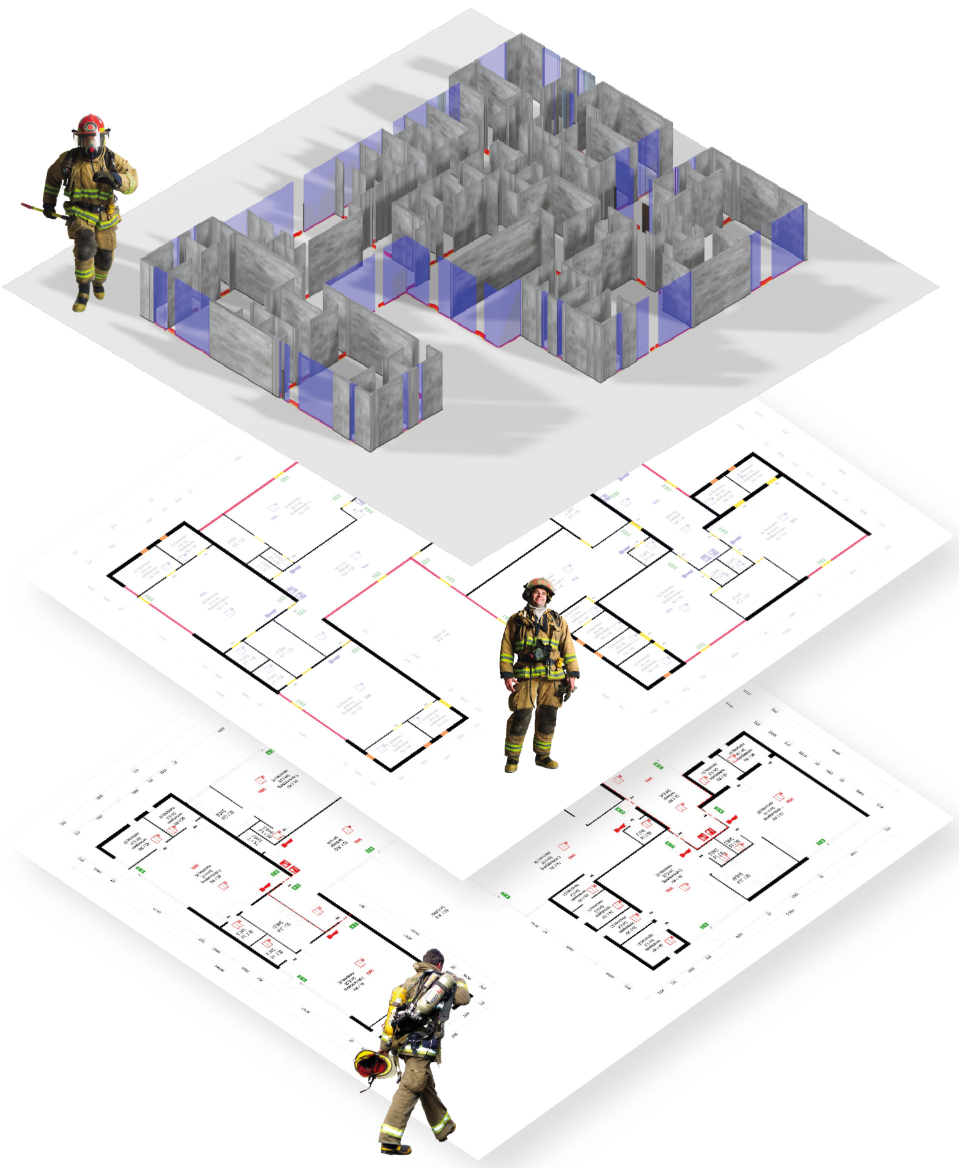
Multi-Unit Floor Plan Recognition and Reconstruction

Master Thesis

Gijs de Jong



Multi-Unit Floor Plan Recognition and Reconstruction

by

Gijs de Jong

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 25, 2022 at 10:00 AM.

Student number:	4723643	
Project duration:	November, 2021 – August, 2022	
Thesis committee:	Prof. Dr. Johan Pouwelse,	TU Delft, supervisor
	Prof. Dr. Jan Rellermeyer,	TU Delft & Leibniz University Hannover, daily supervisor
	Dr. Monique Arkesteijn,	TU Delft

Preface

I want to thank my daily supervisor Dr. Jan Rellermeyer for his trust, guidance, and especially the patience and freedom to pursue my research goals. I am grateful for the people at DE-RISC for giving me the opportunity to research both my thesis and work together with a fun group of people. I want to thank Monique Arkesteijn and Mathilda Du Preez for always lightening up my mood to keep me motivated. I thank Jos'e and Meric for enduring the annotating struggle together with me. I also thank Dr. Johan Pouwelse, for taking the time to review my work.

I would like to thank the all of the people involved in the DE-RISC project for their valuable comments and insights, especially the safety region of Rotterdam who provided the multi-unit floor plans.

Finally, I want to express my gratitude for my girlfriend, friends and family for supporting me, keeping me motivated, and offering help when needed during these last months.

Gijs de Jong
Delft, August 2022

Summary

Automatically deriving 3D representations of buildings is a challenging problem which is at the base of a wide range of applications. The DE-RISC project aims to generate a 3D model of the entire city of Rotterdam in The Netherlands, enabling many of these applications. Generating a 3D model of a building can be done in a variety of ways, of which only few are robust, scalable and generalizable. Recognition and reconstruction of architectural floor plans is such a scalable method long researched in literature. Although these methods generalized poorly initially, recent breakthroughs in computer vision have allowed for the application of deep learning based approaches. Recent floor plan processing methods have shown promising results on single-unit floor plans. Single-unit floor plans are floor plans of single apartments of relatively low complexity. In contrast, multi-unit floor plans describe entire buildings, and are thus significantly larger and of higher complexity. Applying single-unit floor plan processing methods to multi-unit floor plans is not trivial, and results in insufficient accuracy. These methods can therefore not be applied to an entire city, limiting the scalability and generalizability.

This thesis proposes a novel multi-scale floor plan recognition and reconstruction method designed to transform floor plans of arbitrary size into their 3D representations. As no multi-unit floor plan datasets exists, a novel floor plan dataset MURF is presented based on multi-unit floor plans from buildings in Rotterdam. MURF considers seven boundary and opening semantic classes, each with distinct physical properties. The recognition part of the method relies on an FCN employing multi-scale skip-connections, an attention mechanism, and a multi-task training objective to reinforce the learning of multi-scale features. The reconstruction part refines predictions from the recognition step by applying post-processing, vectorization, and visualization in Blender.

The proposed method is compared to floor plan processing models from literature and general state-of-the-art segmentation models by a quantitative and qualitative evaluation. Experimental results show that the proposed method is significantly outperforms existing floor plan processing methods, and performs best out of general segmentation models. A case study on the EMC in Rotterdam demonstrates the generalizability of the proposed method.

Our code is available at: `https://github.com/TheOnlyError/2d3d`.

Contents

1 Introduction

Over the recent years, 3D representations of buildings have become increasingly useful across several domains such as indoor remodelling [1], construction [2], facilities management [3], and emergency models [4]. The DE-RISC[1] project is an example of an initiative heavily relying on the information contained in 3D models. The DE-RISC project aims to develop a fully integrated 3D model of Rotterdam called the *Digital Twin*, combining static building information with real-time data. Examples of static and real-time data include 3D representations of all buildings, infrastructure, traffic data and population crowd density. With support of the Municipality of Rotterdam and the Rotterdam-Rijnmond Security Region (VRR), the digital twin of Rotterdam represents a future-proof solution for the manageability of the continuously changing environment. Such a digital twin can be used in various domains, for example security, planning, traffic control and construction. A Proof of Concept using 3D point clouds of the digital twin has been developed for a single building, which showed that scaling this approach to the entirety of Rotterdam is difficult due to, among others, model complexity and limited data availability. Creating a 3D representation of the entirety of Rotterdam is the first step in creating a digital twin, and what this thesis will focus on.

1.1. Problem Statement

Various approaches for generating 3D models of buildings of various levels of detail have been proposed, making use of different data sources to do so. Among others, these include 3D point clouds [5, 6, 7], 3D annotations obtained through Virtual Reality (VR) [8], Computer Aided Design (CAD) drawings [9, 10], and architectural floor plans [1, 11, 12]. Although effective for single buildings, the majority of these approaches are not scalable to an entire city due to data availability, required human intervention, privacy sensitivity, cost or complexity. Data availability is a crucial part of scalability, as many methods require extensive data of a building in order to automatically generate its 3D model [5, 6, 7, 8, 10]. However, getting sufficient data is costly and sometimes impossible [13, 14]. This thesis opts to use architectural floor plans as its sole source to generate 3D models, as these are widely available and require no further human intervention [15].

Architectural floor plan are dense 2D representations of the geometric and semantic information of floors of buildings. The level of complexity in floor plans varies significantly, and can range from describing the material and structure of each wall, to considering all wall properties and structure equal. The conversion of 2D floor plans into 3D models has long been a topic of active research [1, 8, 16, 17] and been shown to be a challenging problem [18]. The main difficulty of generating a 3D representation lies in generalizing across varying drawing notations and increasing levels of unrelated noise in drawings [1]. It is also important to note that generating an actual 3D representation from a single floor plan is impossible, as height information regarding walls is almost always omitted. The height of walls and other elements in generated 3D models is often assumed, and thus a 2.5D representation would be a more accurate description. However, to prevent confusion this thesis will refer to 2.5D representations as 3D for the remainder of the content.

[1] https://convergence.nl/nl/how-do-we-protect-a-large-city-with-many-high-rise-buildings-against-emergencies/

Initial floor plan analysis relied on a predefined set of heuristics used to determine the semantic elements in floor plans [19, 20, 21, 22]. Since the rules were derived from the input data, these solutions generalized poorly and required significant human intervention to process a different floor plan dataset. To automatically derive an effective set of heuristics from a dataset, and thus eliminating the need for human intervention, machine learning based methods were later applied [23, 24]. Although less human interaction was required, these solutions still required retraining before processing a new floor plan dataset.

With recent advances in computer vision, deep learning based approaches have been proposed for floor plan analysis [16, 17, 25, 26, 27]. *Convolutional Neural Networks* (CNNs) in particular have been shown to be very effective at identifying the semantic elements in floor plans required for 3D visualization. The main advantage of using a neural network to process floor plans is that the inference on such a network can be scaled significantly better compared to traditional processing techniques. This thesis will thus also focus on developing a CNN to process floor plans.

The main limitations of existing approaches lie in their constraints on input size and considered semantic elements in floor plans. The current state-of-the-art solutions are trained on datasets where the size and complexity of floor plans is limited. The floor plans in these datasets are often sourced from single-unit apartments in residential areas [1, 16, 17, 26, 28]. The complexity of single-unit residential apartments is significantly lower compared to other types of floor plans, for instance rural floor plans [27]. Although more complex, rural floor plans are still limited regarding their size. *Multi-unit floor plans* are distinct from single-unit floor plans, as they represent entire building floors instead of single apartments. Single-unit floor plans are often more aligned with the axes of the plan, and thus provide a more regular set of room shapes and boundary elements. In contrast, multi-unit floor plans almost always contain elements not aligned with the axes or contain non-rectangular shapes. As a result, multi-unit floor plans are significantly larger and more complex than single-unit floor plans, presenting a more difficult image segmentation task. Although geometric approaches have been proposed, multi-unit floor plans have not been evaluated in a deep learning setting in literature [14, 29].

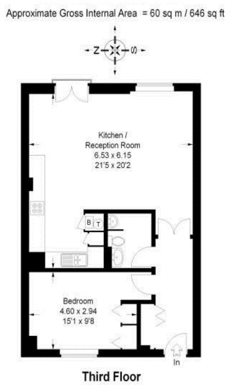
Effectively processing floor plans of arbitrary size and complexity is especially important to convert a city of different buildings into their respective 3D representations. A naive solution would be to apply an ensemble of segmentation models that operate on different floor plan scales. This approach would, however, not scale well with increasing floor plan size. Figure 1.1 shows the differences between two examples of a single-unit and multi-unit floor plan. In addition to the significantly larger size, the multi-unit floor plan contains more annotations unrelated to the structural elements that can be identified. Compared to the single-unit floor plan, the size of structural elements, such as windows, varies more in the multi-unit floor plan. The size, level of noise and feature complexity all contribute to the difficult segmentation case multi-unit floor plans present.

The semantic classes that state-of-the-art methods consider also limit the generalizability across other types of floor plans. Although recent works consider sufficient objects in floor plans, the structural elements important for a representative 3D model are lacking [17, 27]. For emergency related applications, different types of walls (e.g. glass walls) and openings (e.g. sliding doors) may be crucial for a realistic generated 3D model. No model in recent literature has attempted to detect these structural elements in single-unit or multi-unit floor plans.

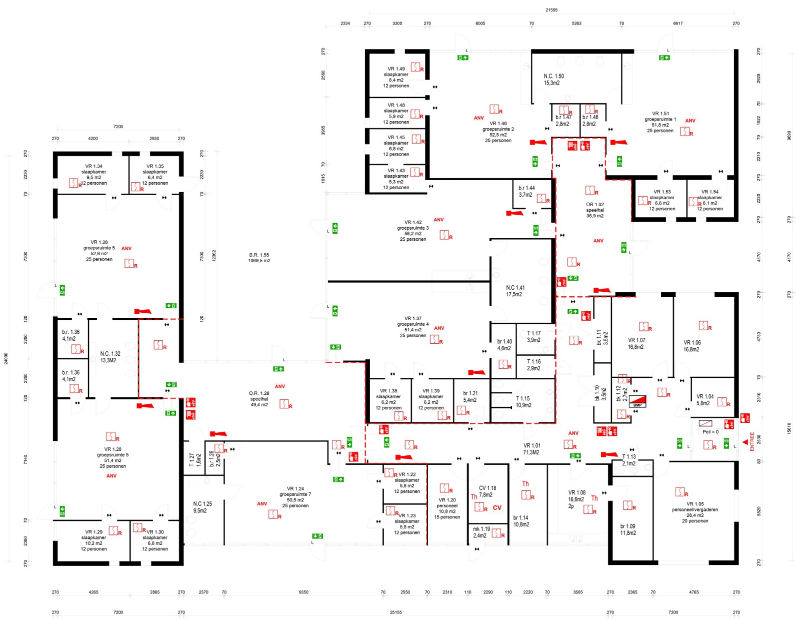
Another limitation of current state-of-the-art floor plan processing models is that they have been trained on a fixed input resolution. A resolution of 512×512 is often used for the proposed networks [16, 17, 27]. Fixing the floor plan size for a model can have three disadvantages. In the case that a dataset is not sampled (e.g. larger floor plans divided into smaller ones), all floor plans in a dataset have to be resized for training. Resizing the floor plans reduces the level of detail significantly, which the model can now no longer capture. The second disadvantage is that fixing the input size drastically limits learning multi-scale features. For instance, features of walls that relate across different samples can no longer be learned. The third disadvantage is related to the architecture and hyperparameters of a model with a fixed input size. State-of-the-art floor plan processing methods often use a Fully Convolutional Network (FCN), because of their ability to process inputs of arbitrary size. However, if some component of an FCN requires a predefined shape, the model is now restricted to one size. This limits the scalability of the model significantly. In addition to this, values for hyperparameters of a model are often found empirically, but could harm performance if the model is tasked to detect larger features than originally trained for. A model that can learn from floor plans of various resolutions can thus help to learn multi-scale features essential for effectively processing multi-unit floor plans.

This thesis aims to fill this gap by proposing: (1) a novel multi-unit floor plan dataset, and (2) a multi-scale CNN architecture that can detect the necessary structural elements in both single-unit and multi-unit floor plans to generate accurate 3D representations. Concretely, the objective of this thesis is to answer the following three research questions:

1. How can a multi-unit floor plan dataset best be defined for effective learning and visualization?
2. What CNN architecture can effectively detect semantic information in architectural floor plans of arbitrary size in a scalable manner?
3. How can produced segmentation masks best be transformed into a 3D representation?



(a) Single-unit floor plan of 349 × 600 pixels.



(b) Multi-unit floor plan of 9606 × 7573 pixels

Figure 1.1: Examples of single-unit and multi-unit floor plan.

1.2. Contribution

The answers to the proposed research questions are formalized by the following three contributions:

1. Introduce *MURF*, a novel multi-unit floor plan dataset consisting of high-quality architectural floor plans with varying degrees of complexity to train and evaluate the proposed model on. Refine existing public datasets in order to create a combined dataset and evaluate generalization across different types of floor plans.
2. Propose a novel multi-task FCN able to effectively process architectural floor plans of arbitrary type, size and complexity.
3. Develop an efficient reconstruction pipeline to transform produced segmentation masks into 3D models.

This thesis will evaluate the proposed model against the state of the art, also considering newer models that have not been evaluated in the floor plan processing domain yet.

2
Related Work

The conversion of 2D floor plans into 3D models has long been an active research topic in the domain of computer vision. Solving this problem requires solving a number of sub-problems, which include noise removal, symbol recognition, vectorization and 3D transformation [17]. In order to derive a 3D representation of a floor plan, the appropriate elements such as walls and openings must be identified. Initially, manual techniques were employed to do so [19, 20, 21, 22]. Over time, the improved access to computational resources allowed for more complex automatable approaches. The research focus of floor plan analysis shifted from feature engineering derived from low-level image processing to deep learning methods that have been shown to generalize better [25, 26].

In addition to detecting the necessary elements to construct a 3D model of a floor plan, other elements in a floor plan can be identified. Existing models described in literature have been trained to detect the room types or fixed furniture in a floor plan.

While a plethora of methods exist for single-unit floor plans, i.e. floor plans of single apartments, there is a distinct lack of scalable approaches for multi-unit floor plans, i.e. floor plans of entire floors of buildings. The majority of the existing methods in literature were designed for and applied to single-unit floor plans. Although multi-unit plans are fundamentally different from single-unit plans, there are semantic similarities between them. Such similarities include the overlap of semantic classes in both types of floor plans. The complexity of multi-unit floor plans in particular is higher, conveying more information than single-unit floor plans. Reviewing the single-unit floor plan processing methods can thus still provide valuable insights, which this thesis will consider.

This chapter first presents an overview of relevant single-unit and multi-unit floor plan processing methods and related datasets. As the focus of this thesis is to develop a scalable solution, only semi-automatic methods are considered. More recent developments in semantic segmentation have not yet been evaluated in the floor plan processing domain. These state-of-the-art segmentation models will be briefly discussed in the end.

2.1. Early Methods

Initial methods of floor plan analysis mainly revolve around extracting the walls and separating text from graphics in single-unit floor plans. The extracted walls can then be combined in a post-processing step to derive the rooms in a floor plan. Under the assumption that walls are of similar notation, the walls can be extracted by analysing low-level image features.

Dosch et al. combined a classical Hough transform with image vectorization to generate a 3D representation of a single-unit floor plan [19]. In order to overcome memory limitations, a tiling method was proposed that divides the images into partially overlapping segments. The visualization method was later improved by Or et al. by generating polygon meshes [20]. This method was applied to multi-unit floor plans. Both solutions were hand tailored to the used floor plans, and thus generalized poorly.

Macé et al. designed a system to detect rooms in single-unit floor plans [21]. This system first extracted walls from the floor plan by similar coupling of Hough transform and vectorization as in [19]. The walls and openings are then used in a room segmentation strategy, which refines room polygons by measuring concavity. The limitation of this approach is that the room detection focused on rectangular rooms, and thus performed degraded on more complex floor plans.

Ahmed et al. proposed a sequence of morphological operations to group walls in a single-unit floor plan into thin, medium and thick lines [22]. The rooms could be identified by combining the extracted walls with a text/graphics segmentation method based on connected components [30]. Ahmed et al. later proposed an extension that also detects the room labels using Optical Character Recognition (OCR) [31]. An obvious limitation of this approach is that it can only be applied to floor plans of similar style in which the walls are represented by thick lines, i.e. bold and of solid filled color.

2.2. Machine Learning Methods

Conventional methods of floor plan analysis rely on heuristics derived from floor plans to recognize elements in it. Since these solutions are tailored to specific datasets, they generalize poorly to a larger range of drawing notations. Walls alone can be represented in different ways, for instance by varying thickness, texture or number of parallel lines. Examples of different wall drawing notations are depicted in Figure 2.1. This means that low-level heuristics have to be redefined for different datasets, which is expensive and limits the scalability. This motivated the development of machine learning methods that could overcome these difficulties.

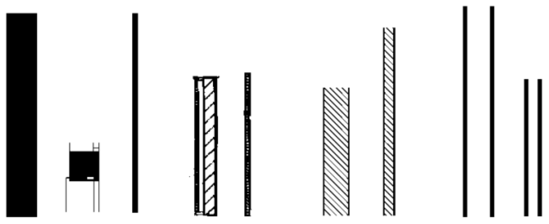


Figure 2.1: Examples of various drawing notations for walls [32].

De Las Heras et al. introduced a patch-based segmentation method to overcome the variability of notations in single-unit floor plans [23]. This method produced a pixel-wise prediction of walls in single-unit floor plans. This prediction is based on the K-Means clustering of patch features that are extracted using Principal Component Analysis [33]. De Las Heras et al. later provided an extension to additionally segment doors and windows [24]. It is important to note that this method still required retraining the parameters for a new dataset, but meant that the method itself would no longer have to be modified as well. The requirement of retraining is the biggest limitation of this approach, as improved adaptability of the solution is not sufficient to generalize well to different drawing notations. Another notable limitation is that walls are assumed to consist of straight lines, meaning curved walls can not be detected.

2.3. Deep Learning Methods

While machine learning methods allowed for more adaptable solutions, generalizability on different drawing notations still proved to be challenging. With the development of deep learning technology, related methods have proved to be more effective at generalization on floor plans. CNNs have been shown to perform well in floor plan analysis in particular. This section will discuss its uses and results.

Dodge et al. were one of the first to use an FCN for wall segmentation in single-unit floor plans [25]. A separate object detection method based on faster R-CNN [34] was employed to determine room sizes. Performance was significantly better compared to earlier methods relying on shallow learners. The retraining for machine learning methods was also no longer required.

Liu et al. trained a neural network to detect junction points in single-unit floor plans [1]. These were combined by integer programming to derive the walls and room types in floor plans. A novel dataset was proposed for evaluation, in which shows that the model outperforms the previous state-of-the-art proposed by Ahmed et al. [31]. The main limitation of this work is the Manhattan assumption, which assumes all lines are either horizontal or vertical. This means recognizing complex elements such as curved walls is impossible.

Zeng et al. proposed a multi-task CNN to detect the room boundaries and room types in single-unit floor plans [16]. Walls and openings (e.g. windows and doors) were considered as room boundaries. The main contribution was to use an improved loss function and novel room-boundary attention mechanism. This attention mechanism reused the room boundary features when determining room types. A limitation of this network is that it is not fully convolutional, and might therefore have problems processing larger mult-unit floor plans.

In recent work, Lv et al. proposed a segmentation architecture that utilizes a combination of deep segmentation and detection networks [17]. An FCN variant of DeepLabV3+ [35] was used to detect walls. A heatmap regression task was proposed to improve the detection of openings. YOLOv4 [36] was used to detect objects and annotations in a floor plan. A novel scale calculation was also proposed, which relies on annotations in a floor plan to determine the size of the 3D representation. Compared to [16], this method increased accuracy for wall and room type segmentation. The architecture was also evaluated on a new single-unit floor plan dataset. Although the results are promising on single-unit plans, the model was not evaluated on multi-unit floor plans.

2.4. Datasets

Over the recent years, several floor plan datasets have been proposed, each with distinct properties. This thesis aims to propose a scalable and generalizable multi-unit floor plan processing approach. The generalizability refers not just to the variability of drawing notation in floor plans, but also regarding their size and complexity. Combining several floor plan datasets for learning and evaluation can help achieve this goal. This section will thus discuss several floor plan datasets.

Recent literature has published the floor plan datasets shown in Table 2.1 [14]. The datasets have been selected based on their published date, annotation, complexity, resolution and number of floor plans. Datasets R3D [26], R2V [1] and CubiCasa5K [28] have been evaluated extensively in related work [17, 27, 37]. Zeng et al. proposed expanded variants of R3D and R2V with pixel-wise labels instead of vectorized ones. Eighteen circularly shaped floor plans were also added to R3D. Although the annotations of R2V are publicly accessible, the floor plans are not [14]. Lv et al. recently published a novel dataset RFP, but this has unfortunately not been made public. The same applies to RuralHomeData introduced by Lu et al. [27]. No further other evaluation on RFP and RuralHomeData has been presented yet apart from the original works.

To the best of our knowledge, no multi-unit floor plan dataset exists in literature. Since there are semantic similarities between single-unit and multi-unit floor plans, the datasets are still useful for learning. This thesis opts to use the extended R3D and CubiCasa5K datasets for its learning and evaluation, because these are publicly available and of distinct size and complexity.

The original R3D dataset contains 214 floor plans crawled from a London rental site [26]. The source of the 18 round-shaped floor plans added by Zeng et al. is not specified, although the annotations are in English [16]. Compared to R2V, the walls in R3D are more complex as they are of nonuniform thickness. This nonuniform representation is essential for a model to learn multi-scale features, crucial for prediction generalizability. In addition to the round-shaped floor plans, this is why R3D has a higher complexity than R2V. Zeng et al. introduced the annotations for R3D that are used in this thesis. The annotations are pixel-wise masks for each structural class and room type.

The CubiCasa5K dataset is based on a large sample of 5000 floor plans sourced from real estate material from Finland [28]. The majority of source floor plans are raster scans of original floor plan drawings. The annotations of floor plans are SVG files, containing a large number of structural elements, room types and objects. It is important to note that while the annotations in CubiCasa5K contain many classes, only a small subset of classes was evaluated by merging them [28]. Another important note is that labels in CubiCasa5K sometimes deviate from the source floor plans, in the case where the real-world state of a respective apartment is different [28]. The level of noise introduced by raster scans, large number of classes, and label deviations are the main reasons why the complexity of CubiCasa5K is considered high.

2.5. State-of-the-Art Segmentation Models

The field of image segmentation is developing quicker than the floor plan processing domain can match with. Several CNNs have been proposed that showed promising results in other domains such as medical image segmentation.

The majority of the state-of-the-art CNNs are improved versions of U-Net [38]. U-Net is one of the

Dataset (year)	Annotation	Complexity	Resolution min-max	Number of plans	Public access
R3D (2015) [26]	Walls, openings, and 6 room types	Low	175-1104	214	✓
R2V (2017) [1]	Walls, doors, windows, stairs, other object types, and 12 room types	Low	96-1920	815	✗
R3D* (2019) [16]	Same as R3D but includes circular shaped plans	Medium	175-1104	232	✓
CubiCasa5K (2019) [28]	Walls, doors, windows, and 80 other object categories	High	50-8000	5000	✓
RFP (2021) [17]	Walls, doors, windows, and 7 room types	Low	180-3615	7000	✗
RuralHomeData (2021) [27]	Walls, doors, windows, stairs, slopes, text, and 21 room types	Medium	1600-2560	800	✗

Table 2.1: Overview of datasets available in recent literature. *R3D has been extended with 18 images by Zeng et al. [16]. The complexity of a dataset is based on the type, annotation and resolution of corresponding floor plans. R2V is not considered publicly accessible, because only the annotations without respective floor plans are available [14].

first FCNs based on an encoder-decoder architecture that utilized skip connections to fuse high-level semantic feature maps with low-level detailed feature maps. The main limitation of the skip connections in U-Net is the large semantic gap between the fused feature maps [39].

U-Net++ [39] was one of the first segmentation models to improve the original design of U-Net. The main improvement of U-Net++ was to replace the plain skip connections of U-Net with nested and dense connections, reducing the semantic gap between the encoder and decoder [39].

U-Net3+ [40] was proposed as an improvement over U-Net++ and introduced full-scale skip connections. These were designed to make full use of multi-scale features by directly fusing the feature maps from different scales. In addition to improved accuracy compared to U-Net++, U-Net3+ requires less network parameters.

The disadvantage of directly aggregating multi-scale feature maps is that this assumes their weights are equal. MACU-Net [41] introduces channel attention blocks in order to combine multi-scale features effectively. MDA-UNet [42] utilized similar channel attention blocks, designed to significantly reduce the network parameters required.

One downside of removing the intermediate neurons of U-Net++ is that the performance of small targets with a small number of samples is limited. U-Net# is designed to combine U-Net++ and U-Net3+, utilizing all three types of skip connections: nested, dense and full-scale connections.

Although MACU-Net, MDA-UNet and U-Net# have been shown to outperform U-Net3+ individually, an analysis evaluating the models against each other has not been performed yet. Thus no clear best architecture is known, and should be decided based on context.

3 MURF

This chapter contains an extensive description and evaluation of the novel *Multi-unit Rotterdam Floor plan* (MURF) dataset proposed in this thesis. The design of MURF is related to RQ1: determining how a multi-unit floor plan dataset can best be defined for effective learning and visualization. The last section will reflect on MURF and the first research question. Note that 'MUFP' in the figures for the remainder of this chapter refers to the MURF dataset.

3.1. Data Source

The MURF dataset is a novel multi-unit floor plan dataset consisting of large, high-quality floor plans of varying complexity. Multi-unit floor plans are distinct from single-unit floor plans as they contain the architectural information of the entire floor of a building, compared to a single apartment. As a result, multi-unit floor plans are larger and more complex than single-unit floor plans. MURF consists of 8 floor plans provided by two industrial partners: the Municipality of Rotterdam and the Rotterdam-Rijnmond Security Region (VRR). The floor plans are of various buildings which include hospitals, schools and large luxury apartment buildings.

MURF consists of rasterized versions of the source floor plans created using Adobe Photoshop. Since the source floor plans were always provided in a PDF file format, the rasterized images could be simply be generated by converting the PDFs into a PNG image. The resolution of a rasterized PNG floor plan was set large enough so that all details of the PDF were captured with sufficient quality. Figure 3.1 contains 3 examples of multi-unit floor plans in MURF.

3.2. Annotation

Annotation of the multi-unit floor plans was manually done using Labelbox[1] by a small group of three Computer Science students at TU Delft (2 BSc and 1 MSc). In an iterative process, the set of semantic classes, annotation guidelines for each class, and resulting annotations were continuously refined. This process involved consulting industrial partners, such as the Fire Brigade of Rotterdam or floor plan experts from TU Delft with sufficient architectural knowledge. The Fire Brigade and similar parties were important to determine which information of a building is useful in a generated 3D model, and thus which semantic classes need to be identified in the 2D floor plan. Floor plan experts helped determine which regions in floor plans actually correspond to a certain semantic class, proving useful in order to create guidelines in order to identify the semantic elements. Since the floor plans in MURF are large, a single annotation took from 7 to 10 hours depending on the complexity of the floor plan. Each floor plan was labelled by one annotator, and reviewed and corrected separately by the other two persons.

Table 3.1 contains the final list of semantic elements considered in MURF. The corresponding colors will be used for visualization for the remainder of this thesis. The boundary and opening descriptions are related to loss functions that will be discussed in a Section 4.1.4. Glass walls refer to walls that are breakable and see-through, but can not be opened like windows. Railings refer to half height walls that

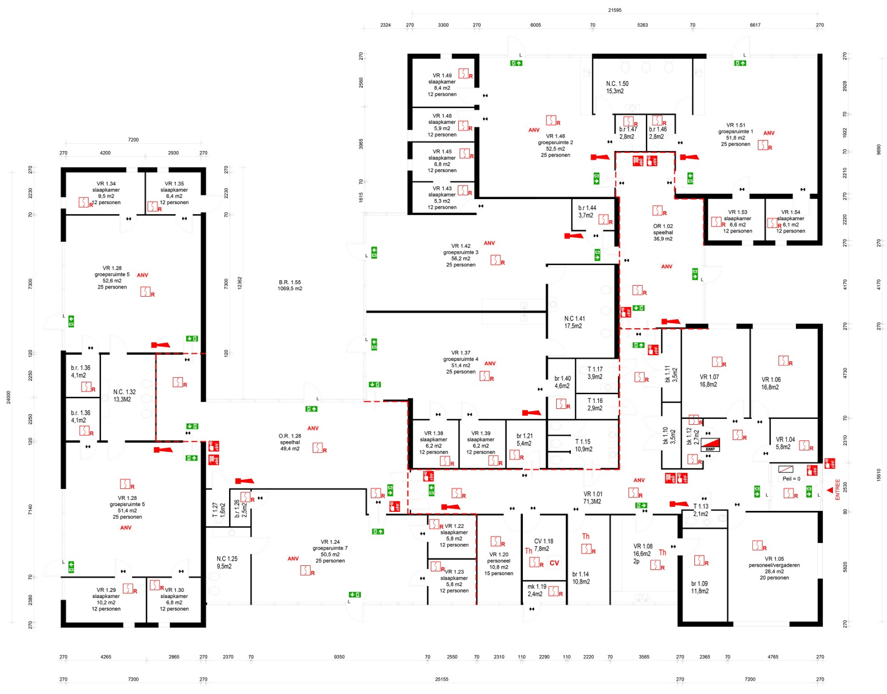
[1] https://labelbox.com/

can be traversed over. Sliding doors refer to doors that open in a distinct manner without a swing path, including both sliding doors that open automatically and doors that do not. The reason for including glass walls and railings in addition to regular walls is related to their semantic difference and distinct use case. The same reason applies to sliding doors, which are added in addition to regular doors. The features of glass walls, railings and sliding doors differ sufficiently for a model to be able to learn to detect them separately. Including glass walls, railings and sliding doors in generated 3D models also improves the overall accuracy, since the 3D representations are now 'closer' to the real-world state of buildings. This in turn improves the applicability of generated 3D models. Railings could for instance help the fire brigade identify the best route for a building by taking the traversable walls into account. Figure 3.2 contains 3 examples of ground-truth multi-unit floor plan labels in MURF.

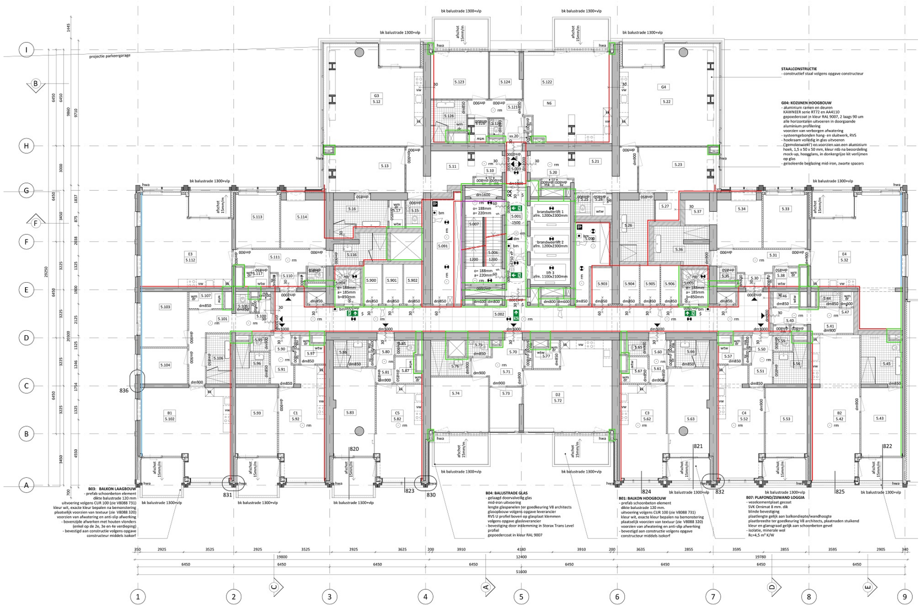
To generate the masks, the Labelbox API was used to export all annotations. The annotations were further processed and combined into a single mask by assigning each pixel in the mask a distinct integer value based on the semantic class. The values per class are shown in Table 3.1. Further processing was only necessary for some of the larger multi-unit floor plans, which could not be fully exported using the API due to unforeseen size restrictions. For these larger plans, the annotation paths from Labelbox had to be manually converted into class polygons used for the generated mask.

Description	Color	Value	Boundary	Opening
Wall	⬛	1	✓	✗
Glass wall	🟥	2	✓	✗
Railing	🟩	3	✓	✗
Door	🟨	4	✗	✓
Sliding door	🟦	5	✗	✓
Window	🟧	6	✗	✓
Stairs	🟦	7	✗	✗

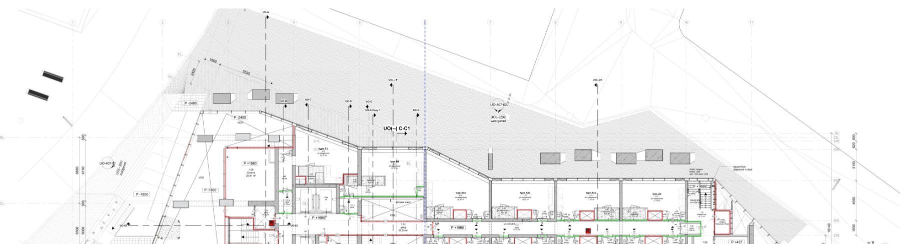
Table 3.1: Structural element classes of MURF with colors, boundary and opening descriptions.



(a)

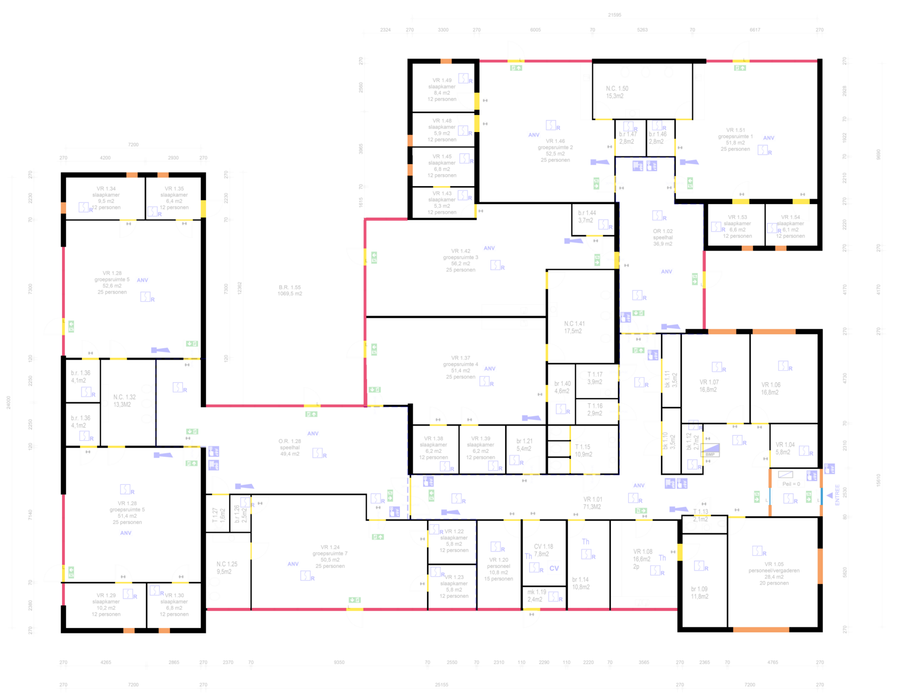


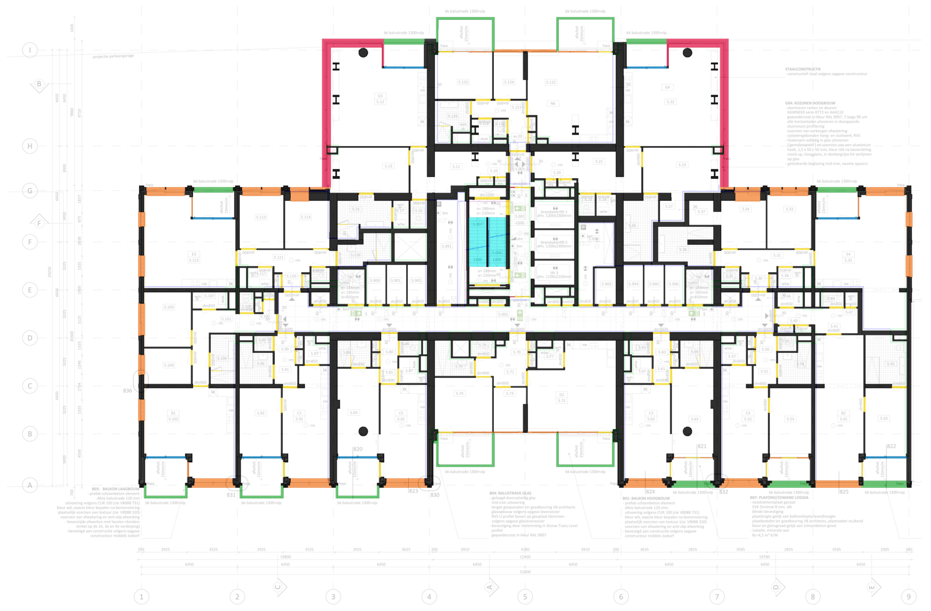
(b)



(c)

Figure 3.1: Three examples of original multi-unit floor plans of varying complexity in the MURF dataset.


(a)


(b)

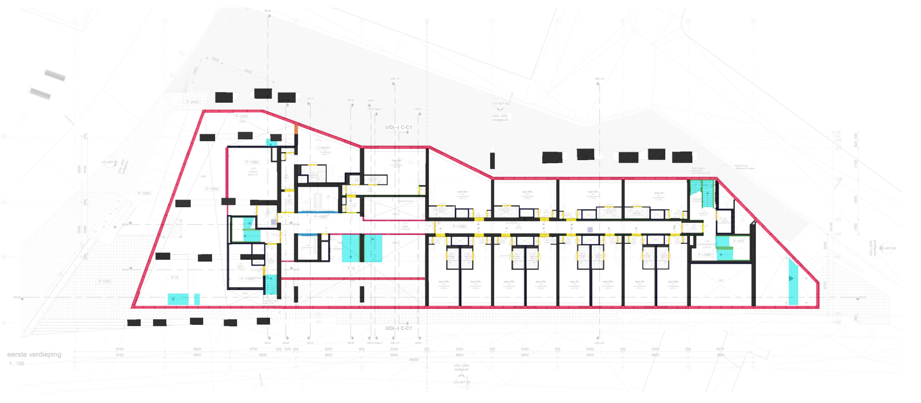

(c)

Figure 3.2: Three examples of ground-truth labels superimposed on multi-unit floor plans in the MURF dataset.

3.3. Dataset Statistics

This section presents statistical information about MURF by discussing three aspects: (1) the distribution of floor plan sizes, (2) the number of class pixels, and (3) the distribution of classes. Since MURF contains a distinct list of semantic classes compared to other floor plan datasets, it will be discussed independent of the others first.

Figure 3.3 shows the different resolutions of the floor plans in MURF. The smallest floor plan consists of 3100 × 9700 pixels, while the largest floor plan has 28160 × 11850 pixels.

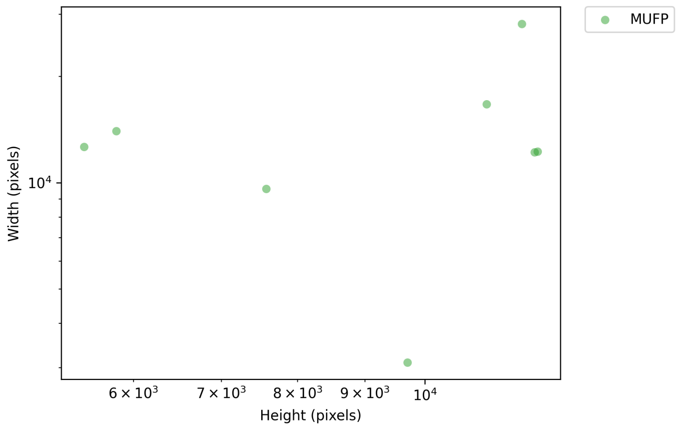


Figure 3.3: Source image resolution of floor plans in MURF.

Figure 3.4 shows the total number of pixels of the semantic classes of MURF. Although there is no distinction between instances of a given class in the annotations, the number of pixels is equal to the area of the combined instances. The area of background pixels make up the large majority of floor plans, causing significant class imbalance. The number of sliding doors pixels in the multi-unit floor plans is especially low, due to both their scarce presence and small dimensions. Stairs occur as often as sliding doors, but make up a larger portion of floor plans in comparison.

Figure 3.5 contains the normalized frequency for each semantic class in MURF. Since there are 8 data points, only a few histogram bins are necessary to represent the results. The class imbalance is also visible here as the background pixels constitute the majority, averaging around 90%. All of the railing and opening classes follow a similar distribution, where the frequency of the higher areas decreases. For the glass walls and stairs the highest frequency is centered around the middle, due to the fact that these class pixels are not present in all the floor plans. For the background and wall pixels the distribution is more uniform-like, since each floor plan has a varying number of them.

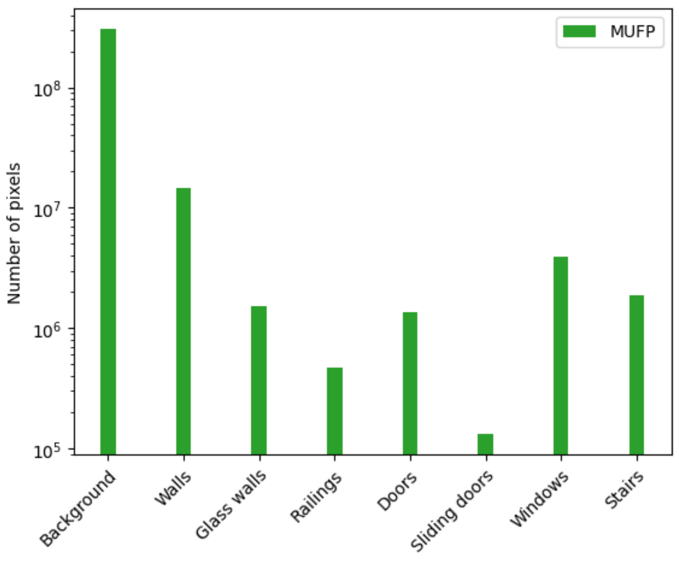
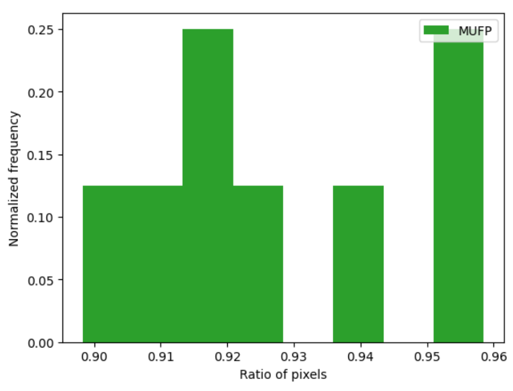
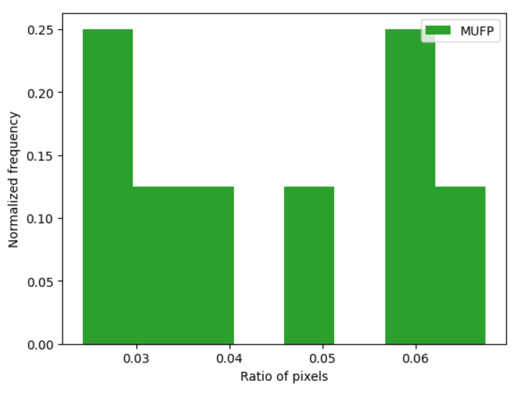


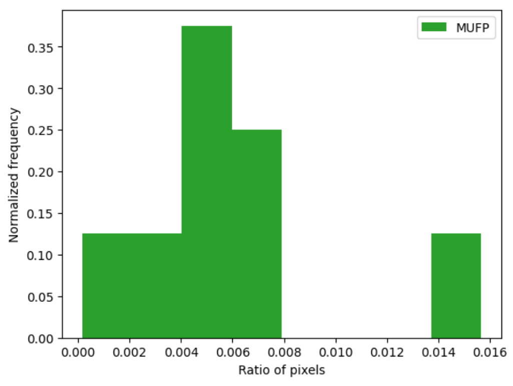
Figure 3.4: Number of pixels per semantic class in MURF.



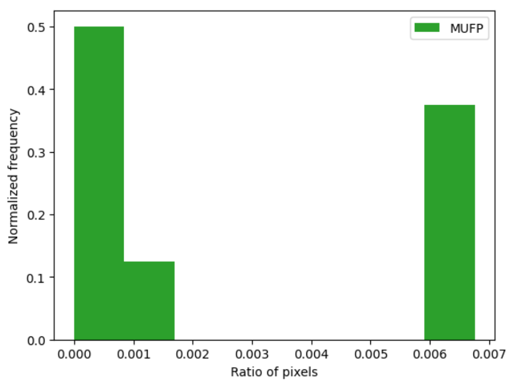
(a) Normalized frequency of background pixels in MURF.



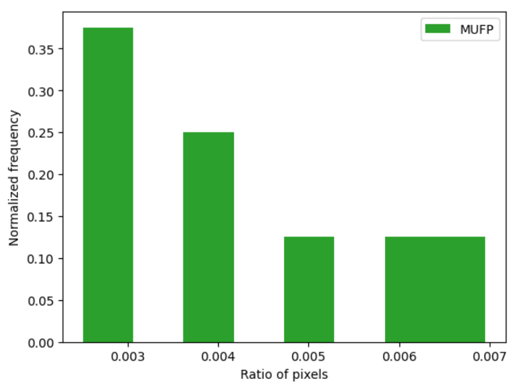
(b) Normalized frequency of walls pixels in MURF.



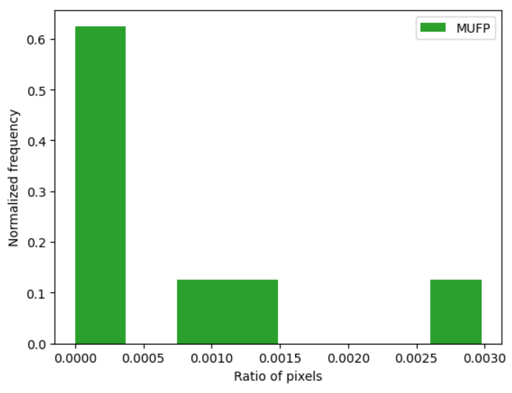
(c) Normalized frequency of glass walls pixels in MURF.



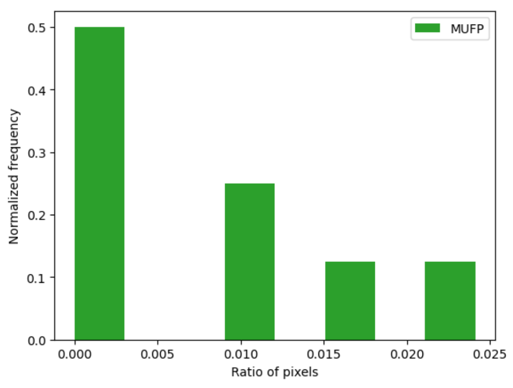
(d) Normalized frequency of railings pixels in MURF.



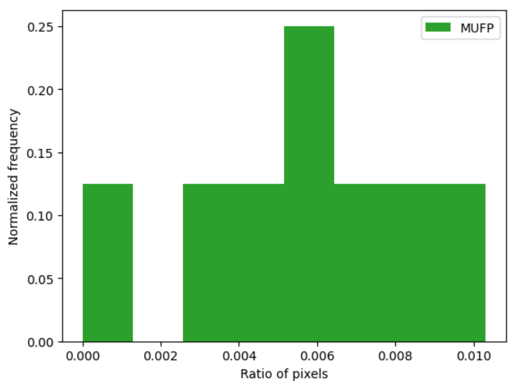
(e) Normalized frequency of doors pixels in MURF.



(f) Normalized frequency of sliding doors pixels in MURF.



(g) Normalized frequency of windows pixels in MURF.



(h) Normalized frequency of stairs pixels in MURF.

Figure 3.5: Normalized frequencies of all semantic classes of MURF.

3.4. Comparative Analysis

This section provides a comparison of MURF and the two public datasets R3D [16] and a refined version of CubiCasa5K [28] named CubiCasa. CubiCasa is proposed to improve the annotation quality of CubiCasa5K and will be discussed further in the next section. The dataset comparison is based on the joint statistics of all datasets. For each dataset three statistics are reported: (1) the distribution of floor plan sizes, (2) the number of class pixels, and (3) the distribution of classes. The set of classes considered for a comparison depends on the datasets included. In the case of R3D, only walls and openings are considered. For the refined version of CubiCasa5K, only walls, railings, doors, windows and stairs are considered. Table 3.2 describes the conversion of MURF labels to the labels of CubiCasa or R3D. The conversion of CubiCasa to R3D is also indirectly shown in Table 3.2. MURF will be compared to R3D and CubiCasa separately in the following subsections. Appendix A contains supplementary examples of floor plans from the two public datasets.

MURF label	CubiCasa label	R3D label
Wall	Wall	Wall
Glass wall	Wall	Wall
Railing	Railing	Wall
Door	Door	Opening
Sliding door	Door	Opening
Window	Window	Opening
Stairs	Stairs	-

Table 3.2: Mapping of semantic classes of MURF to CubiCasa or R3D. Symbol '-' denotes that a class is ignored.

Figure 3.6 shows the different resolutions of the floor plans in MURF compared to R3D [16] and CubiCasa. The center points of the three dataset clusters show that while R3D and CubiCasa somewhat overlap, the floor plans in MURF are significantly larger. The data points of R3D are more structured compared to CubiCasa, likely due to manual preprocessing of floor plans beforehand. The range of R3D is also much smaller compared to CubiCasa, which contains floor plans of a larger range of sizes. The range of floor plan sizes in MURF is limited to large dimensions, but still describes a relatively large range considering it only contains 8 data points.

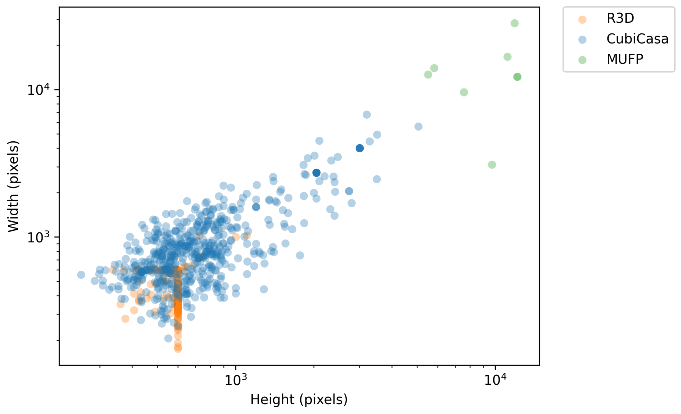


Figure 3.6: Source image resolution of floor plans in MURF compared to R3D [16] and CubiCasa.

3.4.1. Refined CubiCasa5K

The overall annotation quality of the CubiCasa5K dataset [28] is significantly lower compared to the quality of R3D. Class labels are often incorrect, incomplete or contain many unrelated artifacts. For a few floor plans, the annotations do not match due to differences in the current real-world state of apartments [28]. These differences would, however, only confuse a model learning to identify its elements. Figure 3.7 contains examples of common annotation problems. The quality of railings annotations in the floor plans is especially poor. Although not mentioned in the original work [28], stairs have been scarcely labelled in the floor plans.

To improve annotation quality, a manual selection of 560 floor plans of varying size and complexity was made to remove duplicates and find annotations that would be relatively cheap to correct. For each floor plan, all labels were manually corrected. Stairs annotations were added to the floor plans that missed them. This thesis proposes CubiCasa as a refined version of CubiCasa5K, which will be used in the remainder of this thesis. Lu et al. showed that a labelled subset of CubiCasa5K can provide sufficient information for effective learning and evaluation [27].

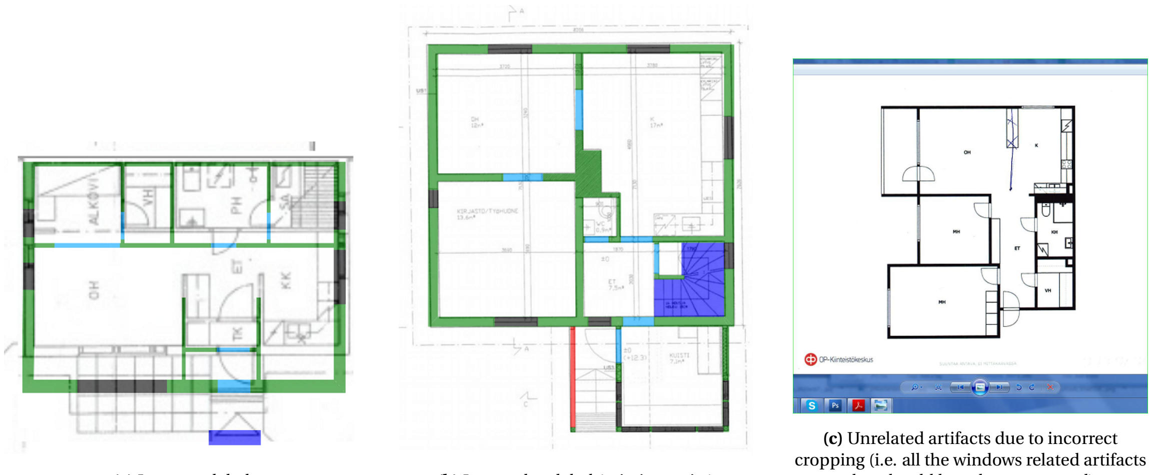
Along with the annotation improvements, CubiCasa also contains rasterized labels in a PNG file format instead of the original SVG files. This was necessary to facilitate a fair comparison between the CubiCasa5K model and the model proposed by this thesis, and to create a combined dataset for learning and evaluation. The former part can be considered fair because the original model proposed by Kalervo et al. effectively rasterized the SVG annotations as well when they are converted into tensors for learning [28].

To generate PNG representations of the source SVG annotations, a small SVG parser was written. The annotations in CubiCasa5K contain many classes and nested relations, of which only a few are evaluated [28]. In order to extract the semantic classes in Table 3.2, the following objects were identified in the SVGs: Stairs, Railing, Wall, Window, Door and Column. The majority of the objects was accompanied by a set of points, which was used to determine the location of the object in the floor plan. When a column was represented by a circular SVG object, the corresponding center point and radius was used. Column objects were merged with wall objects, as they are semantically identical to each other. The parsed SVG files were further processed to generate the PNG masks.

Transforming the parsed SVG files into PNG representations was not trivial due to overlapping class labels. For instance, windows are contained in walls in the annotations, which could result in small wall pixels along the boundaries of windows in generated PNG masks. Another example are railings, which are sometimes contained in stairs classes in the annotations. These overlapping labels had to be corrected because they do not represent the ground-truth accurately and can inhibit the learning process. Two measures were taken to achieve this: (1) a small dilation operation with a 3×3 kernel was applied to all windows, and (2) the annotations were then combined into a single mask, relying on a predefined ‘drawing’ order to fix part of the overlap. The drawing order used to convert the SVG files into PNG representations is shown in Table 3.3.

CubiCasa5K label	Drawing order
Wall	1
Stairs	2
Railing	3
Door	4
Window	5

Table 3.3: Drawing order of CubiCasa5K labels applied to fix overlapping classes. The lowest value is drawn first.



(a) Incorrect label **(b)** Incomplete label (missing stairs) **(c)** Unrelated artifacts due to incorrect cropping (i.e. all the windows related artifacts that should have been removed)

Figure 3.7: Three examples of annotation inconsistencies in the CubiCasa5K dataset.

3.4.2. R3D Comparison

To compare the statistics of MURF and CubiCasa to R3D, the mapping in Table 3.2 is first applied. Figure 3.8 shows the number of pixels per semantic class of each dataset. CubiCasa contains the most data on each of the classes, followed by MURF and R3D. The important observation to make here is that the number of floor plans does not solely dictate the amount of data in a dataset. The size and content of the floor plans are equally important. MURF contains 8 high resolution floor plans, resulting in a comparable number of pixels compared to CubiCasa which is 70 times larger. MURF also exceeds the number of pixels in R3D, which contains 225 floor plans respectively. The ratios of the number of class pixels are identical for the three datasets; the background class has the most pixels followed by walls and openings.

Figure 3.5 shows the normalized frequencies of each dataset considering the three semantic classes of R3D. The distributions of all semantic classes are similar for the three datasets. The distribution of CubiCasa is most varied, followed by R3D and MURF. This variance is partly due to the dataset sizes.

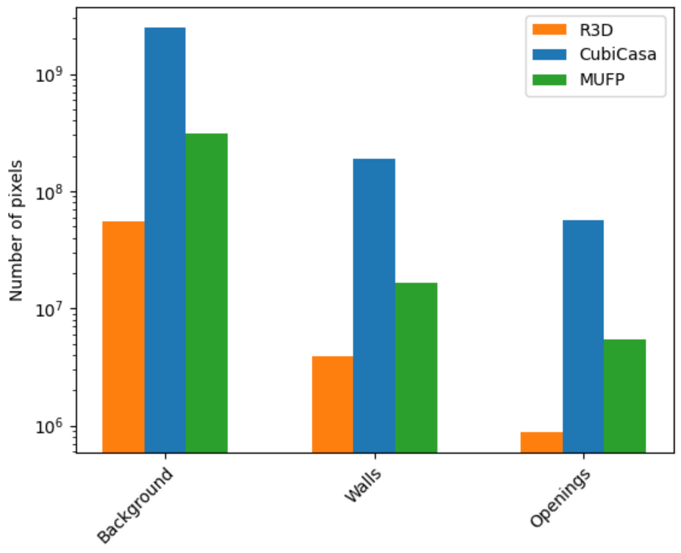
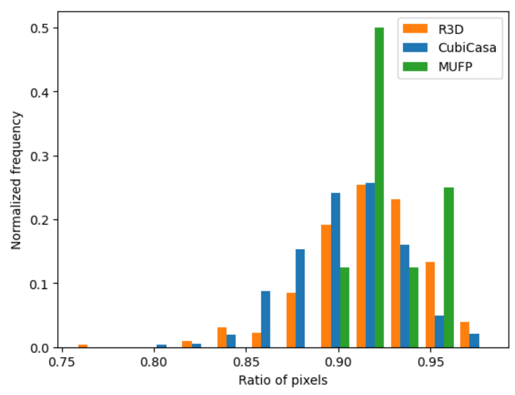
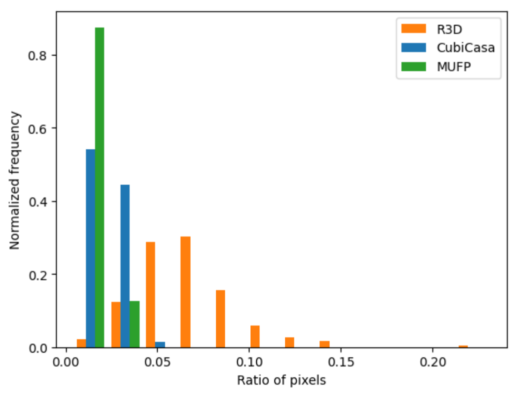


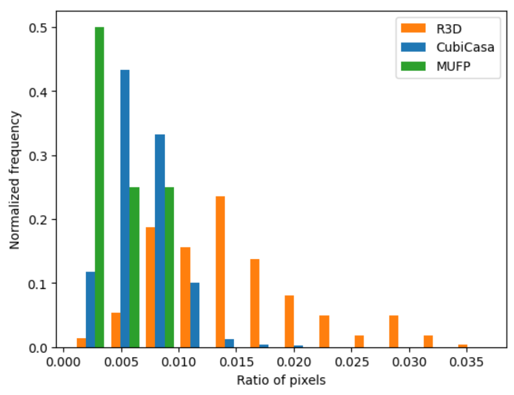
Figure 3.8: Number of pixels per semantic class in MURF compared to R3D [16] and CubiCasa.



(a) Normalized frequency of background pixels in MURF compared to R3D [16] and CubiCasa.



(b) Normalized frequency of walls pixels in MURF compared to R3D [16] and CubiCasa.



(c) Normalized frequency of opening pixels in MURF compared to R3D [16] and CubiCasa.

Figure 3.9: Normalized frequencies of all semantic classes of MURF compared to R3D [16] and CubiCasa.

3.4.3. CubiCasa Comparison

This section compares MURF to CubiCasa by applying the mapping in Table 3.2. Since R3D does not have sufficient semantic classes for this mapping, it is left out of the comparison. Figure 3.10 shows difference in the number of class pixels between MURF and CubiCasa. The ratios of class pixels are almost identical, apart from stairs and door pixels which are swapped with a small difference.

Figure 3.11 shows the normalized frequencies of each semantic class of MURF compared to CubiCasa. The distributions for each class are similar, except for the center points of the background pixels and door pixels in MURF, which are shiften more to the right and left respectively compared to CubiCasa. The frequencies of class pixels in CubiCasa are distributed over a larger range, which is likely due to the large difference in the number of floor plans.

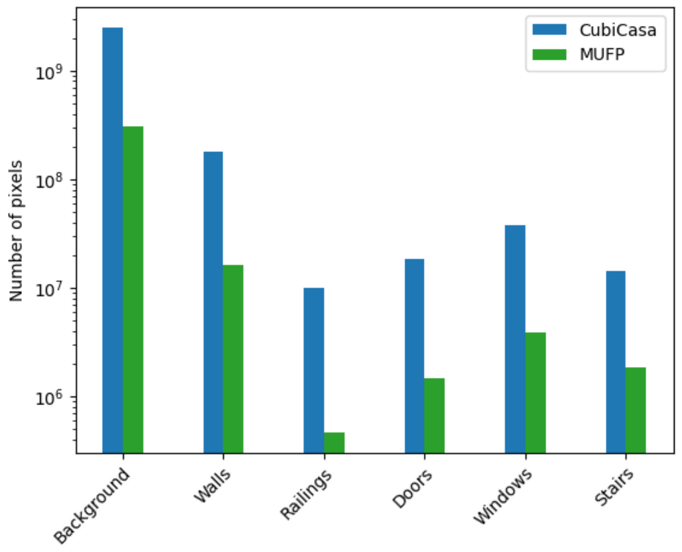
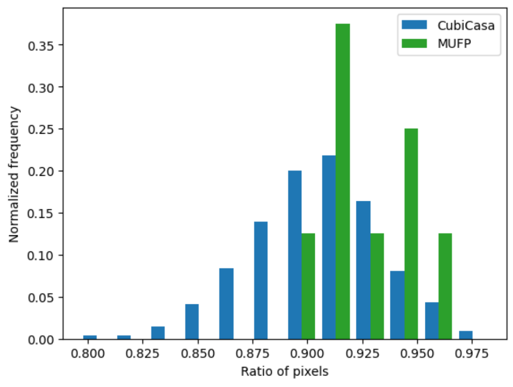
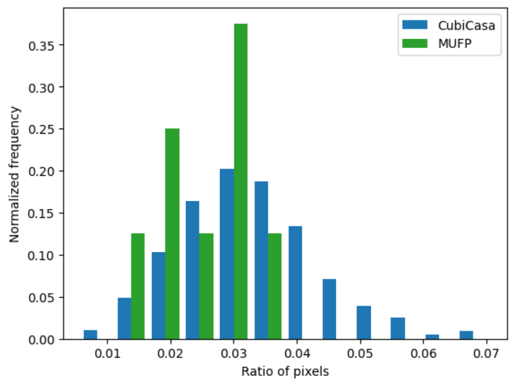


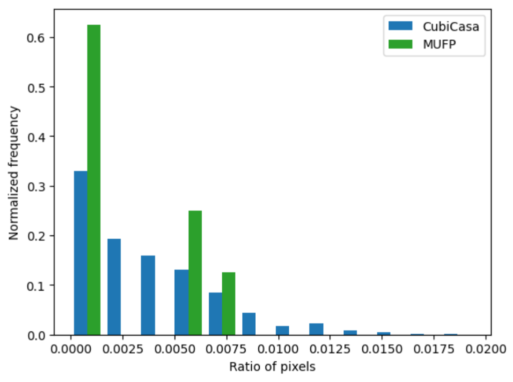
Figure 3.10: Number of pixels per semantic class in MURF compared to CubiCasa.



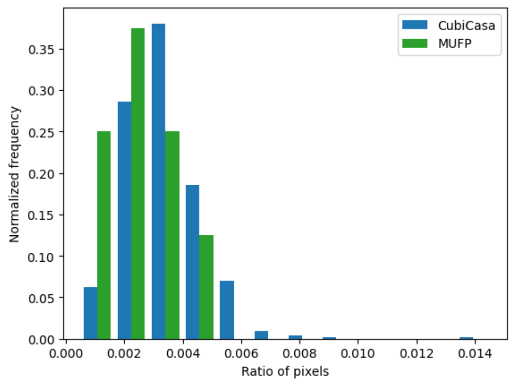
(a) Normalized frequency of background pixels in MURF compared to CubiCasa.



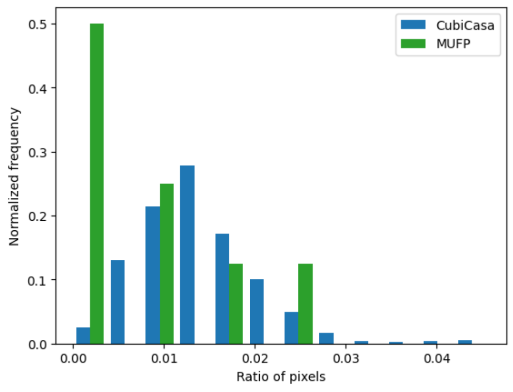
(b) Normalized frequency of walls pixels in MURF compared to CubiCasa.



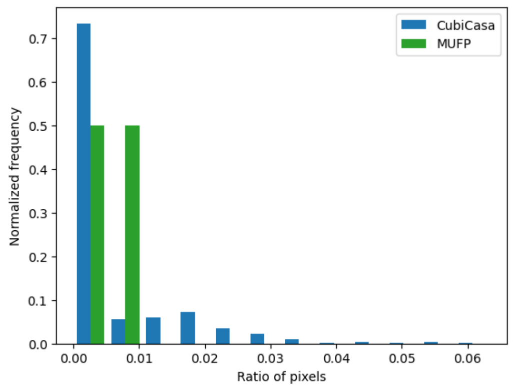
(c) Normalized frequency of glass walls pixels in MURF compared to CubiCasa.



(d) Normalized frequency of railings pixels in MURF compared to CubiCasa.



(e) Normalized frequency of doors pixels in MURF compared to CubiCasa.



(f) Normalized frequency of stairs pixels in MURF compared to CubiCasa.

Figure 3.11: Normalized frequencies of all semantic classes of MURF compared to CubiCasa.

3.5. Reflection on RQ1

This chapter introduced MURF: a novel multi-unit floor plan dataset. This chapter described its data source, annotation process and relevant statistics compared to existing floor plan datasets. RQ1 was posed to answer the following: how can a multi-unit floor plan dataset best be defined for effective learning and visualization? This chapter showed the following:

- A relatively small number of 8 multi-unit floor plans provides sufficient annotation data, compared to R3D and CubiCasa, for a model to generalize on its features in a realistic learning setting.
- The semantic elements in Table 3.1 were determined to be sufficiently semantically distinct for learning, and most useful for related applications of visualized 3D models. The use case of these elements is based on their properties and interaction (e.g. see through, has swing path, traversable, connected to other floors).

4 Method

This chapter discusses the network architecture of the model and its variants proposed in this thesis, including relevant loss functions. Figure 4.1 contains a schematic overview of the entire floor plan processing pipeline, which shows that the proposed method is divided into a recognition (bottom) and reconstruction (top) part. The recognition part focuses on identifying semantic elements in the floor plan. The reconstruction part further processes the detected elements in order to transform them into a 3D model. The recognition and reconstruction parts focus on RQ2 and RQ3 respectively, for which a reflection is discussed in the last section.

4.1. Recognition

This section elaborates on the relevant semantic classes considered in floor plans and the proposed model architectures responsible for the recognition part of the model. The recognition part relies on an FCN based on U-Net3+ [40], and a multi-task training objective designed to capture pixel, patch and multi-scale losses. A deeply supervised version of the proposed architectures will also be discussed.

4.1.1. Semantic Classes

The model proposed in this thesis only considers the detection of structural elements in floor plans. Room types are not considered for two reasons: (1) their definition is often unclear in multi-unit floor plans, and (2) they are not necessary to generate 3D models. Object types such as furniture are not considered for identical reasons. Table 3.1 contains the list of semantic elements the model supports, in addition to the boundary and opening categories which will be used in the training objective for the recognition part.

4.1.2. Proposed Architectures

To develop an effective model architecture, an extensive literature review has been conducted to evaluate 11 recently published state-of-the-art segmentation models [39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]. The goal of this literature review was to create an overview of differences between segmentation models, design choices made, and ultimately come up with a set of potential architectures for models that could prove effective at floor plan analysis. The segmentation models that were considered in the literature review can be separated into two types: FCNs and typical non-fully convolutional CNNs. Two widely used models of both types are U-Net [38] and Mask R-CNN [50], which are able to perform semantic segmentation or instance segmentation respectively. Instance segmentation can be seen as a more challenging task than semantic segmentation, since it treats multiple objects of the same class as a single entity. To determine what type of model architecture might perform best, U-Net and Mask R-CNN were first examined. Only boundary and opening classes were considered for the initial evaluation.

Initial results on Mask R-CNN showed that there are two main disadvantages of using a CNN. The first disadvantage is that the generated masks of Mask R-CNN are of insufficient quality, even when performing semantic segmentation. This is a known side effect of Mask R-CNN, due to the fixed size of 28×28 pixels for regions that are used to generate masks [51]. This produces inaccurate results for objects larger than this size, which is often the case since multi-unit floor plans contain objects of arbitrary size. For

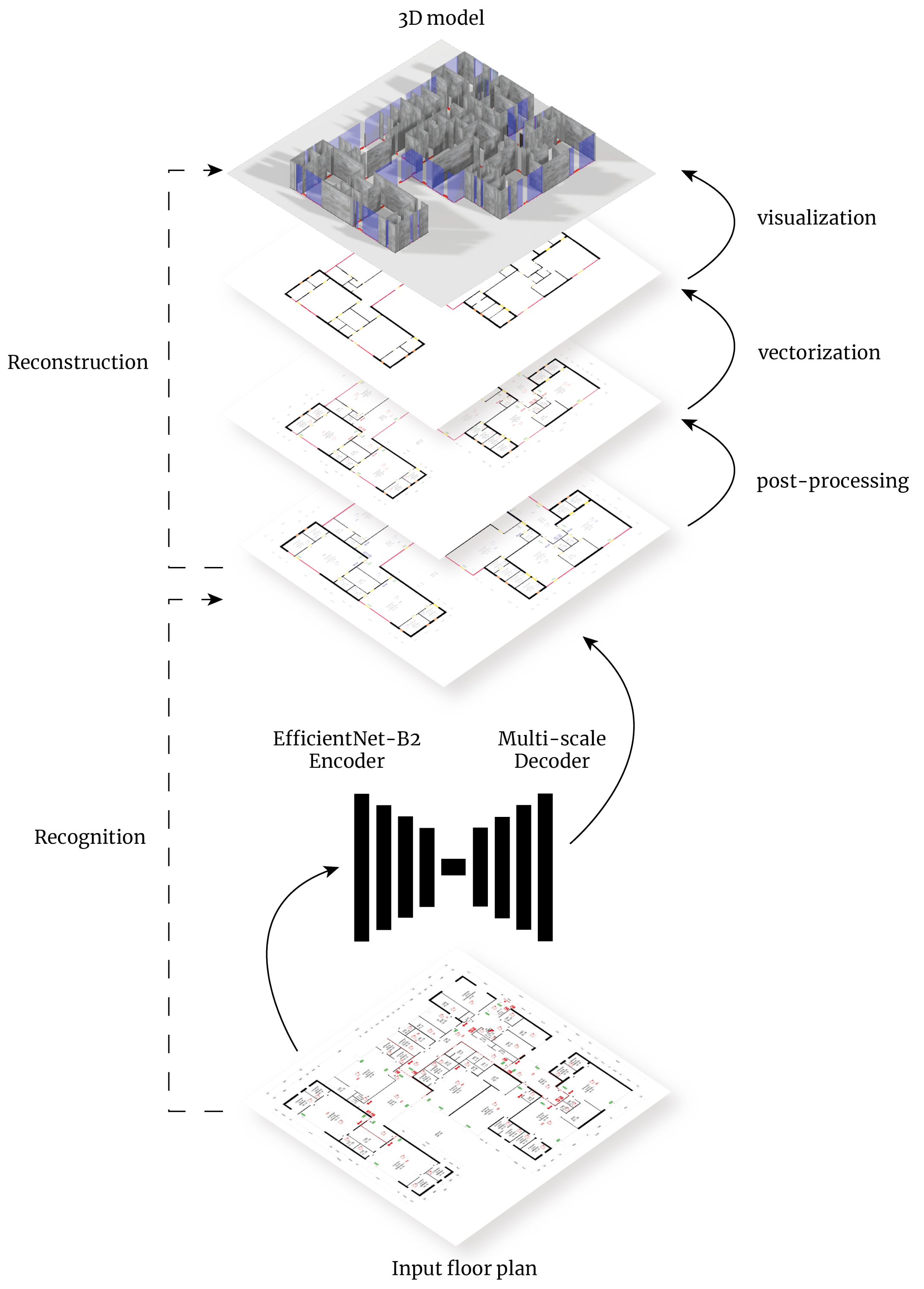
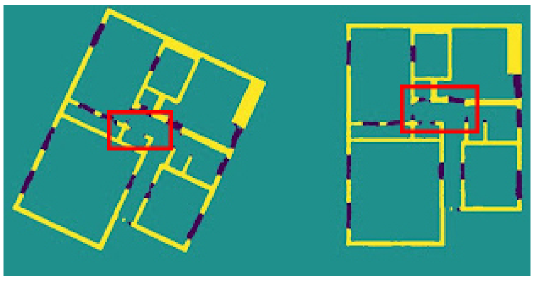


Figure 4.1: Schematic diagram illustrating the steps of the proposed floor plan processing method. The recognition step (bottom part) produces a segmentation mask of the floor plan by using a custom FCN relying on an EfficientNet-B2 encoder. The segmentation mask is subsequently refined through the reconstruction step (top part), which applies post-processing, vectorization and, finally, visualization using Blender.

instance, walls surrounding an entire building are significantly larger than a single door serving as its entrance. The second disadvantage of Mask R-CNN that a non-fully convolutional network cannot infer a mask of larger floor plans than it has been trained on. This means that a sampling approach has to be used to process much larger multi-unit floor plans, limiting scalability. PointRend [51] was proposed to improve the quality of the generated masks of Mask R-CNN. Although PointRend indeed produces more accurate results, the improved accuracy was still significantly lower than an FCN could achieve. Due to the lower accuracy and limited scalability, this thesis only considers FCNs for its model architectures.

As shown in Figure 4.2, U-Net showed more promising results compared to PointRend on early predictions of single-unit floor plans. The highlighted area shows missing opening predictions of PointRend compared to U-Net. The generated masks were of sufficient accuracy and precision, which lead to further evaluation on larger multi-unit plans. The results on multi-unit floor plans of U-Net were significantly poorer. Although boundaries could still be detected fairly well, openings were no longer detected reliably. To improve segmentation performance, two variants of U-Net were examined next. Specifically, U-Net++ [39] and U-Net3+ [40] were considered, both improving upon the original U-Net architecture. The two models showed promising results on an initial evaluation on a small sample of multi-unit floor plans, with U-Net3+ performing best. Due to the improved performance, and reduced parameters compared to U-Net++ [40], U-Net3+ was selected as the starting point for the proposed architectures.


(a) Segmentation mask of PointRend on single-unit floor plan

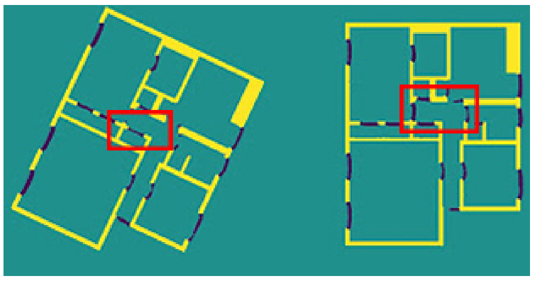

(b) Segmentation mask of U-Net on single-unit floor plan

Figure 4.2: Early results of PointRend and U-Net on single-unit floor plan considering walls and openings highlighted in yellow and blue. Problematic areas of PointRend highlighted in red.

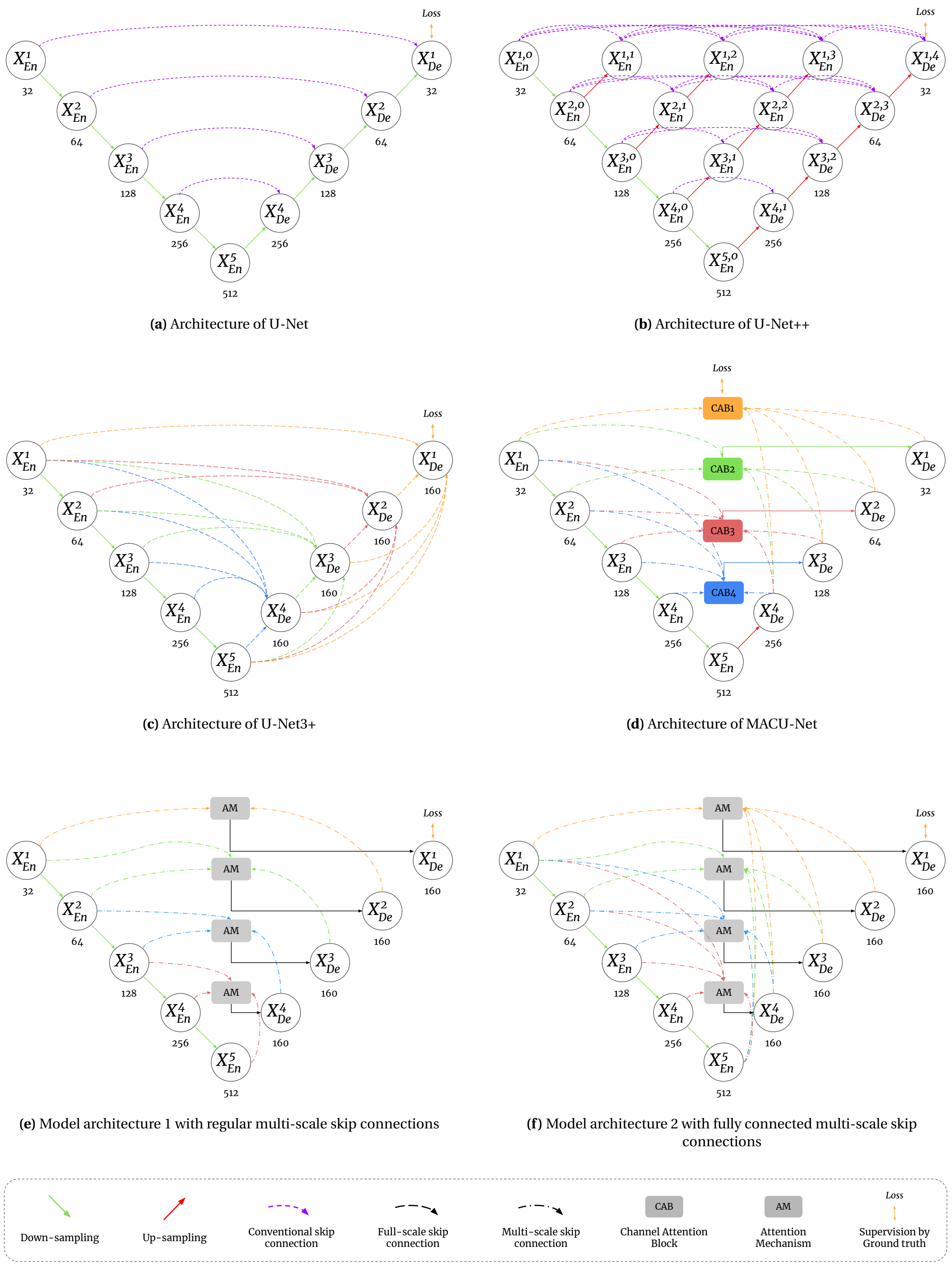
The architectures for the main semantic model of this thesis are thus based on U-Net3+ [40]. Figure 4.3 depicts the two model architectures considered in comparison with the existing architectures of U-Net [38], U-Net++ [39], U-Net3+ [40] and MACU-Net [41]. The remainder of this subsection will dive into the specific differences between existing models and the model architectures proposed in in this thesis. The following will be discussed: normalization, re-designed skip-connections, convolution blocks, attention mechanism and the encoder backbones used.

Normalization

Due to GPU memory constraints on our machine, the maximum batch size of floor plans is limited to small values (≤ 2). Using native Batch Normalization [52] or Instance Normalization [53] would not be an effective normalization method due to inaccurate mean and variance statistics used [54]. Two alternative normalization techniques can mitigate this problem: Group Normalization (GN) [54] and Filter Response Normalization (FRN) [55]. GN computes its mean and variance over a predefined group size, smaller than the number of channels for a given feature map [54]. FRN computes statistics over the width and height for a single image, eliminating the dependence on other batch elements [55]. FRN has been shown to marginally outperform GN on small batch sizes, and would therefore be the logical choice. Nevertheless, this thesis opts to use GN instead. The reason for this is that the dependence of FRN on the width and height of an input restricts a model to only operate on a fixed input size. This would, however, interfere with multi-scale feature learning and, more importantly, multi-scale inference. FRN normalization is thus incompatible with the model proposed in this thesis, which will use GN instead.

Multi-scale skip connections

The design and use of skip connections is an important part of the architecture of an effective FCN. Skip connections can combat the negative side-effects that an FCN introduce, such as spatial information loss, convergence instability and vanishing gradients [56]. Several types of skip connections can be identified from existing literature: short, long, nested, dense, full-scale, and multi-scale skip connections [38, 39, 40, 41, 57]. Short and long skip were proposed first and have been evaluated extensively. Nested,


Loss
Down-sampling
Up-sampling
Conventional skip connection
Full-scale skip connection
Multi-scale skip connection
CAB
Channel Attention Block
AM
Attention Mechanism
Supervision by Ground truth
CAB1
CAB2
CAB3
CAB4
32
64
128
256
512
160

(a) Architecture of U-Net

(b) Architecture of U-Net++

(c) Architecture of U-Net3+

(d) Architecture of MACU-Net

(e) Model architecture 1 with regular multi-scale skip connections

(f) Model architecture 2 with fully connected multi-scale skip connections

Figure 4.3: Two model architectures in comparison with existing segmentation models: (e) regular multi-scale skip connections, and (f) fully connected multi-scale skip connections.

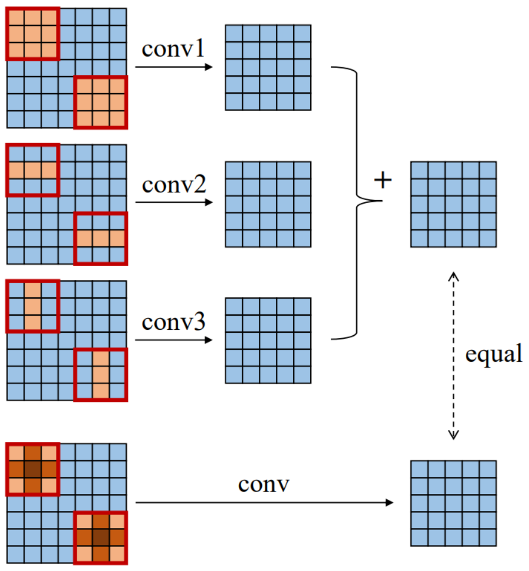
dense and full-scale skip connections are specific variants of short or long skip connections. Applying a combination of long and short skip connections in an FCN is generally beneficial, improving spatial feature utilization, convergence time, and combating vanishing gradients [56]. It is, however, important to realize that the performance of the skip connections in a specific architecture depends heavily on the learning context including the dataset, feature shape and feature scale [58]. Too many skip connections can, for instance, have a negative impact on the segmentation accuracy [58]. State-of-the-art segmentation models have proposed several distinct types and combinations of skip connections, none of which have been evaluated on floor plan datasets. This thesis will therefore propose two model architectures with different skip-connections based on literature. The best performing architecture will be determined through empirical evaluation.

Several types of re-designed skip connections have been proposed to close the semantic gap between features in the original U-Net architecture, improving overall segmentation performance. U-Net++ uses nested and dense connections shown in Figure 4.3b to reduce this semantic gap, but at the cost of significant computational overhead [39]. U-Net3+, shown in Figure 4.3c achieves comparable performance to U-Net++ by using fully connected full-scale skip connections. Recent works introduce *multi-scale* skip connections depicted in Figure 4.3d, similar to full-scale connections, but without the assumption that all channels of feature maps share equal weights [41]. Multi-scale connections thus seem most promising from related work, which will be used for the model architectures considered in this thesis.

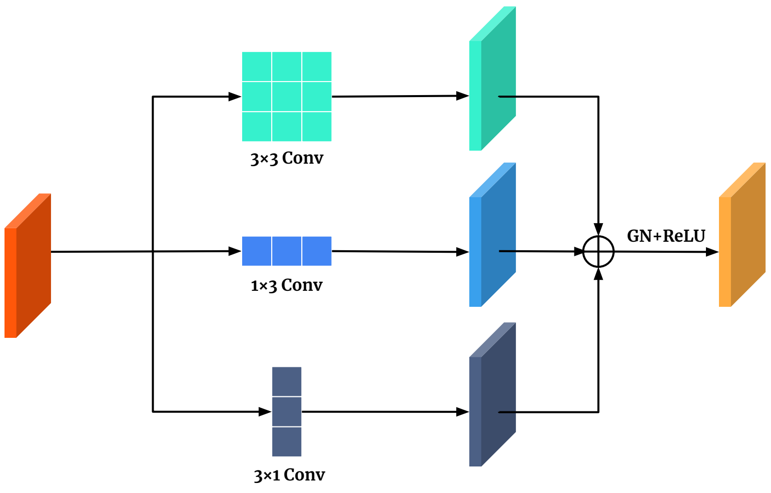
There are various ways of combining multi-scale skip connections, which can have a significant impact on segmentation performance. Furthermore, a model architecture using more skip connections does not necessarily perform better than one with fewer skip connections [58]. MACU-Net, shown in Figure 4.3d, uses fully connected multi-scale connections, where features from multiple decoder neurons are fused with multiple decoder neurons [41]. MDA-UNet applies similar multi-scale connections where multiple encoder neurons are fused with a single decoder neuron [42]. The encoders combined with decoder neurons are from the same or previous (i.e. shallower) depth. Since MACU-Net and MDA-UNet have not been evaluated on floor plan datasets, this thesis evaluates both by incorporating them into the two model architectures. Figure 4.3e and Figure 4.3f show the two model architectures considering regular and fully connected multi-scale skip connections.

Asymmetric convolution block

To better capture learned feature maps, different types of convolution blocks can be used. In addition to the normalization and activation functions paired with convolutional layers, the design and use of kernels play an important role in improving the performance of a CNN [59]. This thesis proposes an Asymmetric Convolution (AC) block, as depicted in Figure 4.4, to replace the standard square convolutions in U-Net3+ [40]. Asymmetric convolution blocks have been shown to enhance skeleton features by replacing standard $k \times k$ kernels with the aggregation of the $k \times k$ kernel and two additional horizontal $1 \times k$ and vertical $k \times 1$ kernels [59]. Directional kernels have also been evaluated on floor plans, but do not improve clear performance improvements compared to horizontal and vertical kernels [16, 60].



(a) Enhancement of skeleton features by asymmetric convolution from [59].



(b) *AC* block using group normalization.

Figure 4.4: Asymmetric Convolution (*AC*) block enhancing skeleton features. ⊕ represents element-wise addition.

Attention Mechanism

Attention mechanisms are an umbrella term for methods that refine generated feature maps to more effectively capture the learned representations. Refining features to improve their representation power can, for instance, be achieved by enhancing important features and suppressing the others [61]. Two types of attention mechanisms that have been extensively evaluated in related work are *Channel Attention Modules* (CAM) and *Spatial Attention Modules* (SAM). Channel attention modules aim to improve feature utilization by weighing feature maps according to the 'importance' of their channels [61]. The importance of a channel can, for instance, refer to how much a channel differs from others. Spatial attention modules aim to weigh feature maps according to extracted refined spatial features. Using a combination of channel and spatial attention modules can improve performance [42, 61]. Based on these findings, this thesis opts to use both a channel and attention block to improve segmentation performance compared to U-Net3+.

The proposed attention module infers a channel attention map M_c and spatial attention map M_s from an intermediate feature F, which is subsequently weighed through multiplication formalized by Equation 4.1. The final refined feature is computed by performing asymmetric convolution on the concatenated channel and spatial refined feature. An overview of the AM is shown in Figure 4.5:

$$\begin{aligned} \mathcal{A}(F) &= AC\Big(\big[F_c \otimes M_c(F), F_s \otimes M_s(F)\big]\Big), \\ F_c &= C^{1\times1}(F), \\ F_s &= C^{1\times1}(F), \end{aligned} \tag{4.1}$$

where F_c and F_s are channel compressed representations of intermediate feature map F using two 1×1 convolutions. Note that the spatial map M_s is derived from the full sized feature F. Concatenation is denoted by $[\cdot]$.

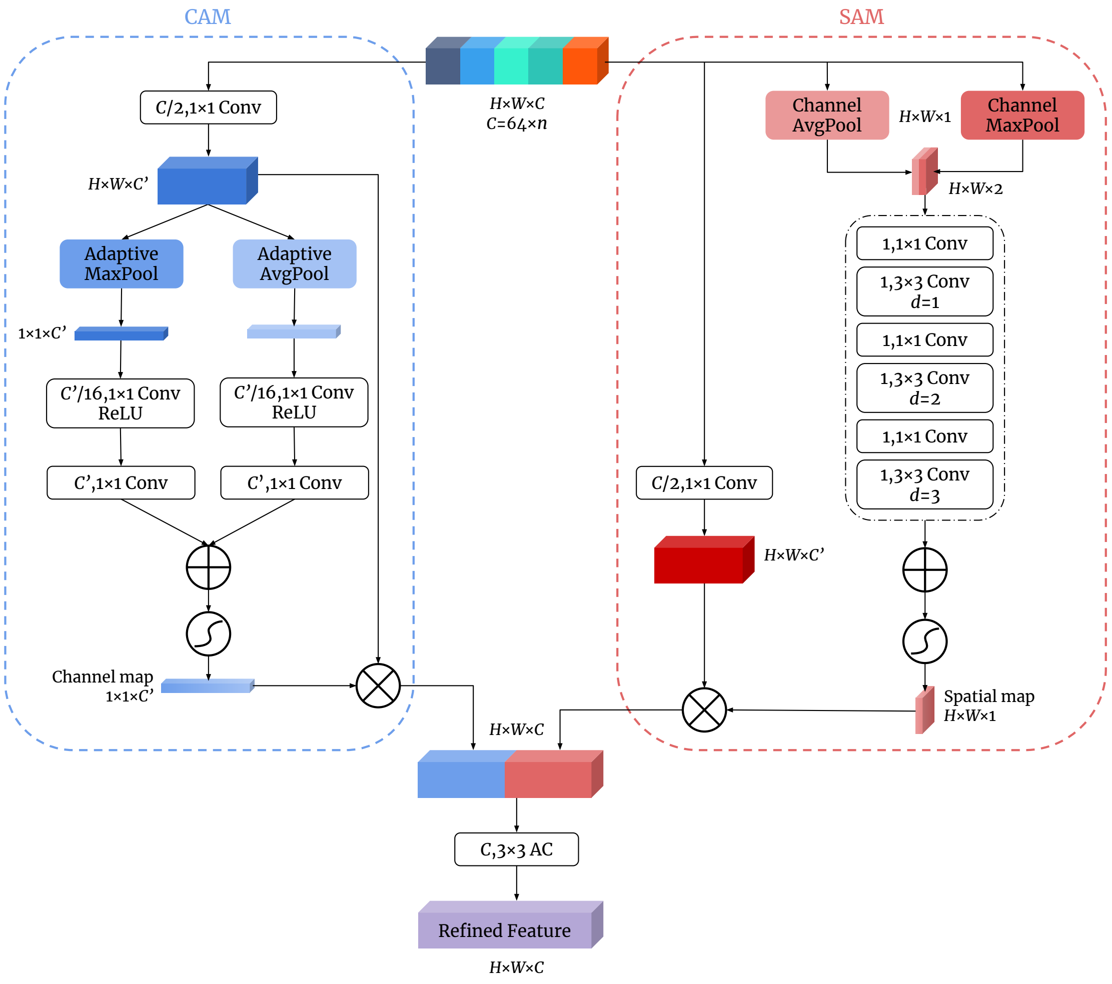


Figure 4.5: Overview of Attention Module consisting of Channel Attention Module (CAM) and Spatial Attention Module (SAM) enhancing channel and spatial features respectively. ⊕, ⊗ and Ⓢ denote element-wise addition, multiplication and the sigmoid operation.

The intermediate feature map F of the model architectures is constructed similar to that of U-Net3+ [40]. For each up-sampling level $i < N$ and respective decoder X^i_{De}, the feature map F^i to be refined consists of a mix of different scaled feature maps from encoder and decoder layers. The first decoder layer is equal to last encoder layer. The scaled feature maps for the other decoders are aggregated through concatenation. Depending on depth i of the decoder, features from other encoder and decoders are first either up- or down-sampled to match the spatial dimensions of F^i. The different sampling operations improve the representation power of the model by merging both high-level and low-level semantic information [40]. Down-sampling and up-sampling are done by a max pooling operation and bilinear interpolation respectively. Other alternatives for sampling layers have been evaluated, such as average pooling and transposed convolution, but performed worse. Directly concatenating the resized feature maps would result in an aggregated feature map consisting of a large number of channels. The asymmetric convolution with 32 filters is used to reduce the number of channels in each feature maps from another layer. The number of filters is set equal to the number of filters in the first encoder block to capture sufficient information. The final intermediate feature map is equal to the concatenation of $32 \times n$ feature maps from the other encoder and decoder layers. The decoder layer X_{De}^{i} is equal to the refined feature map of F^{i} obtained through the attention mechanism. The intermediate feature map of the fully connected model architecture 2 is formalized by Equation 4.2 and depicted in Figure 4.6. The intermediate feature map of the regularly connected architecture is constructed similarly, but only consists of the feature maps from the encoders and decoders of the same and previous depth. Equation 4.3 formalizes the feature map of model architecture 1.

$$X_{De}^{i} = \begin{cases} X_{En}^{i} & , i = N \\ \mathcal{A}\left(\left[\underbrace{AC\Big(\mathcal{D}(X_{En}^{k})\Big)_{k=1}^{i-1}, AC(X_{En}^{i}),}_{\text{Scales:}1^{\text{th}}\sim i^{\text{th}}} \underbrace{AC\Big(\mathcal{U}(X_{De}^{k})\Big)_{k=i+1}^{N}}_{\text{Scales:}(i+1)^{\text{th}}\sim N^{\text{th}}}\right]\right) & \text{, i=1, ..., i=N-1} \end{cases} \tag{4.2}$$

$$X_{De}^{i} = \begin{cases} X_{En}^{i} & , i = N \\ \mathcal{A}\left(\left[\underbrace{AC\Big(\mathcal{D}(X_{En}^{i-1})\Big), AC(X_{En}^{i}),}_{\text{Scales:}1^{\text{th}}\sim i^{\text{th}}} \underbrace{AC\Big(\mathcal{U}(X_{De}^{i+1})\Big)}_{\text{Scales:}(i+1)^{\text{th}}\sim N^{\text{th}}}\right]\right) & \text{, i=1, ..., i=N-1} \end{cases} \tag{4.3}$$

where $\mathcal{A}$ is the attention mechanism, AC is the asymmetric convolution, $\mathcal{D}(\cdot)$ and $\mathcal{U}(\cdot)$ are up- and down-sampling, $[\cdot]$ represents concatenation, and $C(\cdot)$ is a regular convolution operation.

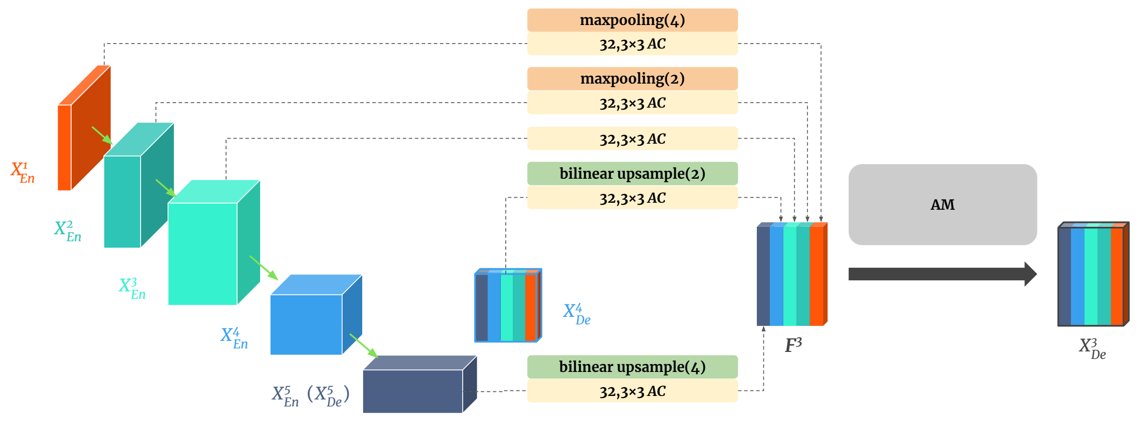


Figure 4.6: Example of intermediate feature map F^{3} of third decoder layer X_{De}^{3}. AC and AM indicate the asymmetric convolution block and attention mechanism.

Channel Attention Module The disadvantage of directly aggregating full-scale features, as proposed by U-Net3+ [40], is that this assumes the weights of the channels of these features are equal. To better utilize the features, a channel attention module is proposed to weigh the feature map $F \in \mathbb{R}^{H\times W\times C}$ according to a channel feature map $M_c \in \mathbb{R}^{1\times 1\times C/2}$, where C indicates the number of channels of the feature map. Multiplying the intermediate feature map with the channel feature map allows the model to reinforce informative channels and restrain indiscriminative channels [62]. A variant of the channel attention block used in MACU-Net [41] will be used by this thesis. The advantage of this approach, compared to others such as MDA-UNet, is that the CAM is designed to process features of arbitrary size. Although the structure of the CAM used in this thesis is similar to that of MACU-Net, there are three notable differences: (1) the contents of the intermediate feature map, (2) the number of filters used in convolution operations, and (3) the application of the channel-refined feature map.

The channel feature map $M_c \in \mathbb{R}^{1\times 1\times C/2}$ is inferred along the spatial dimensions of an intermediate feature, as formalized by Equation 4.4. Due to the large number of channels in the original input feature map, a compressed version F_c with half the number of channels is first computed by 1×1 convolution. The spatial dimensions of F_c are then squeezed through separate adaptive average pooling and max pooling producing F_c^{avg} and F_c^{max} of $1\times 1\times C/2$. Separate and simultaneous average and max pooling has been shown to improve representation power of a model [61]. The extraction of important channels is done by compressing the pooled feature maps into one-sixteenth of their original size using a convolution and ReLU operation. To produce the channel feature map, the channel features are expanded to $C/2$ channels and aggregated by element-wise addition. The merged channel map is normalized by the sigmoid function.

$$M_c(F) = \sigma\Big(C^{1\times 1}\Big(\text{ReLU}\Big(C^{1\times 1}\Big(F_c^{\text{avg}}\Big)\Big)\Big) + C^{1\times 1}\Big(\text{ReLU}\Big(C^{1\times 1}\Big(F_c^{\text{max}}\Big)\Big)\Big)\Big) \tag{4.4}$$

where σ, ReLU and C denote the sigmoid function, ReLU activation, and a convolution operation.

Spatial Attention Module The aim of a spatial attention module is to highlight important features by weighing the feature map $F \in \mathbb{R}^{H\times W\times C}$ according to a spatial feature map $M_s \in \mathbb{R}^{H\times W\times 1}$, where H, W and C are the height, width and number of channels of the feature map. MDA-UNet has applied a variant of a spatial attention module to better fuse multi-scale features in U-Net3+ [42]. The limitation of the spatial attention module in MDA-UNet is that it was designed to refine a single encoder feature. This thesis proposes an improved spatial attention module, considering both regular and fully connected multi-scale features. Figure 4.5 depicts the spatial attention module used in the model architectures.

The spatial attention module (SAM) aims to learn a spatial feature map $M_s \in \mathbb{R}^{H\times W\times 1}$ by inferring along the input features' channel dimension. The value of the spatial map is formalized by Equation 4.5. To compute the spatial attention map, average and max channel pooling [63] is first applied to produce $F_s^{\text{avg}} \in \mathbb{R}^{H\times W\times 1}$ and $F_s^{\text{max}} \in \mathbb{R}^{H\times W\times 1}$ respectively. Channel pooling is used because it has been shown to be effective at enhancing spatial features [61]. The concatenated pooled feature maps are then passed to the spatial convolutional layer C_s which enhances spatial features using different dilation rates. Concatenating average and max-pooled feature maps has been shown effective at reinforcing spatial features [42]. Different dilation rates have been used to capture features from different-sized receptive fields, reinforcing both small and large spatial features. The spatially enhanced features are aggregated through element-wise addition and normalized by the sigmoid function to obtain the final spatial map.

$$\begin{aligned} M_s(F) &= \sigma\Big(C_s\Big(\Big[F_s^{\text{avg}}, F_s^{\text{max}}\Big]\Big)\Big), \\ C_s(F) &= \sum_{i=1}^{3}\Big(C^{1\times 1}(F) + C^{3\times 3, d=i}(F)\Big) \end{aligned} \tag{4.5}$$

where σ and C_s denote the sigmoid function and spatial convolutional block respectively.

Backbone

Variants of U-Net have been proposed as backbone-agnostic extensions. Improved variants of U-Net have considered different backbone models responsible for encoding the input features, which include VGG, ResNet and DenseNet encoders [39, 40, 48]. This thesis opts to use EfficientNet as the backbone model for the encoder, which has been shown to outperform the aforementioned encoders [64]. Eight versions of EfficientNet ranging from B0 to B7 with increasing complexity have been proposed. This thesis opts to use EfficientNet-B2, offering a good balance between the complexity and accuracy.

4.1.3. Deep Supervision

Lee et al. proposed deeply-supervised nets that improved the learning ability of intermediate layers [65]. Applying deep supervision also combats vanishing gradients and slow convergence [39, 40, 65]. Deep supervision has shown to be promising in state-of-the-art segmentation models, improving the use of multi-scale features [39, 40]. U-Net++ [39] proposed a deeply supervised variant applying losses to the shallowest decoders. Unet3+ [40] applied deep supervision to all decoder neurons, which are first upsampled to the match spatial dimensions. At inference time, an average of all supervised neurons is often used. This thesis proposes to use deep supervision to reinforce multi-scale feature learning by applying a different loss function to each decoder neuron of a distinct scale. The loss function applied to each decoder differs by varying the scales the heatmap regression loss considers. Similar to U-Net3+, the average up-sampled predictions of the proposed architectures are also used for inference. Up-sampling is done by bilinear interpolation. Figure 4.7 shows the deeply supervised variant of model architecture 2. The deeply supervised variant of proposed architecture 1 is constructed similarly.

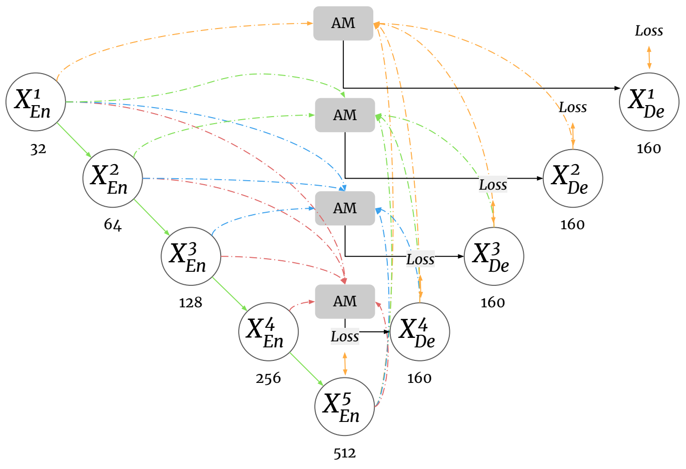


Figure 4.7: Deeply supervised variant of model architecture 2. A loss term is computed for each decoder neuron.

4.1.4. Training Objective

The training objective of the segmentation model consists of multiple tasks, each designed for a specific purpose: handling class imbalance, utilizing semantic relations in labels, and utilizing multi-scale predictions. Each task can also be seen as a form of pixel, patch and multi-scale loss respectively, helping to process floor plans of arbitrary size.

Unified focal loss

Unified focal loss is used to handle class imbalance by suppressing the background loss and enhancing the foreground loss [66]. The class imbalance in the floor plan datasets is mainly due to the background pixels. For instance, in the augmented CubiCasa datset, the ratio foreground to background is 1 : 16. Fundamentally, the unified focal loss relies on a generalization of Focal loss and Focal Tversky loss, improved versions of cross entropy and dice loss respectively [66]. Two key advantages of using the unified focal loss is that (1) the focal hyperparameters for each loss term have been grouped, reducing their number significantly, and (2) the focal losses have been modified so that they are applied separately per class. The latter advantage is distinct from the regular Focal loss, which applies the focal effect to all classes.

The asymmetric variant of unified focal loss $\mathcal{L}_{\mathrm{uAF}}$ as formalized by Equation 4.6 was shown to be most effective and is used here [66]. Term λ is the focal hyperparameter weighing the background suppression and foreground enhancement.

$$\mathcal{L}_{\mathrm{aUF}} = \lambda \mathcal{L}_{\mathrm{maF}} + (1-\lambda)\mathcal{L}_{\mathrm{maFT}} \tag{4.6}$$

where $\mathcal{L}_{\mathrm{maF}}$ and $\mathcal{L}_{\mathrm{maFT}}$ are the modified asymmetric Focal loss modified asymmetric Focal Tversky loss respectively.

Adaptive affinity field loss

Unified focal loss applies a form of unary supervision, which lacks the spacial discrimination to exploit semantic structure in labels [67]. To capture this structure, affinity field loss is used. Affinity field loss employs both a grouping force $\mathcal{L}_{\mathrm{group}}$ and separating force $\mathcal{L}_{\mathrm{separate}}$ in the prediction map $\hat{y}$. The degree of each force depends on whether a pixel and its neighbours belong to the same category c in the ground-truth label y.

$$\mathcal{L}^{cki}_{\text{group}} = \sum_{j \in \mathcal{N}_k(i)} \begin{cases} D_{\text{KL}}(\hat{y}_j(c) \,||\, \hat{y}_i(c)) & \text{if } y_i(c) = y_j(c) \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

$$\mathcal{L}^{cki}_{\text{separate}} = \sum_{j \in \mathcal{N}_k(i)} \begin{cases} \max\{0, m - D_{\text{KL}}(\hat{y}_j(c) \,||\, \hat{y}_i(c))\} & \text{if } y_i(c) \neq y_j(c) \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

where $\mathcal{N}_k(i)$ denotes the neighbourhood for a pixel i of size k. The neighbourhood $\mathcal{N}_k$ is defined by the 8 endpoints in the 3×3 kernel with dilation rate k. For $k = 3$ this is equal to the Moore neighbourhood of a pixel. $D_{\text{KL}}(\cdot)$ s the Kullback-Leibler divergence between two distributions.

The adaptive affinity field loss $\mathcal{L}_{\text{AAF}}$ learns the optimal kernel size out of set K per class through training by formulating the loss as a minimax problem. Maximizing the affinity field loss ensures the most critical neighbourhood sizes are used. This approach is chosen since the optimal set of kernels is context dependent and can be expensive to find [67].

$$\begin{gathered} \mathcal{L}_{\text{AAF}} = \max_{w} \Big\{ \sum_{c}^{C} \sum_{k}^{K} \sum_{i} (w^{\text{group}}_{ck} \mathcal{L}^{cki}_{\text{group}} + w^{\text{separate}}_{ck} \mathcal{L}^{cki}_{\text{separate}}) \Big\}, \\ \text{s.t. } \sum_{k}^{K} w^{\text{group}}_{ck} = \sum_{k}^{K} w^{\text{separate}}_{ck} = 1 \text{ and } w^{\text{group}}_{ck}, w^{\text{separate}}_{ck} \geq 0 \end{gathered} \tag{4.9}$$

Multi-scale heatmap regression loss

Relying on a soft loss function can help stabilize training by guiding a network during training [68, 69]. Heatmap regression can be an effective measure to compute such a loss, which shows promising results for human pose estimation. Lv et al. introduce opening regression loss, utilizing predicted heatmaps to regress the boundaries of openings [17].

This thesis introduces a multi-scale heatmap regression loss for openings that considers the regressed opening predictions based on multi-scale heatmaps derived from ground-truth labels. This is distinct from the approach proposed by Lv et al. [17], as no heatmaps are directly generated by the model. Instead, the heatmaps are indirectly derived from the segmentation masks. It has been shown that a similar approach reduces the memory footprint of a model significantly [70].

Another difference between the proposed approach and that of Lv et al. [17] is how the ground-truth heatmaps are created. In [17] the endpoints of openings are used to create heatmaps, which are manually defined for their new dataset. Instead of manually adding these endpoints to the existing datasets, this thesis derives the endpoints from the opening labels by further processing them. Generating the heatmaps is composed of the following three steps, also depicted in Figure 4.8.

1. Find the connected components of category c using [71] (Figure 4.8b).
2. Find the contour of each component of step 1 using [72] (Figure 4.8c).
3. Given scale s and β, compute the value of each pixel i of category c using Equation 4.10 (Figure 4.9)

$$H^s_{c\beta}(i) = \max_{j \in \mathcal{O}_c} \exp\Big(-\frac{||y_i(c) - y_j(c)||^2_2}{\beta^2}\Big) \tag{4.10}$$

where $\mathcal{O}_c$ is the set of endpoints of opening c, and β controls the spread of the peak of the values in the heatmap.

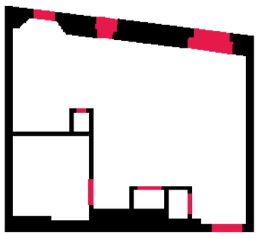
Given the heatmaps H, the heatmap regression loss $\mathcal{L}_{\text{MHR}}$ is a simple average of the squared euclidean distance between the prediction map $\hat{y}$ and the heatmap over all scales S, openings O and pixels i in the floor plan. This is formalized by Equation 4.11. The reason for using the average heatmap $\frac{1}{|B|}\sum^B_\beta(H^s_{c\beta}(i))$ is that this motivates the model to keep refining its openings prediction. An example of such a combined heatmap is shown in Figure 4.10.

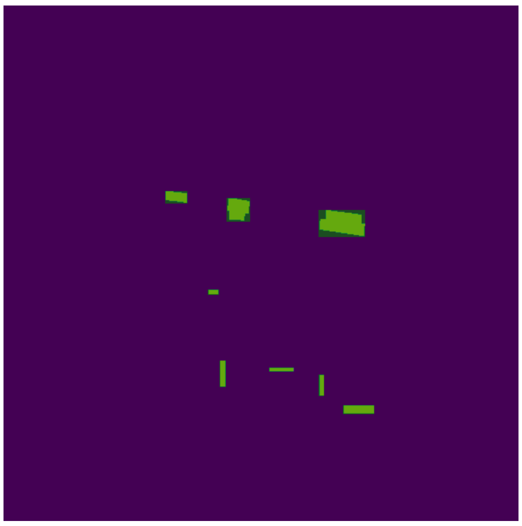
$$\mathcal{L}_{\text{MHR}} = \frac{1}{|S|}\sum_{s}^{S} \frac{1}{|O|}\sum_{c}^{O} \frac{1}{|\hat{y}|}\sum_{i} |\hat{y}_i(c) - \frac{1}{|B|}\sum_{\beta}^{B}(H^s_{c\beta}(i))| \tag{4.11}$$

where $O \subset K$ is the subset of opening categories consisting of the elements described in Table 3.1. Note that only $|\hat{y}_i(c) - \cdot|$ refers to the absolute difference as apposed to the length.

The values controlling the spread in the heatmap in B must be chosen carefully. If too high, values in the resulting heatmap equal or larger than 0.5 outside of an opening could confuse the model trying to accurately detect it. Thus, the combined mean heatmap of values B should have a sharp decline towards 0.5, followed by a slower decrease in value in order to facilitate learning. The mean heatmap with $B = \{2, 10\}$, as illustrated in Figure 4.11, can model this effectively.

The different scales S play an important role in deep supervision, for which the heatmaps can be used to train the model to learn opening features across multiple scales [69]. The heatmap loss can then be computed for each decoder, with increasing scale as the resolution of the decoder increases. Figure 4.12 shows the same heatmap generated at different scales by down-sampling. If deep supervision is not used, only one scale of heatmaps is considered for the loss.


(a) Ground-truth label, boundaries and openings highlighted in black and red respectively.


(b) Opening connected components highlighted in green.

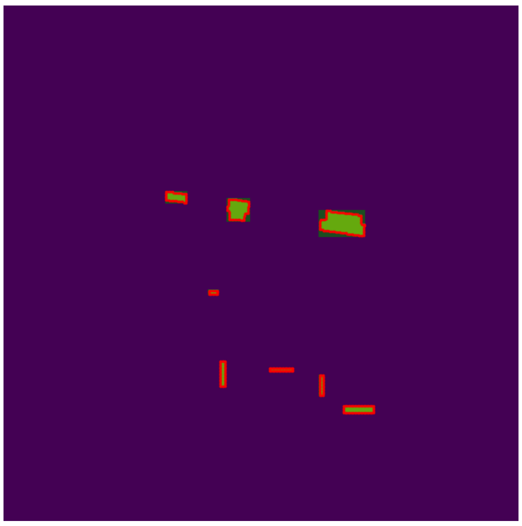
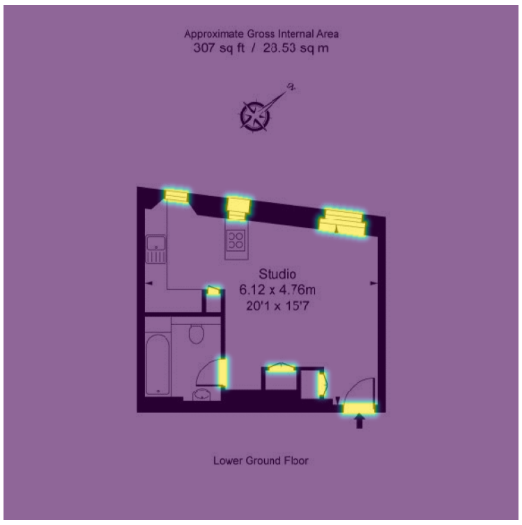
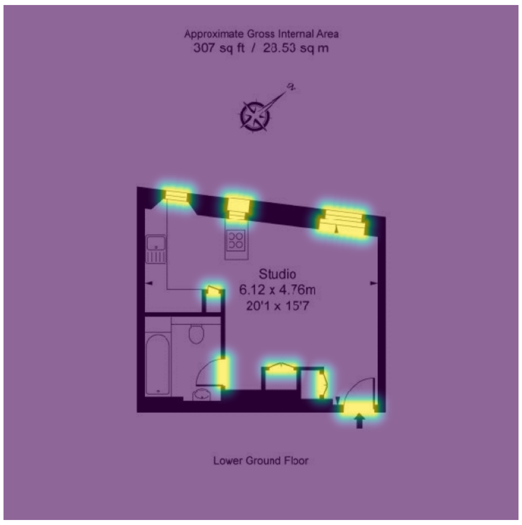

(c) Contours of connected components highlighted in red.

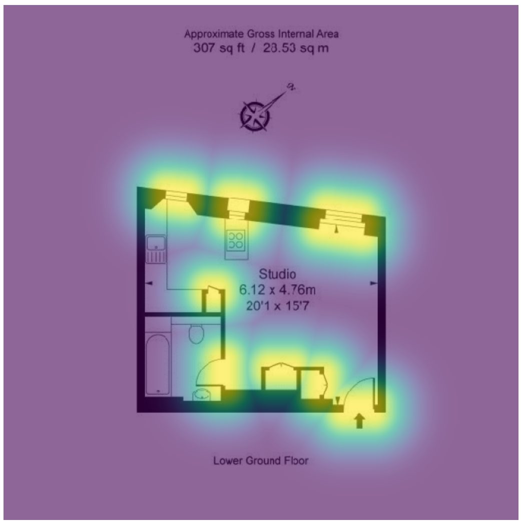
Figure 4.8: Ground-truth opening label and the first two steps to derive opening endpoints.



(a) $\beta = 5$



(b) $\beta = 10$



(c) $\beta = 40$

Figure 4.9: Three examples of generated heatmaps of the opening class for increasing values for β superimposed on the original floor plan.

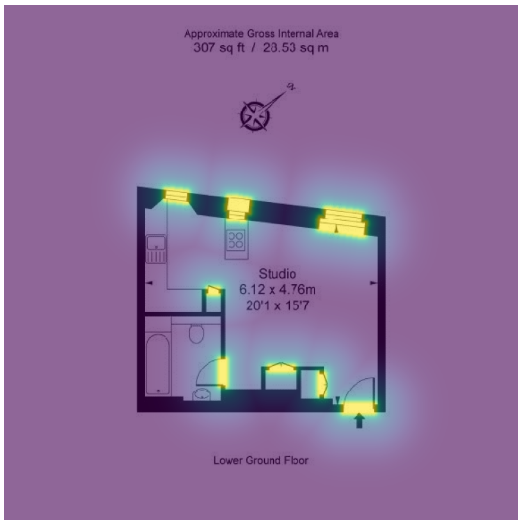


Figure 4.10: Example of average heatmap computed using $B = \{5, 10, 40\}$ superimposed on the original floor plan.

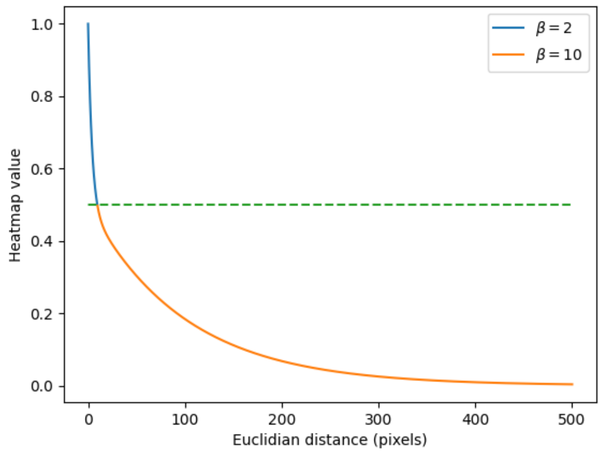
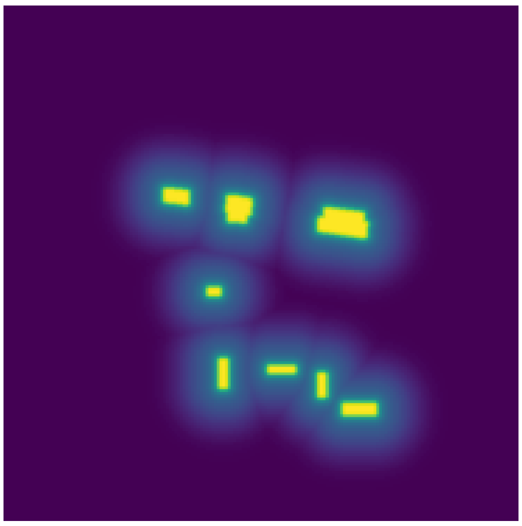
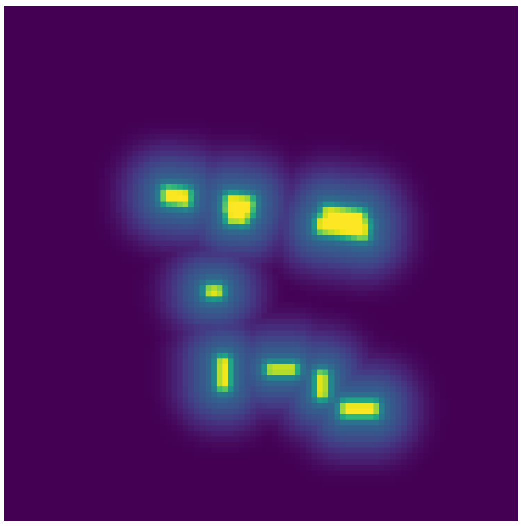


Figure 4.11: Heatmap values for increasing distances with $B = \{2, 10\}$. Term $\beta = 2$ marked in blue is responsible for the sharp decline towards 0.5 necessary to prevent contradictory learning. Term $\beta = 10$ ensures there is a slower decline after 0.5 to facilitate further learning. Heatmap value 0.5 has been marked in green for reference.


(a) $s = \frac{1}{4}$


(b) $s = \frac{1}{8}$

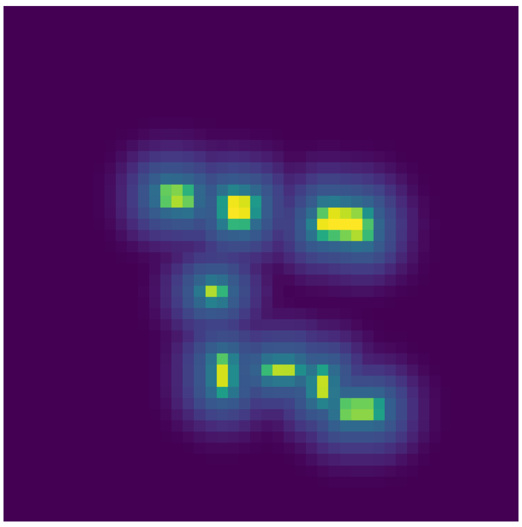

(c) $s = \frac{1}{16}$

Figure 4.12: Three examples of generated heatmaps for decreasing values of scale *s*.

Total loss

The final multi-task loss is a weighted sum of all previously defined loss functions $\mathcal{T} = \{\mathcal{L}_{\text{aUF}}, \mathcal{L}_{\text{AAF}}, \mathcal{L}_{\text{MHR}}\}$. Manually tuning the weights for the losses can be a difficult and time-consuming task. Kendall et al. showed that task weights can be modelled as homoscedastic uncertainty terms and thus learned during training by applying appropriate regularization [73]. The regularization term prevents the model from finding trivial solutions. A refined version of this approach proposed by Liebel et al. is used here [74].

$$\mathcal{L} = \sum_{\tau \in \mathcal{T}} \frac{1}{2\sigma_\tau^2} \cdot \mathcal{L}_\tau + \ln(1 + \sigma_\tau^2) \tag{4.12}$$

4.2. Reconstruction

The reconstruction step of the model is responsible for transforming the segmentation mask into a 3D representation. The reconstruction step consists of post-processing, vectorization and 3D visualization. This section will discuss all three methods. The entire reconstruction pipeline takes seconds for small floor plans, and in the order of minutes for the largest multi-unit floor plans.

4.2.1. Post-Processing

The post-processing step is used to reduce noise in the segmentation mask produced in the recognition step. The goal of post-processing is to increase the model accuracy as measured by segmentation and recognition metrics such as pixel accuracy and intersection over union. The post-processing step first transforms the segmentation mask into approximate polygons derived from the union of all semantic class masks. The polygons consist of approximated contours of the elements in the joint mask, aimed to reduce noise in the segmentation mask. The approximate polygons are then refined by merging and transforming vertices and polygons according to their distance and angle. The refined polygons are transformed back into their original class masks by dividing the polygons into smaller polygons until their semantic class can be determined with sufficient accuracy. Finally, several heuristics are applied to the post-processed mask. The heuristics resolve any potential conflicting predictions, such as glass walls neighbouring windows which do not occur in MURF. The post-processing steps are depicted in Figure 4.14. Figure 4.15 shows the post-processed segmentation mask of a multi-unit floor plan.

Zeng et al. proposed a simpler method using two types of horizontal and vertical kernels that are applied to the boundary masks to fill in pixel gaps by applying a morphological close operation [16]. Equation 4.13 shows the two types of horizontal kernels of size 5. The vertical kernels are a simple transpose of the horizontal ones. While this approach is effective at refining 'straight' boundaries (orthogonal to the kernels), the kernels fail to refine rotated boundaries. Figure 4.13 shows post-processing problems that occur when the kernels are applied to an example floor plan with rotated boundaries. For this reason, this thesis proposes a post-processing method that, while more computationally expensive, can process floor plans rotated by any angle.

$$k_5^{h1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad k_5^{h2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.13}$$

Approximate polygons

To derive the approximate polygons from the segmentation mask, one joint mask consisting of the union of all semantic classes is first created. The joint mask of an example segmentation mask is shown in Figure 4.14a and Figure 4.14b. The following steps are executed on the joint mask until no further approximation is necessary:

1. Find the initial connected components in prediction $\hat{y}$ using [71].
2. While there are components left, find the rotated rectangle bb with the minimum area enclosing the component using [75], and calculate the uncertainty of bb according to Equation 4.14
   (a) If uncertain, divide the bounding box bb into two smaller rectangles as shown in Figure 4.16, add the new components of each polygon, and repeat step 2.

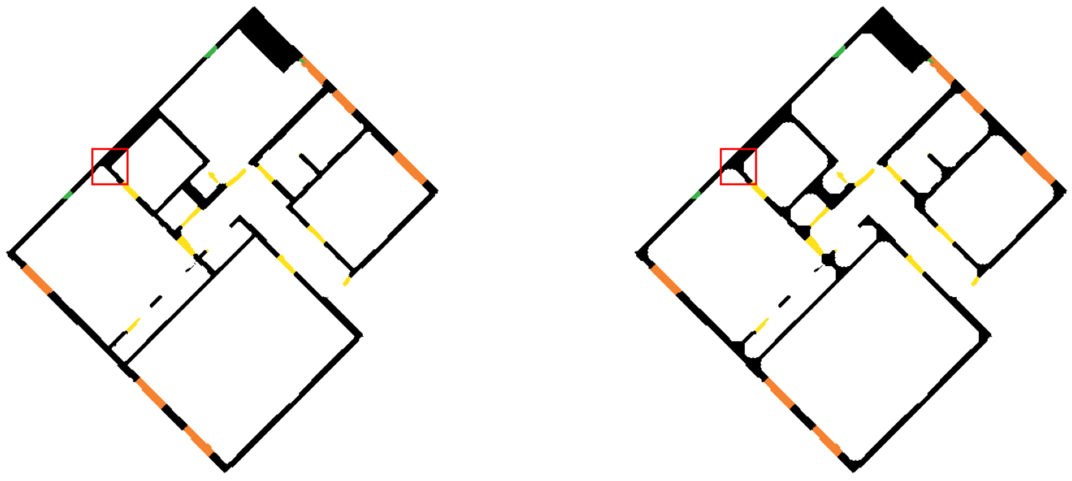

(a) Original segmentation mask **(b)** Post-processed segmentation mask

Figure 4.13: Reduced accuracy on boundary junctions when applying simple closing kernels on rotated boundaries. Example problem area marked in red.

(b) If certain, add the approximate polygon to the list of polygons.

$$\text{uncertain}(bb) = \left(\frac{1}{|bb|}\sum_{i\in bb}\hat{y}(i)\right) < \epsilon_u \tag{4.14}$$

where ϵ_u is a constant uncertainty threshold set to 0.5 determined empirically. The value was found to be high enough to produce accurate approximations, and low enough to generalize over noisy predictions.

Refined polygons

The refined polygons are obtained by reducing the approximate polygons in two steps according to two rules similar to Lv et al. [17]. The first step is to merge all vertices of which the euclidean distance is smaller then or equal to a pre-defined distance threshold ϵ_d. Vertices are continuously merged until no longer possible. The second step refines the polygons by removing vertices approximately collinear to two other vertices. A vertex is considered collinear to two other vertices if the angle between them is greater then or equal to a threshold ϵ_a. Distance and angle thresholds ϵ_d and ϵ_a have been found empirically and set to 4 and $\cos(14°)$. The two rules are executed as follows:

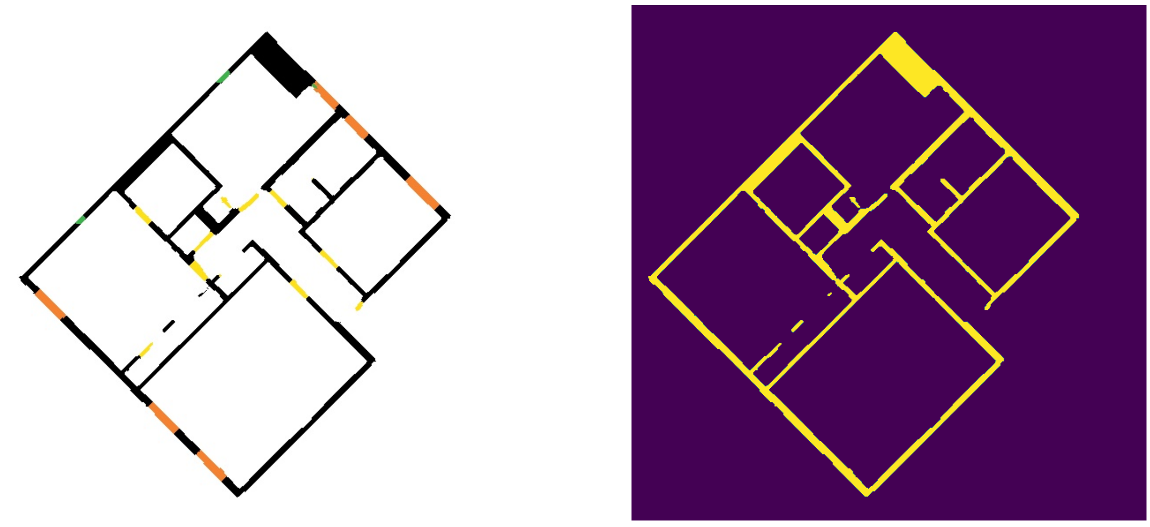
1. Repeat until there are no vertices to merge:
   (a) Construct a KD-tree of the vertices V of all polygons
   (b) For each vertex $v_i \in V$, compute the set of close vertices V_c such that for each vertex $v_j \in V_c$, it holds that either v_i and v_j are from a different polygon and $||v_i - v_j||_2 \le \epsilon_d$, or $v_i = v_j$. This prevents small polygons from being collapsed into a single vertex.
      i. Pick a vertex $v_j \in V_c$ and replace v_i and v_j with $\frac{v_i+v_j}{2}$ in the respective polygons.

   ,

2. Repeat until there are no vertices to remove:
   (a) For each distinct triplet of vertices v_i, v_j, v_k in each polygon, remove the middle vertex v_j if the angle between the vertices $|\cos(\overrightarrow{v_i v_j}, \overrightarrow{v_j v_k})| \ge \epsilon_a$

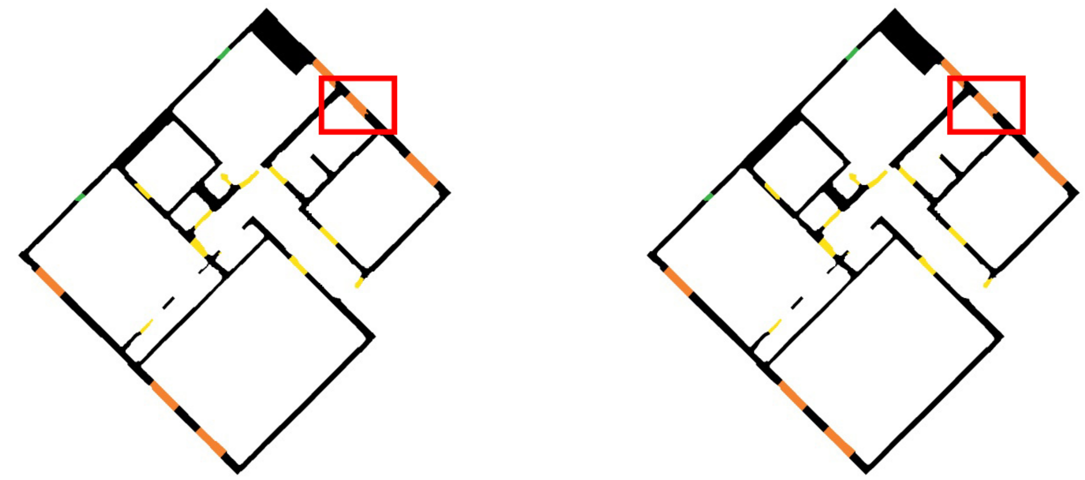
Heuristics

Although the level of noise of the refined segmentation masks is significantly lower, applying predefined heuristics can further improve accuracy. Three heuristics are used to correct the segmentation masks if necessary. The first heuristic replaces windows with glass walls if their pixels are neighbouring. This heuristic is derived from the annotated multi-unit floor plans, in which glass walls are never directly



(a) Original segmentation mask with detected walls, windows, doors and railings in black, orange, yellow and green respectively.

(b) Joint mask equal to union of all semantic class masks.



(c) Post-processed segmentation mask with approximate polygons.

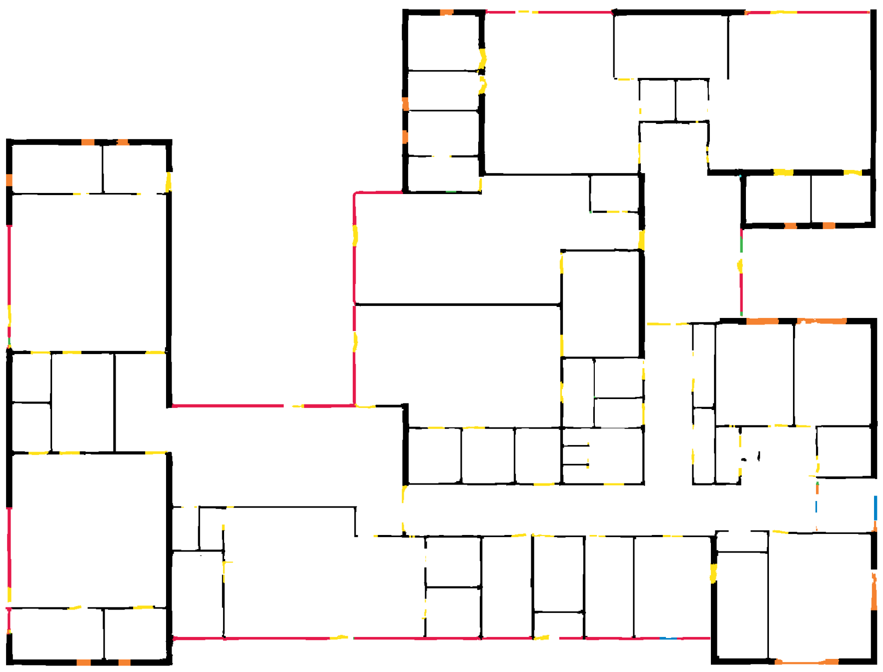
(d) Final post-processed segmentation mask after refining polygons and applying heuristics.

Figure 4.14: Post-processing steps on example segmentation mask. The differences between the final post-processed mask and approximated polygons mask is small, but noticeable in the reduced noise in walls and other straight elements. Example of refined window polygon highlighted in red.

neighbouring a window. Since glass walls and windows share the same physical properties, the real-world effect of this heuristic is small. The second heuristic is similar but resolves conflicting predictions between sliding doors and regular doors. Either sliding doors or doors are used to replace the pixels, depending on which is in the majority in the neighbourhood. The third heuristic replaces the pixels neighbouring stairs that are not background pixels with either walls or railings depending on which of the latter is already present. This heuristic is derived from the CubiCasa and MURF dataset, for which all stairs are either surrounded by background pixels, walls or railings. For all heuristics, the neighbourhood of a pixel is defined by a $k \times k$ kernel centered on the pixel. A kernel size of $k = 6$ was found to be effective. The effect of applying the heuristics on a segmentation mask is shown in Figure 4.14d.

4.2.2. Vectorization

The vectorization step is responsible for converting the post-processed segmentation mask into polygons that can be used to visualize the original floor plan. A method different from Lv et al. [17] is used since they require wall junctions for vectorization. The vectorization method is similar to the post-processing,


(a) Original segmentation mask.

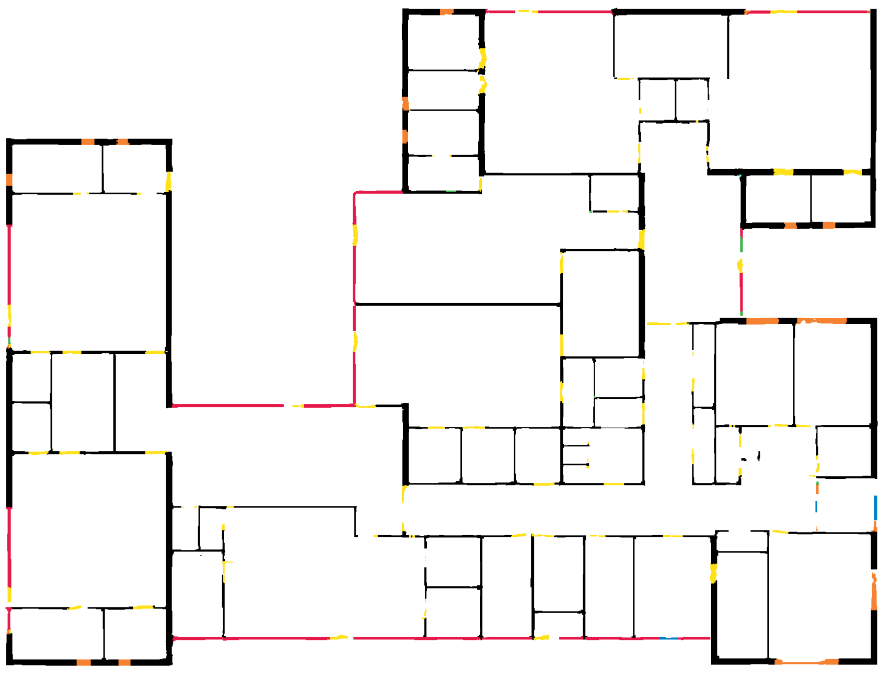
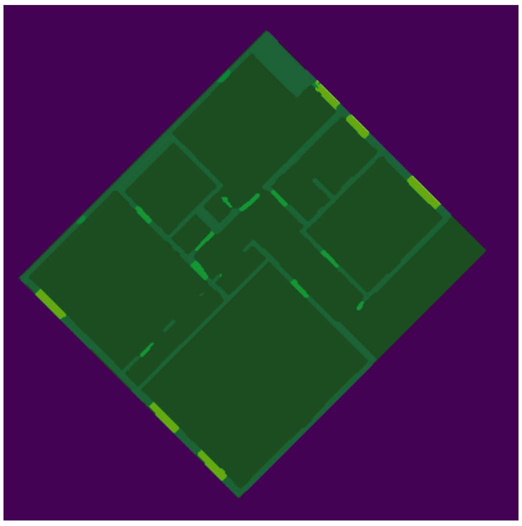

(b) Post-processed segmentation mask.

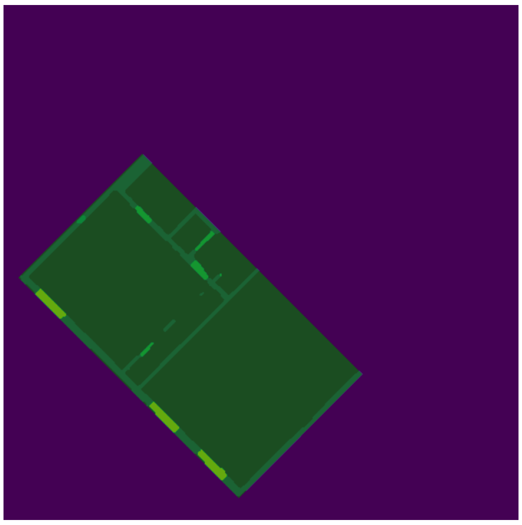
Figure 4.15: Example of post-processed segmentation mask of multi-unit floor plan.

described by the following steps:

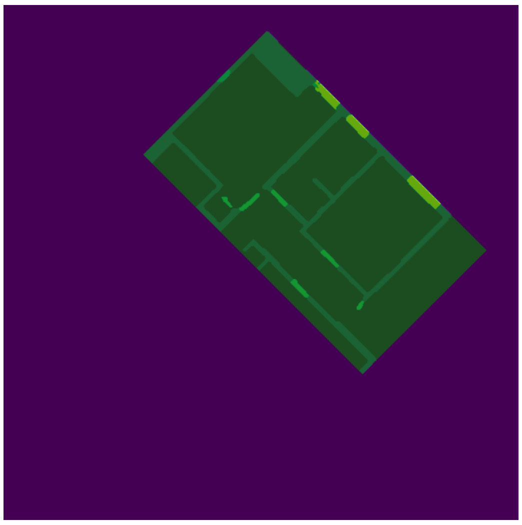
1. Find the connected components of each semantic class *c* using [71].
2. Find the contour of each component of step 1 using [72].
3. Convert the contour into a 2D polygon by compressing all horizontal, vertical and diagonal coordinates.



(a) Initial bounding rectangle.



(b) First new bounding box of first half.



(c) Second new bounding box of second half.

Figure 4.16: Example of bounding box and smaller bounding boxes generated to derive new components. The smaller bounding boxes divide the largest side of the initial bounding box in two.

4.2.3. Visualization

Using a vectorized representation of the segmentation mask, a 3D visualization can be generated. Each polygon of a segmentation mask is first transformed into a 3D object by generating the appropriate vertices and faces. A predefined height is used depending on the class. This height is set equal for each class other than the stairs category. The height of stairs is set to a higher value to make them semantically distinct from other elements in the generated 3D model. Finally, each 3D object is assigned a fixed texture derived from the semantic class. Blender [76] is used to visualize the 3D objects, as it allowed for fast visualization without user interaction.

4.3. Reflection on RQ2 and RQ3

This chapter described the recognition and reconstruction part of the architectures proposed in this thesis. The recognition part of the model is responsible for detecting the semantic elements in floor plans of arbitrary size, which is related to RQ2: what CNN architecture can effectively detect semantic information in architectural floor plans of arbitrary size in a scalable manner? The following four observations on RQ2 were made in this chapter:

- An FCN is, in comparison to a typical CNN, more effective at processing multi-unit floor plans, due to its ability to operate on input of arbitrary size.
- The performance of skip connections depends heavily on the context, and should be evaluated accordingly. Multi-scale skip connections offer a good trade-off between computational cost and improved accuracy, which will be used in this thesis.
- An attention mechanism can further improve segmentation performance by increasing the representation power of features by weighing them according to channels or spatial features.
- Compared to a single-task training objective, a multi-task training objective optimizing pixel, patch and multi-scale loss can improve segmentation performance by incorporating structural reasoning into the learning process.

The reconstruction part of the model aims to answer part of RQ3: how can produced segmentation masks best be transformed into a 3D representation? This chapter showed the following on RQ3:

- Conventional morphological operations are not sufficient to refine floor plans with rotated elements.
- A post-processing method relying on refined rotated rectangles of components in a produced segmentation mask is effective at reducing noise in predictions.
- Heuristics can further improve accuracy by resolving conflicting predictions.
- Blender can efficiently transform a vectorized segmentation mask into a 3D model.

5 Experiments

This chapter describes the experiments performed in this thesis and the floor plan datasets that are used. The datasets used in the experiments are first described, related to RQ2: the design and use of a multi-unit floor plan dataset. Afterwards a quantitative and qualitative evaluation based on five datasets is performed to assess the performance of the novel model architectures proposed in this thesis, related to RQ1 and RQ3 respectively.

5.1. Datasets

Five augmented datasets have been used in the experiments in this chapter. This section will briefly discuss each dataset and the augmentation pipeline used to expand the floor plan datasets. The datasets have been selected for their size, variety and usage in literature. Table 5.1 contains the used datasets with their respective number of images, resolution, and augmented size. Chapter 3 and Appendix A contain example floor plans of MURF and the two public datasets R3D and CubiCasa.

The first is based on R3D [26] and its extension proposed by Zeng et al. [16]. This dataset consists of mostly rectangular single-unit floor plans and only considers walls and opening classes. The dataset also contains 18 floor plans with round-shaped layouts. A total of 7 images was removed from this dataset, as incorrect padding values were used. These floor plans could otherwise confuse the model when learning to distinguish background from foreground pixels. R3D considers two semantic classes: boundaries (e.g. walls) and openings (e.g. doors).

The second dataset is CubiCasa, a refined version of the CubiCasa5K dataset [28] as described in Section 3.4.1. CubiCasa is a larger single-unit floor plan dataset with varying levels of complexity. CubiCasa considers five semantic classes: walls, railings, doors, windows and stairs.

The third dataset is the novel multi-unit floor plan dataset MURF proposed as part of this thesis. It is based on images of large buildings in the city of Rotterdam in the Netherlands. Due to their size and level of noise, the floor plans in MURF are of high complexity compared to R3D and CubiCasa. MURF considers seven semantic classes: walls, glass walls, railings, doors, sliding doors, windows and stairs.

The fourth dataset is a combination of CubiCasa and MURF. Although the list of semantic classes of CubiCasa is smaller, the classes of MURF will not be merged. This allows for a full evaluation on all semantic elements. The purpose of this combined dataset is to evaluate to what degree a model can generalize across different drawing notations and floor plan sizes.

The fifth and last dataset is MURF$_{\text{test}}$, which is a variant of MURF used solely for the qualitative evaluation. MURF$_{\text{test}}$ consists of the same floor plans in MURF but derived by using larger samples during augmentation. Evaluation on this dataset should give a better indication of the performance of a model processing multi-unit floor plans, as it poses two extra challenges. First, the floor plans are significantly larger than what the models have been trained on, and second, the boundaries of floor plans, missing in MURF due to sampling, will now have to be predicted as well. The reason for not using the MURF$_{\text{test}}$ dataset during training is that the floor plans are too large to fit into GPU memory.

Dataset	Images	Resolution	Augmented images
R3D	225	175-1104	957
CubiCasa5K	560	206-6768	7265
MURF	124	3100-28160	578
CubiCasa5K+MURF	684	206-28160	7843
MURF$_{\text{test}}$	8	3100-28160	32

Table 5.1: The five datasets used in the experiments with their respective number of floor plan images, original resolution and number of augmented images. The MURF dataset contains the number of sampled images in the images column. The MURF$_{\text{test}}$ is an unsampled variant of MURF, used only for qualitative evaluation.

5.1.1. Augmentation

Since segmentation models are not invariant to translation, rotation and scale [77], an augmentation pipeline was employed to expand the floor plan datasets. The augmentation pipeline consists of applying each combination of the following steps:

1. Optionally flip the floor plan around the x-axis.
2. Optionally rotate the floor plan randomly by $r \in [-90, 90]$ degrees.
3. Optionally apply a sampling step if the image is larger than the predefined maximum size.
4. Resize the floor plan randomly within the predefined minimum and maximum size.

For the larger floor plans, a sampling step is performed in step 3. This sampling step divides a floor plan into random smaller tiles within a minimum and maximum size so that each pixel of the original floor plan is present in the resulting set of tiles. The tiles are generated by randomly sampling their center points over a floor plan. A similar sampling approach has been applied to create CubiCasa5K and other floor plan datasets [27, 28]. Without sampling, the augmentation pipeline quadruples the number of floor plan images in a dataset. The number of tiles that are generated depend on the floor plan size.

The sampling step generates tiles from a predefined set of sizes equal to $\{512, 576, ..., 832\}$. A step size of 64 pixels for tile sizes is used to support the stride length of each model considered in the following experiments. A minimum and maximum of 512×512 and 832×832 is used because these are large enough to capture sufficient floor plan information and small enough to still fit into GPU memory. For a floor plan with width w and height h, the lower and upper bound on the number of tiles is formalized by Equation 5.1. The lower bound is equal to dividing the dimension over the largest tile size with minimum overlap. Deriving the upper bound is more complex. The upper bound assumes the worst first center position of a tile is first picked, leaving a single row and column of pixels near one corner of the floor plan uncovered. These rows, and the remaining pixels, can be covered by dividing the floor plan dimensions by half the tile size. In practice, the average number of tiles needed for a floor plan scales according to Equation 5.2. The value 448 is derived from the number of pixels the average tile $\frac{512+832}{2}$ covers with an average overlap of $672 \cdot \frac{2}{3}$. The average overlap relies on a factor $\frac{2}{3}$ due to the fact that this increases with the number of tiles already present. Appendix B contains examples of the average number of tiles for floor plans of various dimensions.

$$\left\lceil \frac{w}{832} \right\rceil \cdot \left\lceil \frac{h}{832} \right\rceil \leq \#\text{tiles} \leq \begin{cases} 1 & \text{if } \Gamma \leq 512 \\ \lceil \frac{2\Gamma}{512+1} \rceil & \text{if } \varepsilon \leq 512 < \Gamma \\ \lceil \frac{2w}{512+1} \rceil \cdot \lceil \frac{2h}{512+1} \rceil & \text{otherwise} \end{cases} \tag{5.1}$$

where $\varepsilon = \min(w, h)$ and $\Gamma = \max(w, h)$.

$$\#\text{tiles} \approx \left\lceil \frac{w}{448} \right\rceil \cdot \left\lceil \frac{h}{448} \right\rceil \tag{5.2}$$

5.2. Experimental Setup

The models proposed in this thesis are trained for a maximum of 200 epochs with a batch size of 1. Memory limits did not allow for larger batches without reducing image quality which would harm performance. Using a larger batch size would also not offer any benefits in terms of model accuracy, since group normalization works well on small batches [54]. For the two model architectures proposed in this thesis, the ADAM optimizer is used to update the model parameters with a learning rate of 1×10^{-4}. If the validation loss does not reduce for 10 epochs, the learning rate is set to half its original value until a minimum of 1×10^{-5} is reached. An early stopping criteria with a patience of 30 epochs is also used. Unless otherwise specified, all compared models have been set up identical to the manner described in their respective works. All models have been pre-trained on the ImageNet dataset [78].

All augmented floor plan datasets are split into train/validation/test sets as 0.6/0.2/0.2 which are used for all models. The training sets are used to calculate and perform the gradient updates, which are evaluated on the validation set used to determine the updates improved model performance. The test sets represent a portion of unseen data used for evaluation. In the case of MURF$_{\text{test}}$, all floor plans are part of the test set.

For both the sampling and augmentation methods, the minimum and maximum floor plan resolution of 512×512 and 832×832 is used respectively. All implementations are based on TensorFlow, and were trained on a machine with the following specifications:

- GPU: 2× NVIDIA GeForce RTX 2080 Ti GPU with 11 GB VRAM
- CPU: 2× AMD EPYC 7542 32-Core Processor
- Storage: 1 TB SSD

5.3. Quantitative Evaluation

The purpose of the quantitative evaluation is to empirically assess the performance of the proposed architectures. This section will discuss the three experiments performed and what metrics are considered.

5.3.1. Evaluation Metrics

This thesis considers three evaluation metrics to assess the models as proposed by Long et al. [79], also commonly used in related literature [16, 17, 27]. The three selected metrics are overall pixel accuracy, per-class pixel accuracy and intersection over union. When applicable, mean metrics are also considered, which are averaged over all semantic classes excluding the background class. The background class is not considered for this mean value, as it can misrepresent results due to the class imbalance. The performance metrics assess either the segmentation quality or recognition quality of the model, and are separated accordingly. For all metrics, n_{ij} is the number of pixels of class i predicted to belong to j and $t_i = \sum_j n_{ij}$ is the total number of pixels of class i. Term $n_c = |C|$, where C denotes the set of semantic classes.

Segmentation quality

The metrics assessing the segmentation quality of a model rely on region-based metrics. The first metric is overall pixel accuracy, formalized by Equation 5.3, which computes the mean accuracy of all pixels independent of the semantic classes. The background class is excluded in the average to focus on the foreground classes. The second metric in Equation 5.4 is the Intersection over Union (IoU) metric, which computes the overlap between the predicted and true pixels of a specific class. The mean IoU in Equation 5.5 is a simple average over all classes.

$$\text{Pixel Acc.} = \frac{\sum_i n_{ii}}{\sum_i t_i} \tag{5.3}$$

$$\text{Class IoU} = \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{5.4}$$

$$\text{Mean IoU} = \frac{1}{n_c} \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{5.5}$$

Recognition Quality

Class accuracy, as formalized by Equation 5.6, is used to evaluate the recognition quality of a model. The class accuracy, more commonly referred to as recall, describing the ratio of true positives over all true positive and false negatives. The mean class accuracy in Equation 5.7 is also considered, which is the mean per-class accuracy.

$$\text{Class Acc.} = \frac{n_{ii}}{t_i} \tag{5.6}$$

$$\text{Mean Acc.} = \frac{1}{n_c}\sum_i \frac{n_{ii}}{t_i} \tag{5.7}$$

5.3.2. Experiment 1: Comparing Recognition with DFPR and CubiCasa5K

The purpose of experiment 1 is to evaluate the performance of the recognition and post-processing part of the proposed architectures. This evaluation is based on a comparative analysis using the datasets described and two floor plan processing models from related work. From all relevant state-of-the-art floor plan processing methods [1, 15, 16, 17, 27, 28, 37], only Zeng et al. [16] and Kalervo et al. [28] have published the details of their model and are thus included in this experiment. The two models, named DFPR and CubiCasa5K respectively, have been modified for a fair comparison, removing fixed input size requirements and room predictions. Room predictions have been removed because these would otherwise only reduce the accuracy of boundary and opening classes. A subscript $\mathcal{B}$ is used for the modified models. For CubiCasa5K$_{\mathcal{B}}$, the balanced entropy loss function of Zeng et al. [16] is adopted, as the original categorical cross entropy loss failed to converge. It should be noted that the CubiCasa5K model is an improved version of Liu et al. [1], so this model is also evaluated indirectly.

The proposed model architectures aim to improve the state-of-the-art on both single-unit and multi-unit floor plans. To determine whether this is the case, the described datasets will be evaluated separately. The following tables contain the results on R3D, CubiCasa, MURF and the combined dataset respectively. The values between brackets represent post-processed results. For fairness, the heuristics were also applied to the DFPR$_{\mathcal{B}}$ and CubiCasa5K$_{\mathcal{B}}$ models. The next sections discuss the results on each dataset and the general observations that can be made.

R3D

Table 5.2 and Table 5.3 contain the accuracy and IoU results on R3D. The results show that the modified variant DFPR$_{\mathcal{B}}$ performs significantly worse than originally reported. This is likely due to the fact that the DFPR$_{\mathcal{B}}$ model has been trained and evaluated on an augmented dataset, consisting of more difficult floor plans. The floor plans in the augmented dataset are more difficult due to, for instance, rotated elements that did not occur before augmentation. Interestingly, the CubiCasa5K$_{\mathcal{B}}$ model performs worse than the DFPR$_{\mathcal{B}}$ model, despite being designed to predict a larger range of more complex semantic elements. One possible reason for this is that the CubiCasa5K$_{\mathcal{B}}$ model was trained with a lower batch size than recommended in the original work due to limited GPU memory. The limited batch size does, however, not fully explain the significant performance difference compared to the original work. A second reason could be related to the architecture of CubiCasa5K, which is designed to process floor plans of a fixed size, failing at predicting multiple sizes of floor plans. The proposed architectures perform best considering accuracy and IoU, with the fully connected model architecture 2 achieving the best performance out of both.

Compared to DFPR$_{\mathcal{B}}$ and CubiCasa5K$_{\mathcal{B}}$, the post-processed values of the proposed architectures show a significant increase in accuracy, but a significant decrease in the intersection over union. It is also interesting to note that architecture 1 achieves a higher post-processed accuracy than architecture 2, although the base accuracy of the former is lower.

		DFPR	DFPR$_{\mathcal{B}}$	CubiCasa5K$_{\mathcal{B}}$	Architecture 1	Architecture 2
Pixel Acc.		0.89 (0.90)	0.84 (0.85)	0.80 (0.83)	0.92 (**0.94**)	**0.93** (0.94)
	Wall	0.98 (0.98)	0.87 (0.88)	0.85 (0.88)	0.95 (0.96)	0.95 (**0.96**)
	Door-and-window	0.83 (0.83)	0.73 (0.73)	0.60 (0.60)	0.87 (**0.89**)	0.87 (0.88)
	Closet	0.61 (0.54)	-	-	-	-
Class	Bathroom & *etc.*	0.81 (0.78)	-	-	-	-
Acc.	Living room & *etc.*	0.87 (0.93)	-	-	-	-
	Bedroom	0.75 (0.79)	-	-	-	-
	Hall	0.59 (0.68)	-	-	-	-
	Balcony	0.44 (0.49)	-	-	-	-
Mean		0.63 (0.66)	0.80 (0.80)	0.73 (0.74)	0.90 (**0.92**)	0.91 (0.92)

Table 5.2: Accuracy on R3D compared to DFPR [16] and CubiCasa5K [28]. The first column is copied from Zeng et al. for reference [16]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

		DFPR	DFPR$_{\mathcal{B}}$	CubiCasa5K$_{\mathcal{B}}$	Architecture 1	Architecture 2
	Wall	0.98 (0.98)	0.83 (0.82)	0.82 (0.82)	0.90 (0.83)	**0.90** (0.83)
	Door-and-window	0.83 (0.83)	0.65 (0.65)	0.55 (0.55)	0.76 (0.71)	**0.77** (0.72)
	Closet	0.61 (0.54)	-	-	-	-
Class	Bathroom & *etc.*	0.81 (0.78)	-	-	-	-
IoU	Living room & *etc.*	0.87 (0.93)	-	-	-	-
	Bedroom	0.75 (0.79)	-	-	-	-
	Hall	0.59 (0.68)	-	-	-	-
	Balcony	0.44 (0.49)	-	-	-	-
Mean		0.63 (0.66)	0.74 (0.74)	0.68 (0.69)	0.83 (0.77)	**0.84** (0.78)

Table 5.3: IoU on R3D compared to DFPR [16] and CubiCasa5K [28]. The first column is copied from Zeng et al. for reference [16]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

CubiCasa

Table 5.4 and Table 5.5 contain the accuracy and IoU results on CubiCasa. The results on CubiCasa show a similar trend compared to R3D. The DFPR$_{\mathcal{B}}$ model outperforms CubiCasa5K$_{\mathcal{B}}$ significantly, especially on stairs. The difference between the accuracy of DFPR$_{\mathcal{B}}$ and the two proposed architectures is more notable on CubiCasa, proving to be a more difficult dataset to learn features from. The proposed architectures achieve a considerably higher per class accuracy and overall pixel accuracy. The background class is the exception, on which the CubiCasa5K$_{\mathcal{B}}$ and DFPR$_{\mathcal{B}}$ models achieve a higher accuracy. This is likely due to wrong predictions of the baseline models of other semantic classes, indirectly increasing the background accuracy. The proposed architectures also improves the IoU compared to the baselines, but to a lesser extent. There is notable difference between architecture 1 and architecture 2 on CubiCasa. Architecture 1 achieves the highest IoU, while architecture 2 achieves the best accuracy. For all models, the railing class is most difficult to predict. This is likely due to their semantic similarity to walls and windows.

The post-processing results on CubiCasa are similar to the results on R3D. The accuracy of the proposed architectures is increased significantly, while the IoU is decreased. The exception is the railing class, which decreases in accuracy. This is likely due to incorrect application of the heuristic that replaces railings with walls when deemed appropriate. Architecture 2 achieves the best overall post-processed performance.

		CubiCasa5K	DFPR$_{\mathcal{B}}$	CubiCasa5K$_{\mathcal{B}}$	Architecture 1	Architecture 2
Pixel Acc.		0.898 (0.886)*	0.71 (0.72)	0.48 (0.50)	0.86 (0.89)	0.87 (**0.89**)
	Wall	0.858 (0.720)	0.76 (0.78)	0.57 (0.61)	0.90 (**0.92**)	0.91 (0.92)
	Railing	0.287 (0.116)	0.36 (0.36)	0.02 (0.01)	0.62 (0.61)	**0.68** (0.66)
	Door	0.598 (0.598)	0.58 (0.58)	0.20 (0.20)	0.71 (0.75)	0.75 (**0.78**)
	Window	0.737 (0.661)	0.66 (0.66)	0.40 (0.40)	0.86 (0.86)	0.85 (**0.86**)
	Stairs	-	0.65 (0.65)	0.18 (0.18)	0.73 (0.79)	0.78 (**0.81**)
	Background*	0.936 (0.932)	0.998 (0.997)	**0.998** (0.998)	0.98 (0.98)	0.98 (0.98)
	Bedroom	0.862 (0.859)	-	-	-	
	Bath	0.734 (0.726)	-	-	-	
	Bathtub	0.301 (0.220)	-	-	-	
	Chimney	0.117 (0.089)	-	-	-	
	Closet	0.776 (0.739)	-	-	-	
Class	Electr. Appl	0.757 (0.709)	-	-	-	
Acc.	Empty	0.993 (0.991)	-	-	-	
	Fire Place	0.404 (0.240)	-	-	-	
	Garage	0.472 (0.481)	-	-	-	
	Hallway	0.712 (0.706)	-	-	-	
	Kitchen	0.799 (0.770)	-	-	-	
	Living Room	0.826 (0.838)	-	-	-	
	Other rooms	0.571 (0.575)	-	-	-	
	Outdoor	0.777 (0.741)	-	-	-	
	Sauna bench	0.742 (0.603)	-	-	-	
	Sink	0.661 (0.598)	-	-	-	
	Storage	0.539 (0.528)	-	-	-	
	Toilet	0.684 (0.618)	-	-	-	
Mean		0.658 (0.611)	0.60 (0.60)	0.27 (0.28)	0.76 (0.80)	0.80 (**0.81**)

Table 5.4: Accuracy on CubiCasa compared to DFPR [16] and CubiCasa5K [28]. The first column is copied from Kalervo et al. for reference [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. *This value includes the background class and can thus not directly be compared to the others. Highest value marked in bold.

		CubiCasa5K	DFPR$_{\mathcal{B}}$	CubiCasa5K$_{\mathcal{B}}$	Architecture 1	Architecture 2
	Wall	0.730 (0.614)	0.72 (0.73)	0.55 (0.58)	**0.79** (0.74)	0.78 (0.74)
	Railing	0.236 (0.105)	0.33 (0.33)	0.02 (0.01)	**0.47** (0.44)	0.45 (0.42)
	Door	0.536 (0.486)	0.53 (0.53)	0.19 (0.18)	**0.57** (0.53)	0.56 (0.53)
	Window	0.668 (0.549)	0.63 (0.62)	0.32 (0.33)	0.73 (0.70)	**0.73** (0.70)
	Stairs	-	0.61 (0.61)	0.14 (0.15)	**0.62** (0.60)	0.61 (0.59)
	Background	0.873 (0.855)	0.98 (**0.99**)	0.97 (0.97)	0.98 (0.98)	0.98 (0.98)
	Bedroom	0.742 (0.733)	-	-	-	
	Bath	0.606 (0.596)	-	-	-	
	Bathtub	0.267 (0.205)	-	-	-	
	Chimney	0.112 (0.086)	-	-	-	
	Closet	0.692 (0.658)	-	-	-	
Class	Electr. Appl	0.660 (0.608)	-	-	-	
IoU	Empty	0.976 (0.970)	-	-	-	
	Fire Place	0.362 (0.224)	-	-	-	
	Garage	0.337 (0.335)	-	-	-	
	Hallway	0.556 (0.553)	-	-	-	
	Kitchen	0.650 (0.625)	-	-	-	
	Living Room	0.666 (0.664)	-	-	-	
	Other rooms	0.414 (0.415)	-	-	-	
	Outdoor	0.664 (0.566)	-	-	-	
	Sauna bench	0.673 (0.553)	-	-	-	
	Sink	0.557 (0.490)	-	-	-	
	Storage	0.448 (0.439)	-	-	-	
	Toilet	0.628 (0.566)	-	-	-	
Mean		0.566 (0.518)	0.56 (0.56)	0.25 (0.25)	**0.64** (0.59)	0.63 (0.58)

Table 5.5: IoU on CubiCasa compared to DFPR [16] and CubiCasa5K [28]. The first column is copied from Kalervo et al. for reference [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

MURF

Table 5.6 and Table 5.7 contain the accuracy and IoU results on MURF. The results show that the CubiCasa5K$_{\mathcal{B}}$ starts to fail on certain semantic classes. The accuracy and IoU of every class except walls and stairs is very low, reaching values close to zero for the railing and door classes. This is likely due to the increased variance in floor plan scales present in the MURF dataset. In contrast, the DFPR$_{\mathcal{B}}$ model still manages to achieve similar performance to R3D and CubiCasa proving the generalize quite well across different floor plan datasets. The accuracy on doors is lower for the DFPR$_{\mathcal{B}}$ model on MURF compared to CubiCasa, probably due to the additional sliding doors class. Similar to CubiCasa, the two proposed architectures achieve a significantly higher accuracy and IoU compared to the baseline architectures. This is especially the case for the more difficult classes, including glass walls, railings and sliding doors. The difference between the regularly connected and fully connected architecture is small, with the latter achieving the best performance on both the accuracy and IoU.

Post-processing increases the accuracy, but decreases the IoU for the proposed architectures. Similar to on R3D, a lower base accuracy sometimes results in a higher post-processed accuracy, which is the case for the wall class of architecture 1. The post-processed results for the baseline models illustrate that small base values cause it to fail. For instance, the low base accuracy of the stairs class of CubiCasa5K$_{\mathcal{B}}$ is reduced further by post-processing. Overall, model architecture 2 performs the best.

		DFPR$_{\mathcal{B}}$	CubiCasa5K$_{\mathcal{B}}$	Architecture 1	Architecture 2
Pixel Acc.		0.71 (0.73)	0.34 (0.39)	0.90 (0.91)	0.92 (**0.93**)
	Wall	0.74 (0.77)	0.51 (0.58)	0.94 (**0.95**)	0.95 (0.95)
	Glass wall	0.66 (0.70)	0.03 (0.14)	0.85 (0.85)	0.86 (**0.86**)
	Railing	0.59 (0.59)	< 0.01 (< 0.01)	0.86 (0.81)	**0.88** (0.82)
Class Acc.	Door	0.44 (0.43)	< 0.01 (< 0.01)	0.79 (0.81)	0.80 (**0.82**)
	Sliding door	0.42 (0.41)	0.01 (< 0.01)	0.83 (0.85)	0.84 (**0.86**)
	Window	0.74 (0.73)	0.05 (0.01)	0.88 (0.89)	0.90 (**0.91**)
	Stairs	0.80 (0.80)	0.11 (0.01)	0.84 (0.86)	0.87 (**0.88**)
Mean		0.63 (0.63)	0.10 (0.11)	0.85 (0.86)	0.87 (**0.88**)

Table 5.6: Accuracy on MURF compared to DFPR [16] and CubiCasa5K [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

		DFPR$_{\mathcal{B}}$	CubiCasa5K$_{\mathcal{B}}$	Architecture 1	Architecture 2
	Wall	0.72 (0.74)	0.48 (0.54)	0.87 (0.81)	**0.87** (0.81)
	Glass wall	0.63 (0.66)	0.02 (0.10)	0.80 (0.76)	**0.80** (0.76)
	Railing	0.52 (0.52)	< 0.01 (< 0.01)	0.75 (0.66)	**0.79** (0.68)
Class IoU	Door	0.40 (40)	< 0.01 (< 0.01)	0.65 (0.61)	**0.67** (0.62)
	Sliding door	0.34 (0.35)	< 0.01 (< 0.01)	0.67 (0.61)	**0.75** (0.70)
	Window	0.66 (0.66)	0.05 (0.01)	0.80 (0.76)	**0.81** (0.77)
	Stairs	0.77 (0.77)	0.09 (0.01)	0.80 (0.78)	**0.83** (0.81)
Mean		0.58 (0.58)	0.14 (0.10)	0.76 (0.71)	**0.79** (0.74)

Table 5.7: IoU on MURF compared to DFPR [16] and CubiCasa5K [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

Combined dataset

Table 5.8 and Table 5.9 contain the accuracy and IoU results on the combined dataset. The results on the combined dataset hint that it is a more difficult dataset to generalize on compared to the previous datasets. The increased class imbalance introduced by combining the datasets is one factor making learning more difficult. The class imbalance is increased because of two reasons: (1) the CubiCasa dataset simply contains more floor plans compared to MURF, and (2) the CubiCasa dataset considers a smaller set of semantic classes than MURF. Combining the two datasets results in, for instance, sliding doors making up a smaller portion of the dataset pixels compared to MURF. Since sliding doors are already underrepresented in MURF, the combined dataset further increases this class imbalance. It is interesting to note that both DFPR$_{\mathcal{B}}$ and CubiCasa5K$_{\mathcal{B}}$ rely on a weighted cross entropy loss function which aims to balance the classes. The results show that this loss function is not sufficient to deal with the class imbalance of the combined dataset. Both DFPR$_{\mathcal{B}}$ and CubiCasa5K$_{\mathcal{B}}$ fail at predicting glass walls and sliding doors, and can no longer be used reliably. Whereas the baseline models fail to generalize, the proposed architectures achieve promising results. There is a notably decreased accuracy and IoU on the glass walls and railings, likely due to missing annotations for glass walls that might be present in the CubiCasa floor plans. The model architecture 2 outperforms architecture 1 by similar margins compared to the previous datasets.

Similar to on CubiCasa, the post-processing method of the baseline models further reduces the accuracy of semantic classes with a low base accuracy. The proposed post-processing method increases the accuracy, but reduces the IoU for the combined dataset. The increase and decrease in accuracy and IoU is similar to the individual MURF and CubiCasa datasets. With post-processing, model architecture 2 still performs the best out of all the models.

		DFPR$_{\mathcal{B}}$	CubiCasa5K$_{\mathcal{B}}$	Architecture 1	Architecture 2
Pixel Acc.		0.53 (0.54)	0.33 (0.36)	0.84 (0.85)	0.85 (**0.86**)
	Wall	0.59 (0.61)	0.47 (0.52)	0.89 (**0.90**)	0.90 (0.90)
	Glass wall	<0.01 (<0.01)	<0.01 (<0.01)	0.51 (0.52)	0.54 (**0.54**)
	Railing	0.04 (0.03)	<0.01 (<0.01)	0.59 (0.57)	**0.62** (0.60)
Class Acc.	Door	0.28 (0.28)	0.02 (0.01)	0.71 (**0.73**)	0.69 (0.72)
	Sliding door	0 (0)	0 (0)	0.34 (0.36)	0.60 (**0.61**)
	Window	0.55 (0.54)	0.03 (0.01)	0.83 (**0.84**)	0.82 (0.83)
	Stairs	0.45 (0.45)	0.02 (0.02)	0.72 (0.74)	0.73 (**0.74**)
Mean		0.27 (0.27)	0.08 (0.08)	0.66 (0.69)	0.70 (**0.71**)

Table 5.8: Accuracy on CubiCasa+MURF compared to DFPR [16] and CubiCasa5K [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

		DFPR$_{\mathcal{B}}$	CubiCasa5K$_{\mathcal{B}}$	Architecture 1	Architecture 2
	Wall	0.57 (0.59)	0.45 (0.49)	0.78 (0.73)	**0.79** (0.73)
	Glass wall	<0.01 (<0.01)	<0.01 (<0.01)	**0.44** (0.41)	0.43 (0.40)
	Railing	0.04 (0.03)	<0.01 (<0.01)	0.44 (0.41)	**0.44** (0.41)
Class IoU	Door	0.27 (0.27)	0.02 (0.01)	0.54 (0.51)	**0.56** (0.52)
	Sliding door	0 (0)	0 (0)	0.21 (0.18)	**0.24** (0.21)
	Window	0.49 (0.49)	0.03 (0.01)	0.71 (0.69)	**0.71** (0.69)
	Stairs	0.43 (0.43)	0.02 (0.01)	0.59 (0.57)	**0.62** (0.60)
Mean		0.26 (0.26)	0.07 (0.08)	0.53 (0.50)	**0.54** (0.50)

Table 5.9: IoU on CubiCasa+MURF compared to DFPR [16] and CubiCasa5K [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

General observations

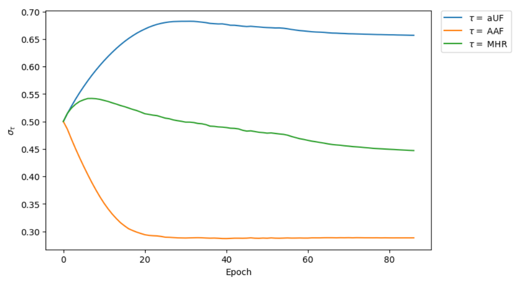
The results on the four datasets have shown that, regarding the baselines, the DFPR$_{\mathcal{B}}$ consistently outperforms the CubiCasa5K$_{\mathcal{B}}$ model, proving to generalize better across more complex and various scaled floor plan features. Generally, the baseline models also perform worse than originally reported. This is likely due to the augmented datasets, which are more difficult to learn from. It is also important to note that the DFPR model [16] has not been evaluated on a separate test set consisting of unseen data. The results of evaluating DFPR$_{\mathcal{B}}$ on a test set are thus expected to be worse. Compared to the baselines the proposed architectures achieve a considerably higher accuracy and IoU for all datasets. Although by a small margin, the fully connected model architecture 2 performs better than the regularly connected architecture 1.

The proposed post-processing method generally increases the accuracy significantly, but decreases the intersection over union. The decreased IoU is likely due to the width of semantic elements the post-processing method increases to deal with noise. Morphological erosion has been applied to mitigate this, but another method could prove more effective. Different values for the distance and angle threshold could also reduce the increased width in the segmentation masks introduced by post-processing.

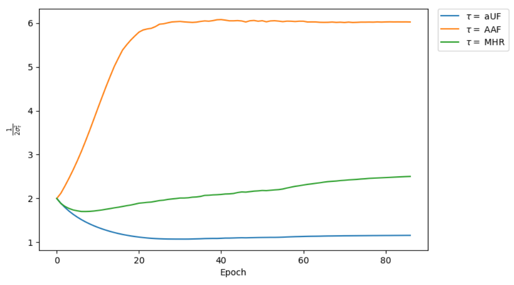
Loss terms

The training objective for the model architectures is to minimize a multi-task loss function, for which the task weights are learned through training. This section presents a brief analysis of how task weights change in order to gain a better understanding of how tasks are used during training. The results from model architecture 2 on the combined dataset are discussed, which follow the same trend of model architecture 1.

The task weights σ_τ balance the unified focal loss, adaptive affinity field loss and multi-scale heatmap regression loss. Figure 5.1 shows how the values of the task weights develop across epochs, with Figure 5.1b showing the scaling parameter that is used during training. The results show that the lowest loss is achieved by reducing the weight of the unified focal loss, while increasing that of the affinity field and heatmap regression loss. The increase of the affinity field loss weight is most apparent, showing that it is useful during training.



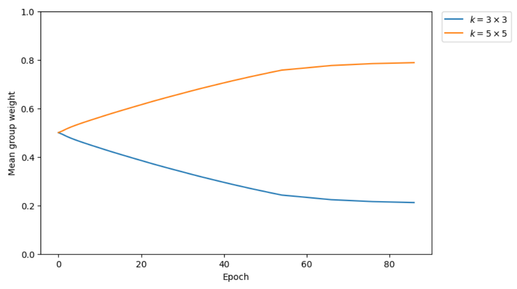
(a) Task weight σ_τ across epochs.



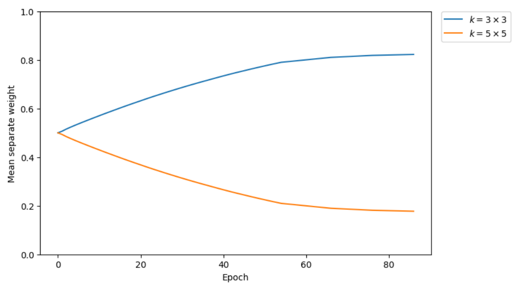
(b) Task weights $\frac{1}{2\sigma_\tau^2}$ across epochs.

Figure 5.1: Task weights σ_τ across epochs, $\frac{1}{2\sigma_\tau^2}$ is shown for reference as this value is used to scale the loss tasks.

In addition to the task weights, the adaptive affinity field loss also learns grouping and separating weights per kernel and class. Figure 5.2 shows the value of the weights across epochs. The results show that the largest kernel size is most effective to group similar pixels, while the smallest kernel is more useful to separate different pixels.



(a) Mean group weight $\sum_c^C w_k^{\text{group}}$ across epochs.



(b) Mean separate weight $\sum_c^C w_k^{\text{separate}}$ across epochs.

Figure 5.2: Grouping and separating weights applied to focus on important areas for calculating the affinity field loss.

5.3.3. Experiment 2: Comparing Recognition with State-of-the-Art

The purpose of experiment 2 is to evaluate how general state-of-the-art segmentation models perform compared to the model architectures proposed in this thesis. The reason for comparing general segmentation models is that these models improve much faster over time compared to floor plan processing methods, which allows for a more fair comparison of models regarding their size and complexity. Namely, one reason for the improved performance of the model architectures compared to DFPR and CubiCasa5K presented in experiment 1 is due to the increased model complexity. Considering segmentation models that have shown very promising results in other fields, such as medical image segmentation, and are of similar size and complexity thus allow for a better understanding of how the model architectures perform.

The two general segmentation models that are considered are U-Net++ [39] and U-Net3+ [40]. Both have shown promising results in medical image segmentation and have source code that is publicly accessible. U-Net [38] is also included in the evaluation as a baseline for the segmentation models. Each model has been modified to be similar to the proposed architectures. The same backbone, feature map sizes and loss function are used used for all models. This sections discusses the aggregated results on each dataset: R3D, CubiCasa, MURF and the combined dataset. The metrics include the pixel accuracy, mean class accuracy and mean IoU, as these are sufficient to evaluate the performance of the proposed architectures. Only the boundary and opening classes considered by the proposed architectures are compared. Post-processed results are not included, as the method is identical for all models. Appendix C contains the full results of experiment 2.

Results

Table 5.10 contains the aggregated results for the four datasets. The results show that the fully connected model architecture 2 outperforms U-Net3+. The exception is the IoU, on which U-Net3+ achieves a higher value for CubiCasa and the combined dataset. The regularly connected architecture performs worse than U-Net3+, likely due to the lower model complexity not being able to capture sufficient information. Compared to experiment 1, the differences between the considered models are smaller. This is expected, because an accuracy of 100% can never be reached which causes the best models to achieve similar performance. It is also interesting to note that U-Net, the baseline model, performs much better than might be expected due to its low complexity compared to the other models. U-Net outperforms U-Net++ on every dataset, although the former is a more complex model. This confirms the hypothesis that more skip-connections do not necessarily improve performance, and that the use of skip-connections are context dependent.

		U-Net	U-Net++	U-Net3+	Architecture 1	Architecture 2
	Pixel Acc.	0.92	0.91	0.93	0.92	**0.93**
R3D	Mean Class Acc.	0.90	0.89	0.90	0.90	**0.91**
	Mean Class IoU.	0.83	0.82	0.83	0.83	**0.84**
	Pixel Acc.	0.86	0.82	0.86	0.86	**0.87**
CubiCasa	Mean Class Acc.	0.78	0.76	0.77	0.78	**0.80**
	Mean Class IoU.	0.64	0.58	**0.64**	0.61	0.63
	Pixel Acc.	0.91	0.83	0.91	0.90	**0.92**
MURF	Mean Class Acc.	0.85	0.71	0.86	0.85	**0.87**
	Mean Class IoU.	0.78	0.63	0.77	0.76	**0.79**
	Pixel Acc.	0.83	0.80	0.85	0.84	**0.85**
Combined	Mean Class Acc.	0.65	0.56	0.69	0.66	**0.70**
	Mean Class IoU.	0.52	0.44	**0.55**	0.48	0.54

Table 5.10: Overall accuracy, mean class accuracy and mean IoU of the proposed architectures and compared segmentation models on four floor plan datasets. Highest value marked in bold.

5.3.4. Experiment 3: Deep Supervision

Deep supervision has been shown to speed up training and improve segmentation performance. The purpose of experiment 3 is to assess whether a deeply supervised variant of the best performing proposed architecture further improves performance. This experiment will only evaluate the combined dataset, as there is sufficient room for improvement compared to the regular proposed architectures. Evaluating a deeply supervised variant on the other datasets will likely not result in improved performance, because the proposed models already perform close to optimal.

Results

Table 5.11 contains the aggregated results of architecture 2 and its deeply supervised variant on the combined dataset. The results show that the deeply supervised variant achieves a higher mean class accuracy, but lower overall pixel accuracy and mean class IoU. Since the increase in mean class accuracy is small, compared to the decrease in IoU, the proposed deep supervision method is not effective. Other training or inference strategies could prove more effective.

	Architecture 2	Architecture 2 DS
Pixel Acc.	**0.85**	0.85
Mean Class Acc.	0.70	**0.71**
Mean Class IoU.	**0.54**	0.52

Table 5.11: Overall accuracy, mean class accuracy and mean IoU of architecture 2 compared to its deeply supervised (abbreviated by DS) variant on the combined dataset.

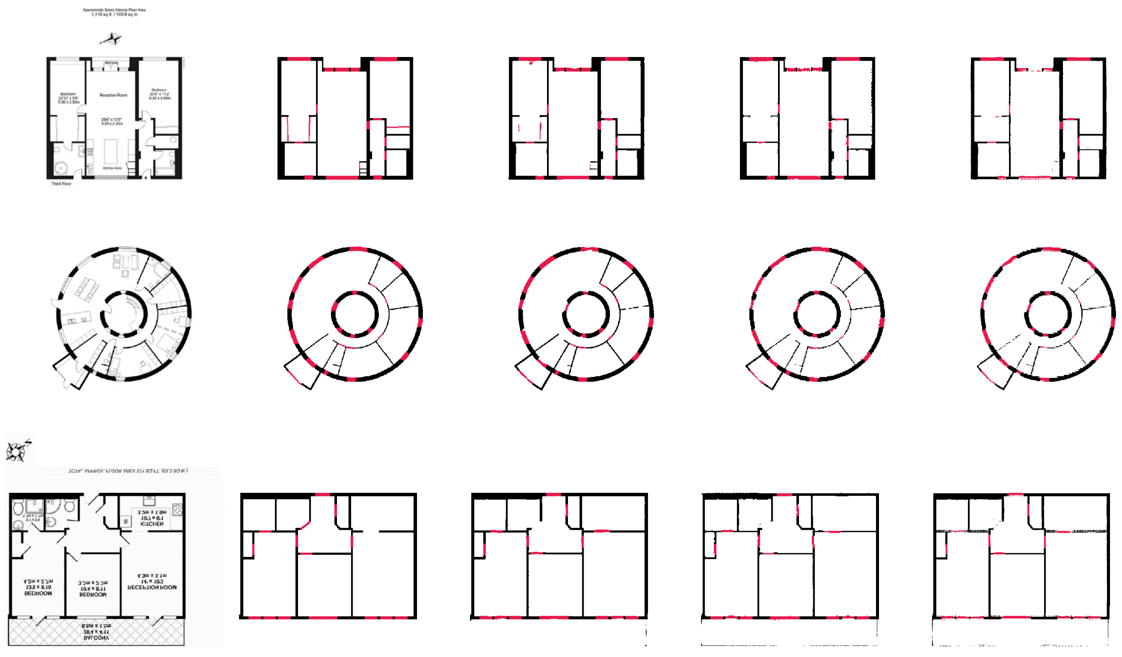
5.4. Qualitative Evaluation

The goal of the qualitative evaluation is to get a deeper understanding of the recognition and reconstruction results. This section presents the results of the best performing recognition architecture on the datasets considered, and an additional case study on a large hospital building in Rotterdam.

5.4.1. Floor Plan Datasets

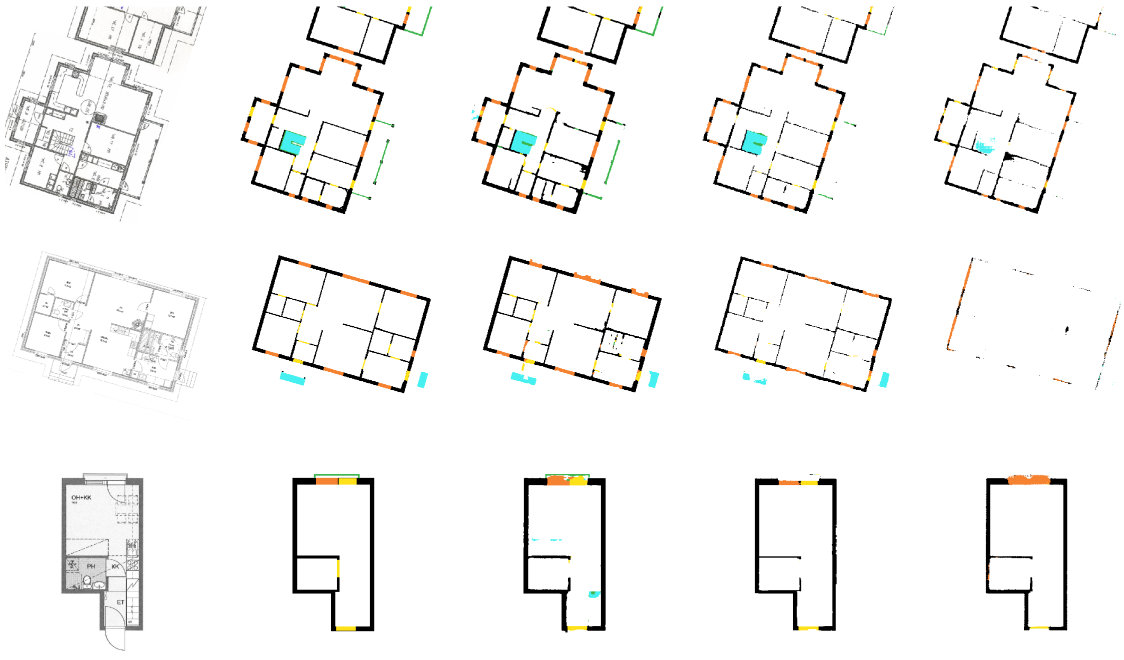
Figure 5.3, Figure 5.4 and Figure 5.5 shows the recognition results for three example images from R3D, CubiCasa and MURF respectively. The results show that the proposed architecture 2 is much more robust to the noise introduced by augmenting the floor plans in the dataset. Especially the CubiCasa5K$_{\mathcal{B}}$ fails to identify sufficient semantic information in the floor plans. Compared to DFPR$_{\mathcal{B}}$, the proposed architecture produces segmentation masks of higher quality. Although the proposed architecture produces more accurate masks, there are still some mispredicted stairs in the masks. The difference in accuracy of the segmentation masks increases for the more complex datasets such as MURF.

In addition to the original segmentation mask of proposed architecture 2, the post-processed segmentation mask, abbreviated by ‘pp’, is also examined by generating its 3D model. Figure 5.6, Figure 5.7 and Figure 5.8 show the 3D models of three post-processed segmentation masks from each of the datasets. Appendix D contains the full-sized 3D models. The results show that the reconstruction method is effective at reducing noise so an accurate 3D model can be generated. Figure 5.9 contains an example of a full-sized 3D models generated from MURF$_{\text{test}}$ by proposed architecture 2. The results from MURF$_{\text{test}}$ show that the features captured by the proposed architecture are sufficient to infer a full-sized multi-unit floor plan.



(a) Input **(b)** Ground truth **(c)** Arch. 2 **(d)** DFPR$_{\mathcal{B}}$ **(e)** CubiCasa5K$_{\mathcal{B}}$

Figure 5.3: Qualitative comparison of segmentation masks from R3D.



(a) Input **(b)** Ground truth **(c)** Arch. 2 **(d)** DFPR$_{\mathcal{B}}$ **(e)** CubiCasa5K$_{\mathcal{B}}$

Figure 5.4: Qualitative comparison of segmentation masks from CubiCasa.

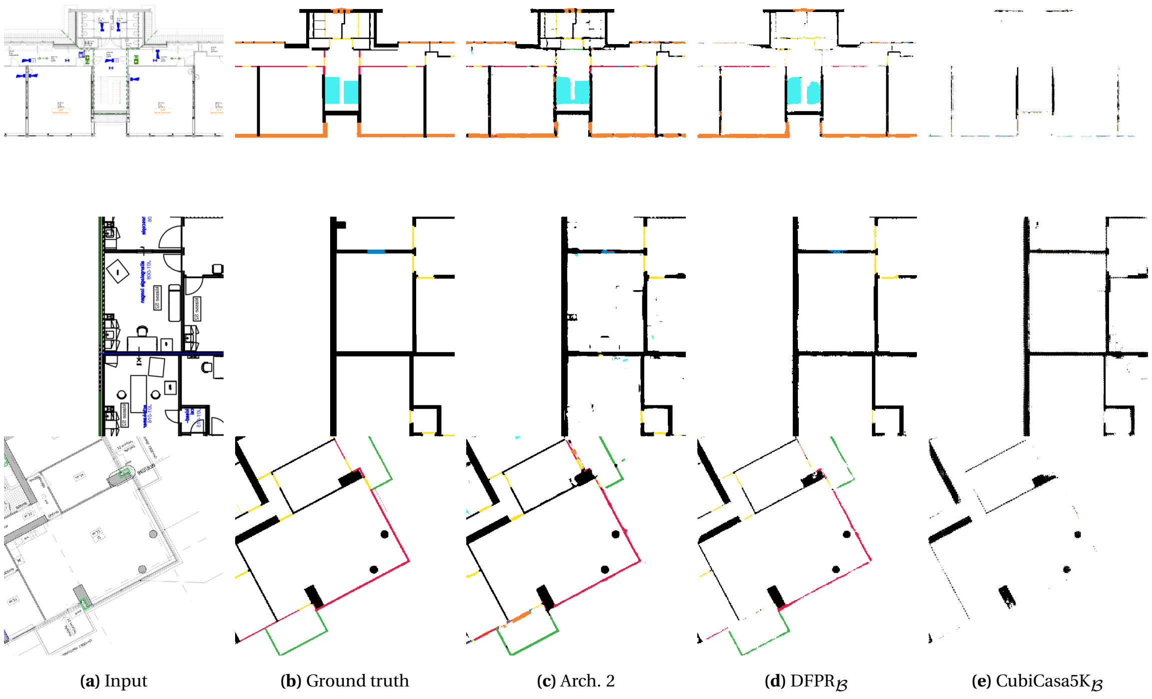

(a) Input (b) Ground truth (c) Arch. 2 (d) DFPR$_{\mathcal{B}}$ (e) CubiCasa5K$_{\mathcal{B}}$

Figure 5.5: Qualitative comparison of segmentation masks from MURF.

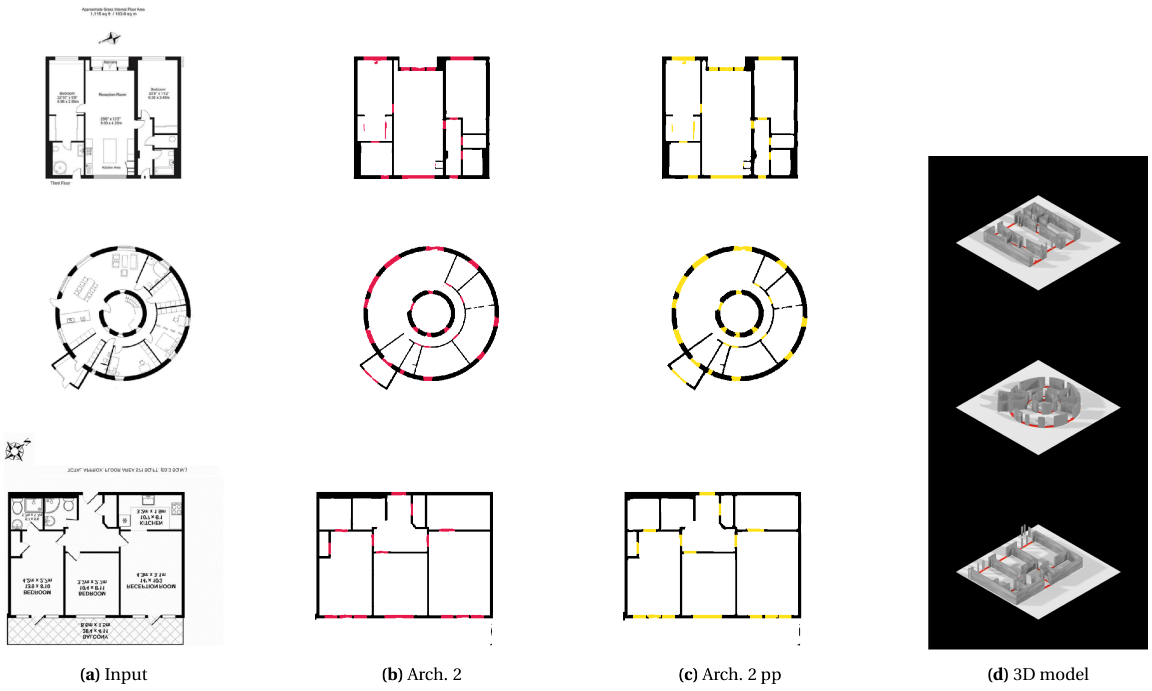

(a) Input (b) Arch. 2 (c) Arch. 2 pp (d) 3D model

Figure 5.6: Qualitative comparison of generated 3D models by proposed architecture 2 from R3D.

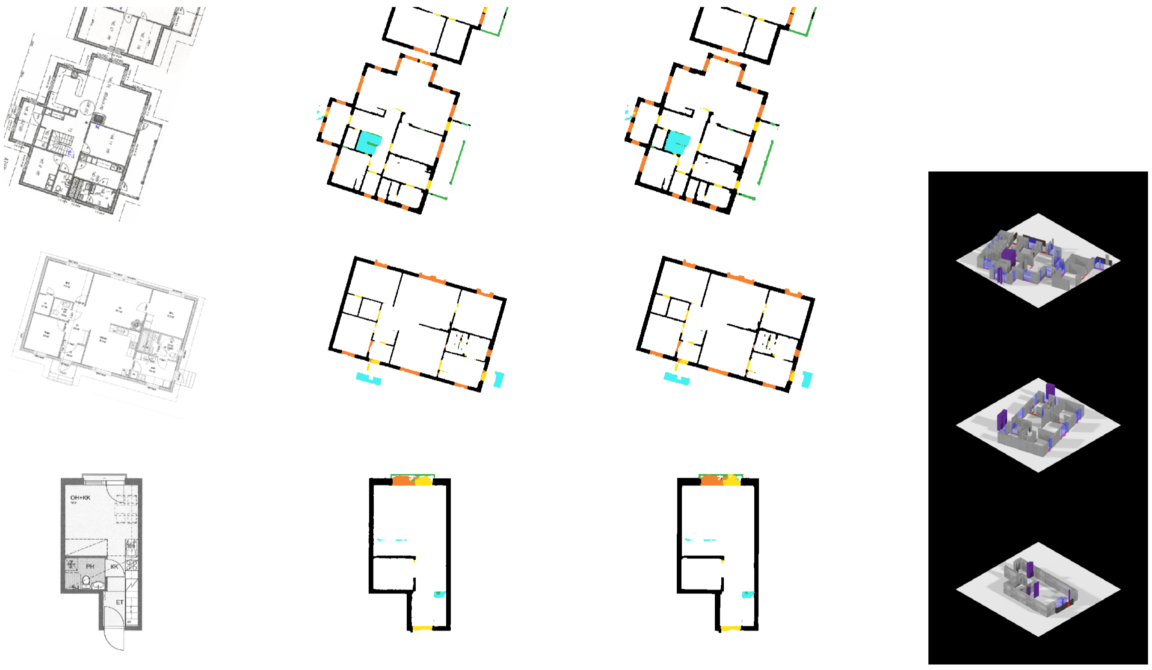

(a) Input **(b)** Arch. 2 **(c)** Arch. 2 pp **(d)** 3D model

Figure 5.7: Qualitative comparison of generated 3D models by proposed architecture 2 from CubiCasa.

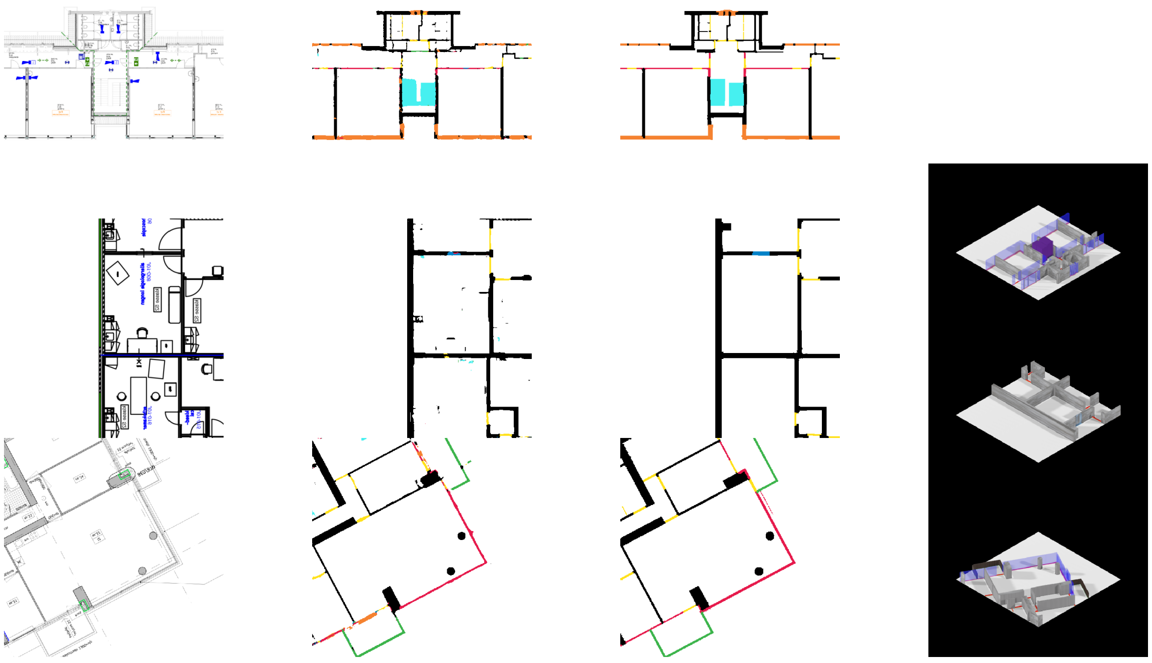
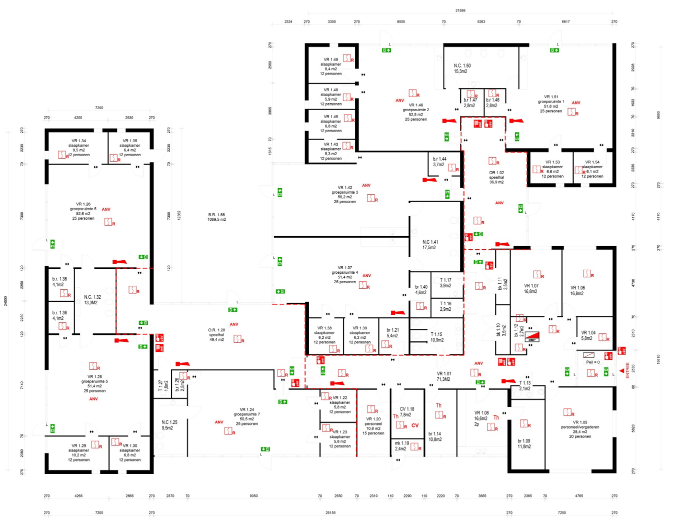
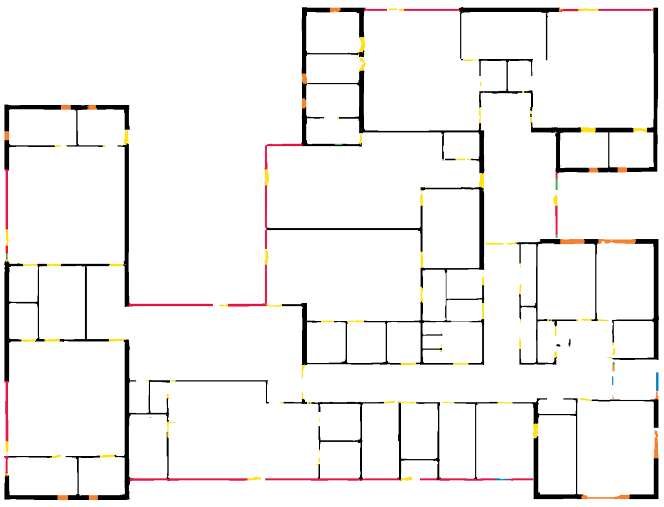

(a) Input **(b)** Arch. 2 **(c)** Arch. 2 pp **(d)** 3D model

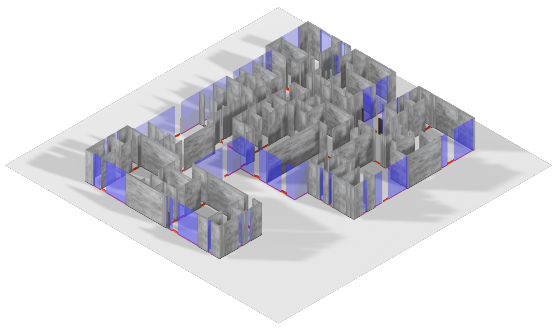
Figure 5.8: Qualitative comparison of generated 3D models by proposed architecture 2 from MURF.



(a) Input floor plan



(b) Post-processed segmentation mask



(c) 3D model

Figure 5.9: 3D model generated from full-sized MURF$_{\text{test}}$ dataset.

5.4.2. Case Study

In addition to the qualitative evaluation on floor plan datasets, this section presents a case study of the proposed floor plan processing method on the Erasmus Medical Center (EMC) in Rotterdam. The goal of the case study on the EMC is to evaluate the differences between a man-made 3D model and an automatically generated 3D model of a part of the building of the EMC. The building of the EMC consists of several parts constructed during different years. A 3D model has only been created for the so-called NG building of the EMC. Unfortunately, the NG part of the EMC is not visible in the floor plan, and thus a similar region of the floor plan is considered in this case study. Figure 5.10 shows the front view of the EMC. The green and red areas correspond to the same areas in the floor plan shown in Figure 5.11. The blue area in Figure 5.11 is the region considered for the 3D model comparison.

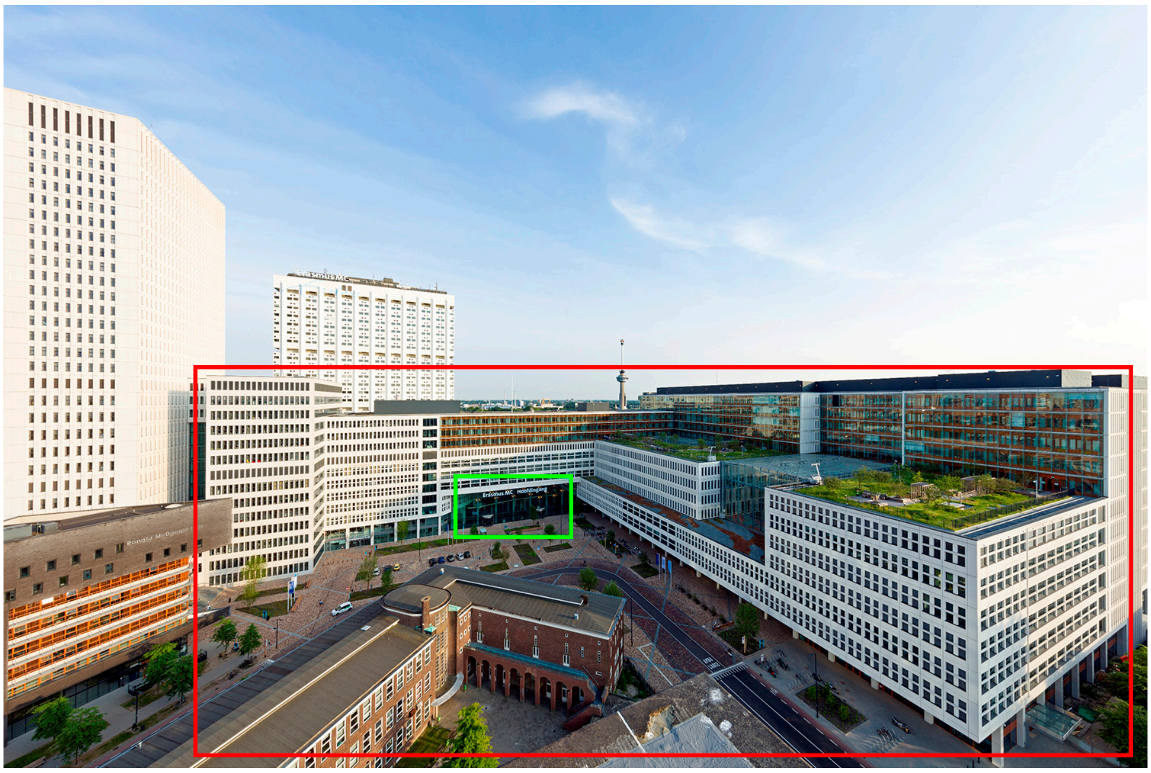


Figure 5.10: Front view of the Erasmus Medical Center. Areas marked in red and green correspond to the same areas in the floor plan (Figure 5.11). Source: Rob van Esch (May 2018).

One notable difference between the floor plan of the EMC and other multi-unit floor plans is the difference in image contrast. A small contrast modification formalized by Equation 5.8 was applied to correct this. The contrast for all pixels within a lower and higher bound based on the floor plan is increased by a factor of 2. Note that the highest value is equal to 255, so to increase contrast the pixel value must be divided instead. Figure 5.12 shows the result of applying the contrast correction on a portion of the EMC floor plan.

$$y_i = \begin{cases} \frac{y_i}{2} & \text{if } \frac{1}{|\{y_i : y_i < 250\}|} \sum_i y_i < y_i < \frac{1}{|y|} \sum_i y_i \\ y_i & \text{otherwise} \end{cases} \tag{5.8}$$

where $y_i \in [0, 255]$ is the value of a pixel i in the floor plan with a value of 255 equal to white.

Results

Figure 5.13 shows a comparison between the man-made 3D model and the generated model of a part of the EMC. The results show that, after applying a simple contrast correction, the overall quality of the generated 3D model is sufficient to recognize similar features between the man-made model and the generated model. For instance, the column structure along the walls is identified, similar to in the man-made model. Several stairs are also correctly identified in the region of the floor plan. There is still a

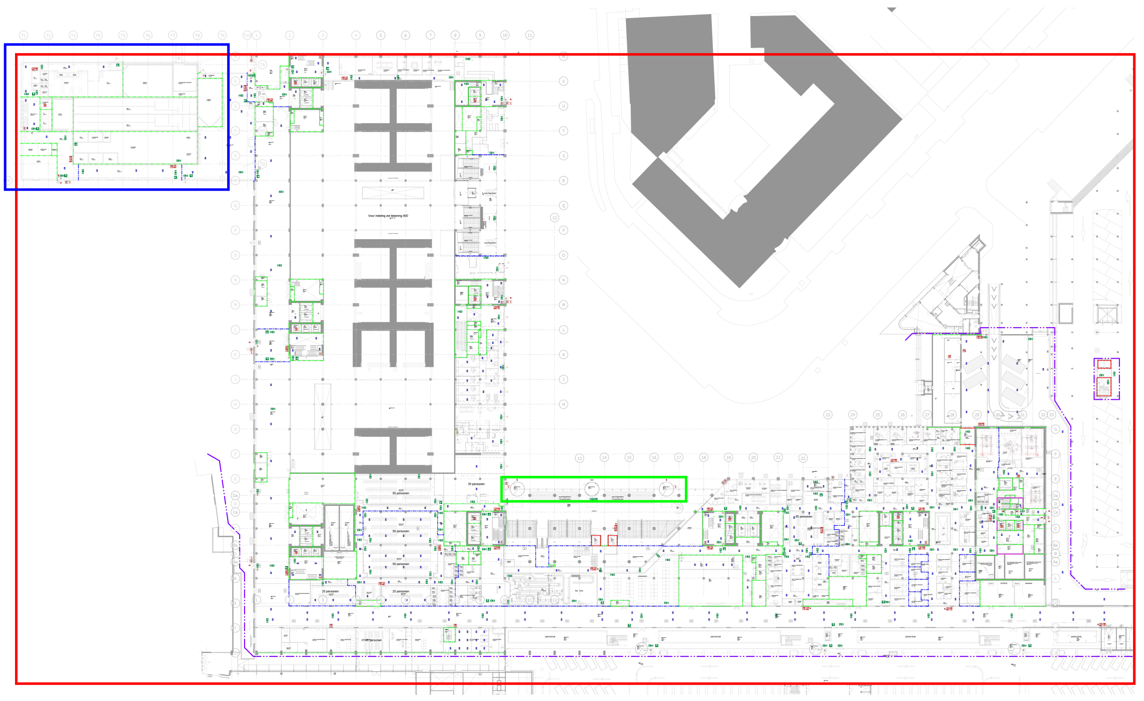
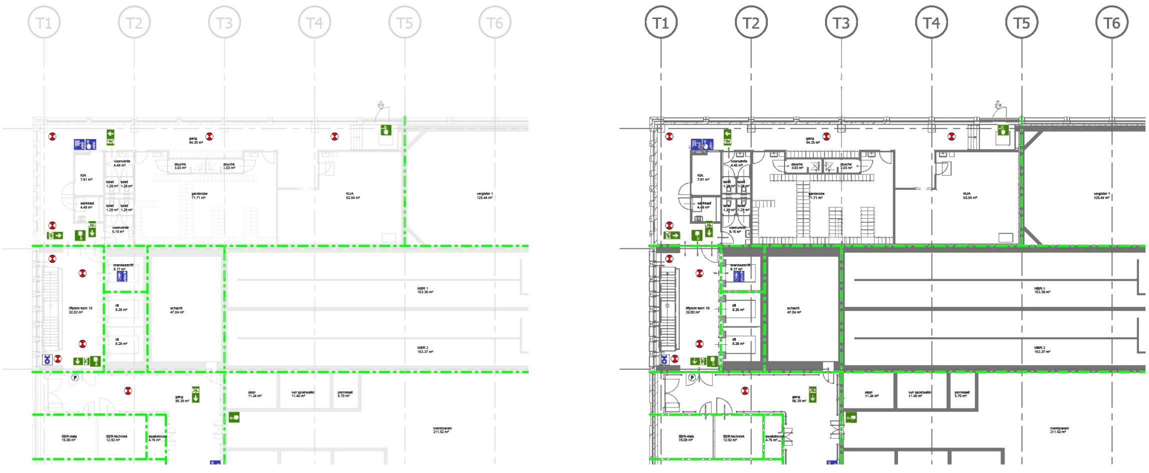


Figure 5.11: Architectural floor plan of the Erasmus Medical Center. Areas marked in red and green correspond to the same areas in the front view (Figure 5.10).



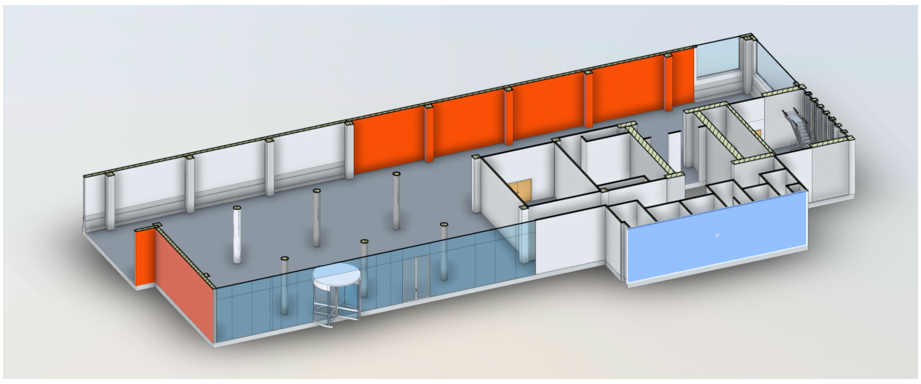
(a) Original contrast in EMC floor plan. **(b)** Corrected contrast in EMC floor plan.

Figure 5.12: Comparison of pixel contrast in EMC floor plan before and after applying contrast correction.

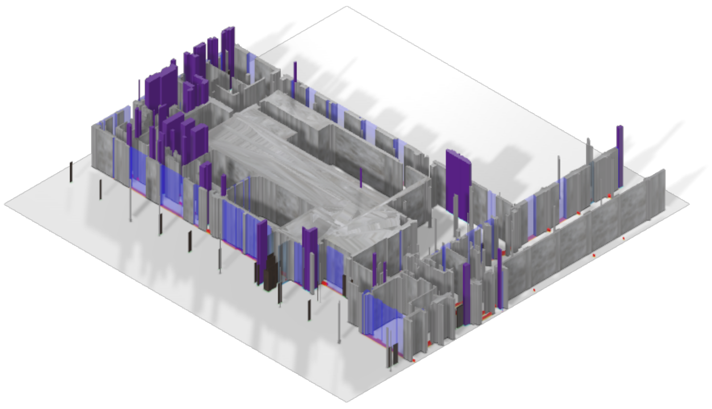
moderate amount of noise, but this can be further reduced by applying an improved contrast correction. Overall the proposed architecture shows promising results on an unseen complex multi-unit floor plan.

5.5. Ablation Studies

Two ablation studies have been conducted to evaluate the design choices made and the impact of added components towards overall performance. This section describes the experiments and their results. Each ablation study has been performed on the combined dataset, as this has been shown to be the most difficult dataset to learn from. This allows for more room to improve a baseline model and thus better



(a) Man-made 3D model of NG part of EMC building.



(b) Generated 3D model of portion of original floor plan.

Figure 5.13: Comparison man-made and generated 3D models of the EMC.

examination of the effect of an added component.

5.5.1. Attention Mechanism

The first ablation study is on the CAM and SAM blocks used in the attention mechanism in the proposed architectures. To investigate the improved accuracy obtained by using the CAM and SAM blocks, each is added separately to a baseline architecture. The baseline architecture is a variant of model architecture 2 that performs a single *AC* convolution to compute the decoder blocks, similar to the square convolution used in U-Net3+ [40]. Table 5.12 contains the results of the ablation study on the attention mechanism.

The results show that using both a CAM and SAM module achieves the best performance. Using only the CAM module increases the mean IoU. The SAM module is likely responsible for the improved pixel accuracy and mean class accuracy.

5.5.2. Multi-Task Loss

The final multi-task loss consists of three losses: asymmetric unified focal loss $\mathcal{L}_{\text{aUF}}$, adaptive affinity field loss $\mathcal{L}_{\text{AAF}}$, and multi-scale heatmap regression loss $\mathcal{L}_{\text{MHR}}$. In order to determine their impact on accuracy, the losses are iteratively combined and evaluated on the combined dataset with architecture 2. Table 5.13 shows the results.

	AC	CAM	SAM	CAM+SAM
Pixel Acc.	0.85	0.85	0.85	**0.85**
Mean Class Acc.	0.66	0.66	0.66	**0.70**
Mean Class IoU.	0.54	**0.55**	0.54	0.54

Table 5.12: Ablation study of CAM and SAM blocks used in the attention mechanism on the combined dataset. The baseline model is a variant of model architecture 2 that uses a single *AC* block.

The results show that the combined loss function achieves the best performance. The affinity field loss increases the mean class accuracy. The added heatmap regression loss further increases the class accuracy and mean IoU.

	$\mathcal{L}_{\text{aUF}}$	$\mathcal{L}_{\text{aUF}} + \mathcal{L}_{\text{AAF}}$	$\mathcal{L}_{\text{aUF}} + \mathcal{L}_{\text{AAF}} + \mathcal{L}_{\text{MHR}}$
Pixel Acc.	0.85	0.85	**0.85**
Mean Class Acc.	0.66	0.68	**0.70**
Mean Class IoU.	0.53	0.54	**0.54**

Table 5.13: Ablation study of combining multi-task loss terms of architecture 2 on the combined dataset. Terms $\mathcal{L}_{\text{aUF}}$, $\mathcal{L}_{\text{AAF}}$ and $\mathcal{L}_{\text{MHR}}$ refer to the asymmetric unified focal loss, adaptive affinity field loss, and multi-scale heatmap regression loss respectively.

5.6. Reflection on RQs

The use of the novel multi-unit floor plan dataset MURF has been evaluated and compared to two public datasets R3D and CubiCasa. RQ1 investigates how a multi-unit floor plan dataset can best be defined, the following was shown:

- A multi-scale sampling method can be used to capture multi-scale features in large multi-unit floor plans. This ensures training samples are small enough to fit into GPU memory, while preserving features useful to learn for inference on full-scale floor plans.
- Augmenting a floor plan dataset can artificially increase the complexity of floor plans, forcing models to learn features that often occur in multi-unit floor plans.
- A set of fixed heights and textures per semantic class is sufficient to allow semantic classes to be distinguished easily in generated 3D models.

This chapter also evaluated the performance of the two proposed model architectures through a quantitative and qualitative evaluation, related to RQ2 and RQ3 respectively. The following observations were made in this chapter:"

- The two proposed architectures improve performance significantly compared to existing floor plan models DFPR [16] and CubiCasa5K [28]. The performance is improved in terms of overall pixel accuracy, class accuracy and IoU values.
- Compared to state-of-the-art methods, the fully connected architecture 2 achieves the best performance. However, the differences are notably smaller compared to the improved performance over DFPR [16] and CubiCasa5K [28].
- A deeply supervised variant of architecture 2 offers a marginal increase in accuracy, at the cost of a lower IoU.
- The post-processing method increases accuracy significantly, but reduces the IoU.
- Blender is an effective tool for visualizing vectorized segmentation masks.

6 Discussion

This chapter presents a further discussion of MURF, and the recognition and reconstruction results of the two proposed model architectures considering the results of Lu et al. [27] and Lv et al. [17]. Limitations of the proposed floor plan processing method and possible use cases are discussed next.

To the best of our knowledge, MURF serves as the first multi-unit floor plan dataset evaluated in a deep learning setting. Although MURF consists of a small number of floor plans compared to R3D and CubiCasa5K, it contains sufficient data for effective learning. MURF is also the first multi-unit floor plan dataset considering multiple types of walls and doors, including glass walls, railings and sliding doors. Lv et al. [17] have proposed a larger single-unit floor plan dataset RFP, which could be used to create a larger combined dataset. Further evaluation is necessary to determine whether this is possible, as RFP does not consider railings or stairs classes.

The recognition results from experiment 1 showed that the two proposed architectures outperform two recent floor plan processing models DFPR [16] and the CubiCasa5K model [28] on floor plan datasets of varying complexity. The difference in performance becomes more significant when the complexity of a floor plan dataset increases, which was especially the case for the combined dataset. This shows that the increased complexity of the proposed architectures allow the models to better capture a larger range of features necessary to identify the elements in floor plans of different sizes.

Lu et al. [27] proposed a different floor plan processing CNN and also evaluated the DFPR model on a subset of the CubiCasa5K dataset. Compared to Lu et al., the proposed architectures in this thesis and DFPR achieve a slightly lower accuracy on the refined CubiCasa dataset. The lower accuracy is expected, because the CubiCasa dataset contains augmented floor plans, in addition to the added stairs class.

Experiment 2 showed that, compared to state-of-the-art segmentation models such as U-Net++ [39] and U-Net3+ [40], only the second proposed fully connected architecture achieves the best performance on floor plan datasets. The regularly connected architecture cannot utilize the feature maps from multiple scales, reducing segmentation accuracy. The increased accuracy of the second architecture is due to the attention mechanism designed to highlight important features by considering the spatial and channel dimensions separately. Since the attention mechanism has been designed to highlight floor plan features specifically, further evaluation is necessary to determine whether the proposed model also improves accuracy in other image segmentation domains. The results of U-Net illustrate that a higher model complexity does not necessarily improve performance, which could mean that, instead of the second architecture, the first model architecture might be more effective in other domains.

A qualitative evaluation further examined the results of the reconstruction pipeline. Lv et al. [17] perform a similar evaluation considering the single-unit floor plans from the original non-augmented R3D and CubiCasa5K datasets. The proposed models in this thesis produce segmentation masks of comparable accuracy. The reconstruction step, however, produces more noisy post-processed segmentation masks and 3D models compared to those of Lv et al. This is likely due to the higher complexity of the Lv et al. model, which generates a set of endpoints per opening class in addition to the regular segmentation masks. The set of endpoints is used in a vectorization step, which refines the segmentation mask in a much more aggressive manner.

6.1. Limitations

One limitation of the proposed MURF dataset is the small number of floor plans compared to other public floor plan datasets. Although it has been shown that MURF contains similar amount of data compared to other datasets, more data could help the model generalize better. Due to the high annotation cost per floor plan, more floor plans could not be labelled and used for training. Model-assisted labelling[1] could speed up the annotating process significantly, but was not explored due to lack of sufficient ground-truth data that it requires.

Although an extensive evaluation was performed on the hardware specified in this thesis, additional computational resources could further improve recognition performance. For instance, a more complex backend model for the encoder such as EfficientNet-B3 may increase model accuracy. More computational power could also permit larger batch sizes for training the model proposed in this thesis. For a larger batch size, batch normalization could further improve performance compared to group normalization. Further hyper parameter tuning could also improve performance. Although a guided approach was applied to optimize hyper parameters, no exhaustive search was performed. A larger set of kernel sizes for the affinity field loss, or different β values to control the spread in the heatmaps for the heatmap regression loss could be considered.

The recognition part of the model purposefully lacks a region of interest (ROI) detection module used to initially crop a floor plan to improve accuracy. The reason for not including this module is that using a popular object detection model such as YOLOv5 [80] is likely sufficient to crop floor plans to their appropriate size. For example, Lv et al. [17] use a variant of YOLOv4 for their ROI module. No further optimizations would be necessary here that can be considered a scientific contribution. Furthermore, an ROI module would not improve performance on the MURF dataset, as the multi-unit floor plans were already cropped to the appropriate regions during annotating. The floor plans in the R3D and CubiCasa datasets are also do not require any additional cropping.

One limitation of the reconstruction method is the visualization of stairs. Currently, stairs are visualized by polygons of equal height, set to a higher value than other semantic classes. Furthermore, the detected stairs do not take the orientation of the stairs into consideration, which can be used in visualization to improve the use of stairs in the generated 3D models.

A second limitation of the reconstruction method is the way round-shaped objects are visualized. Currently, the segmentation masks are transformed into a set of polygons with the endpoints of each connected component as the vertices. The disadvantage of this approach is that it generates staircase like artifacts due to the discrete values in the segmentation mask. These artifacts are especially apparent for round-shaped objects.

6.2. Applications

Architectural floor plans play a crucial role in architectural firms and engineering offices. As a result, methods transforming architectural floor plans into other formats can have many applications. Compared to single-unit floor plan processing methods, the proposed method in this thesis can be applied to both single-unit and multi-unit floor plans, and thus to more buildings. This section discusses three exciting applications within the industry that could be explored.

The first application is related to why the proposed method was designed to begin with: the digital twin of Rotterdam. The proposed method can be applied to the entirety of Rotterdam due to two improvements. The first improvement is the larger range of floor plan sizes that the proposed model can process compared to existing methods. The second improvement is the scalability of the proposed method, both in terms of inference time and required data. The proposed method can generate the 3D model of any sized floor plan within minutes on commodity hardware. A point-cloud based method would, for instance, require more time, but more importantly, more data to generate a 3D model. A 3D representation of Rotterdam could prove useful for the fire brigade of Rotterdam, which currently relies exclusively on floor plans for the majority of the buildings in Rotterdam in case of an emergency.

The second application would be to apply the floor plan processing method in the residential market. Rasterizing and visualizing floor plans is a growing market with a predicted value of $43bn by 2025 [14]. An automatic approach to detect a larger set of semantic elements in floor plans compared to existing approaches could pose as a valuable starting point for generating 3D models used in the rental market.

[1] https://docs.labelbox.com/docs/model-assisted-labeling

The third application is related to BIM reconstruction. BIM models are dense 3D representations of buildings, describing additional properties such as material, thermal and other properties. BIM models allow for several use cases, such as structural analysis, thermal simulations, and emergency route planning. The produced 3D models produced by the method in this thesis can be converted into BIM models by augmenting the model with additional data if available.

7 Conclusion

The goal of this thesis is to propose a floor plan processing method capable of processing floor plans of arbitrary size in a scalable manner, focusing on identifying the necessary structural elements to create an accurate 3D representation. A novel multi-unit floor plan dataset, MURF, is introduced and used to evaluate the proposed method in this thesis.

The MURF dataset is compared to two other datasets, namely R3D and CubiCasa. MURF showed that a small number of multi-unit floor plans are necessary compared to many single-unit floor plans to represent a comparable amount of data. A list of 7 semantic classes were found to be sufficient to generate realistic 3D models that can be used for various use cases. To facilitate effective learning, a multi-scale sampling method was used to split the multi-unit floor plans into distinct tiles. Learning from multi-scale samples reinforced models to adopt multi-scale features.

The recognition part of the proposed method is responsible for identifying the semantic elements in floor plans. Multi-unit floor plans are very challenging to process, mainly due to their size and increased complexity compared to single-unit floor plans. A key part in being able to handle both is to learn from floor plans at different scales. Different model architectures combining multi-scale features have been evaluated to see which skip connections performed best. An additional attention mechanism is employed to improve the representation power of the multi-scale features, highlighting feature maps separately based on their spatial or channel importance.

The recognition method is evaluated by comparing it to DFPR [16] and CubiCasa5K [28] on augmented variants of four datasets: R3D, a refined version of CubiCasa5K [28] called CubiCasa, MURF, and a combination of CubiCasa and MURF. Compared to the state-of-the-art floor plan processing methods, the proposed architectures show promising results, increasing accuracy and intersection over union significantly on all datasets. The difference in performance is larger for the floor plan datasets of higher complexity.

A second experiment is conducted to investigate the performance of the proposed architectures compared to U-Net++ [39] and U-Net3+ [40]: two general state-of-the-art segmentation models that have shown promising results in other research domains. The results show that, while the difference in performance is smaller compared to the first experiment, the second proposed architecture using fully connected multi-scale features performs best overall. A deeply supervised variant of architecture 2 is also evaluated which supervised the intermediate decoder layers. Applying deep supervision resulted in a marginal increase in model accuracy, at the cost of a decreased IoU.

The reconstruction part of the proposed method transforms the produced segmentation masks into a 3D model by applying a pipeline consisting of post-processing, vectorization and visualization. The post-processing method is designed to refine elements of arbitrary orientation in the floor plan. This is a necessary improvement over post-processing methods in literature, because multi-unit floor plans almost always contain elements not aligned with the floor plan axes. The vectorization and visualization steps generate a list of polygons from the post-processed segmentation mask and render their combined 3D representations using Blender respectively. Similar to related work [16, 17, 28], this is actually a 2.5D representation of the floor plan, as the height of elements is set to a pre-defined value depending on the semantic class. Stairs are, for instance, rendered with a higher height to improve their visibility in a 3D model.

A qualitative evaluation on the reconstruction part is presented based on the floor plan datasets and a case study on the Erasmus Medical Center (EMC) in Rotterdam. Compared to DFPR [16] and CubiCasa5K [28], the proposed architectures produce segmentation masks of higher accuracy. Post-processing further reduces inaccurate predictions and can be effectively paired with vectorization and visualization to generate 3D models. The case study on the EMC shows that the proposed method generalizes well enough to generate a 3D model of a large building without any additional required training on data of the building. With the exception of a different type of door, the generated 3D model of the EMC looks sufficiently similar to a man-made 3D model created for a part of the building.

7.1. Answers to RQs

The first research question is: how can a multi-unit floor plan dataset best be defined for effective learning and visualization? This thesis showed that, compared to single-unit floor plans, a small number of multi-unit floor plans are necessary to represent a comparable amount of semantic data. A sampling method is necessary to split the large floor plans into smaller tiles, to ensure the images fit in to GPU memory. Sampling from multiple sizes can facilitate multi-scale learning, improving overall model performance. Regarding visualization, 7 semantic classes were found to be sufficient, each semantically distinct and offering additional information about, for instance, opacity or height. Object or room types in multi-unit floor plans was found to be unclear or contradictory, and were thus deemed too difficult to learn.

The second research question is: what CNN architecture can effectively detect semantic information in architectural floor plans of arbitrary size in a scalable manner? Three important design choices were made to achieve this. The first was to use an FCN, which can process inputs of arbitrary size. The second was to adopt a multi-task training objective, which optimizes pixel, patch and multi-scale loss simultaneously. The third and last design choice was to use an attention mechanism block, highlighting and suppressing important and indiscriminative spatial and channel dimensions. Although training the proposed FCN is computationally expensive, inference is not and thus considered scalable.

The third and last research question is: how can produced segmentation masks best be transformed into a 3D representation? A reconstruction pipeline was proposed to achieve this, refining a segmentation mask and deriving approximate polygons used to visualize a floor plan in Blender. A pre-defined height and set of textures was used to better illustrate the physical differences between semantic classes in a floor plan.

7.2. Future Directions

The focus of this thesis was to detect structural elements in multi-unit floor plans. Several research directions have been purposefully left unexplored, to show a relatively simple model considering only structural elements can generate accurate 3D representations of floor plans. This section will elaborate on four possible future research directions that could further improve the performance of the proposed model.

The first future direction could be to augment the model to also detect object types in floor plans. This may not be necessary in the use cases described in this thesis, but could prove useful in other domains such as indoor modelling. The challenge here would be to come up with a list of object types, and sufficient data for each object, for the model to generalize well. Detecting object types in a multi-unit floor plan dataset is particularly challenging, as many floor plans would be necessary to supply the model with sufficient data on each object class for the model to generalize well.

The second future direction could be to integrate additional data sources into the floor plan processing pipeline, further improving accuracy or allowing for more complex use cases. Examples of integrating data include deriving textures from public images of buildings for visualization, or adding real-time sensor data of a building to the generated 3D model to visualize crowds.

The third future direction would be to expand the evaluation with transformers. Transformers have recently been shown to be effective at image segmentation, and could prove effective in the domain of floor plan processing as well. Transformer-based approaches usually process an input image in patches. Developing a scalable method of processing the large multi-unit floor plans in patches would pose as an additional challenge.

A fourth and last direction would be to improve inference speed by performing kernel fusion of the asymmetric convolution blocks, as proposed in ACNet [59]. Since the separate horizontal and vertical kernels are no longer necessary after training, they can be merged by element-wise summation with the regular square kernel used in the convolutional layers. Merging the kernels results in the same inference-time performance compared to a model without using the asymmetric convolution block, since no extra computations are required.

Bibliography

[1] Chen Liu et al. “Raster-to-vector: Revisiting floorplan transformation”. In: *Proceedings of the IEEE International Conference on Computer Vision.* 2017, pp. 2195–2203.

[2] Chengshuang Sun et al. “A literature review of the factors limiting the application of BIM in the construction industry”. In: *Technological and Economic Development of Economy* 23.5 (2017), pp. 764–779.

[3] Yusuf Arayici, Timothy Onyenobi, and Charles Egbu. “Building information modelling (BIM) for facilities management (FM): The MediaCity case study approach”. In: *International Journal of 3-D Information Modeling (IJ3DIM)* 1.1 (2012), pp. 55–73.

[4] Timo Hartmann and Martin Fischer. “Applications of BIM and Hurdles for Widespread Adoption of BIM”. In: *2007 AISC-ACCL eConstruction Roundtable Event Rep* (2008).

[5] Brian Okorn et al. “Toward automated modeling of floor plans”. In: *Proceedings of the symposium on 3D data processing, visualization and transmission.* Vol. 2. 2010.

[6] Charles Thomson and Jan Boehm. “Automatic geometry generation from point clouds for BIM”. In: *Remote Sensing* 7.9 (2015), pp. 11753–11775.

[7] Vladeta Stojanovic et al. “Generation of Approximate 2D and 3D Floor Plans from 3D Point Clouds.” In: *VISIGRAPP (1: GRAPP).* 2019, pp. 177–184.

[8] Junfang Zhu, Hui Zhang, and Yamei Wen. “A new reconstruction method for 3D buildings from 2D vector floor plan”. In: *Computer-aided design and applications* 11.6 (2014), pp. 704–714.

[9] Hichem Barki et al. “BIM models generation from 2D CAD drawings and 3D scans: an analysis of challenges and opportunities for AEC practitioners”. In: *Building Information Modelling (BIM) in Design, Construction and Operations* 149 (2015), pp. 369–380.

[10] Jalaj Pandey and Ojaswa Sharma. “Fast and robust construction of 3D architectural models from 2D plans”. In: (2016).

[11] Yunfan Zhao, Xueyuan Deng, and Huahui Lai. “Reconstructing BIM from 2D structural drawings for existing buildings”. In: *Automation in Construction* 128 (2021), p. 103750.

[12] RG Kippers et al. “AUTOMATIC 3D BUILDING MODEL GENERATION USING DEEP LEARNING METHODS BASED ON CITYJSON AND 2D FLOOR PLANS.” In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* (2021).

[13] Xuetao Yin, Peter Wonka, and Anshuman Razdan. “Generating 3d building models from architectural drawings: A survey”. In: *IEEE computer graphics and applications* 29.1 (2008), pp. 20–30.

[14] Pablo N Pizarro et al. “Automatic floor plan analysis and recognition”. In: *Automation in Construction* 140 (2022), p. 104348.

[15] Seongyong Kim et al. “Deep floor plan analysis for complicated drawings based on style transfer”. In: *Journal of Computing in Civil Engineering* 35.2 (2021), p. 04020066.

[16] Zhiliang Zeng et al. “Deep floor plan recognition using a multi-task network with room-boundary-guided attention”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 9096–9104.

[17] Xiaolei Lv et al. “Residential floor plan recognition and reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2021, pp. 16717–16726.

[18] Ruiyun Zhu et al. “Training strategies for cnn-based models to parse complex floor plans”. In: *Proceedings of the 2020 9th International Conference on Software and Computer Applications.* 2020, pp. 11–16.

[19] Philippe Dosch et al. “A complete system for the analysis of architectural drawings”. In: *International Journal on Document Analysis and Recognition* 3.2 (2000), pp. 102–116.

[20] Siu-hang Or et al. “Highly automatic approach to architectural floorplan image understanding & model generation”. In: *Pattern Recognition* (2005), pp. 25–32.

[21] Sébastien Macé et al. “A system to detect rooms in architectural floor plan images”. In: *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*. 2010, pp. 167–174.

[22] Sheraz Ahmed et al. “Improved automatic analysis of architectural floor plans”. In: *2011 International Conference on Document Analysis and Recognition*. IEEE. 2011, pp. 864–869.

[23] Lluis-Pere de las Heras et al. “Wall patch-based segmentation in architectural floorplans”. In: *2011 international conference on document analysis and recognition*. IEEE. 2011, pp. 1270–1274.

[24] Llus-Pere de las Heras et al. “Statistical segmentation and structural recognition for floor plan interpretation”. In: *International Journal on Document Analysis and Recognition (IJDAR)* 17.3 (2014), pp. 221–237.

[25] Samuel Dodge, Jiu Xu, and Björn Stenger. “Parsing floor plan images”. In: *2017 Fifteenth IAPR international conference on machine vision applications (MVA)*. IEEE. 2017, pp. 358–361.

[26] Chenxi Liu et al. “Rent3d: Floor-plan priors for monocular layout estimation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3413–3421.

[27] Zhengda Lu et al. “Data-driven floor plan understanding in rural residential buildings via deep recognition”. In: *Information Sciences* 567 (2021), pp. 58–74.

[28] Juho Kannala. “CubiCasa5K: A Dataset and an Improved Multi-task Model for Floorplan Image Analysis”. In: *Image Analysis: 21st Scandinavian Conference, SCIA 2019, Norrköping, Sweden, June 11–13, 2019, Proceedings*. Vol. 11482. Springer. 2019, p. 28.

[29] Dany Alejandro Cabrera Vargas. “Wall extraction and room detection for multi-unit architectural floor plans”. PhD thesis. 2018.

[30] Sheraz Ahmed et al. “Text/graphics segmentation in architectural floor plans”. In: *2011 International Conference on Document Analysis and Recognition*. IEEE. 2011, pp. 734–738.

[31] Sheraz Ahmed et al. “Automatic room detection and room labeling from architectural floor plans”. In: *2012 10th IAPR international workshop on document analysis systems*. IEEE. 2012, pp. 339–343.

[32] Llus-Pere de las Heras et al. “Unsupervised wall detector in architectural floor plans”. In: *2013 12th International Conference on Document Analysis and Recognition*. IEEE. 2013, pp. 1245–1249.

[33] Svante Wold, Kim Esbensen, and Paul Geladi. “Principal component analysis”. In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[34] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).

[35] Liang-Chieh Chen et al. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

[36] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “Yolov4: Optimal speed and accuracy of object detection”. In: *arXiv preprint arXiv:2004.10934* (2020).

[37] Hanme Jang, Kiyun Yu, and JongHyeon Yang. “Indoor reconstruction from floorplan images with a deep learning approach”. In: *ISPRS International Journal of Geo-Information* 9.2 (2020), p. 65.

[38] 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*. Springer. 2015, pp. 234–241.

[39] Zongwei Zhou et al. “Unet++: Redesigning skip connections to exploit multiscale features in image segmentation”. In: *IEEE transactions on medical imaging* 39.6 (2019), pp. 1856–1867.

[40] Huimin Huang et al. “Unet 3+: A full-scale connected unet for medical image segmentation”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 1055–1059.

[41] Rui Li, Chenxi Duan, and Shunyi Zheng. “MACU-Net Semantic Segmentation from High-Resolution Remote Sensing Images”. In: *arXiv preprint arXiv:2007.13083* (2020).

[42] Alyaa Amer, Tryphon Lambrou, and Xujiong Ye. “MDA-Unet: A Multi-Scale Dilated Attention U-Net for Medical Image Segmentation”. In: *Applied Sciences* 12.7 (2022), p. 3676.

[43] Wen Liu, Yankui Sun, and Qingge Ji. “Mdan-unet: multi-scale and dual attention enhanced nested u-net architecture for segmentation of optical coherence tomography images”. In: *Algorithms* 13.3 (2020), p. 60.

[44] Mehreen Mubashar et al. “R2U++: a multiscale recurrent residual U-Net with dense skip connections for medical image segmentation”. In: *Neural Computing and Applications* (2022), pp. 1–17.

[45] Xuezhi Xiang et al. “FCDNet: A Change Detection Network Based on Full-Scale Skip Connections and Coordinate Attention”. In: *IEEE Geoscience and Remote Sensing Letters* 19 (2022), pp. 1–5.

[46] Dashuai Tian. “A Change Detection Method Based on Full-scale Skip Connections and Mixed Pooling Module”. In: *Journal of Physics: Conference Series*. Vol. 2258. 1. IOP Publishing. 2022, p. 012059.

[47] Yang Liu et al. “Mixed-UNet: Refined Class Activation Mapping for Weakly-Supervised Semantic Segmentation with Multi-scale Inference”. In: *arXiv preprint arXiv:2205.04227* (2022).

[48] Ledan Qian et al. “UNet#: A UNet-like Redesigning Skip Connections for Medical Image Segmentation”. In: *arXiv preprint arXiv:2205.11759* (2022).

[49] Chuanbo Qin et al. “Improved U-Net3+ with stage residual for brain tumor segmentation”. In: *BMC Medical Imaging* 22.1 (2022), pp. 1–15.

[50] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[51] Alexander Kirillov et al. “Pointrend: Image segmentation as rendering”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9799–9808.

[52] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[53] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Instance normalization: The missing ingredient for fast stylization”. In: *arXiv preprint arXiv:1607.08022* (2016).

[54] Yuxin Wu and Kaiming He. “Group normalization”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[55] Saurabh Singh and Shankar Krishnan. “Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11237–11246.

[56] Michal Drozdzal et al. “The importance of skip connections in biomedical image segmentation”. In: *Deep learning and data labeling for medical applications*. Springer, 2016, pp. 179–187.

[57] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[58] Haonan Wang et al. “Uctransnet: rethinking the skip connections in u-net from a channel-wise perspective with transformer”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 3. 2022, pp. 2441–2449.

[59] Xiaohan Ding et al. “Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1911–1920.

[60] Yuli Zhang et al. “The direction-aware, learnable, additive kernels and the adversarial network for deep floor plan recognition”. In: *arXiv preprint arXiv:2001.11194* (2020).

[61] Sanghyun Woo et al. “Cbam: Convolutional block attention module”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[62] Rui Li et al. “ABCNet: Attentive bilateral contextual network for efficient semantic segmentation of Fine-Resolution remotely sensed imagery”. In: *ISPRS journal of photogrammetry and remote sensing* 181 (2021), pp. 84–98.

[63] Zhanyu Ma et al. “Fine-grained vehicle classification with channel max pooling modified CNNs”. In: *IEEE Transactions on Vehicular Technology* 68.4 (2019), pp. 3224–3233.

[64] Mingxing Tan and Quoc Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[65] Chen-Yu Lee et al. “Deeply-supervised nets”. In: *Artificial intelligence and statistics*. PMLR. 2015, pp. 562–570.

[66] Michael Yeung et al. “Unified focal loss: Generalising dice and cross entropy-based losses to handle class imbalanced medical image segmentation”. In: *Computerized Medical Imaging and Graphics* 95 (2022), p. 102026.

[67] Tsung-Wei Ke et al. “Adaptive affinity fields for semantic segmentation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 587–602.

[68] Adrian Bulat and Georgios Tzimiropoulos. “Human pose estimation via convolutional part heatmap regression”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 717–732.

[69] Lipeng Ke et al. “Multi-scale structure-aware network for human pose estimation”. In: *Proceedings of the european conference on computer vision (ECCV)*. 2018, pp. 713–728.

[70] Aiden Nibali et al. “3d human pose estimation with 2d marginal heatmaps”. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 1477–1485.

[71] Federico Bolelli et al. “Spaghetti labeling: Directed acyclic graphs for block-based connected components labeling”. In: *IEEE Transactions on Image Processing* 29 (2019), pp. 1999–2012.

[72] Satoshi Suzuki et al. “Topological structural analysis of digitized binary images by border following”. In: *Computer vision, graphics, and image processing* 30.1 (1985), pp. 32–46.

[73] Alex Kendall, Yarin Gal, and Roberto Cipolla. “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7482–7491.

[74] Lukas Liebel and Marco Körner. “Auxiliary tasks in multi-task learning”. In: *arXiv preprint arXiv:1805.06334* (2018).

[75] Godfried T Toussaint. “Solving geometric problems with the rotating calipers”. In: *Proc. IEEE Melecon*. Vol. 83. 1983, A10.

[76] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: `http://www.blender.org`.

[77] Jeff Heaton. *Ian goodfellow, yoshua bengio, and aaron courville: Deep learning*. 2018.

[78] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[79] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[80] G Jocher et al. “ultralytics/yolov5: v6. 1-TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference”. In: *Zenodo, Feb* 22 (2022).

A
Dataset Samples

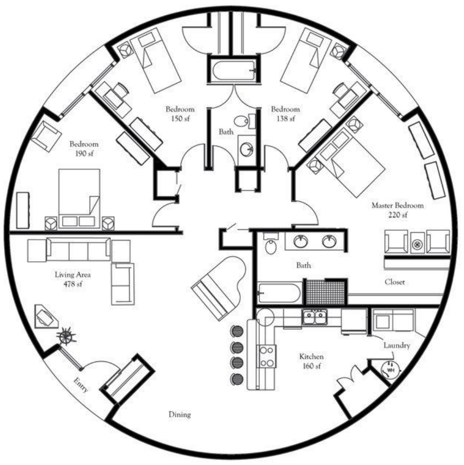
This chapter contains supplementary examples of floor plans from the two public datasets considered in this thesis: R3D [16] and a refined version of CubiCasa5K [28] called CubiCasa.

A.1. R3D

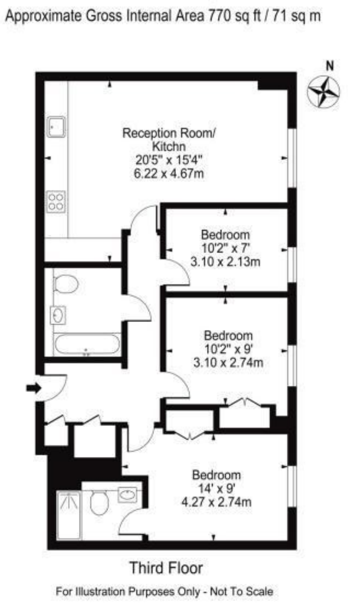
Figure A.1 contains three examples of floor plans in R3D.

A.2. CubiCasa

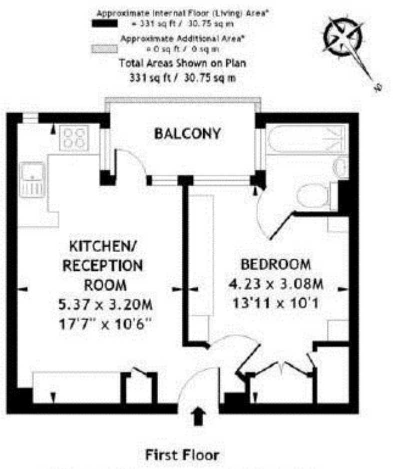
Figure A.2 contains three examples of floor plans in CubiCasa.



(a)

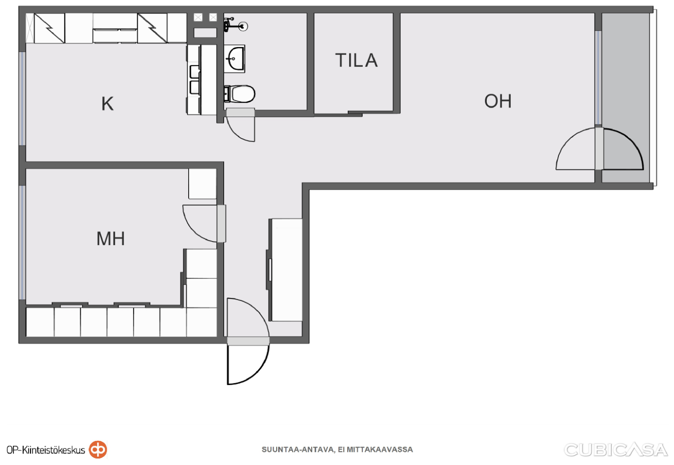


(b)

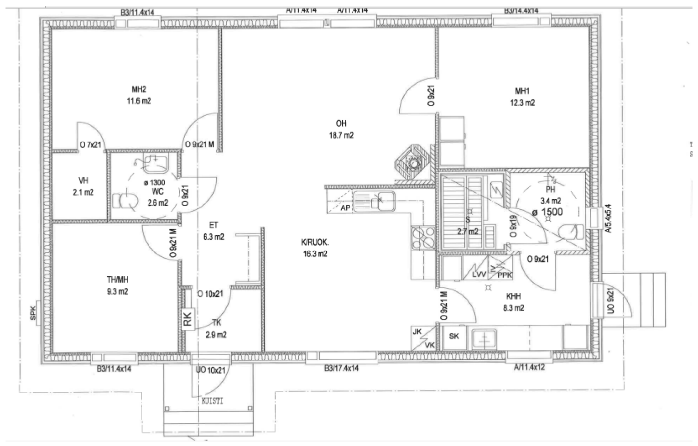


(c)

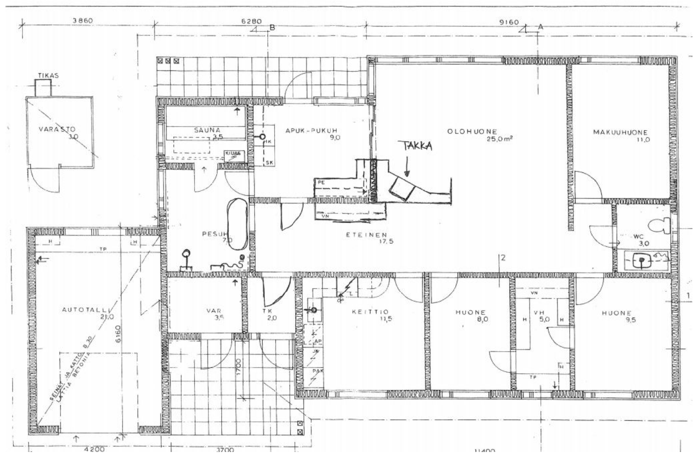
Figure A.1: Three supplementary examples of floor plans in the R3D dataset.



(a)



(b)

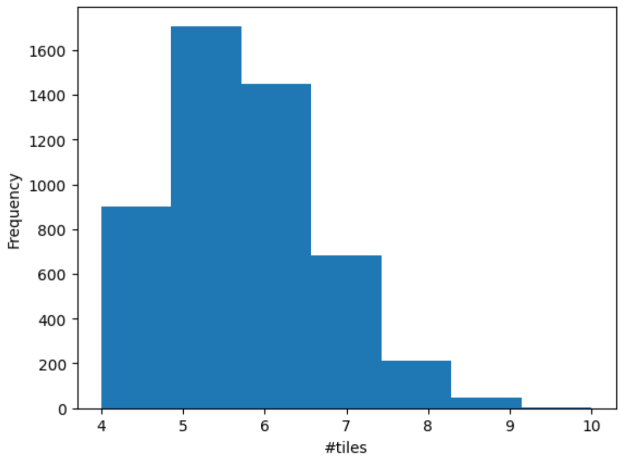


(c)

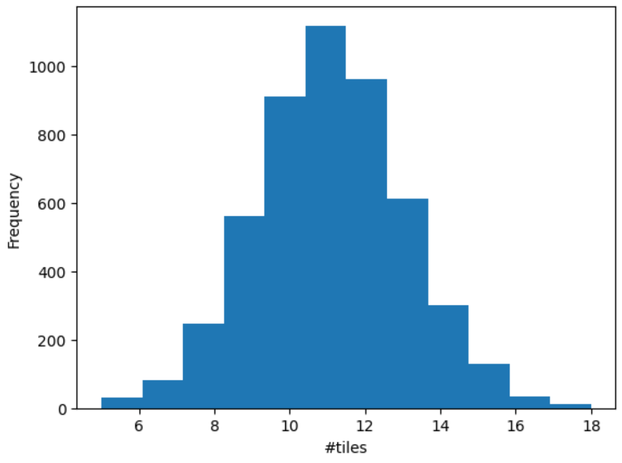
Figure A.2: Three supplementary examples of floor plans in the CubiCasa dataset.

B
Average Bound on Sampling Method

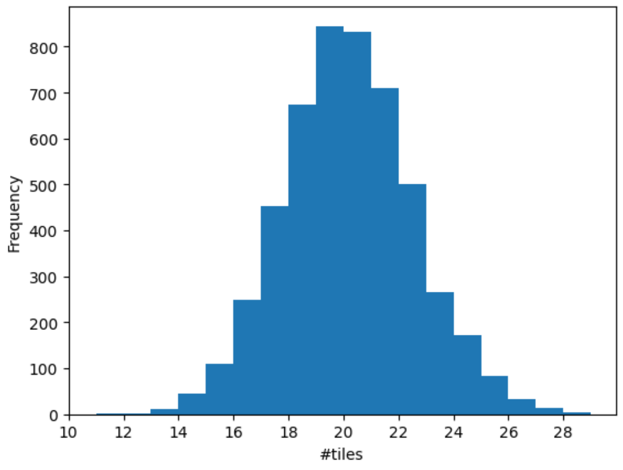
This chapter contains supplementary examples of the average number of tiles needed for floor plans of various resolutions. Figure B.1a, Figure B.1b and Figure B.1c shows the number of tiles for floor plans of 1000×1000, 1500×1500 and 2000×2000 respectively.



(a) $w = h = 1000$



(b) $w = h = 1500$



(c) $w = h = 2000$

Figure B.1: Three examples of the distribution of the average number of files for various floor plans of width w and height h.

C
Full Results Experiment 2

This section contains the full results of experiment 2, containing all class-related metrics for each of the four datasets considered.

R3D
Table C.1 and Table C.2 contain the accuracy and IoU results on R3D.

		U-Net	U-Net++	U-Net3+	Architecture 1	Architecture 2
Pixel Acc.		0.92	0.91	0.93	0.92	**0.93**
Class Acc.	Wall	0.94	0.93	0.95	0.95	**0.95**
	Door-and-window	0.87	0.85	0.86	0.87	**0.87**
Mean		0.90	0.89	0.90	0.90	**0.91**

Table C.1: Accuracy on R3D compared to DFPR [16] and CubiCasa5K [28]. The first column is copied from Zeng et al. for reference [16]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

		U-Net	U-Net++	U-Net3+	Architecture 1	Architecture 2
Class IoU	Wall	0.90	0.88	**0.90**	0.90	0.90
	Door-and-window	0.77	0.76	0.76	0.76	**0.77**
Mean		0.83	0.82	0.83	0.83	**0.84**

Table C.2: IoU on R3D compared to DFPR [16] and CubiCasa5K [28]. The first column is copied from Zeng et al. for reference [16]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

CubiCasa
Table C.3 and Table C.4 contain the accuracy and IoU results on CubiCasa.

		U-Net	U-Net++	U-Net3+	Architecture 1	Architecture 2
Pixel Acc.		0.86	0.82	0.86	0.86	**0.87**
Class Acc.	Wall	0.88	0.85	0.91	0.90	**0.91**
	Railing	0.67	0.66	0.64	0.62	**0.68**
	Door	0.71	0.68	0.70	0.71	**0.75**
	Window	**0.88**	0.79	0.86	0.86	0.85
	Stairs	0.78	**0.80**	0.73	0.73	0.78
Mean		0.78	0.76	0.77	0.76	**0.80**

Table C.3: Accuracy on CubiCasa compared to DFPR [16] and CubiCasa5K [28]. The first column is copied from Kalervo et al. for reference [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. *This value includes the background class and can thus not directly be compared to the others. Highest value marked in bold.

		U-Net	U-Net++	U-Net3+	Architecture 1	Architecture 2
Class IoU	Wall	**0.80**	0.75	0.79	0.79	0.78
	Railing	0.46	0.38	**0.48**	0.47	0.45
	Door	**0.58**	0.58	0.58	0.57	0.56
	Window	0.73	0.67	**0.74**	0.73	0.73
	Stairs	**0.63**	0.54	0.62	0.62	0.61
Mean		0.64	0.58	**0.64**	0.64	0.63

Table C.4: IoU on CubiCasa compared to DFPR [16] and CubiCasa5K [28]. The first column is copied from Kalervo et al. for reference [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

MURF

Table C.5 and Table C.6 contain the accuracy and IoU results on MURF.

		U-Net	U-Net++	U-Net3+	Architecture 1	Architecture 2
Pixel Acc.		0.91	0.83	0.91	0.90	**0.92**
	Wall	0.91	0.88	0.93	0.94	**0.95**
	Glass wall	0.85	0.78	0.86	0.85	**0.86**
	Railing	0.85	0.66	0.86	0.86	**0.88**
Class Acc.	Door	0.80	0.68	**0.80**	0.79	0.80
	Sliding door	0.83	0.37	0.84	0.83	**0.84**
	Window	0.86	0.72	0.89	0.88	**0.90**
	Stairs	0.84	**0.89**	0.86	0.84	0.87
Mean		0.85	0.71	0.86	0.85	**0.87**

Table C.5: Accuracy on MURF compared to DFPR [16] and CubiCasa5K [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

		U-Net	U-Net++	U-Net3+	Architecture 1	Architecture 2
	Wall	0.86	0.81	0.85	0.87	**0.87**
	Glass wall	0.80	0.67	0.78	0.80	**0.80**
	Railing	0.78	0.56	0.75	0.75	**0.79**
Class IoU	Door	0.65	0.58	0.65	0.65	**0.67**
	Sliding door	0.71	0.34	0.72	0.67	**0.75**
	Window	0.81	0.64	0.80	0.80	**0.81**
	Stairs	0.82	0.79	0.83	0.80	**0.83**
Mean		0.78	0.63	0.77	0.76	**0.79**

Table C.6: IoU on MURF compared to DFPR [16] and CubiCasa5K [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

Combined dataset

Table C.7 and Table C.8 contain the accuracy and IoU results on the combined dataset.

		U-Net	U-Net++	U-Net3+	Architecture 1	Architecture 2
Pixel Acc.		0.83	0.80	0.85	0.84	**0.85**
	Wall	0.89	0.84	0.89	0.89	**0.90**
	Glass wall	0.46	0.27	0.57	0.51	**0.54**
	Railing	0.61	0.59	0.60	0.59	**0.62**
Class Acc.	Door	0.70	0.62	0.69	**0.71**	0.69
	Sliding door	0.33	0.04	0.43	0.34	**0.60**
	Window	0.83	0.82	**0.83**	0.83	0.82
	Stairs	0.75	0.77	**0.79**	0.72	0.73
Mean		0.65	0.56	0.69	0.66	**0.70**

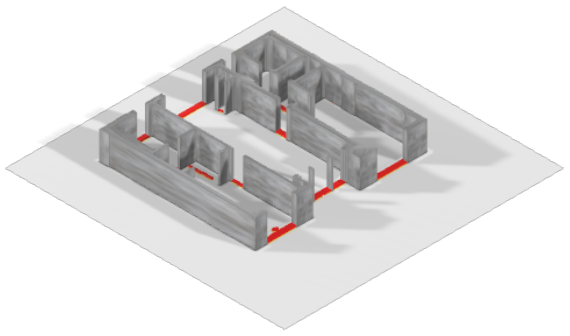
Table C.7: Accuracy on CubiCasa+MURF compared to DFPR [16] and CubiCasa5K [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset.

		U-Net	U-Net++	U-Net3+	Architecture 1	Architecture 2
	Wall	0.78	0.75	**0.79**	0.78	0.79
	Glass wall	0.38	0.18	**0.46**	0.44	0.43
	Railing	0.44	0.38	0.44	0.44	**0.44**
Class IoU	Door	0.55	0.55	**0.57**	0.54	0.56
	Sliding door	0.17	0.02	**0.27**	0.21	0.24
	Window	0.71	0.62	**0.72**	0.71	0.71
	Stairs	0.61	0.60	0.61	0.59	**0.62**
Mean		0.52	0.44	**0.55**	0.53	0.54

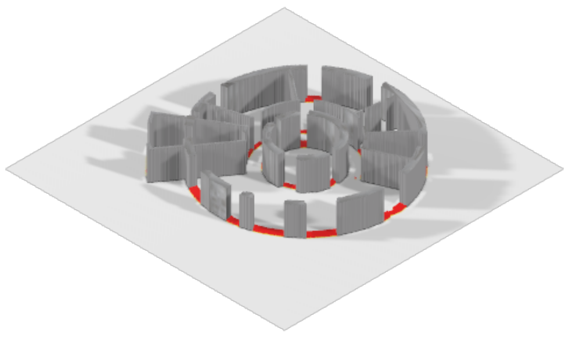
Table C.8: IoU on CubiCasa+MURF compared to DFPR [16] and CubiCasa5K [28]. Subscript $\mathcal{B}$ refers to the model variant only considering boundary and opening elements that was evaluated on the augmented dataset. Highest value marked in bold.

D
Reconstruction Results

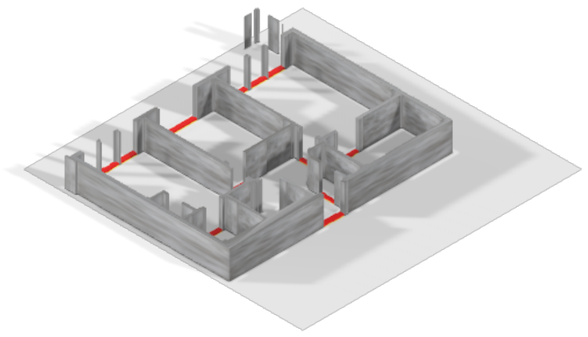
This chapter contains the full-sized 3D models of the qualitative evaluation.



(a)

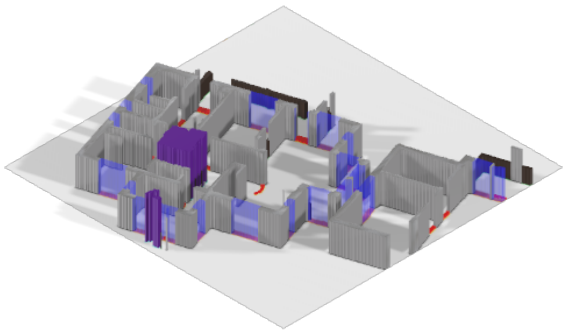


(b)

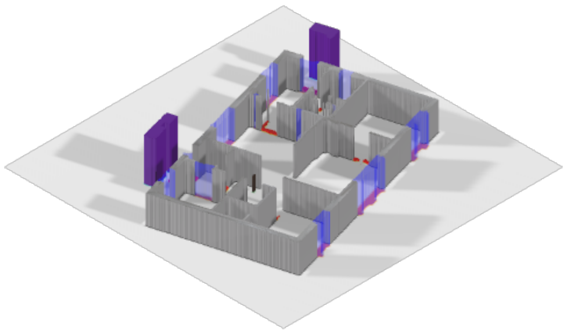


(c)

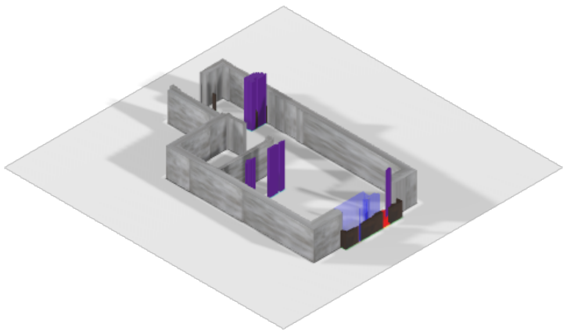
Figure D.1: 3D models from R3D.



(a)

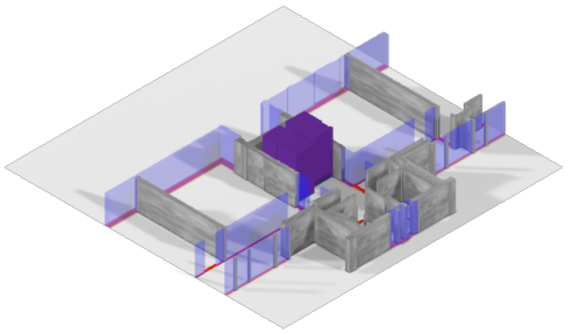


(b)

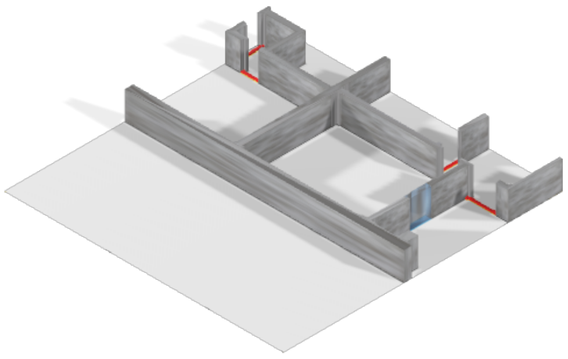


(c)

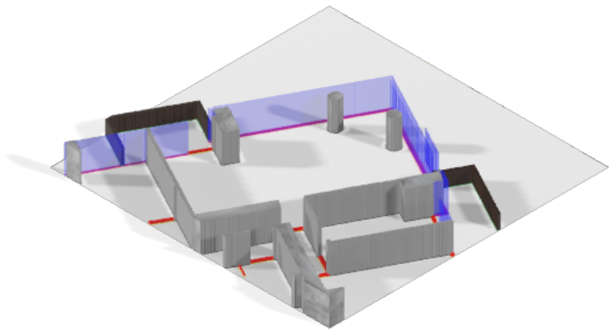
Figure D.2: 3D models from CubiCasa.



(a)



(b)



(c)

Figure D.3: 3D models from MURF.