.2	28.5
YOLACT	Fast	550^2	34.8	28.8	32.1	29.7
Mobile R-CNN	Fast	600	29.2	34.3	31.3	28.9
Mobile R-CNN	Fast	550	27.5	36.4	30.7	28.3

lightweight methods, with minimal increases in inference time. In fact, Mobile R-CNN (550) increases box and mask mAP by 0.7 and 1.6, respectively, over FBNet (xirb16d), all while being 2.5 ms faster.

6.5.3.1 COCO 2017

In Table 6.4, we retrain Mobile R-CNN on the COCO 2017 dataset, and evaluate on the `testdev` split for comparison with YOLACT [Bolya et al., 2019b]. As can be seen in the listed results, Mobile R-CNN and YOLACT offer somewhat similar performance overall; Mobile R-CNN is marginally faster, but YOLACT offers higher accuracy. We investigate this further in the following.

We evaluate on the `val` split in Table 6.5 so that we can include YOLACT with a standard NMS, as leveraged in Mobile R-CNN, as well as with their proposed Fast NMS, which is an approximation of non-maximal suppression (NMS) for faster inference. For symmetry, we also include results for Mobile R-CNN with Fast NMS to compare. In all configurations where the same style of NMS is used, Mobile R-CNN exhibits significantly faster speeds, but with a lower accuracy.

In their paper introducing YOLACT, Bolya et al. [2019b] state, "two-stage methods require re-pooling features for each ROI and processing them with with subsequent computations, which make them unable to obtain real-time speeds". However,

Table 6.6: A comparison of differing group-sizes in our Mobile R-CNN backbone. The number of paramaters, inference speed and accuracy of each group-size configuration is detailed. The selected configuration is marked with an *. Worth noting that each of these configurations is trained for a shortened training schedule.

Group-Size	# Params	Latency (ms)	mAP	
			Box	Mask
1	12.9M	43.4	25.0	23.4
8	13.3M	47.7	25.5	23.9
16	13.8M	43.9	25.9	24.1
32*	14.8M	42.2	26.1	24.2
64	16.7M	41.9	26.6	24.7

the two-stage Mobile R-CNN demonstrates that this is not true. In fact, when their Fast NMS is used in Mobile R-CNN so that the inherent structure of the two methods is being compared more directly, Mobile R-CNN exhibits inference speeds over 16% faster than YOLACT, with a loss of only 0.8 in box and mask mAP.

6.5.4 Group Selection

In this section, we perform a study to determine the group sizes used in the network backbone. We test at a resolution such that the minimum length of images is 800 pixels after training each configuration for a 1x schedule. Note that this is shorter than schedule used in Section 6.5.3, however, it still able to provide insight into the relative speed accuracy trade-offs between configurations.

The results are listed in Table 6.6. As can be seen from the performance, a group-size of 32 across the network offers a good balance of fast inference and high accuracy without an unreasonable increase in the number of parameters. Notably, the inference times across configurations remain consistent with the initial investigation into the marginal wall-clock speed-up for a single layer in Figure 6.3, with a group-size of 32 offering faster inference than some smaller ranks of group convolution.

6.5.5 Image Resolution

Here we evaluate the impact that input image resolution has on the accuracy and speed of Mobile R-CNN. We also include results for the MobileNetV2 FPN backbone for reference. The results are given in Figure 6.7. Changes to image resolution offer an easy and effective tool to select a speed/trade off for specific performance requirements. However, it seems that the accuracy of Mobile-RCNN somewhat saturates



Figure 6.6: Qualitative detection results on COCO2014 minival.

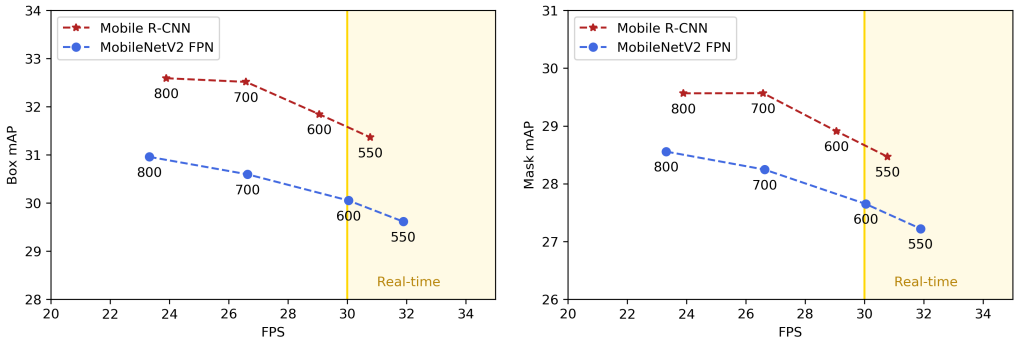


Figure 6.7: Impact of input resolution on inference speed and latency. We include results for both Mobile R-CNN and the depthwise MobileNetV2 FPN, for reference. Each point is labelled with its corresponding input resolution.

	Accuracy (%)		
Model	Latency (ms)	Top 1	Top 5
MobileNetV2	11.9	71.7	90.4
Expanded	13.3	73.9	91.8

Table 6.7: ImageNet classification results for the original MobileNetV2 and our proposed version with expanded groups. Latency is recorded as the forward pass time for a batch of 16 images on a NVidia 1080ti GPU.

above 700 pixel images. Further improvements to accuracy, therefore, would likely require a more expressive backbone.

6.6 Image Classification

In this section, we also explore how our backbone network performs for the task of image classification on the ImageNet dataset [Krizhevsky et al., 2012]. We retrain MobileNetV2 and our expanded version from scratch and evaluate the results.

Training is conducted over a total of 150 epochs, with a batch size of 128. An SGD optimiser is used with an initial learning rate of 0.05, momentum as 0.9 and weight decay at $4e-5$. The learning rate is decayed following a cosine annealing procedure without restarts. Data augmentation for training is performed as a 224×224 random crop from the randomly re-scaled original image. During training, a dropout of 0.2 is also applied to the linear layers in the predictor head.

The image classification results are given in Table 6.7. It can be seen that the proposed group expansion offers a significant improvement in classification accuracy that is able to be achieved by this still lightweight network. This is accompanied by only a slight increase in the latency.

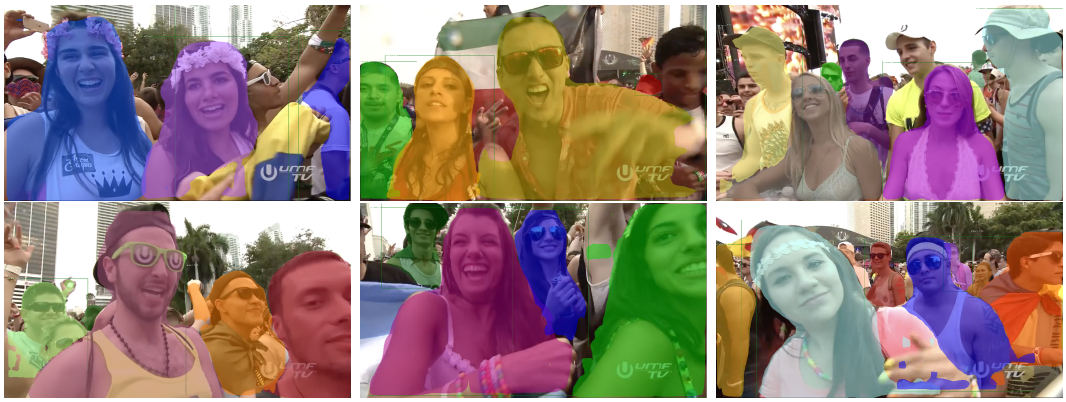


Figure 6.8: Example people detections in festival crowds using Mobile R-CNN. Even without any fine-tuning for these crowded images Mobile R-CNN is able to perform the person instances segmentation task well, and in real time.[4].

6.7 Down-stream Application

In this section, we describe a down-stream applications of Mobile R-CNN which leverages the low cost instance segmentation, albeit in a remarkably different setting than outlined in Section 6.1.

6.7.1 Aphex Twin Set Art at Coachella 2019

The real-time performance of Mobile R-CNN was also used to create live art for Aphex Twin's set at Coachella 2019. We were approached by the musician with their conception for digital set art for use at Coachella, a music festival that is widely considered to be among the largest and most famous in the world[5].

They aimed to incorporate the live audience into the performance. This, therefore, required a system to detect and segment individual members of the audience using a live feed from a camera man. Much as with MoveCare, Mobile R-CNN was then the only, to the best of our knowledge, instance segmentation method that could meet these real-time requirements with high accuracy.

Mobile R-CNN enabled real-time on a laptop computer which received a live feed from a camera in the audience. People visible in the camera feed were then detected and their outline inferred. Some example festival crowd images and Mobile R-CNN's detections are depicted in Figure 6.8. These people detections were then processed

[4]Images are taken from Oliver Heldens on Youtube: https://www.youtube.com/watch?v=AiGGFDn7ac4

[5]https://www.timeout.com/music/the-50-best-music-festivals-in-the-world



Figure 6.9: Mobile R-CNN used in real-time at Coachella 2019. The network was modified to detection only the person class and was fine-tuned for this setting. Its predictions were then processed to create the digital set art depicted.

to create the digital set art and displayed on the stage's screens. Captures from the set recording are included in Figure 6.9.

6.8 Conclusion

In this chapter, we have presented Mobile R-CNN, a method for real-time instance segmentation. Mobile R-CNN achieves its fast and accurate inference through the creation of a lightweight backbone which exploits the real-world performance characteristics of group convolutions. Unlike other instance segmentation methods which target real-time inference [Bolya et al., 2019b; Jetley et al., 2017; Uhrig et al., 2018], Mobile R-CNN does not achieve real-time inference speeds through any significant reformulation of two-stage instance segmentation methods. Instead, Mobile R-CNN demonstrates that highly accurate and very fast inference is possible with the same general inference structure proposed in Mask R-CNN [He et al., 2017].

Mobile R-CNN serves as our first exploration of group convolutions for efficient architecture design. Rather than exploring how architectures could be made as efficient as possible with depthwise convolutions, we approached the design problem from an

alternative perspective instead seeing how low-cost networks could be improved in terms of accuracy. We found that by increasing the rank of group-convolutions, the expressive power of a network could be improved, while the theoretical increase in complexity is not reflected in wall-clock time. Therefore, we are able to make MobileNetV2 greatly more accurate as a backbone in an instance segmentation network, without experiencing significant slow-down over the original version.

One notable limitation, however, was that during the design process we found that exploration of the group convolution was time consuming and ultimately relied heavily on heuristics. An example of this can be seen with the limited exploration of network topologies in Section 6.5.4. In this experiment, uniform settings of group-sizes across the network are chosen, leaving a multitude of possible configurations which remain untested. This was the driving motivation for the work in the next chapter which aims to create tools for validation of grouped architecture design and to enable full exploration of network topologies.

Mobile R-CNN was developed for the MoveCare Horizon 2020 project. The results and application of this method validates its performance and usefulness. Further, the design demonstrates the effectiveness of grouped convolution for high accuracy, while maintaining fast inference.

Chapter 7

Enabling Exploration of Group Topologies in Convolutional Neural Networks

7.1 Introduction

As the work in the previous chapter demonstrated, grouped convolution can provide significant performance improvements over depthwise convolution with minimal additional overhead in terms of latency. However, exploring this design space was cumbersome. We, therefore, turn our attention to enabling architecture search on grouped designs.

We introduce GroSS (Group-size Series) which represents grouped convolutions as a sum series of increasing ranks. We demonstrate the power of GroSS in the context of network acceleration through low-rank factorisation using block term decomposition (BTD). This forms a natural application because the result of BTD on a convolutional layer is analogous to forming a bottleneck architecture with a grouped convolution [Yunpeng et al., 2017]. Other pieces of work have used BTD for network acceleration [Kim et al., 2015; Wang et al., 2018a], however, rank selection for the decomposition has either been purely heuristic [Wang et al., 2018a], or not validated [Kim et al., 2015].

With GroSS, however, we are able to decompose each layer in the network into a number of candidate ranks approximations, and train each simultaneously through a weight-sharing scheme. We then can use a simple search to select a rank configuration from this trained design space. Our searching objective is to find a configuration which is more accurate than the baseline selection, while having cheaper inference. The results included within this chapter show that we are able to consistently find

configurations which achieve this.

7.1.1 Contributions

- We present GroSS decomposition – a novel formulation of tensor decomposition as a sum series of rank approximations.
- GroSS enables differentiable switching between grouped convolution ranks. Therefore, to the best of our knowledge, it is the first simultaneous training of differing numbers of groups within a single layer, as well as the all possible configurations between layers. This makes feasible, for the first time, a search for rank selection for group convolution.
- We explore this concurrently-trained architecture space in the context of network acceleration. We factorise a small network, as well VGG-16 [Simonyan and Zisserman, 2014] and ResNet-18 [He et al., 2016], and propose exhaustive and breadth-first searches on CIFAR-10 [Krizhevsky et al., 2014] and ImageNet [Krizhevsky et al., 2012].

GroSS: Group-Size Series Decomposition for Grouped Architecture Search

Henry Howard-Jenkins, Yiwen Li, and Victor Adrian Prisacariu

Active Vision Laboratory, University of Oxford, UK
{henryhj, kate, victor}@robots.ox.ac.uk

Abstract. We present a novel approach which is able to explore the configuration of grouped convolutions within neural networks. Group-size Series (GroSS) decomposition is a mathematical formulation of tensor factorisation into a series of approximations of increasing rank terms. GroSS allows for dynamic and differentiable selection of factorisation rank, which is analogous to a grouped convolution. Therefore, to the best of our knowledge, GroSS is the first method to enable simultaneous training of differing numbers of groups within a single layer, as well as all possible combinations between layers. In doing so, GroSS is able to train an entire grouped convolution architecture search-space concurrently. We demonstrate this through architecture searches with performance objectives on multiple datasets and networks. GroSS enables more effective and efficient search for grouped convolutional architectures.

Keywords: Group Convolution, Network Acceleration, Architecture Search

1 Introduction

In recent years, there has been a flurry of deep neural networks (DNNs) producing remarkable results on a broad variety of tasks. In particular, grouped convolution has become a widely used tool in some prevalent networks. ResNeXt [28] used grouped convolution for improved accuracy over the analogous ResNets [7]. On the other hand, Xception [3], MobileNet [8], and various others [29, 19] have used depthwise convolutions, which are the special case of grouped convolutions where the number of groups is equal to the number of in channels, in a for extremely low-cost inference. With these architectures, grouped convolution has proven to be a valuable design tool for high-performance and low-cost design alike. But, its application to these contrasting performance profiles has so far, to the best of our knowledge, remained relatively unexplored.

Finding a heuristic or intuition for how combinations of grouped convolutions with varying numbers of groups interact within a network is challenging. Grouped convolution, therefore, presents itself as an ideal candidate for Neural Architecture Search (NAS), which has provided an alternative to hand designed networks. NAS allows for the search and even direct optimisation of the network's structure. But, the search space for architectures is often vast, with potentially limitless design choices. Furthermore, each configuration must undergo

some training or fine-tuning for its efficacy to be determined. This has led to the development of methods which lump multiple design parameters together, which reduce the search space in a principled manner [23], as well as creating the need for sophisticated search algorithms [17, 27], which can more quickly converge to an improved design. Both techniques reduce the number of search iterations and ultimately reduce the number of required training/fine-tuning stages.

In this work, however, we do not wish to make assumptions about the grouped convolution manifold. We achieve this with the introduction of a Group-size Series (GroSS) decomposition. GroSS allows us to train the entire search space of architectures *simultaneously*. In doing so, we shift the expense of architecture search with respect to groups away from decomposition and training, and towards cheaper test-time sampling. This allows for the exploration of possible configurations, while significantly reducing the need for imparting bias on the group design hyperparameter selection.

The contributions of this paper can be summarised as follows:

1. We present GroSS decomposition – a novel formulation of tensor decomposition as a series of rank approximations. This provides a mathematical basis for grouped convolution as a series of increasing rank terms.
2. GroSS provides the apparatus for differentiably switching between grouped convolution ranks. Therefore, to the best of our knowledge, it is the first simultaneous training of differing numbers of groups within a single layer, as well as the all possible configurations between layers. This makes feasible, for the first time, a search for rank selection for network compression.
3. We explore this concurrently-trained architecture space in the context of network acceleration. We factorise a small network, as well VGG-16 and ResNet-18, and propose exhaustive and breadth-first searches on CIFAR-10 and ImageNet. We demonstrate the efficacy of the GroSS for rank selection search over a more conventional approach of partial training schedules.

2 Related Work

Grouped convolution has had a wide impact on neural network architectures, particularly due to its efficiency. It was first introduced in AlexNet [14] as an aid for the single network to be trained over multiple GPUs. Since then, it has had a wide impact on DNN architecture design. Deep Roots [9] was the first to introduce group convolution for efficiency, while ResNeXt [28] used grouped convolutions synonymously with concept of *cardinality*, ultimately exploiting the efficiency of grouped convolutions for high-accuracy network design. The reduced complexity of grouped convolution allowed for ResNeXt to incorporate deeper layers within the ResNet-analogous residual blocks [7]. In all, this allowed higher accuracy with a similar inference cost as an equivalent ResNet. The efficiency of grouped convolution has also led to several low-cost network designs. Sifre [20] first introduced depthwise separable convolutions, which were later utilised by Xception [3]. MobileNet [8] utilised a ResNet-like bottleneck design with depthwise convolutions for an extremely efficient network with mobile applications

in mind. ShuffleNet [29] was also based on a depthwise bottleneck, however, pointwise layers were also made grouped convolutions.

Previous work [10, 5, 15, 25] have applied low-rank approximation of convolution for network compression and acceleration. Block Term Decomposition (BTD) [4] has recently been applied to the task of network factorisation [2], where it was shown that the BTD factorisation of a convolutional weight was equivalent to a grouped convolution within a bottleneck architecture. Wang *et al.* [26] applied this equivalency for network acceleration. Since decomposition is costly, these methods have relied on heuristics and intuition to set hyperparameters such as the rank of successive layers within the decomposition. In this paper, we present a method for decomposition which allows for exploration of the decomposition hyperparameters and all the combinations.

Existing architecture search methods have overwhelmingly favoured reinforcement learning. Examples of this include, but are not limited to, NASNet [30], MNasNet [22], ReLeq-Net [6]. In broad terms, these methods all set a baseline structure, which is manipulated by a separate controller. The controller optimises the structure through and objective based on network performance. There has also been work in differentiable architecture search [27, 17] which makes the network architecture manipulations themselves differentiable. In addition, work such as [23] aims to limit the network scaling within a performance envelope to a single parameter.

These methods all have a commonality: the cost of re-training or fine-tuning at each stage motivates the recovery of the optimal architecture in as few training steps as possible, whether this is achieved through a trained controller, direct optimisation or significantly reducing the search space. In this work, however, GroSS allows efficient weight-sharing between varying grouped architectures, thus enabling them to be trained at once. This is similar to the task of one-shot architecture search. SMASH [1] use a hypernetwork to predict weights for each architecture. The work of Li *et al.* [16] bares most resemblance to this paper, where randomly sampled architectures are used to train shared-weights.

3 Method

In this section, we will first introduce Block Term Decomposition (BTD) and detail how its factorisation can be applied to a convolutional layer. After that, we will introduce GroSS decomposition, where we formulate a unification of a series of ranked decompositions so that they can dynamically and differentially be combined. We detail the training strategy for training the whole series at once. Finally, we detail the methodology of our exhaustive and breadth-first search.

3.1 General Block Term Decomposition

Block Term Decomposition (BTD) [4] aims to factorise a tensor into the sum of multiple low rank-Tuckers [24]. That is, given an N^{th} order tensor $\mathbf{X} \in$

$\mathbb{R}^{d_1 \times d_2 \times \ldots \times d_N}$, BTD factorises $\mathbf{X}$ into the sum of R terms with rank $(d'_1, d'_2, ..., d'_N)$:

$$\mathbf{X} = \sum_{r=1}^{R} \mathbf{G}_r \times_1 \mathbf{A}_r^{(1)} \times_2 \mathbf{A}_r^{(2)} \times_3 ... \times_N \mathbf{A}_r^{(N)} \quad \text{where} \begin{cases} \mathbf{G} \in \mathbb{R}^{d'_1 \times d'_2 \times \ldots \times d'_N} \\ \mathbf{A}_r^{(n)} \in \mathbb{R}^{d_n \times d'_n}, n \in \{1, ..., N\} \end{cases} \tag{1}$$

In the above, $\mathbf{G}$ is known as the *core* tensor and we will refer to $\mathbf{A}$ as *factors* matrices. We use the usual notation $\times_n$ to represent the *mode-n* product [4].

3.2 Converting a Single Convolution to a Bottleneck Using BTD

Here, we can restrict discussion from a general, N-mode, tensor to the 4-mode weights of a 2D convolution as follows: $\mathbf{X} \in \mathbb{R}^{t \times u \times v \times w}$, where t and u represent the number of input and output channels, and v and w the spatial size of the filter kernel. Typically the spatial extent of each filter is small and thus we only factorise t and u. To eliminate superscripts, we define $\mathbf{B} = \mathbf{A}^{(1)}$ and $\mathbf{C} = \mathbf{A}^{(2)}$. Therefore, the BTD for convolutional weights is expressed as follows:

$$\mathbf{X} = \sum_{r=1}^{R} \mathbf{G}_r \times_1 \mathbf{B}_r \times_2 \mathbf{C}_r \quad \text{where} \begin{cases} \mathbf{G} \in \mathbb{R}^{t' \times u' \times v \times w} \\ \mathbf{B} \in \mathbb{R}^{t \times t'} \\ \mathbf{C} \in \mathbb{R}^{u \times u'} \end{cases} \tag{2}$$

This factorisation of the convolutional weights into R groups forms a three-layer bottleneck-style structure [2]: a pointwise (1×1) convolution $\mathbf{P} \in \mathbb{R}^{t \times (Rt') \times 1 \times 1}$, formed from factor $\mathbf{B}$; followed by a grouped convolution $\mathbf{R} \in \mathbb{R}^{t' \times (Ru') \times v \times w}$, formed from core $\mathbf{G}$ and with R groups; and finally another pointwise convolution $\mathbf{Q} \in \mathbb{R}^{(Ru') \times u \times 1 \times 1}$, formed from factor $\mathbf{C}$. With careful selection of the BTD parameters, the bottleneck approximation can be applied to any standard convolutional layer. This is visualised in Figure 1.

In Table 1, we detail how the dimensions of the bottleneck architecture are determined from its corresponding convolutional layer, and indicate how properties such as stride, padding and bias are applied within the bottleneck for

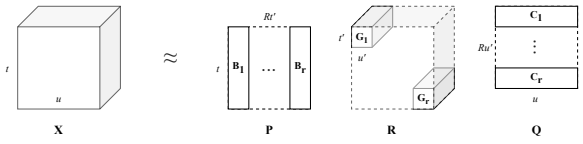


Fig. 1: Formation of bottleneck layers $\mathbf{P}$, $\mathbf{R}$ and $\mathbf{Q}$ from BTD cores and factors.

Table 1: Convolution to grouped bottleneck. The table states how the convolutional parameters are used in the equivalent bottleneck

	Filter Size	C_{in}	C_{out}	Groups	Bias	Stride	Padding
Convolution	$v \times w$	t	u	1	B	S	P
	1×1	t	Rt'	1	-	1	0
Bottleneck	$v \times w$	Rt'	Ru'	R	-	S	P
	1×1	Ru'	u	1	B	1	0

equivalency with the original layer. It is worth noting that we often refer to the quantities t' or u' as the group-size; this quantity determines the number of channels in each group and is equivalent to the rank of the decomposition.

3.3 Group-size Series Decomposition

Group-size Series (GroSS) decomposition unifies multiple ranks of BTD factorisations. This is achieved by defining each successive factorisation relative to the lower order ranks. Thus we ensure that higher rank decompositions only contain information that was missed by the lower order approximations. Therefore the i^{th} approximation of $\mathbf{X}$ is given as follows:

$$\mathbf{X} = \sum_{r=1}^{R_i} [(\mathbf{g}_r)_i + (\mathbf{G}'_r)_{i-1}] \times_1 [(\mathbf{b}_r)_i + (\mathbf{B}'_r)_{i-1}] \times_2 [(\mathbf{c}_r)_i + (\mathbf{C}'_r)_{i-1}]$$

$$\text{where} \begin{cases} (\mathbf{g}_r)_i,\ (\mathbf{G}'_r)_{i-1} \in \mathbb{R}^{t'_i \times u'_i \times v \times w} \\ (\mathbf{b}_r)_i,\ (\mathbf{B}'_r)_{i-1} \in \mathbb{R}^{t \times t'_i} \\ (\mathbf{c}_r)_i,\ (\mathbf{C}'_r)_{i-1} \in \mathbb{R}^{u \times u'_i} \end{cases} \tag{3}$$

Omitting r from the notation, $\mathbf{g}_i$, $\mathbf{b}_i$ and $\mathbf{c}_i$ represent the additional information captured between the $(i-1)^{\text{th}}$ and i^{th} rank of approximation, and $\mathbf{G}'_{(i-1)}$, $\mathbf{B}'_{(i-1)}$ and $\mathbf{C}'_{(i-1)}$ to represent total approximation from lower rank approximations in the form of cores and factors. However, both the core and factors must be recomputed so that the dimensions match the ranks required R_i, which is not a trivial manipulation.

Instead, we introduce a function, $\Psi_{g \to h}()$, which allows the weights of a grouped convolution to be "expanded". The expanded weight from a convolution with group-size g can be used in a convolution with group-size h, where $h > g$, giving identical outputs:

$$\mathbf{W}_g *_g \boldsymbol{F} \equiv \Psi_{g \to h}(\mathbf{W}_g) *_h \boldsymbol{F} \tag{4}$$

where $\mathbf{W}_g$ is the weight for a grouped convolution, $*_g$ refers to convolution with group-size g, and $\boldsymbol{F}$ is the feature map to which the convolution is applied. We provide an example visualisation of $\Psi()$ in Figure 2. This allows us to conveniently reformulate the GroSS decomposition in terms of the successive convolutional weights obtained from BTD, rather than within the cores and factors

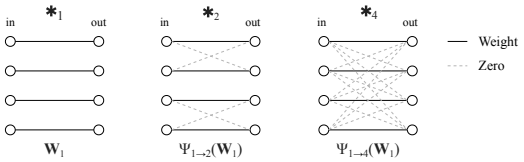


Fig. 2: Visualisation of a depthwise weight, $\mathbf{W}_1$, expanded for convolutions with groups of size 2 and 4.

directly. More specifically, we define the bottleneck weights for the N^{th} order GroSS decomposition with group-sizes, $S = \{s_1, ..., s_N\}$, as follows:

$$\begin{aligned} \mathbf{R}_N &= \Psi_{s_1 \to s_N}(\mathbf{R}_1) + \sum_{i=2}^{N} \Psi_{s_i \to s_N}(\mathbf{r}_i) \\ \mathbf{P}_N = \mathbf{P}_1 + \sum_{i=2}^{N} \mathbf{p}_i, &\quad \mathbf{Q}_N = \mathbf{Q}_1 + \sum_{i=2}^{N} \mathbf{q}_i \end{aligned} \tag{5}$$

$\mathbf{R}_1$, $\mathbf{P}_1$ and $\mathbf{Q}_1$ represent the weights obtained from the lowest rank decomposition present in the series. $\mathbf{r}_i$, $\mathbf{p}_i$ and $\mathbf{q}_i$ represent the additional information that the i^{th} rank decomposition contribute to the bottleneck approximation:

$$\mathbf{p}_i = \mathbf{P}_i - \mathbf{P}_{(i-1)}, \quad \mathbf{r}_i = \mathbf{R}_i - \Psi_{s_{(i-1)} \to s_i}(\mathbf{R}_{(i-1)}), \quad \mathbf{q}_i = \mathbf{Q}_i - \mathbf{Q}_{(i-1)}. \tag{6}$$

This formulation involving only manipulation of the convolutional weights is exactly equivalent to forming the bottleneck components $\mathbf{r}_i$, $\mathbf{p}_i$ and $\mathbf{q}_i$ from $\mathbf{g}_i$, $\mathbf{b}_i$ and $\mathbf{c}_i$, as in the general BTD to bottleneck case.

Further, the grouped convolution weight expansion, $\Psi()$, enables us to dynamically, and differentiably, change the group-size of a convolution. In itself, this is not particularly useful: a convolution with a larger group-size is requires more operations and more memory, while yielding identical outputs. However, it allows for direct interaction between differently ranked network decomposition and, therefore, the representation of one rank by the combination of lower ranks. Thus, GroSS treats the decomposition of the original convolution as the sum of successive order approximations, with each order contributing additional representational power.

Training GroSS Simultaneously The expression of a group-size s_i decomposition as the combination of lower rank decompositions is useful because it enables the group-size to be dynamically changed during training. The expansion and summation of convolutional weights is differentiable and so training at a high rank, also optimises the lower rank approximations simultaneously. To the best of our knowledge GroSS is the first method that allows weight-sharing between, and training of, convolutions with varying numbers of groups.

We leverage the series form of the factorisation during training, by randomly sampling a group-size for each decomposed layer at each iteration. We sample a group-size s_i for each decomposed layer uniformly. Through uniform sampling, we are able to train each network configuration equally.

3.4 Search

The objective for all the searches performed within in this paper is: given a base configuration, we aim to find an alternative configuration which is more accurate, but offers the same or cheaper inference. We implement two forms of search to achieve this leveraging GroSS decomposition: exhaustive and breadth-first.

Exhaustive. Within the exhaustive search, all possible configurations are evaluated. In this search, we simply filter any configuration with multiply accumulates (MACs) above the respective base configuration. After filtering, we can select the highest accuracy remaining.

Breadth-first Search. Where an exhaustive search is not feasible due to the sheer number of possible configurations, we use a greedy breadth-first search. We first randomly select a configuration which requires fewer operations for inference than the base configuration. We evaluate all neighbouring configurations—those which only require one layer to have it's group-size changed—of the currently selected configuration. We select the neighbour with the highest accuracy that does not exceed the number of MACs as the base configuration for the next step. We repeat this step for a maximum of 25 times, or until there are no more accurate neighbours not exceeding the cost of the base configuration.

This is repeated 20 times. The most accurate configuration from all of the 20 runs is considered the result of the search. Since the search for a base configuration with fewer MACs is contained within the search-space with a higher limit, we perform them incrementally. This results in the same search process, but the first 10 runs are initialised using the top-10 highest accuracy results from the smaller search. We found that this generally led to faster stopping.

4 Application of GroSS

In this section, we explain how we apply GroSS to a several models across datasets. We first detail the dataset on which evaluation is conducted. Next, we describe the network architecture on which perform GroSS decomposition. Finally, we list the procedure for the decomposition and fine-tuning.

4.1 Datasets

We perform our experimental evaluation on CIFAR-10 [13] and ImageNet [14]. CIFAR-10 is a dataset consisting of 10 classes. The size of each image is 32×32. In

total there are 60,000 images, which are split into 50,000 train images and 10,000 testing images. We further divide the training set into a training and validation splits with 40,000 and 10,000 images, respectively. ImageNet consists of 1000 classes, with 1.2 million training images and 50,000 validation images. Since the test annotations are not available, we report our accuracy on the validation set.

4.2 Models

In this paper, we perform on three general network architectures: a custom 4-layer network, VGG-16 [21], and ResNet-18 [7]. Here, we provide an overview of the network definitions, with more details in the supplementary material.

Our 4-layer network has four convolutional layers, with output channel dimensions of 32, 32, 64 and 64, followed by two fully-connected layers of size 256 and 10. In our ImageNet experiments, we use a standard VGG-16 and ResNet-18, identical to those in [21] and [7], respectively. However, we make some changes to the fully connected structure in VGG-16 for training and inference on CIFAR-10. The convolutional layers instead followed by a 2×2 max-pooling and two fully-connected layers of size 512 and 10, respectively. A ReLU layer and dropout with probability of 0.5 is applied between the fully-connected layers.

4.3 Decomposition

We perform GroSS decomposition on our small 4-layer network, as well as VGG-16 [21]. In each case we decompose all convolutional layers in the network aside from the first. Unless otherwise stated, we set the bottleneck width equal to the number of input channels. For the 4-layer network, group-sizes are set to all powers of 2 which do not exceed the bottleneck width for that respective layer. This leads to a total of 252 configurations represented by our decomposition. We decompose each layer in VGG-16 and ResNet-18 into 4 group-sizes: (1, 4, 16, 32). This leads to a total of 4^{12} and 4^{16} configurations represented by the decomposed VGG-16 and ResNet-18, respectively.

Our formulation of GroSS decomposition as a series of convolutional weight differences (expanded weights in the case of the grouped convolution), as detailed by Equation 5 means that we are able to use an off-the-shelf BTD framework [12]. For each group-size, we set the stopping criteria for BTD identically: when the decrease in approximation error between steps is below 1×10^{-6} for the 4-layer network and 1×10^{-5} for VGG-16, or 5×10^{5} steps have elapsed. We define approximation error as the Frobenius norm between the original tensor and the product of the BTD cores and factors divided by the Frobenius norm of the original tensor. For the 4-layer network, we perform this decomposition 5 times.

4.4 Fine-tuning

CIFAR-10. After we have performed GroSS decomposition on the network, we then fine-tune on the classification task. For the 4-layer network, we tune

for 150 epochs with a batch-size of 256, an initial learning rate of 0.0001 and momentum 0.9. We decay the learning rate by a factor of 0.1 after both 80 and 120 epochs. For VGG-16, we fine-tune with the same SGD parameters and batch-size, however we train for 200 epochs, and decay the learning rate after 100 and 150 epochs. All network parameters are frozen aside from the GroSS decomposition weights. During training, there is a 0.5 probability of horizontal flipping, zero-padding of size 2 is applied around all borders and a random 32×32 crop is taken from the resulting image.

ImageNet. We decompose VGG-16 and ResNet-18 before funetuning on ImageNet. For VGG-16, we train using SGD for a total of 4 epochs with a batch-size of 128, leading to approximately 10^4 iterations. The initial learning rate is set to 10^{-5}, which is decayed by a factor of 0.1 after 2 epochs. Momentum is set to 0.9. Again, all the network parameters are frozen, aside from the decomposition weights. For ResNet-18, we train for 8 epochs in total with a batch size of 512. The initial learning rate is set to 5×10^{-5}, with decay every 2 epochs The images are resized so that the smallest side is of size 256. During training, the resized images are flipped horizontally with a probability of 0.5 and a random 224×224 crop is taken. During testing, we simply take a centre crop from the resized image, hence evaluating 1-crop accuracy.

Individual Configurations. For the decomposition in the conventional manner, *i.e.* a singular group-size configuration, we decompose using exactly the same routine as with GroSS. However, the fine-tuning schedules are slightly modified. On CIFAR-10, we reduce the schedule for our 4-layer network and our CIFAR VGG-16 to 100 epochs. The initial learning rate is increased to 0.001, and decayed at 80 epochs. On Imagenet, we do not freeze the non-decomposed layers. The VGG-16 configurations have a schedule of 6 epochs, with learning rate decay occurring after every 2 epochs. The initial learning rate is kept at 10^{-4}. For ResNet-18 configurations, we increase batch size to 512 and again unfreeze all layers. We run for a total of 12 epochs, with initial learning rate 10^{-3} and decay after 8 and 10 epochs. Due to this being the conventional BTD factoristation strategy, we often refer to this as the *true* accuracy of a configuration.

5 Results

In this section, we demonstrate the effectiveness of GroSS. First, we explore group-size selection for network acceleration through search on our GroSS decomposition. Secondly, we justify the design of GroSS over a more simply using partial training schedules for group configurations.

5.1 Group-size Search

Here, we evaluate the performance of our search. We split our results by dataset, with our CIFAR-10 results being followed by our results on ImageNet.

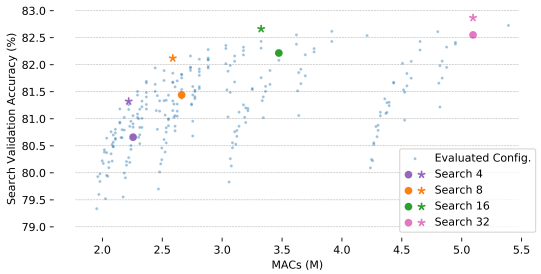


Fig. 3: Exhaustive search for the 4-layer network on CIFAR-10. Each search is colour coded. The circles and stars mark the performance of the baseline and found configuration, respectively.

For each search, we report the change in accuracy of the found configuration over the baseline, as well as the percentage reduction in MACs compared to the baseline. We are primarily concerned with exploring the impact of the number of groups on the performance of the network, rather than other design parameters such as the bottleneck dimensions. In our experimental setup, the inference cost between configurations varies only in the number of groups in each bottleneck. However, there is significant overhead from other layers in the network that remains constant between configurations. Therefore, we report the reduction in total MACs, as well as the contribution from the group layers alone. This provides greater insight into grouped architecture design and the performance of the searches using GroSS.

Table 2: Exhaustive search for our 4-layer network on CIFAR-10

4-Layer Network Configuration	**MACs**	**Accuracy** GroSS	True	**Δ**Acc.	**ΔMACs** Total	G.Conv
Full	5.13M	-	83.99 (0.53)	-	-	-
Baseline: 32 32 32	***5.09M***	*82.55 (0.07)*	*83.70 (0.05)*	-	-	-
32 16 64	**5.09M**	**82.87 (0.14)**	**84.05 (0.07)**	↑ **0.35**	0.00%	0.00%
Baseline: 16 16 16	*3.47M*	*82.22 (0.10)*	*82.94 (0.06)*	-	-	-
VBMF [18]: 16 8 16	**3.33M**	81.76 (0.10)	82.83 (0.10)	↓ 0.11	↓ **4.25%**	↓ **9.09%**
8 16 64	**3.33M**	**82.66 (0.11)**	**83.88 (0.10)**	↑ **0.94**	↓ **4.25%**	↓ **9.09%**
Baseline: 8 8 8	*2.66M*	*81.44 (0.16)*	*82.86 (0.10)*	-	-	-
2 16 32	**2.59M**	**82.12 (0.16)**	**83.50 (0.07)**	↑ **0.64**	↓ **2.77%**	↓ **9.09%**
Baseline: 4 4 4	*2.26M*	*80.66 (0.13)*	*82.37 (0.11)*	-	-	-
1 8 16	**2.22M**	**81.32 (0.16)**	**82.45 (0.15)**	↑ **0.08**	↓ **1.63%**	↓ **9.09%**
Depthwise: 1 1 1	*1.95M*	*79.34 (0.14)*	*81.70 (0.32)*	-	-	-

Table 3: Breadth-first search on our VGG-16 network on CIFAR-10

VGG-16 (CIFAR) Configuration	**MACs**	**Accuracy** GroSS	True	**ΔMACs** ΔAcc.	Total	G.Conv
Full	314M	-	91.52	-	-	-
32 32 32 32 32 32 32 32 32 32 32 32 32	*121M*	*90.97*	***91.57***	-	-	-
VBMF [18]	118M	90.97	91.31	↓ 0.26	↓ 2.68%	↓ **6.18%**
32 4 16 32 16 16 32 16 4 16 32 1	**103M**	**91.31**	91.41	↓ 0.16	↓ **14.6%**	↓ **33.7%**
16 16 16 16 16 16 16 16 16 16 16 16 16	*94.6M*	*91.13*	*91.19*	-	-	-
4 4 16 32 16 16 1 32 4 32 16 1	**86.7M**	**91.28**	**91.31**	↑ **0.12**	↓ **8.36%**	↓ **30.2%**
4 4 4 4 4 4 4 4 4 4 4 4 4	*74.9M*	*90.43*	*90.90*	-	-	-
1 1 1 16 16 4 1 1 4 4 1 4	**74.1M**	**90.97**	**91.14**	↑ **0.24**	↓ **1.11%**	↓ **12.6%**
1 1 1 1 1 1 1 1 1 1 1 1 1	*70.0M*	*90.24*	*90.66*	-	-	-

CIFAR-10. The results of the exhaustive search on the 4-layer network are shown in Table 2, where the decomposition and tune is performed 5 times for each configuration and the mean and standard deviation are reported decompositions with uniform rank values across layers (4, 8, 16, and 32) are chosen as the baseline configurations for the search, such that we perform search across the range of possible configurations. For each baseline configuration we are able to find an alternative that is more accurate whilst requiring fewer operations. The results of the search are also visualised in Figure 3.

In the case of our CIFAR-10 VGG-16 network, the 4^{12} configurations produced by our GroSS decomposition are too many to feasibly enable exhaustive evaluation. We, therefore, perform a breadth-first search. The full details of how this search is performed are described in Section 3.4. Results for this search on VGG-16 are shown in Table 3.

For the searches on the 4, 8 and 16 baselines, we are able to find configurations which meet the objective. However, in the case of the search below the 32 baseline, the found configuration's true accuracy is less than that of the baseline. We speculate that this is because 32 is the maximum rank in the decomposition. Therefore, the rank of each layer can never be increased above that of the baseline. This means that configurations have less room to manoeuvre in targeting more heavy-duty layers at key stages of the network.

We also include a configuration found through Variational Bayesian Matrix Factorisation (VBMF) [18], which is used for one-shot rank selection in [11]. For both networks, we were able to find more accurate configurations which require fewer or the same number of operations than the VBMF rank selection. In fact, Kim *et al.* [11] note that, although they achieve good network compression results with the result of VBMF, they had not investigated whether this method of rank selection was optimal. The results in Table 2 demonstrate that VBMF is not optimal in this case, and GroSS is an effective tool to determine this.

Table 4: Breadth-first search for VGG-16 and ResNet-18 on ImageNet. * denotes the configuration is using the decomposition structure from [26]

ImageNet Configuration	MACs	Accuracy GroSS	Accuracy True	ΔAcc.	ΔMACs Total	ΔMACs G.Conv
VGG-16 (Full)	15.49B	-	71.59	-	-	-
16 16 16 16 16 16 16 16 16 16 16 16	*4.75B*	*70.25*	*70.77*	-	-	-
1 32 32 32 16 1 32 16 32 4 32 16	**4.70B**	**70.40**	**70.82**	↑ **0.04**	↓ **0.99**%	↓ **3.65**%
4 4 4 4 4 4 4 4 4 4 4 4	*3.78B*	*69.73*	*70.51*	-	-	-
1 1 4 4 4 4 32 16 1 1 32 1	**3.78B**	**69.97**	**70.63**	↑ **0.12**	↓ **0.14**%	↓ **1.69**%
1 1 1 1 1 1 1 1 1 1 1 1	*3.54B*	*68.98*	*70.28*	-	-	-
[26]: *11 10 14 9 15 16 16 29 33 56 56 56* *	*1.16B*	*62.64*	*66.85*	-	-	-
11 10 14 9 15 32 64 58 3 56 7 7 *	**1.16B**	**63.11**	**67.22**	↑ **0.37**	↓ **0.39**%	↓ **4.89**%
ResNet-18 (Full)	1.82B	-	69.76	-	-	-
Baseline 16s	*738M*	*60.77*	*65.80*	-	-	-
16 16 4 32 32 32 32 32 16 32 32 32 16 4 32 16	**715M**	**61.25**	**65.84**	↑ **0.04**	↓ **3.18**%	↓ **11.5**%
Baseline 4s	*586M*	*60.02*	*65.46*	-	-	-
1 4 4 1 16 4 1 4 32 4 4 4 16 4 4 4	**585M**	**60.31**	**65.44**	↑ **0.18**	↓ **0.08**%	↓ **0.88**%
1 1 1 1 1 1 1 1 1 1 1 1	*547M*	*58.61*	*65.16*	-	-	-

ImageNet. We now move to a larger, more complex dataset in ImageNet. We perform the same GroSS decomposition and breadth-first search on conventional VGG-16 and ResNet-18 structures. We search against baseline configurations of uniform 4s and 16s. The results are listed in Table 4 and we provide visualisation of the search on VGG-16 in Figure 4.

We also include results for an alternative decomposition structure and group configuration identical to that used in [26], which we detail in the supplementary material. This decomposition structure aggressively reduces widths in bottleneck layers to achieve a large compression ratio. In our search, we are able to show that the original configuration of groups within this structure is not optimal,

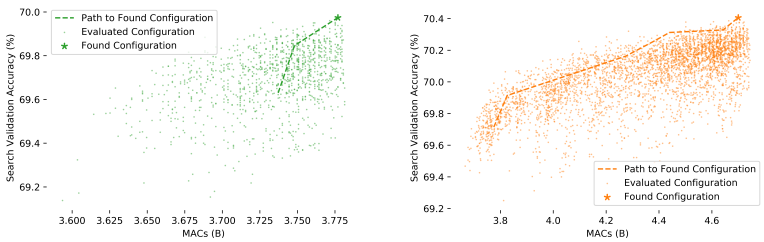


Fig. 4: Visualisation of the VGG-16 breadth-first search on ImageNet. (**Left**) and (**Right**) are the searches against the 4s and 16s baselines, respectively.

with our found configuration leading to a significant improvement in accuracy as well as a slight speed up.

The results show that exploration of group selection with GroSS generalises well across datasets and architectures. In every search performed, we found configurations that met the objective of increased accuracy with lower inference cost.

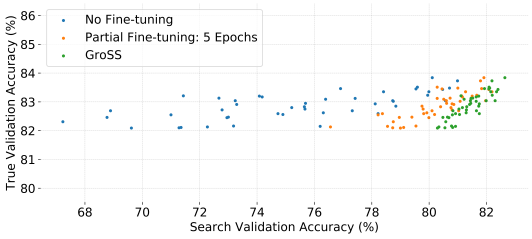


Fig. 5: Search-space vs true validation accuracy for our 4-layer network on CIFAR-10. Here we plot the accuracy of 45 random configurations of our 4-layer network for 3 different methods of obtaining a search-space. The accuracy of each configuration is plotted against its true validation accuracy.

5.2 GroSS Vs. Conventional Fine-tuning

In this section, we justify the need for GroSS by evaluating it against using a partial fine-tuning strategy for each individual configuration. For this, we select 45 random group-size configurations of our 4-layer network and fine-tine according to our individual schedule, which is outlined in Sec. 4.4, giving us a true validation accuracy for each configuration. We can then evaluate the validation accuracy of these same configurations in our GroSS search-space. This procedure allows us to visualise how representative GroSS is of the true accuracy.

For comparison, we also include the validation accuracy of the same configurations with no fine-tuning, as well as with a shortened schedule of 5 epochs. The partial fine-tune could be considered a reasonable solution to reducing the burden of training while performing a configuration search.

We visualise the search-space against the true validation accuracies in Figure 5. Qualitatively, it can be seen that the validation accuracies produced by GroSS produce a significantly more consistent search space. The points appear to be more tightly distributed and closer to the ideal distribution ($y = x$). To measure this quantitatively, we compute the top-5 average precision of the search spaces. We simulate searches across the entire range of configurations by evaluating the average precision at multiple slices through the search-space. This allows for comparison across the space, not just the most accurate group configurations.

Table 5: Average precision across the range of the search space. We compute the average precision using the top 5 true validation accuracies as positive recalls. "$X \downarrow$" refers to the average precision computed after the top-X configurations have been removed from the search

Fine-tune Strategy	**Average Precision (Top 5)**				
	All	**10 ↓**	**20 ↓**	**30 ↓**	**Mean**
No Fine-tuning	43.0	43.8	86.3	69.8	60.7
Partial Fine-tuning	35.2	**64.0**	42.5	66.4	52.0
GroSS	**44.1**	63.8	**94.3**	**82.5**	**71.2**

Table 5 lists the results of this average precision computation. GroSS is consistently as good or better than no fine-tuning and the partial schedule at each slice. This leads to a significant improvement in search performance across the range of configurations which is highlighted by the mean average precision.

When making the comparison between GroSS and a partial training strategy, it is worth considering the computational requirements of each. Running inference for a new configuration in either of the conventional decompositions requires a new network to be initialised, and weights to be loaded. However, since group-sizes are handled dynamically within a GroSS decomposition, switching between them is essentially free, with no structure change or weight loading. This leads to GroSS having a significant speed improvement for running inference (7s vs 287s) over the 45 configurations. This only increases with more configurations tested. For example, the inference for the exhaustive search on the GroSS decomposition of the 4-layer network took only 9s for 252 configurations. Similarly, the total number of training epochs for the partial training strategy increase linearly with the number of configurations, but remain constant for GroSS. With larger search-spaces, such as those visualised in Figure 4, the accuracy and performance benefits of GroSS combine to make grouped architecture search feasible where it might not have been before.

6 Conclusions

In this paper, we have presented GroSS, a series BTD factorisation which allows for the dynamic assignment and simultaneous training of differing numbers of groups within a layer. We have shown how GroSS-decomposed layers can be combined to train an entire grouped convolution search space at once. We confirmed the value of these configurations through an exhaustive search and a breadth-first search. We further demonstrate that, without GroSS, these searches would be less effective and dramatically less efficient.

Acknowledgements We gratefully acknowledge the European Commission Project Multiple-actOrs Virtual Empathic CARegiver for the Elder (MoveCare) for financially supporting the authors for this work.

References

1. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: SMASH: One-shot model architecture search through hypernetworks. arXiv Preprint arXiv:1708.05344 (2017)
2. Chen, Y., Jin, X., Kang, B., Feng, J., Yan, S.: Sharing residual units through collective tensor factorization to improve deep neural networks. In: IJCAI. pp. 635–641 (2018)
3. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1251–1258 (2017)
4. De Lathauwer, L.: Decompositions of a higher-order tensor in block terms—part II: Definitions and uniqueness. SIAM Journal on Matrix Analysis and Applications **30**(3), 1033–1066 (2008)
5. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in Neural Information Processing Systems. pp. 1269–1277 (2014)
6. Elthakeb, A.T., Pilligundla, P., Yazdanbakhsh, A., Kinzer, S., Esmaeilzadeh, H.: ReLeQ: A reinforcement learning approach for deep quantization of neural networks. arXiv Preprint arXiv:1811.01704 (2018)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
8. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv Preprint arXiv:1704.04861 (2017)
9. Ioannou, Y., Robertson, D., Cipolla, R., Criminisi, A.: Deep roots: Improving cnn efficiency with hierarchical filter groups. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1231–1240 (2017)
10. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. arXiv Preprint arXiv:1405.3866 (2014)
11. Kim, Y.D., Park, E., Yoo, S., Choi, T., Yang, L., Shin, D.: Compression of deep convolutional neural networks for fast and low power mobile applications. arXiv Preprint arXiv:1511.06530 (2015)
12. Kossaifi, J., Panagakis, Y., Anandkumar, A., Pantic, M.: Tensorly: Tensor learning in python. The Journal of Machine Learning Research **20**(1), 925–930 (2019)
13. Krizhevsky, A., Nair, V., Hinton, G.: The CIFAR-10 dataset. Online: Http://www. Cs. Toronto. Edu/Kriz/Cifar. HTML **55** (2014)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
15. Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., Lempitsky, V.: Speeding-up convolutional neural networks using fine-tuned CP-decomposition. arXiv Preprint arXiv:1412.6553 (2014)
16. Li, L., Talwalkar, A.: Random search and reproducibility for neural architecture search. arXiv Preprint arXiv:1902.07638 (2019)
17. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable architecture search. arXiv Preprint arXiv:1806.09055 (2018)
18. Nakajima, S., Sugiyama, M., Babacan, S.D., Tomioka, R.: Global analytic solution of fully-observed variational Bayesian matrix factorization. Journal of Machine Learning Research **14**(Jan), 1–37 (2013)

19. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018)
20. Sifre, L., Mallat, S.: Rigid-motion scattering for image classification (2014)
21. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv Preprint arXiv:1409.1556 (2014)
22. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: MnasNet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2820–2828 (2019)
23. Tan, M., Le, Q.V.: EfficientNet: Rethinking model scaling for convolutional neural networks. arXiv Preprint arXiv:1905.11946 (2019)
24. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3), 279–311 (1966)
25. Vanhoucke, V., Senior, A., Mao, M.Z.: Improving the speed of neural networks on CPUs (2011)
26. Wang, P., Hu, Q., Fang, Z., Zhao, C., Cheng, J.: DeepSearch: A fast image search framework for mobile devices. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) **14**(1), 6 (2018)
27. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 10734–10742 (2019)
28. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1492–1500 (2017)
29. Zhang, X., Zhou, X., Lin, M., Sun, J.: ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6848–6856 (2018)
30. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8697–8710 (2018)

GroSS: Supplementary Material

A Appendix

A.1 Network Definitions

In this section, we will provide the explicit definitions of the networks used for our experiments. In Table 1, we detail the structure of our 4-layer network. Table 2 lists the non-standard classifier structure used for VGG-16 on CIFAR-10. All convolutional and fully-connected layers are followed by a ReLU non-linearity, with the exception of the final fully-connected layers in the classifiers.

Table 1: Architecture of our 4-layer network. Each convolution has a 3×3 kernel and is followed by a ReLU non-linearity and a 2×2 max pooling layer

Conv 1	**Conv 2**	**Conv 3**	**Conv 4**	**Classifier**
conv($3 \to 32$)	conv($32 \to 32$)	conv($32 \to 64$)	conv($64 \to 64$)	fc(256→256) fc(256→10)

Table 2: VGG-16 classifier structure for CIFAR-10 and ImageNet

Dataset	**Layers**
CIFAR-10	fc(512→512) fc(512→10)
ImageNet	fc(25088→4096) fc(4096→4096) fc(4096→1000)

A.2 Training From Scratch

4-layer Network. Convolutional weights in the network are initialised with the He initialisation [1] in the "fan out" mode with a ReLU non-linearity. The weights of the fully-connected layers are initialised with a zero-mean, 0.01-variance normal distribution. All bias terms in the network are initialised to 0. The network is trained from scratch on our CIFAR-10 training split for 100 epochs using stochastic gradient descent (SGD). We adopt a initial learning rate of 0.1 and

momentum of 0.9. The learning rate is decayed by a factor of 0.1 after 50 and 75 epochs. We train the network 5 times and use the weights with median accuracy for further experiments.

VGG-16. For our CIFAR-10 variant of VGG-16, the weights are initialised with identical strategy to the 4-layer network. We train this full network on CIFAR-10 for a total of 200 epochs, again using stochastic gradient descent. The initial learning rate is set to 0.05 and momentum to 0.9. The learning rate is decayed by a factor of 0.1 after 100 and 150 epochs. For ImageNet, we take the pretrained model from the Pytorch (Torchvision) [2] model zoo. Specifically, we take the variant without batch-normalisation layers.

ResNet-18. We again make use of the Torchvision model zoo, and use their ResNet-18 model trained on ImageNet.

A.3 High-Compression Decomposition Structure

We recreate the exact structure used for VGG-16 acceleration in [3] with GroSS, which is listed in 3. The constraint that is used in our other experiments, where bottlenecks should be constant width, is relaxed. Since the group-size must be a factor of both bottleneck dimensions (in and out), the bottleneck dimensions chosen by Wang *et al.*do limit the choice of ranks in GroSS. We perform decomposition as with our other VGG-16 experiments, without any bells or whistles. We found that a longer fine-tuning schedule was required to best recover accuracy. Therefore, finetuning consists of 14 epochs with a learning rate of 5×10^{-4} and decay after 8 and 12 epochs. This led to an accuracy consistent with [3].

Table 3: Decomposition structure as used in [3]. Group-sizes marked with * represent the original choice

Layer	Bottleneck (in → out)	Group-sizes
`conv1_2`	11 → 18	11*
`conv2_1`	10 → 24	5 10*
`conv2_2`	28 → 28	1 7 14* 28
`conv3_1`	36 → 48	3 9* 18
`conv3_2`	60 → 48	15* 30 60
`conv3_3`	64 → 56	16* 32 64
`conv4_1`	64 → 100	16* 32 64
`conv4_2`	116 → 100	29* 58 116
`conv4_3`	132 → 132	3 33* 66
`conv5_1`	224 → 224	7 28 56* 112
`conv5_2`	224 → 224	7 28 56* 112
`conv5_3`	224 → 224	7 28 56* 112

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1026–1034 (2015)
2. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
3. Wang, P., Hu, Q., Fang, Z., Zhao, C., Cheng, J.: DeepSearch: A fast image search framework for mobile devices. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) **14**(1), 6 (2018)

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	GroSS: Group-Size Series Decomposition for Grouped Architecture Search
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	**Howard-Jenkins, H.,** Li, Y., and Prisacariu, V. A. (2020). GroSS: Group-Size Series Decomposition for Grouped Architecture Search. In *European Conference on Computer Vision (ECCV)*.

Student Confirmation

Student Name:	Henry Howard-Jenkins		
Contribution to the Paper	• Conception and validation of the idea • Design of the architecture and framework • Implementation and running of all experiments included in the paper. • Joint research of prior work • Writing and presentation of the paper		
Signature	HHJ	Date	31/01/2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Prof Victor Adrian Prisacariu		
Supervisor comments	The description is accurate		
Signature	VPA	Date	09/02/2022

This completed form should be included in the thesis, at the end of the relevant chapter.

7.3 Conclusion

Within this chapter, we presented GroSS as a method to formalise the design process for network architectures of differing group convolutional structure. We validated it in the context of network factorisation, where we were consistently able to find more accurate group topologies that have cheaper inference.

In Chapter 6, our design of Mobile R-CNN demonstrated that through manipulation of group structure alone there are significant gains to be had with minimal additional cost. However, the design process relied heavily on limited heuristics with little opportunity for validation (due to the prohibitive training cost to thoroughly explore the design space). We believe that this difficulty in exploration has been reflected in the literature, where group convolutions are frequently limited to just their depthwise case, or to a simple and homogeneous design across the network, such as in the case of ResNeXt [Xie et al., 2017].

Through its novel expression of group convolutions as decision blocks, where the rank can be changed on-the-fly and weights shared between ranks, GroSS was able to offer a framework for more comprehensive exploration of the group convolution design space. We have demonstrated that it is a useful tool for designing group topology across a network, as well as to validate the existing methods for this design. Ultimately, GroSS makes the training and evaluation of group design feasible where it may not have been before.

Chapter 8

Conclusion

Each chapter within this thesis has presented a conclusion to the work it contained. Generally, this conclusion has been targeted towards providing a brief summary of the work completed, but often with a particular view towards bridging the narrative between chapters when appropriate. Although, this offers a very fair reflection of the motivation that guided the work covered in the thesis, it can, however, limit discussion of the contributions offered in each chapter. In this section, we will discuss the main accomplishments of the work described in this thesis with a view towards wider discussion, other methods that have followed our own and scope for future development.

8.1 Achievements and Future Work

Detection and Regression of 3D Planes. In Chapter 3, we proposed the reformulation of room layout estimation as a detection task. In doing so, our Planar R-CNN was the first, to the best of our knowledge, detection-based paradigm for prediction of planar structure in a image. Most prior detection methods had focused solely on the detection of *things*, *i.e.* objects or living beings that are countable. However, for our task the semantic categories of interest, *e.g.* walls, floor and ceiling, would mostly be considered uncountable *stuff* for which semantic segmentation methods would generally be preferred. This is reflected in prior deep learning methods for inference planar structure in images being conducted as a segmentation task [Liu et al., 2018a; Yang and Zhou, 2018]. After our work was published—in what was likely concurrent work—Liu et al. [2019] introduced PlaneRCNN, which also leveraged the detection paradigm for planar structure inference in images, albeit targeting detection of all visible planes for depth estimation, rather than architectural ones for room layout estimation.

Unconstrained Room Layout Estimation from a Single Perspective View. To the best of our knowledge, Planar R-CNN in Chapter 3 also was the first method to offer recovery of general room layout from single perspective images. Since layout components can often occlude each other, either wholly or partially, room layout estimation from perspective images can be a very challenging task. Methods that approached the task of room layout previously, particularly those which are deep learning based, dealt with this difficulty by imposing strong assumptions about room geometry, such as that the room is a simple 3D cuboid [Lee et al., 2017; Mallya and Lazebnik, 2015; Schwing and Urtasun, 2012]. In fact, this assumption remains common even in more recent work [Chen et al., 2019b; Hirzer et al., 2020; Huang et al., 2018; Zhang et al., 2019]. However, it is easy to think of room layouts which do not fit this box approximation. Our work addressed this through its detection paradigm for layout estimation, and thus was the perhaps the first able to predict informative layouts, both as a 2D segmentation and as a 3D projection, without having to apply constraints on room geometry (other than that its structure is planar). Recently, this challenging task has received a growing level of interest [Yang et al., 2022; Zhang et al., 2021], with Stekovic et al. [2020] notably leveraging plane detection in their method.

Multi-modal Implicit Representation of Room Layout. In Chapter 4, we first demonstrated that, rather than requiring explicit prediction, unconstrained layouts can be represented implicitly. While Kim et al. [2019] introduced the concept of a metric latent space for representing inherent similarity of layout structure across images, their exploration was limited to box rooms and only considered views belonging to one layout *category* at that. We conducted evaluation of multiple representations of room layout to uncover 3D Chamfer Distance as an extremely expressive metric to quantify the inherent similarity between these unconstrained room layouts. In Chapter 5, we furthered this work by demonstrating that implicit 3D room layout representations could be inferred directly from 2D floor plans, which opens up the possibility of leveraging the vast quantity of architectural plans we already have for the built environment.

An interesting extension for these architectural representations, and particularly the structural comprehension from floor plans in Chapter 5, would be to condition large-scale indoor scene generation and view synthesis. Recently, DeVries et al. [2021] introduce a method for generation of, and view synthesis within, complex scenes through the local conditioning of NeRF [Mildenhall et al., 2020] radiance fields. The

local codes take the form of a 2D grid, which is itself generated by sampling from a general learned prior of scenes and decoding. We envisage that this style of generation combined with the ability to implicitly reason 3D structure from 2D floor plans demonstrated in Chapter 5 could enable realistic scene and view generation from specific architectural plans. Further, in ever changing domestic environments, decoupling the representation of a scene's architectural and content (*i.e.* furniture and other clutter) appearance would allow for a principled update of the learned map of a scene such that only non-permanent structure should change.

Cross-modal Matching and Alignment of Layouts for Localisation. In Chapters 4 and 5, we present methods to perform localisation in scenes that have not previously been visited. Each method hinges on the ability to learn a generalised concept of room layout that can be inferred from multiple data modes, namely across both images and structure inferred from floor plans. In Chapter 4, we showed the previously discussed implicit representation of room layout offered a highly expressive layout space which could be accurately reasoned across both panorama images and layout renderings. Notably, we propose fine-grained cross-modal alignment between the images and renderings through a gradient-based optimisation enabled by differentiable rendering. Although differentiable rendering had previously been shown to be effective for object pose estimation [Chen et al., 2019a; Palazzi et al., 2018], it required a like-for-like rendering loss, *e.g.* pixelwise error between rendered and target images, depths or silhouettes. Therefore, these methods rely on the expensive capture of very high-fidelity textured models of the target object and thus have seen limited success in practice. We, however, demonstrated that the latent space itself is expressive enough as an alignment for accurate pose estimation. In doing so, our method was able to perform accurate localisation between images and a low-fidelity representation of the environment, *i.e.* its floor plan. On top of validating its feasibility, we demonstrated that this implicit formulation outperformed alignment of explicit predictions of layouts, further cementing the value of this high-level structural understanding learned in our methods.

Our work in Chapter 5 extended this further. We demonstrated that rendering could be removed entirely, with the network instead able to hallucinate the 3D structure directly from a 2D floor plan. In contrast to Chapter 4 where rendering was used as an explicit translation between the top-down floor plan and an egocentric rendering of layout, this hallucination was implicit and structural inference was still reasoned solely in the layout latent space.

The localisation performance exhibited by our method demonstrates that the high-level structural reasoning is able to bridge extreme disparity between views as well as fidelities. This is an exciting result as it provides a step towards better reflecting human-level reasoning about our surroundings. Using an example from the chapter to illustrate this, one is able to determine an exit route from a fire safety map without having to explicitly imagine an egocentric view at each junction. In practical terms, this work brought with it a significant improvement in accuracy (37% and 29% increase on Structured3D [Zheng et al., 2020a] and Zillow Indoor Dataset [Cruz et al., 2021], respectively) and a dramatic improvement in speed (37× speed up) over the rendering-based method, all while requiring fewer assumptions about the scene.

We hope that these contributions will be applied to other domains, such as object pose estimation, enabling application with prior models that are either cheap to acquire or, similar to floor plans for indoor environments, priors that *have already been acquired*. In addition, it would be interesting to see how this methodology could be applied on a larger scale. One recent example of this is the work conducted by Tomešek et al. [2022] for localisation in the Alps. They follow a rendering-based pipeline leveraging pre-existing topological data to localise images in the mountainscape.

Real-time Instance Segmentation. In Chapter 6, we presented Mobile R-CNN, which is a high accuracy, real-time instance segmentation method. It was among the first methods that could achieve accuracy inline with results that had been accomplished in real-time object detection, *e.g.* [Redmon and Farhadi, 2018], while also being able to predict object masks. With a remarkably simple approach, Mobile R-CNN achieves its fast and accurate inference with the same general inference structure as Mask R-CNN [He et al., 2017]. Specifically, we created an accurate and low-latency backbone which exploits interesting real-world performance characteristics of group convolutions. This is in contrast to other prior and concurrent pieces of work on real-time instance segmentation, which rely on significant reformulation away from the two-stage R-CNN [Bolya et al., 2019b; Jetley et al., 2017; Uhrig et al., 2018]. Although Mobile R-CNN was not published, the code was made available leading to it being used for real-time set art at Coachella in 2019, as well as it forming a core component of the Horizon 2020 MoveCare project.

Since Mobile R-CNN's development, real-time instance segmentation has received a great level of interest as a research task. This has led to a number of very recent methods that are able to offer significantly higher accuracy at real-time speeds [Bolya

et al., 2019a; Cao et al., 2020; Chen et al., 2020; Lee and Park, 2020]. However, we believe that these new methods do not detract from the insight gained from the development of Mobile R-CNN, namely that group convolution can be a very influential design tool in neural networks and, importantly, one that had largely been overlooked.

Enabling Comprehensive Exploration of Group Convolution. In Chapter 7 we presented GroSS, a sum series formulation of group convolution such that weights can be shared between multiple ranks for group convolution. We leveraged GroSS for the purpose of network accleration through BTD network factorisation. We showed how GroSS enabled the dynamic assignment and simultaneous training of groups within a layer and therefore non-standard topologies across the network. Prior network design [Chollet, 2017; Howard et al., 2017; Xie et al., 2017] or acceleration [Kim et al., 2015; Wang et al., 2018a] that leveraged group convolutions had generally relied on simple topologies or heuristics for rank selection with limited (or no) validation of the choice. However, we demonstrated that, with GroSS, we are able to train an entire search space of group topologies at once. We confirmed the value of these configurations through remarkably simple search strategies to uncover network structure with the desired performance characteristics. We further demonstrated that, without GroSS, these searches would be less effective and dramatically less efficient. Ultimately, GroSS enables exploration and validation of group convolutional designs that may not have been feasible before.

We hope that GroSS will be applied beyond the narrow test cases that we explored in Chapter 7. For example, it could be used to validate the cardinality design choice of networks such as ResNeXt [Xie et al., 2017], or in an adaptive run-time complexity selection method [Wang et al., 2018b; Wu et al., 2018] where group rank can be selected dynamically based on the predicted difficulty of the task. Most of all, we hope that GroSS is leveraged, or at least serves as inspiration, for more nuanced group structures in deep neural networks.

Bibliography

Anantharaman, R., Velazquez, M., and Lee, Y. (2018). Utilizing mask r-cnn for detection and segmentation of oral diseases. In *2018 IEEE international conference on bioinformatics and biomedicine (BIBM)*, pages 2197–2204. IEEE.

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307.

Armagan, A., Hirzer, M., Roth, P. M., and Lepetit, V. (2017). Learning to align semantic segmentation and 2.5 d maps for geolocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3425–3432.

Arth, C., Pirchheim, C., Ventura, J., Schmalstieg, D., and Lepetit, V. (2015). Instant outdoor localization and slam initialization from 2.5 d maps. *IEEE Transactions on Visualization & Computer Graphics*, 21(11):1309–1318.

Balntas, V., Li, S., and Prisacariu, V. (2018). Relocnet: Continuous metric learning relocalisation using neural nets. *European Conference on Computer Vision*.

Bender, G., Kindermans, P.-J., Zoph, B., Vasudevan, V., and Le, Q. (2018). Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559. PMLR.

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.

Blum, H., Stiefel, J., Cadena, C., Siegwart, R., and Gawel, A. (2020). Precise robot localization in architectural 3d plans. *arXiv preprint arXiv:2006.05137*.

Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. (2019a). Yolact++: Better real-time instance segmentation. *arXiv preprint arXiv:1912.06218*.

Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. (2019b). Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9157–9166.

Boniardi, F., Valada, A., Mohan, R., Caselitz, T., and Burgard, W. (2019). Robot localization in floor plans using a room layout edge extraction network. *arXiv preprint arXiv:1903.01804*.

Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., and Rother, C. (2017). Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692.

Brachmann, E. and Rother, C. (2018). Learning less is more-6d camera localization via 3d surface regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4654–4662.

Brachmann, E. and Rother, C. (2019). Expert sample consensus applied to camera re-localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7525–7534.

Brahmbhatt, S., Gu, J., Kim, K., Hays, J., and Kautz, J. (2018). Geometry-aware learning of maps for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625.

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. (2017). Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*.

Budvytis, I., Sauer, P., and Cipolla, R. (2019a). Semantic localisation via globally unique instance segmentation. In *British Machine Vision Conference 2018, BMVC 2018*.

Budvytis, I., Teichmann, M., Vojir, T., and Cipolla, R. (2019b). Large scale joint semantic re-localisation and scene understanding via globally unique instance coordinate regression. *arXiv preprint arXiv:1909.10239*.

Cabral, R. and Furukawa, Y. (2014). Piecewise planar and compact floorplan reconstruction from images. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 628–635. IEEE.

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

Cao, J., Anwer, R. M., Cholakkal, H., Khan, F. S., Pang, Y., and Shao, L. (2020). Sipmask: Spatial information preservation for fast image and video instance segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 1–18. Springer.

Cavallari, T., Golodetz, S., Lord, N. A., Valentin, J. P., Di Stefano, L., and Torr, P. H. (2017). On-the-fly adaptation of regression forests for online camera relocalisation. In *CVPR*, volume 2, page 7.

Chen, H., Sun, K., Tian, Z., Shen, C., Huang, Y., and Yan, Y. (2020). Blendmask: Top-down meets bottom-up for instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8573–8581.

Chen, S., Wang, Z., and Prisacariu, V. (2021). Direct-posenet: Absolute pose regression with photometric consistency. *arXiv preprint arXiv:2104.04073*.

Chen, W., Gao, J., Ling, H., Smith, E. J., Lehtinen, J., Jacobson, A., and Fidler, S. (2019a). Learning to predict 3d objects with an interpolation-based differentiable renderer. *arXiv preprint arXiv:1908.01210*.

Chen, Y., Huang, S., Yuan, T., Qi, S., Zhu, Y., and Zhu, S.-C. (2019b). Holistic++ scene understanding: Single-view 3d holistic scene parsing and human pose estimation with human-object interaction and physical commonsense. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8648–8657.

Chen, Y., Jin, X., Kang, B., Feng, J., and Yan, S. (2018). Sharing residual units through collective tensor factorization to improve deep neural networks. In *IJCAI*, pages 635–641.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

Coughlan, J. M. and Yuille, A. L. (1999). Manhattan world: Compass direction from a single image by bayesian inference. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 941–947. IEEE.

Cruz, S., Hutchcroft, W., Li, Y., Khosravan, N., Boyadzhiev, I., and Kang, S. B. (2021). Zillow indoor dataset: Annotated floor plans with 360deg panoramas and 3d room layouts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2133–2143.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. A., and Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, volume 2, page 10.

Danielczuk, M., Matl, M., Gupta, S., Li, A., Lee, A., Mahler, J., and Goldberg, K. (2019). Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7283–7290. IEEE.

Dasgupta, S., Fang, K., Chen, K., and Savarese, S. (2016). Delay: Robust spatial layout estimation for cluttered indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 616–624.

De Brabandere, B., Neven, D., and Van Gool, L. (2017). Semantic instance segmentation for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 7–9.

De Lathauwer, L. (2008). Decompositions of a higher-order tensor in block terms—part ii: Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1033–1066.

De Lathauwer, L., De Moor, B., and Vandewalle, J. (2000). On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM journal on Matrix Analysis and Applications*, 21(4):1324–1342.

DeVries, T., Bautista, M. A., Srivastava, N., Taylor, G. W., and Susskind, J. M. (2021). Unconstrained scene generation with locally conditioned radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14304–14313.

Ding, M., Wang, Z., Sun, J., Shi, J., and Luo, P. (2019). Camnet: Coarse-to-fine retrieval for camera re-localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2871–2880.

Elthakeb, A. T., Pilligundla, P., Yazdanbakhsh, A., Kinzer, S., and Esmaeilzadeh, H. (2018). Releq: A reinforcement learning approach for deep quantization of neural networks. *arXiv preprint arXiv:1811.01704*.

Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer.

Ethayarajh, K. and Jurafsky, D. (2020). Utility is in the eye of the user: A critique of nlp leaderboards. *arXiv preprint arXiv:2009.13888*.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Flint, A., Murray, D., and Reid, I. (2011). Manhattan scene understanding using monocular, stereo, and 3d features. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2228–2235. IEEE.

Glocker, B., Izadi, S., Shotton, J., and Criminisi, A. (2013). Real-time rgb-d camera relocalization. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 173–179. IEEE.

Gupta, A., Hebert, M., Kanade, T., and Blei, D. M. (2010). Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *Advances in neural information processing systems*, pages 1288–1296.

Gupta, S. and Akin, B. (2020). Accelerator-aware neural network design using automl. *arXiv preprint arXiv:2003.02838*.

Ha, D., Dai, A., and Le, Q. V. (2016). Hypernetworks. *arXiv preprint arXiv:1609.09106*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hedau, V., Hoiem, D., and Forsyth, D. (2009). Recovering the spatial layout of cluttered rooms. In *Computer vision, 2009 IEEE 12th international conference on*, pages 1849–1856. IEEE.

Hentschel, M. and Wagner, B. (2010). Autonomous robot navigation based on openstreetmap geodata. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1645–1650. IEEE.

Hile, H. and Borriello, G. (2008). Positioning and orientation in indoor environments using camera phones. *IEEE Computer Graphics and Applications*, 28(4):32–39.

Hirzer, M., Lepetit, V., and ROTH, P. (2020). Smart hypothesis generation for efficient and robust room layout estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2912–2920.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, S., Feng, M., Nguyen, R. M., and Lee, G. H. (2018). Cvm-net: Cross-view matching network for image-based ground-to-aerial geo-localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7258–7267.

Hu, S. and Lee, G. H. (2020). Image-based geo-localization using satellite imagery. *International Journal of Computer Vision*, 128(5):1205–1219.

Huang, S., Qi, S., Xiao, Y., Zhu, Y., Wu, Y. N., and Zhu, S.-C. (2018). Cooperative holistic scene understanding: Unifying 3d object, layout, and camera pose estimation. *arXiv preprint arXiv:1810.13049*.

Huang, Z., Huang, L., Gong, Y., Huang, C., and Wang, X. (2019). Mask scoring r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6409–6418.

Ioannou, Y., Robertson, D., Cipolla, R., and Criminisi, A. (2017). Deep roots: Improving cnn efficiency with hierarchical filter groups. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1231–1240.

Irschara, A., Zach, C., Frahm, J.-M., and Bischof, H. (2009). From structure-from-motion point clouds to fast location recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2599–2606. IEEE.

Izadinia, H., Shan, Q., and Seitz, S. M. (2017). Im2cad. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2422–2431. IEEE.

Jaderberg, M., Vedaldi, A., and Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*.

Jetley, S., Sapienza, M., Golodetz, S., and Torr, P. H. (2017). Straight to shapes: real-time detection of encoded shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6550–6559.

Jin, H., Song, Q., and Hu, X. (2019). Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1946–1956.

Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., and Xing, E. P. (2018). Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems*, 31.

Kendall, A., Cipolla, R., et al. (2017). Geometric loss functions for camera pose regression with deep learning. In *Proc. CVPR*, volume 3, page 8.

Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946.

Kim, S., Seo, M., Laptev, I., Cho, M., and Kwak, S. (2019). Deep metric learning beyond binary supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2288–2297.

Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L., and Shin, D. (2015). Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*.

Krizhevsky, A., Nair, V., and Hinton, G. (2014). The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lateef, F. and Ruichek, Y. (2019). Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, 338:321–348.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V. (2014). Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*.

Lee, C.-Y., Badrinarayanan, V., Malisiewicz, T., and Rabinovich, A. (2017). Roomnet: End-to-end room layout estimation. *arXiv preprint arXiv:1703.06241*.

Lee, D. C., Hebert, M., and Kanade, T. (2009). Geometric reasoning for single image structure recovery. In *2009 IEEE conference on computer vision and pattern recognition*, pages 2136–2143. IEEE.

Lee, Y. and Park, J. (2020). Centermask: Real-time anchor-free instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13906–13915.

Leung, K. Y. K., Clark, C. M., and Huissoon, J. P. (2008). Localization in urban environments by matching ground level video images with an aerial image. In *2008 IEEE International Conference on Robotics and Automation*, pages 551–556. IEEE.

Li, L. and Talwalkar, A. (2020). Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pages 367–377. PMLR.

Lin, T.-Y., Cui, Y., Belongie, S., and Hays, J. (2015). Learning deep representations for ground-to-aerial geolocalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5007–5015.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, C., Kim, K., Gu, J., Furukawa, Y., and Kautz, J. (2019). Planercnn: 3d plane detection and reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4450–4459.

Liu, C., Yang, J., Ceylan, D., Yumer, E., and Furukawa, Y. (2018a). Planenet: Piece-wise planar reconstruction from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2588.

Liu, H., Simonyan, K., and Yang, Y. (2018b). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. (2018c). Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Lukierski, R., Leutenegger, S., and Davison, A. J. (2017). Room layout estimation from rapid omnidirectional exploration. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6315–6322. IEEE.

Mallya, A. and Lazebnik, S. (2015). Learning informative edge maps for indoor scene layout prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 936–944.

Massa, F. and Girshick, R. (2018). maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer.

Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

Ng, T., Lopez-Rodriguez, A., Balntas, V., and Mikolajczyk, K. (2021). Reassessing the limitations of cnn methods for camera pose regression. *arXiv preprint arXiv:2108.07260*.

Noda, M., Takahashi, T., Deguchi, D., Ide, I., Murase, H., Kojima, Y., and Naito, T. (2010). Vehicle ego-localization by matching in-vehicle camera images to an aerial image. In *Asian Conference on Computer Vision*, pages 163–173. Springer.

Palazzi, A., Bergamini, L., Calderara, S., and Cucchiara, R. (2018). End-to-end 6-dof object pose estimation through differentiable rasterization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pink, O. (2008). Visual map matching and localization using a global feature map. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–7. IEEE.

Pomerleau, F., Colas, F., and Siegwart, R. (2015). A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104.

Prisacariu, V. A., Kähler, O., Golodetz, S., Sapienza, M., Cavallari, T., Torr, P. H., and Murray, D. W. (2017). Infinitam v3: A framework for large-scale 3d reconstruction with loop closure. *arXiv preprint arXiv:1708.00783*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Rigamonti, R., Sironi, A., Lepetit, V., and Fua, P. (2013). Learning separable filters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2754–2761.

Runz, M., Buffier, M., and Agapito, L. (2018). Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20. IEEE.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520.

Sarlin, P.-E., Cadena, C., Siegwart, R., and Dymczyk, M. (2019). From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12716–12725.

Sarlin, P.-E., Debraine, F., Dymczyk, M., Siegwart, R., and Cadena, C. (2018). Leveraging deep visual descriptors for hierarchical efficient localization. In *Conference on Robot Learning*, pages 456–465. PMLR.

Sarlin, P.-E., DeTone, D., Malisiewicz, T., and Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947.

Sattler, T., Leibe, B., and Kobbelt, L. (2017). Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):1744–1756.

Sattler, T., Maddern, W., Toft, C., Torii, A., Hammarstrand, L., Stenborg, E., Safari, D., Okutomi, M., Pollefeys, M., Sivic, J., et al. (2018). Benchmarking 6dof outdoor visual localization in changing conditions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8601–8610.

Sattler, T., Zhou, Q., Pollefeys, M., and Leal-Taixe, L. (2019). Understanding the limitations of cnn-based absolute camera pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3302–3312.

Schindler, G., Brown, M., and Szeliski, R. (2007). City-scale location recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE.

Schwing, A. G. and Urtasun, R. (2012). Efficient exact inference for 3d indoor scene understanding. In *European Conference on Computer Vision*, pages 299–313. Springer.

Selvam, S., Ganesan, V., and Kumar, P. (2021). Fuseconv: Fully separable convolutions for fast inference on systolic arrays. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 651–656. IEEE.

Senlet, T. and Elgammal, A. (2011). A framework for global vehicle localization using stereo images and satellite and road maps. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 2034–2041. IEEE.

Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., and Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937.

Sifre, L. and Mallat, S. (2014). Rigid-motion scattering for texture classification. *arXiv preprint arXiv:1403.1687*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Stekovic, S., Hampali, S., Rad, M., Sarkar, S. D., Fraundorfer, F., and Lepetit, V. (2020). General 3d room layout from a single view by render-and-compare. In *European Conference on Computer Vision*, pages 187–203. Springer.

Sun, C., Hsiao, C.-W., Sun, M., and Chen, H.-T. (2019). Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1047–1056.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.

Thrun, S. (2002). Probabilistic robotics. *Communications of the ACM*, 45(3):52–57.

Tomešek, J., Čadík, M., and Brejcha, J. (2022). Crosslocate: Cross-modal large-scale visual geo-localization in natural environments using rendered modalities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3174–3183.

Uhrig, J., Rehder, E., Fröhlich, B., Franke, U., and Brox, T. (2018). Box2pix: Single-shot instance segmentation by assigning pixels to object boxes. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 292–299. IEEE.

Unicomb, J., Ranasinghe, R., Dantanarayana, L., and Dissanayake, G. (2018). A monocular indoor localiser based on an extended kalman filter and edge images from a convolutional neural network. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE.

Valentin, J., Nießner, M., Shotton, J., Fitzgibbon, A., Izadi, S., and Torr, P. H. (2015). Exploiting uncertainty in regression forests for accurate camera relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4400–4408.

Vojir, T., Budvytis, I., and Cipolla, R. (2020). Efficient large-scale semantic visual localization in 2d maps. In *Proceedings of the Asian Conference on Computer Vision*.

Wald, J., Sattler, T., Golodetz, S., Cavallari, T., and Tombari, F. (2020). Beyond controlled environments: 3d camera re-localization in changing indoor scenes. In *European Conference on Computer Vision*, pages 467–487. Springer.

Wang, P., Hu, Q., Fang, Z., Zhao, C., and Cheng, J. (2018a). Deepsearch: A fast image search framework for mobile devices. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(1):6.

Wang, S., Fidler, S., and Urtasun, R. (2015). Lost shopping! monocular localization in large indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2695–2703.

Wang, X., Yu, F., Dou, Z.-Y., Darrell, T., and Gonzalez, J. E. (2018b). Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424.

Watanabe, Y., Amaro, K. R., Ilhan, B., Kinoshita, T., Bock, T., and Cheng, G. (2020). Robust localization with architectural floor plans and depth camera. In *2020 IEEE/SICE International Symposium on System Integration (SII)*, pages 133–138. IEEE.

Winterhalter, W., Fleckenstein, F., Steder, B., Spinello, L., and Burgard, W. (2015). Accurate indoor localization for rgb-d smartphones and tablets given 2d floor plans. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3138–3143. IEEE.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. (2019). Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742.

Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K., and Feris, R. (2018). Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826.

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

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Xu, J., Stenger, B., Kerola, T., and Tung, T. (2017). Pano2cad: Room layout from a single panorama image. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 354–362. IEEE.

Yang, C., Zheng, J., Dai, X., Tang, R., Ma, Y., and Yuan, X. (2022). Learning to reconstruct 3d non-cuboid room layout from a single rgb image. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2534–2543.

Yang, F. and Zhou, Z. (2018). Recovering 3d planes from a single image via convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100.

Yunpeng, C., Xiaojie, J., Bingyi, K., Jiashi, F., and Shuicheng, Y. (2017). Sharing residual units through collective tensor factorization in deep neural networks. *arXiv preprint arXiv:1703.02180*.

Zhang, J., Kan, C., Schwing, A. G., and Urtasun, R. (2013). Estimating the 3d layout of indoor scenes and its clutter from depth sensors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1273–1280.

Zhang, W., Zhang, W., and Gu, J. (2019). Edge-semantic learning strategy for layout estimation in indoor environment. *IEEE transactions on cybernetics*, 50(6):2730–2739.

Zhang, X., Wang, C.-K., Deng, K., Yago-Vicente, T., and Arora, H. (2021). Roomstructnet: Learning to rank non-cuboidal room layouts from single view. *arXiv preprint arXiv:2110.00644*.

Zhang, X., Zhou, X., Lin, M., and Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856.

Zhang, Y., Song, S., Tan, P., and Xiao, J. (2014). Panocontext: A whole-room 3d context model for panoramic scene understanding. In *European Conference on Computer Vision*, pages 668–686. Springer.

Zhang, Y., Yu, F., Song, S., Xu, P., Seff, A., and Xiao, J. (2015). Large-scale scene understanding challenge: Room layout estimation. *accessed on Sep*, 15.

Zheng, J., Zhang, J., Li, J., Tang, R., Gao, S., and Zhou, Z. (2020a). Structured3d: A large photo-realistic dataset for structured 3d modeling. In *European Conference on Computer Vision*, pages 519–535. Springer.

Zheng, W., Lu, J., and Zhou, J. (2020b). Structural deep metric learning for room layout estimation. In *European Conference on Computer Vision*, pages 735–751. Springer.

Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710.

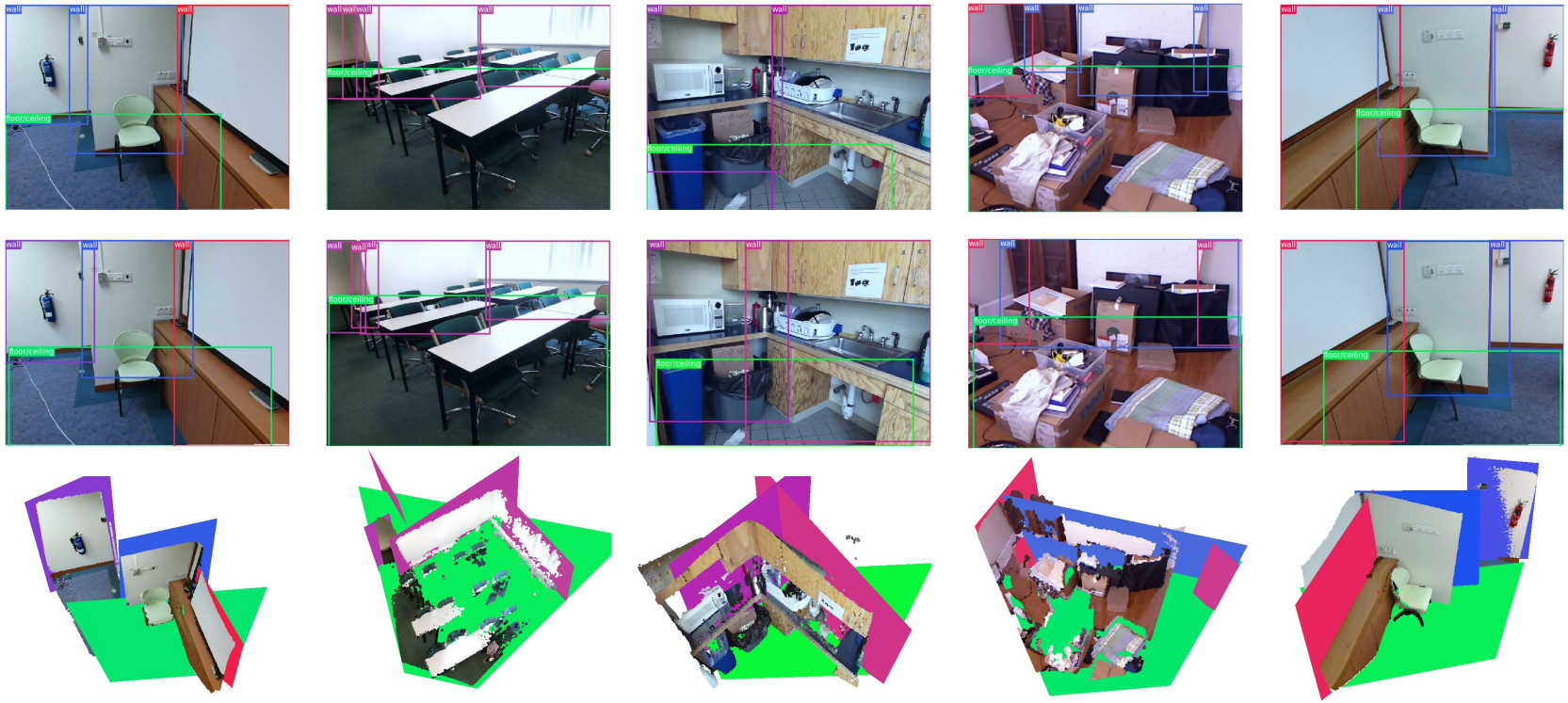
Zou, C., Colburn, A., Shan, Q., and Hoiem, D. (2018). Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2059.

Zou, Z., Shi, Z., Guo, Y., and Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*.

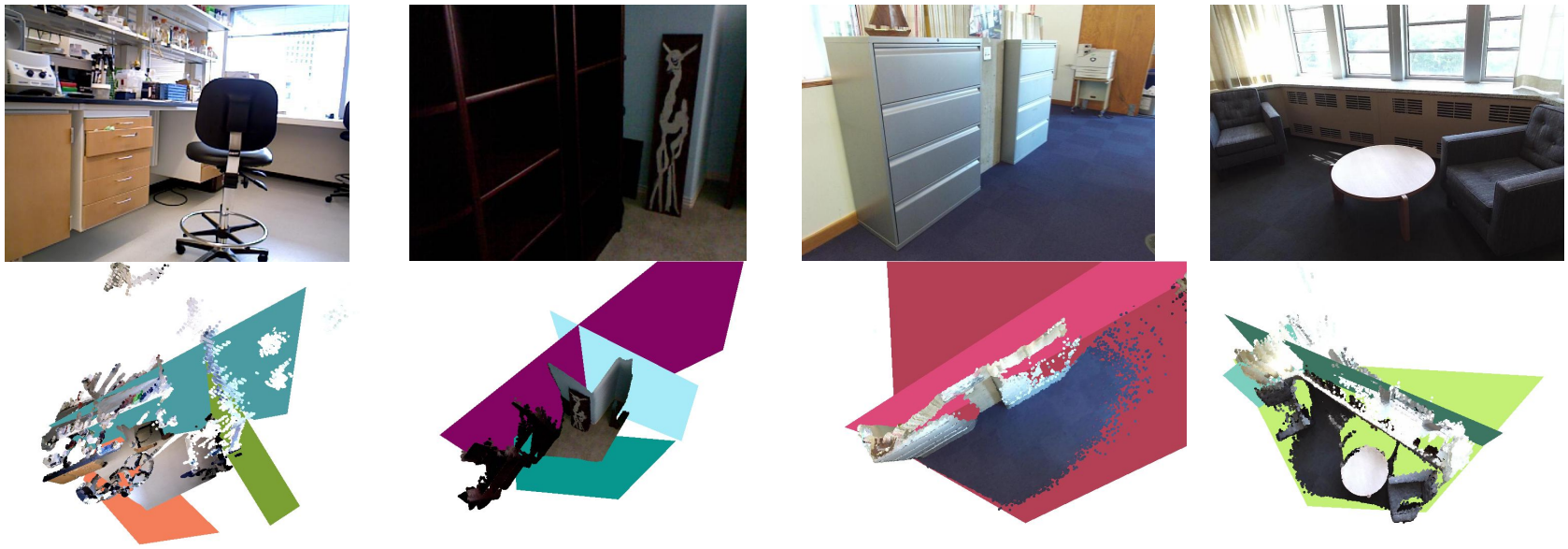
Appendix A

Enlarged Figures

In order to aid viewing in print, or any other scenario where zoom may not be available, this appendix contains select figures from the main body of this thesis that have been reproduced at a larger scale. Within this thesis, the captions of these figures had been marked with a magnifying glass symbol, 🔍.



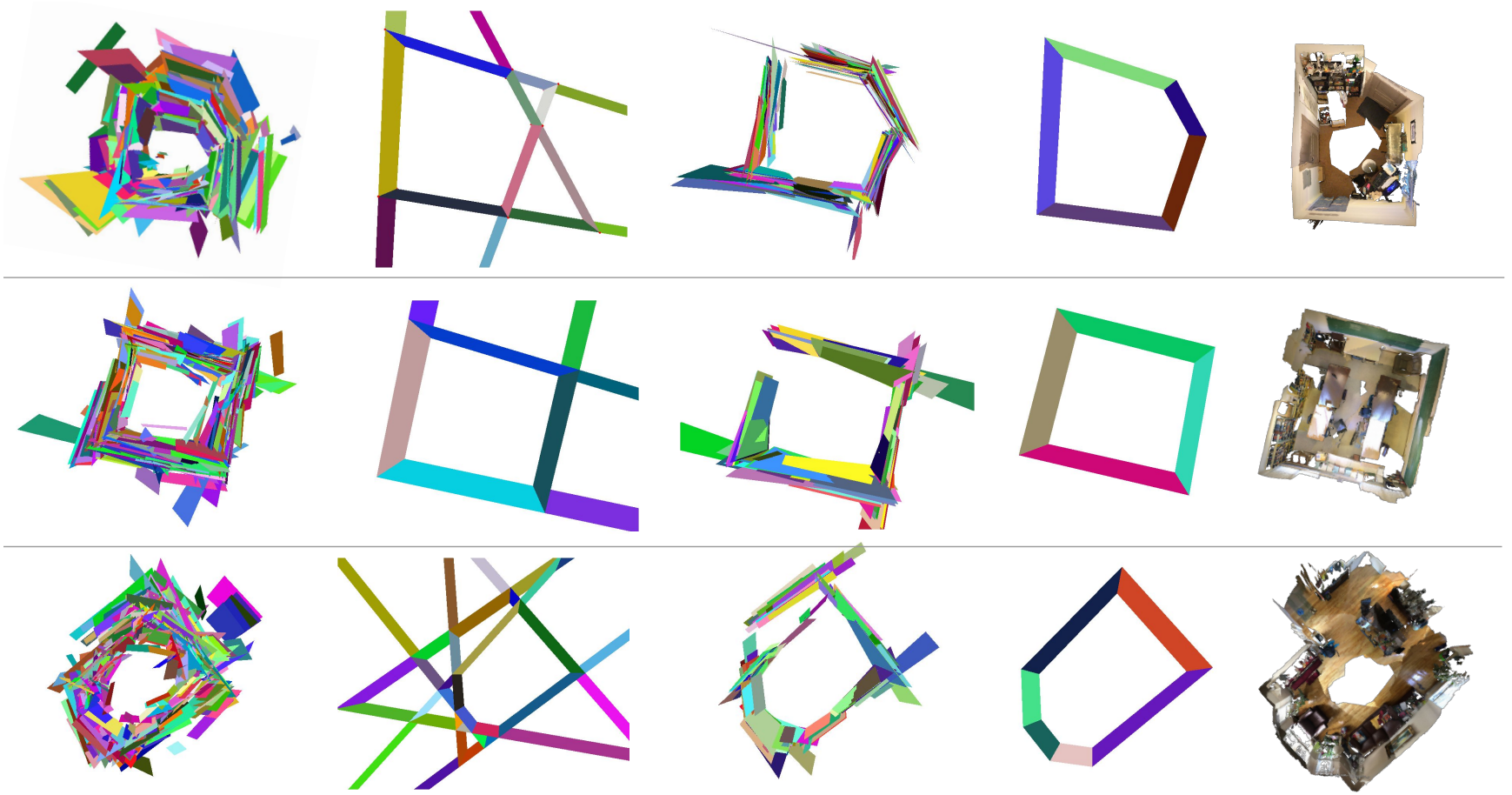
Chapter 3 Fig. 3: The first and second rows show the image overlaid with the ground truth and network detections, respectively. The last row shows the predicted bounding-boxes projected onto their matching regressed planes to produce the informative plane segments. For additional 3D context, a point cloud from a depth image matching the input RGB image is displayed with the plane segments



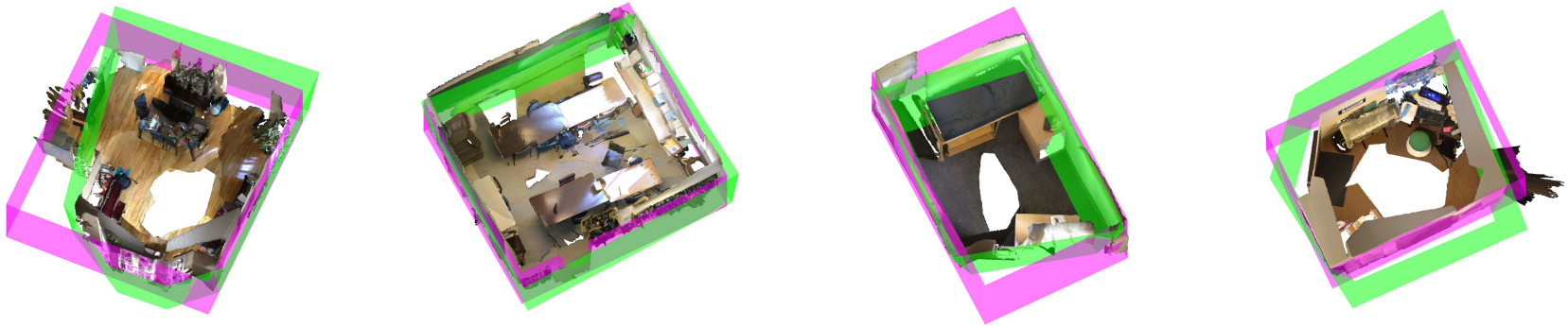
Chapter 3 Fig. 4: Examples demonstrating the discussed advantages: The first column shows a plane being accurately regressed in a cluttered environment; The second and third columns show examples walls are regressed behind shelves and cabinets; In the forth column a wall is correctly estimated behind a chair and a non-planar curtain.



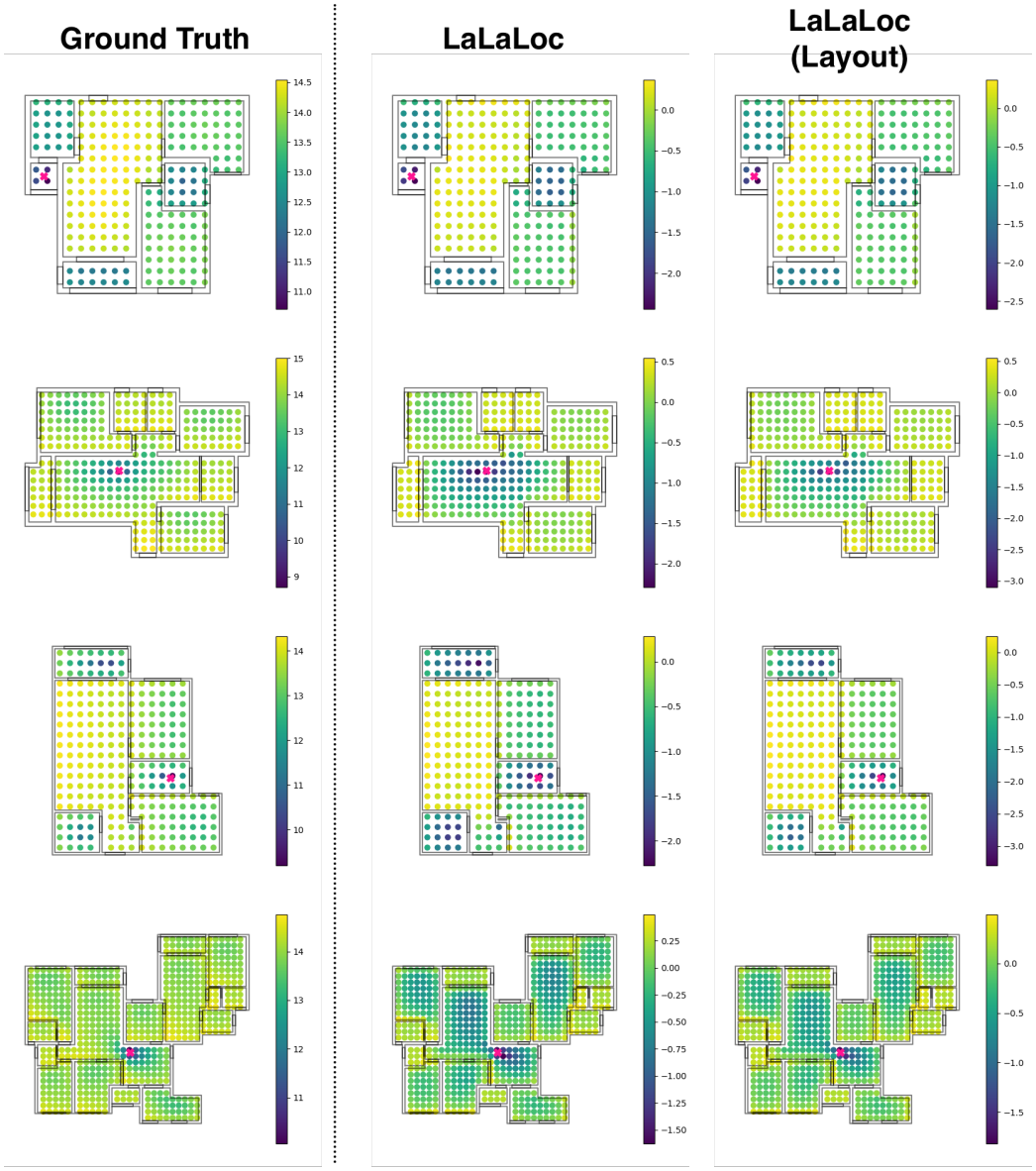
Chapter 3 Fig. 5: Example 2D room layouts produced by our method on the NYUDv2 303 dataset. The first row shows the input image. The second and third rows show the ground truth and predicted room layout segmentations, respectively. The fourth row shows the prediction overlaid onto the input image. It is worth noting that in the last two columns the rooms in the input are not box-shaped. We believe that our prediction better reflects the room shape in these cases than the ground truth.



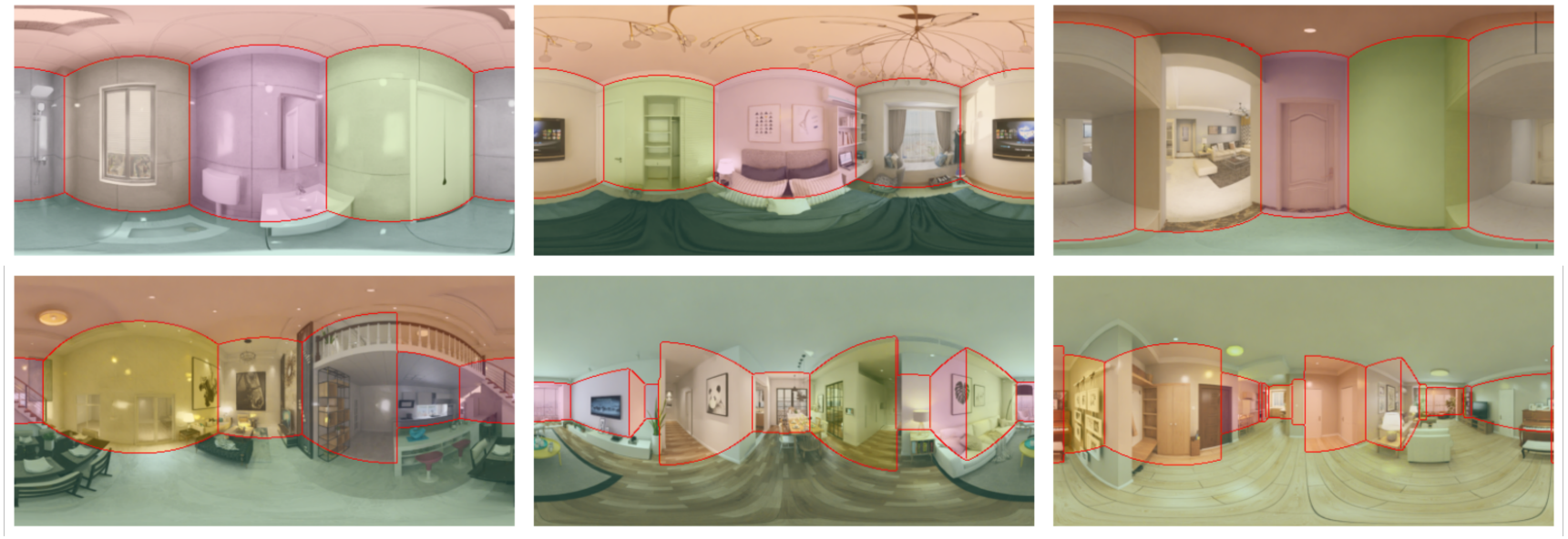
Chapter 3 Fig. 6: Each row visualises the steps of our spatial voting method for determining the room layout from Planar R-CNN measurements on a ScanNet sequence. The first column shows a subset of the raw measurements. After inferring the underlying room planes, we intersect to find the set of wall segment candidates, shown in the second column, as well as the top n measurements from each cluster with the highest likelihood to form the group of voter patches, shown in the third column. The fourth and fifth column show the final layout estimation and the ground truth mesh, respectively.



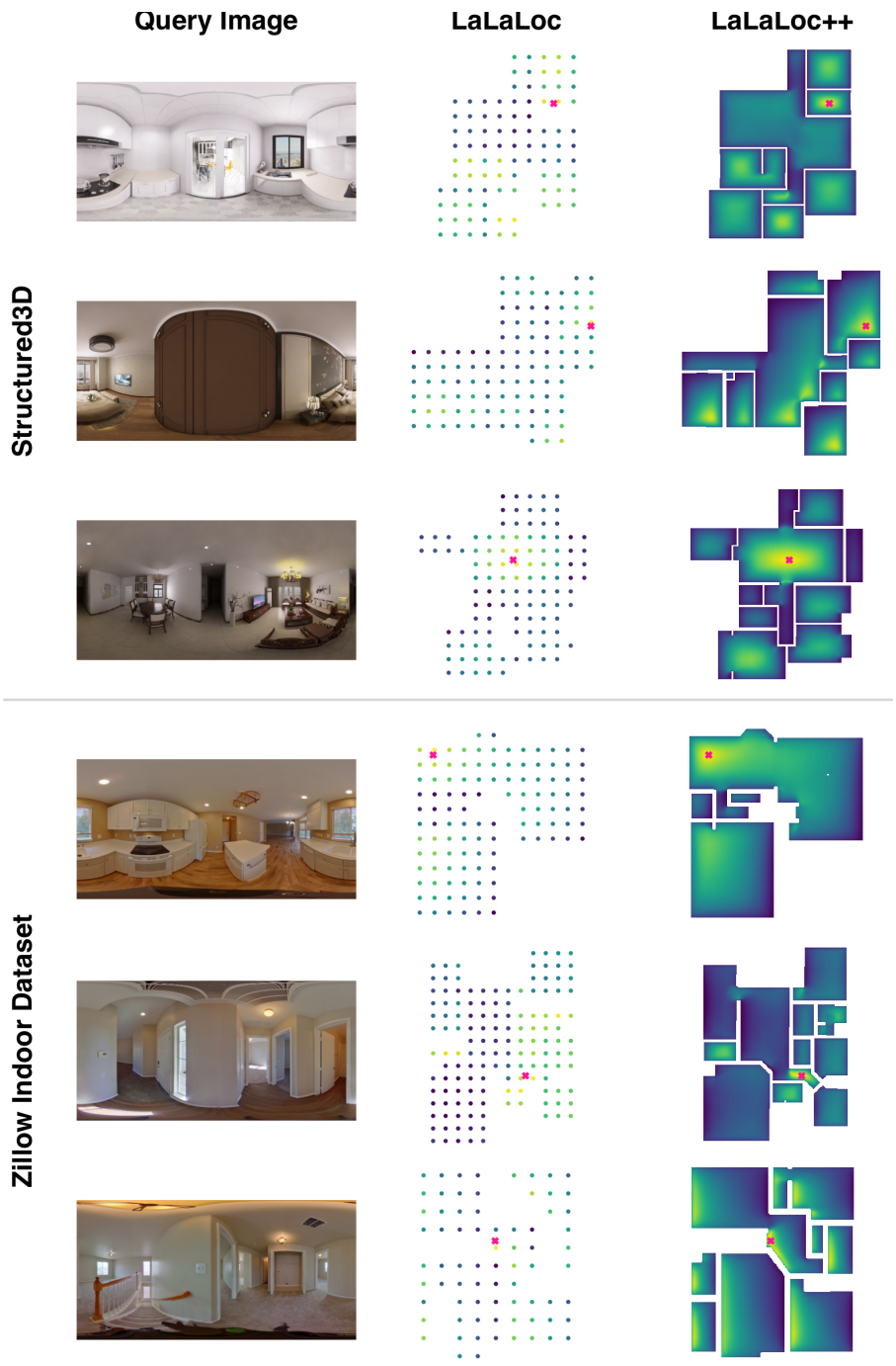
Chapter 3 Fig. 7: Manhattan boxes (magenta) fitted to the ground truth point clouds using the method provided in SUN RGB-D shown with our own unconstrained layouts (green).



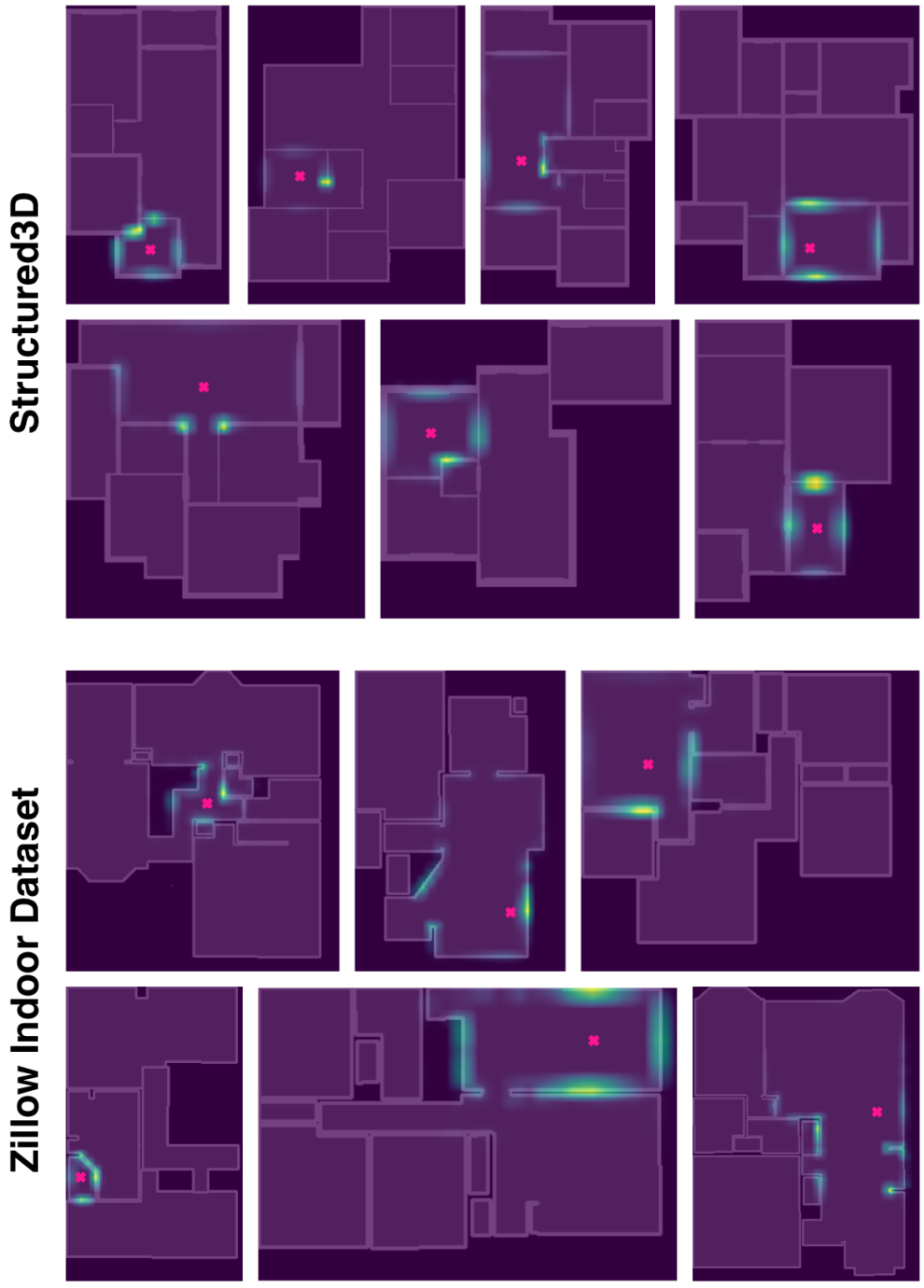
Chapter 4 Fig. 3: Qualitative depiction of the layout distances between a query pose, marked with a pink "X", and the grid of sampled poses across the scene. For each sampled pose, we colour it by the log of its respective distance from the query. *Left:* we plot the ground-truth layout distances. *Middle:* the layout distance is that predicted by LaLaLoc. *Right:* predicted by LaLaLoc in its layout-to-layout configuration.



Chapter 4 Fig. 4: Example failures. *Top:* Cases where the wrong room has been retrieved due to ambiguity of layout, but the alignment is generally coherent with the layout seen in the query image. *Bottom:* The correct room is retrieved, but the alignment is incorrect. In these cases, it is often seen that room corners are incorrectly aligned to the edges of objects such as tables and counter tops.



Chapter 5 Fig. 4: Qualitative comparison of the floor plan comprehension differences between LaLaLoc++ and LaLaLoc. Visualised as a plot of the latent distance between the query image and embeddings across the floor plan in each of the two methods. While LaLaLoc++'s Φ_{plan} is able to infer layout densely within the scene, LaLaLoc instead just has a sparse sampling (in this case every 0.5m in each direction). The difference in the resulting understanding of the environment is stark.



Chapter 5 Fig. 5: Investigation into floor plan saliency for inferring layout at a given location from a global floor plan. The query location is marked with a pink cross, and computed saliency is overlaid onto the floor plan. The floor plan comprehension module demonstrably learns to differentiate between visible and invisible structure for the query locations when computing their descriptor.