MSc thesis in Geomatics

Detailed Facade Reconstruction for Manhattan-world Buildings

Linjun Wang

2022

MSc thesis in Geomatics

Detailed Facade Reconstruction for Manhattan-world Buildings

Linjun Wang

June 2022

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

3D geoinformation group
Delft University of Technology

Supervisors: Liangliang Nan
Nail Ibrahimli
Co-reader: Hugo Ledoux

Abstract

3D building models play an important role in many real-world applications. Different models are suitable for different application scenarios based on their levels of detail. LOD3 models with facade details are crucial for many applications, such as virtual reality and urban simulation. Currently, 3D building models with lower LOD are largely available, but the number of LOD3 models is very limited. Most LOD3 reconstruction methods depend on manual operation, which is very time-consuming. How to automatically reconstruct the detailed facade for building models has remained a problem in computer vision. The problem can be seen as an image processing problem, but how to convert the 2D results into 3D smoothly should also be considered.

In this project, we proposed a method to automatically reconstruct the detailed building models based on the Faster R-CNN. The method starts from a set of street view images, and the results are models with facade elements. A 3D point cloud can be extracted from the images using SfM and MVS, and the camera parameters can also be recovered. We take advantage of the high-quality facade images and parse the facades to detect their bounding boxes. The bounding boxes can match pretty well with the rectangular shape of the facade elements. The 2D facade elements can be added to the 3D building model based on the camera parameters. The process is very efficient and automatic. The regularity of the facade elements will be reserved, making the result more convincing. Our method includes four main steps: (1) coarse model reconstruction, (2) facade image selection and rectification, (3) facade element detection and regularization, (4) detailed facade reconstruction.

Experiment results show that our method can produce reliable building models with facade details for many different situations. It can work for both the multi-face building blocks and the street side buildings. Our test shows that the window detection performance is pretty good. The object detection is extremely fast, and the whole pipeline is lightweight and efficient. In theory, the method can also be extended to reconstruct large-scale city models, which means it has broad application prospects.

Acknowledgements

With the completion of this thesis, the time of studying in the Netherlands finally came to an end. The days in the Netherlands are very long in June. I occasionally go out for a bike ride after dinner, traveling through the towns and cities around Delft on the familiar roads. Almost every corner has left traces in my memory. They are indescribable warmth and loneliness that accompanied my alone time in the past two years. I will often compare them to the wind whizzing overhead when I ride my bike, and I will never forget them anyway.

First of all, I would like to thank my first mentor, Liangliang Nan. My laziness often makes the progress of the thesis not smooth, but he never gave up on my guidance. The core idea of this paper is also inspired by him. Without his previous research as the basis, this paper would not have been completed so smoothly. At the same time, I would like to thank my second mentor, Nail, whose patient answers helped me to quickly deepen my understanding of the relevant principles. The two supervisors made very valuable suggestions for the revision of this paper. For six months, on Monday mornings every two weeks, our discussions pull me out of a state of chaos time and time again. If the completion of the entire graduation project is a trip in the fog, then the two teachers must be the light of a searchlight shining in front of me.

The two years in the Netherlands cannot be said to be so colorful. The epidemic in the Netherlands lasted for a year and a half. I completed all the courses and graduation thesis at home. The two studios on Roland Holstlaan and Barbarasteeg carry most of my joys and sorrows. Therefore, in the past two years, I am afraid that it is not as simple as getting a master's degree. It is my first time staying alone for such a long time, and it's the first time I have had a long talk with myself seriously. This kind of gain may not be what everyone generally expects. I didn't gain too many foreign friends, and my conversations with them were mostly limited to the Internet. What I gained is some memories of self-talk, in the constant self-consistency and inconsistency, and my inner peace. It's more like a pity to talk about, but I always think it's something more precious.

Thanks to all the friends I met in Delft, we depend on each other abroad, and we can provide each other with warmth at any time. Thanks to all the Geomatics students, we almost only meet in online meetings, but you guys with bright smiles on the screen will also be part of my fond memory. Thanks to the three Chinese restaurants in Delft. Thanks to everything in the Netherlands that has helped me. Thanks to the warm North Atlantic current, the continuous winter rain or the spring breeze.

Thanks to my parents for providing financial support for this journey of my life, and thanks to my family members. I am full of spring in the WeChat group chats during the holidays, and my room is never deserted. Thank you for the unpredictable destiny, thank you for these two years.

Contents

List of Figures

List of Tables

Acronyms

MVS	Multi View Stereo	1
SfM	Structrue From Motion	1
LOD	Level of Detail	2
RANSAC	random sample consensus	7
UAV	unoccupied aerial vehicle	8
HOG	Histogram of Oriented Gradients	9
RNN	recurrent neural network	10
MRF	Markov Random Field	10
NCC	Normalized Cross Correlation	10
PCA	Principal Component Analysis	14
LMS	Least Median of Squares	15
RPN	Region Proposal Network	20
DFCN	Deep Fully Convolutional Net-work	20
IOU	Intersection Over Union	33

1 Introduction

This chapter starts with a brief introduction to the background and motivation of this graduation project. The social value and scientific contribution of this project will be discussed. Then, the overview of the whole pipeline is introduced. Based on the motivation, the main research questions to be answered in this project are defined. Finally, general information about every chapter in this thesis is presented in the last section.

1.1 Background and motivation

A 3D city model is a representation of urban environment with a 3D geometry of common urban objects and structures [Biljecki et al., 2015]. Visualisation dominated the early use of 3D city models. However, with the development of the technology of 3D modeling and virtual reality, 3D city models are becoming more important in various applications, such as urban planning, utility management, spatial analysis, and 3D cadastre [Kolbe, 2009] (Figure 1.1).

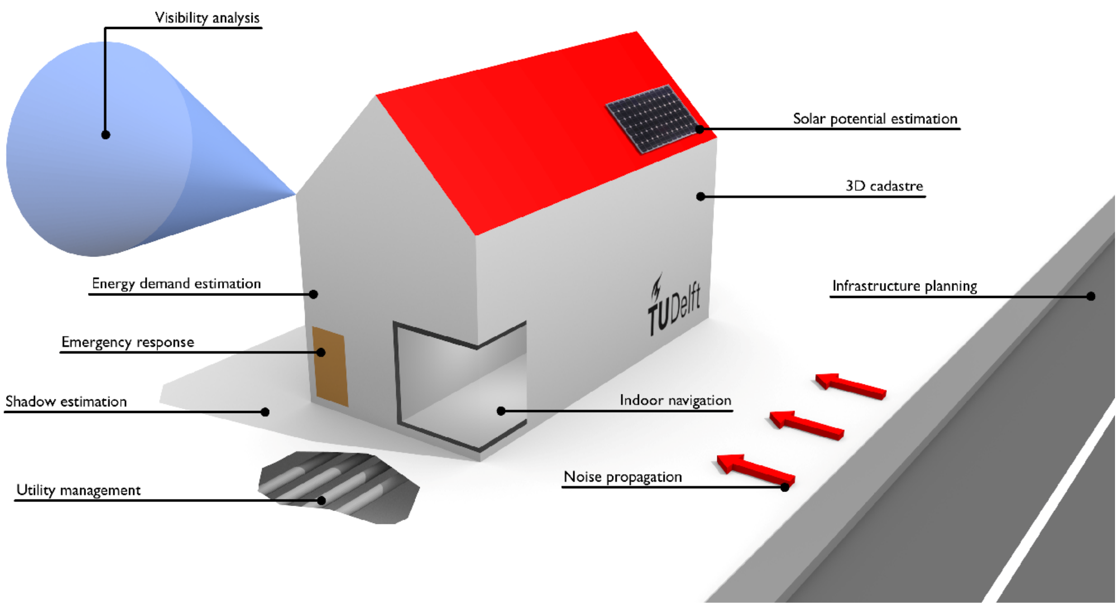


Figure 1.1: 3D city models are applied in many different application domains [Biljecki et al., 2015].

A typical 3D city model can be derived from many different techniques. One common method of obtaining 3D city models is photogrammetry, which will acquire massive point clouds data through Structrue From Motion (SfM) and Multi View Stereo (MVS) from a set of images, and then reconstruct the 3D surface meshes based on the point clouds. Another way is to acquire the point cloud directly from the laser scanning, like LiDAR, and reconstruct the model. The variety of techniques to generate the 3D city models will lead to the diversity of the model characteristics. Different models are suitable for different applications based on their Level of Detail (LOD). The CityGML standard defines five levels of detail (LODs), generally used in 3D city modelling (Figure 1.2):

- **LOD0**: A horizontal polygon representing the footprint.
- **LOD1**: A block model with horizontal and planar roof which is resulting from the extrusion of the building footprint.
- **LOD2**: The generalised roof shape and larger roof superstructures are presented.
- **LOD3**: A detailed architectural model containing windows, doors, chimneys, and other facade details.
- **LOD4**: A detailed model of the building, including indoor features.

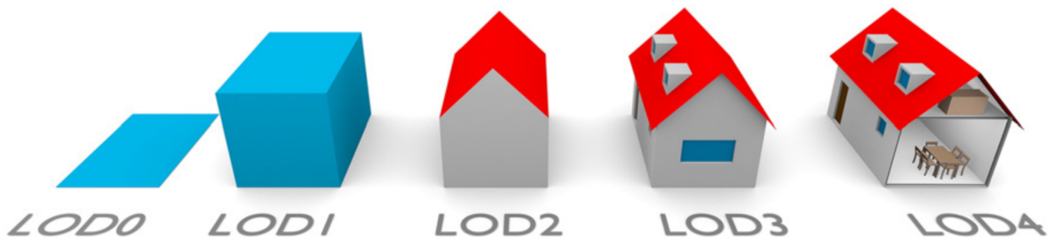


Figure 1.2: Levels of detail [Biljecki et al., 2016].

Currently, 3D city models with lower LOD (LOD1 and LOD2) are largely available, and the number of LOD3 models is very limited. However, LOD3 models with fine details of facade elements (e.g., windows, balconies, doors) are crucial for many advanced real-world applications such as virtual reality and urban simulation. In general, there are two main approaches to generating the LOD3 models [Zhang et al., 2019]. The first is directly reconstruct the LOD3 models from 3D point clouds, such as the O-Snap [Arikan et al., 2013]. However, such methods need many manual efforts to give instructions and correct the result. The second is to add facade details to the LOD2 model to extend it to LOD3, which reconstructs the facade based on the facade texture images or the images of the textured model. However, such reconstruction may have many errors because of the low quality of the texture images. Figure 1.3 shows an example of extending the model to LOD3.

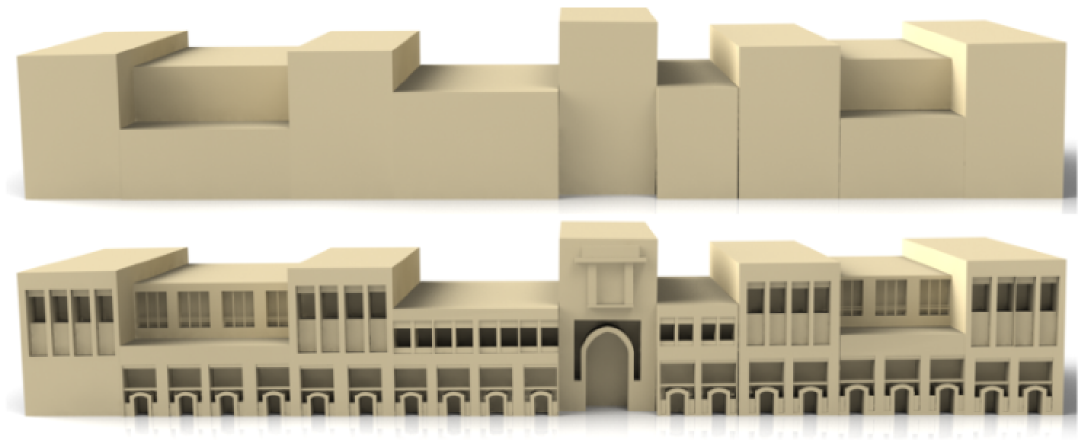


Figure 1.3: Enrich the coarse model with facade details [Nan et al., 2015].

In order to obtain high-quality facade details, we will implement the detection on facade images in this project. Building facade elements detection has become an important topic for many applications in computer vision—for example, the 3D urban scenes and automatic drive [Liu et al., 2020]. The problem with facade element detection is automatically identifying and classifying facade elements, including windows, doors, balconies, air-conditioning, and others. The detection can provide semantic information about the facades, which can also be served as a set of facade templates or the urban environment information. However, the detection is challenging since there will be occlusions and illuminations in the facade images. At the same time, the diversity of facade elements in different buildings and the irregularity of facade elements distribution may also lead to inconsistent detection results (Figure 1.4).

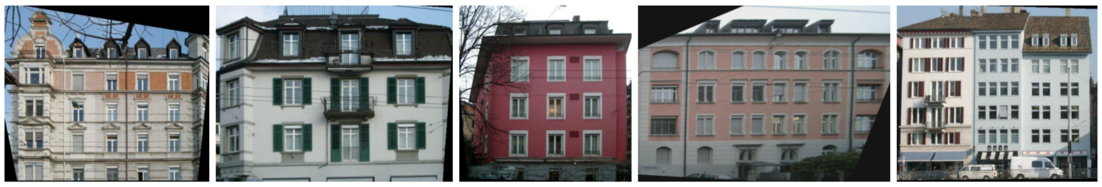


Figure 1.4: Examples of facade images. Data from the CMP facade database [Tyleček and Šára, 2013].

The traditional facade parsing method will segment the facade images at a pixel level, and such results can only provide semantic information, but the geometric information is very limited. When we want to model the facade elements, geometric information like the position and sizes of the elements is as crucial as semantic classification. Object detection algorithms can provide the bounding boxes of specific elements in images, which is suitable for generating geometric information (Figure 1.5). In addition, most facade elements are rectangular so the bounding boxes can be used as a good representation of the facade elements directly.

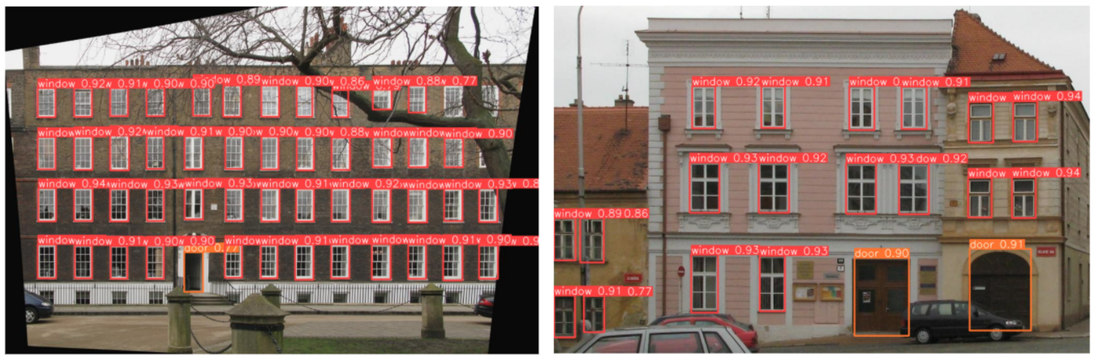


Figure 1.5: Examples of object detection result in facade images.

The goal of this project is to enrich the coarse model of LOD1 or LOD2, which means parsing and adding facade details into the polygon faces of the coarse models. The input is a set of street view images, and the output is a simplified LOD3 building models. We propose a new pipeline which starts from the image acquisition. Reconstruct the coarse building models based on the point clouds from SfM and MVS. Generate the front view images for every facade face of the model. Discover the facade parsing based on the object detection algorithm to make the facade layout regular and match the actual situation as much as possible. The regularization of the layout will be considered an optimization problem. Finally, add the facade elements based on the layout to make the model have a higher LOD.

1.2 Reasearch questions

Based on the background of the problem and the motivations, our main research question is:

How can we bring more facade details into the image-based urban reconstruction pipeline?

As we mentioned in section 1.1, this project aims to discover the method to automatically add the facade details to a model of LOD1 or LOD2. It will be a whole pipeline, including the reconstruction of the coarse model based on the street view images, the automatic generation of the front view images, and the regularization of the facade layout. The challenging part is how to ensure that the final result matches the actual situation as close as possible. So a sub-question also needs to be addressed as follows:

1. *How can we detect facade elements from the facade images?*
2. *How can we ensure the regularity of the facade elements' layout?*

Scope of the thesis

- We assume that all the buildings obey the *Manhattan-world* assumption, so that our approach can be possible to implement. In real-world, many buildings will obey this assumption [Coughlan and Yuille, 1999]. It states that there is a predominance of a triple of mutually orthogonal directions in the scene. This assumption has been used to reveal the regularity for 3D reconstruction methods of buildings.
- The reconstructed coarse model will not contain the roof structure and will only have the wall faces since it's generated based on the street view images. In general, all the coarse models used in this project are LOD1 models, but the proposed method can also be applied to the LOD2 model.
- We will not focus on reconstructing specific 3D facade element models in this project. All the facade elements will be treated as rectangles that can be intruded or extruded on the wall.

1.3 Thesis outline

The main content of the thesis can be divided as five chapters:

Chapter 2 introduces and analyses scientific research related to this graduation project. It contains three parts. The first part is the related researches about the building model reconstruction. The second part talks about researches about facade parsing.

Chapter 3 presents the proposed method of adding facade details to the coarse building model.

Every processing step is described with an example building. The method consists of four components: (1) *coarse model reconstruction*, (2) *facade images selection and rectification*, (3) *facade elements detection and regularization*, (4) *facade elements addition*

Chapter 4 provides the implementation details of the methodology in practice, including the datasets, the programming specifics. The last section provides some solutions for problems in the implementation.

Chapter 5 shows results of the pipeline and evaluations of the results. The first part is about the evalution of the object detection results. The second shows the final reconstruction results, and some analysis are provided based on the results.

Chapter 6 answers the research questions proposed in Chapter 1. The second part is the discussion about the whole pipeline. The contributions of this project are listed. Finally, the future work are discussed in the last section.

2 Related work

In this chapter, some scientific research related to this topic will be introduced and analysed. Since the reconstruction of the coarse model will also be implemented in this project, we first review some work related to the building model reconstruction in Section 2.1. Then the main part of this chapter is about facade modelling methods, in Section 2.2.

2.1 Building model reconstruction

The reconstruction of buildings in the urban environment can be divided into two main categories: single building reconstruction and urban scene reconstruction. For the first category, the polygonal surface reconstruction algorithms are focused on one building or 3D model. Most work are based on point clouds. Arikan et al. [2013] proposed the O-Snap, which is an automatic polygonal reconstruction that can be refined by users interactively. It extracts planar primitives along with their boundary polygons using random sample consensus (RANSAC) and then snaps polygon elements together. The optimization-based snapping algorithm can be participated by the user interaction to fix the polygons. Users can also edit the generated coarse model to add details like windows (see Figure 2.1).

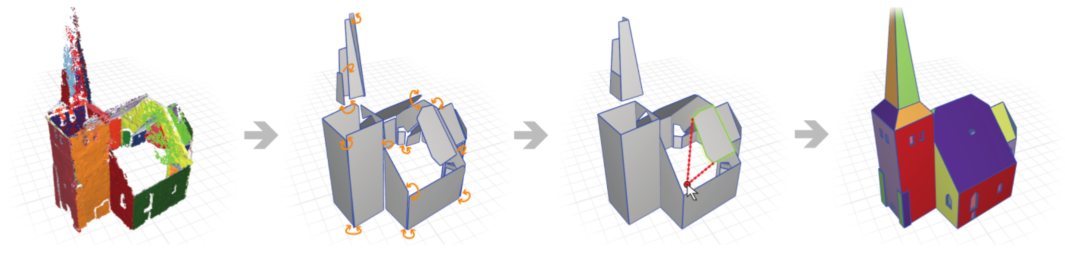


Figure 2.1: Optimization-based snapping for modelling architecture [Arikan et al., 2013]

Nan and Wonka [2017] proposed PolyFit to reconstruct the polygonal surface from SfM point clouds. The algorithm also strat with detecting planar primitives based on RANSAC. Then, the intersection of the planes will produce a set of candidate faces. The face selection problem for the final model is solved as an energy optimization problem. Holzmann et al. [2017] proposed a hybrid method between generic 3D reconstruction and plane-based urban reconstruction. The point cloud is divided into a set of tetrahedra. The tetrahedra set will be labelled with an energy optimization problem to determine which of them needs to be reconstructed. Jonsson [2016] discovered the method to filter and simplify 3D building mesh models, which is suitable for preserving the piecewise structures and sharp features.

As for the urban scene reconstruction, the entire urban scene will be reconstructed in the process. Verdie et al. [2015] introduced an approach to reconstruct 3D urban scenes in the form of LODs, which can generate large scale urban scenes with meaningful LoDs while being robust and scalable.

Li et al. [2016] presented an automatic reconstruction method from unoccupied aerial vehicle (UAV) images. They segment the point cloud and filter the building cluster. A set of footprints are extracted based on the building point cloud, which is regularized with the Douglas-Peucker algorithm. The whole urban scene of buildings can be reconstructed by extruding the footprints. Kelly et al. [2017] apply three different data sources to reconstruct the urban scene: the building surface mesh, the street-level facade image, and the buildings' footprints. They use this data to generate a set of planar segments representing the facade, a group of edges representing the ground plane and a set of line segments representing the building frame. They finally combine all the elements using optimization (see Figure 2.2).

Many of these methods depend on multiple sources of information that are not always available(e.g., point clouds, aerial imagery). Meanwhile, most reconstruction methods don't consider the fine details of the buildings, but the results of such methods can be served as the basic step for the facade details reconstruction.

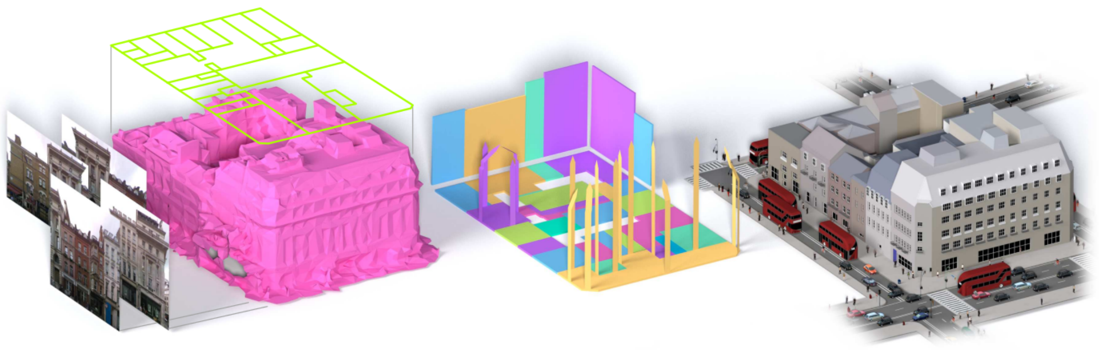


Figure 2.2: Large-scale structured urban reconstruction [Kelly et al., 2017].

2.2 Detailed facade reconstruction

Photogrammetry-based reconstruction

The photogrammetry-based reconstruction of the facade relies on image-based techniques, which obtain dense point clouds and textured models from a set of photos. Xiao et al. [2008] discovered the depth enhanced facade modelling based on the SfM. They assume that the facade elements are essentially 2.5D rectangular elements on the top of the facades. They decomposed the facades into patches, and then augmented a depth value from SfM for each patch (see Figure 2.3). Xiao et al. [2009] designed an automatic method to reconstruct building blocks with facade details from a set of street view images. They segment the building images at pixel level to distinguish the facade and the other areas. They also proposed a method to separate the building point cloud into different parts based on the building blocks. For every building block, they used a patch-based approach to model the facade based on the point cloud directly, which also applies the depth map in the input image space. Wu et al. [2012] proposed a schematic algorithm for reconstruction from sparse point clouds. The result representation can be extremely concise, and a displacement map can be put on top of the schematic surface, to make it possible to recover fine details like facade elements.

Interactive facade reconstruction

Buildings usually consist of an assembly of basic primitive shapes. Most of the works focus on the re-

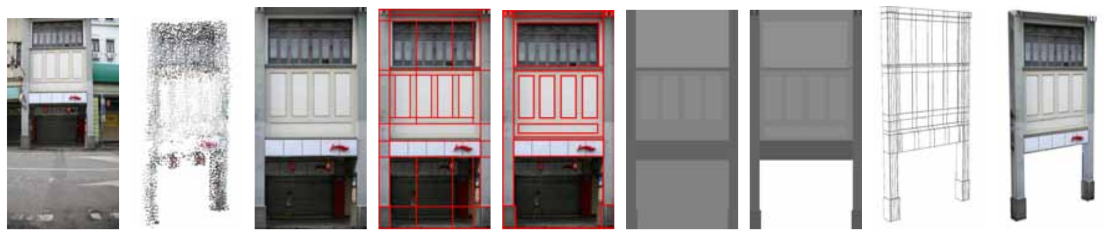


Figure 2.3: Image-based facade modeling[Xiao et al., 2008]

construction of the 3D primitives interactively. Nan et al. [2010] proposed an interactive tool called SmartBoxes to quickly assemble detailed 3D primitives balancing between data fitting and structural regularity based on the regularity of facades. Arikan et al. [2013] proposed an optimization-based interactive tool that can reconstruct relatively detailed building models from sparse point clouds. Lin et al. [2013] segment the urban scenes into different categories using supervised learning and reconstruct 3D models using prior knowledge. This method can create building models that are merely approximated with a small number of textured planes. Nan et al. [2015] reconstruct building details by automatically assembling 3D templates on coarse textured building models. They first discover the method to optimize the coarse model. Then, use the Histogram of Oriented Gradients (HOG) feature to detect the positions of the facade elements based on the facade texture images. The detection method is optimization-based and can balance the image matching and structural regularity. Finally, the detailed 3D templates are added to the coarse model based on their positions. Unlike the works above, this method focus on including geometric details into the coarse models based on the template assembly (see Figure 2.4).

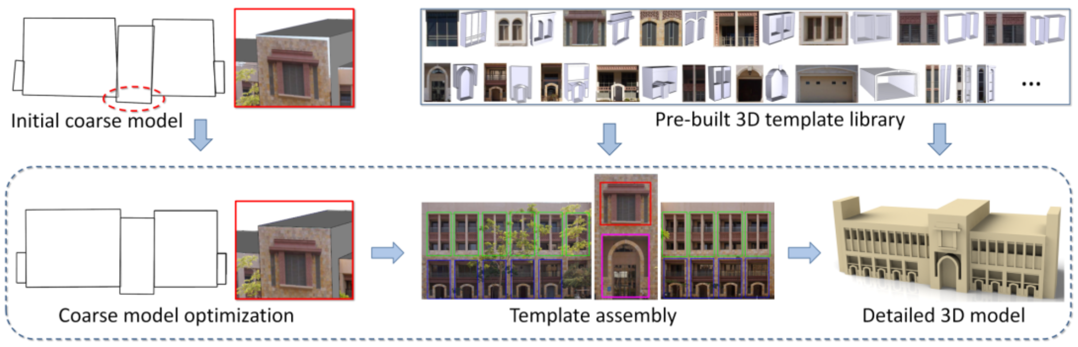


Figure 2.4: Template assembly for detailed urban reconstruction [Nan et al., 2015]

2.3 Facade element detection and layout regularization

Facade parsing

Facade parsing is an important task in computer vision. The goal of facade parsing is to segment the facade images into regions for different facade elements. Facade parsing can be seen as a part of the facade reconstruction, but many work also treat it as a independent problem.

One typical form of facade parsing is classifying each pixel in facade images into a specific semantic category. Deep learning has shown its power in facade parsing tasks. There is a large body of work about the facade image segmentation problems based on deep learning methods. Cohen et al. [2014]

used a dynamic programming algorithm with extentsions to find labels of facade. The global optimality certificates will be obtained if the individual algorithms remain independent. Martinović et al. [2012] proposed a three-layered approach to parse facade. They merged the pixel-level results from recurrent neural network (RNN) and the object detection together using a 2D Markov Random Field (MRF). This method can apply information about semantic segmentation and facade objects at the same time. They also proposed some layout rules for different facade types to constraint the distribution of the facade elements. Schmitz and Mayer [2016] applied a fully convolutional network approach on facade image patches. They use a relatively small dataset to train the network based on the transfer learning. Liu et al. [2020] proposed a novel symmetric loss for the deep convolutional neural network and tested the method on two facade datasets. The result of deep neural networks is combined with the bounding boxes generated by the Reginal Proposal Network. The method can use both the power of deep learning and the building facade structures (see Figure 2.5). However, the pixel-based facade parsing can only segment the facade image at the pixel-level, which is not suitable to generate the layout of the facade elements.

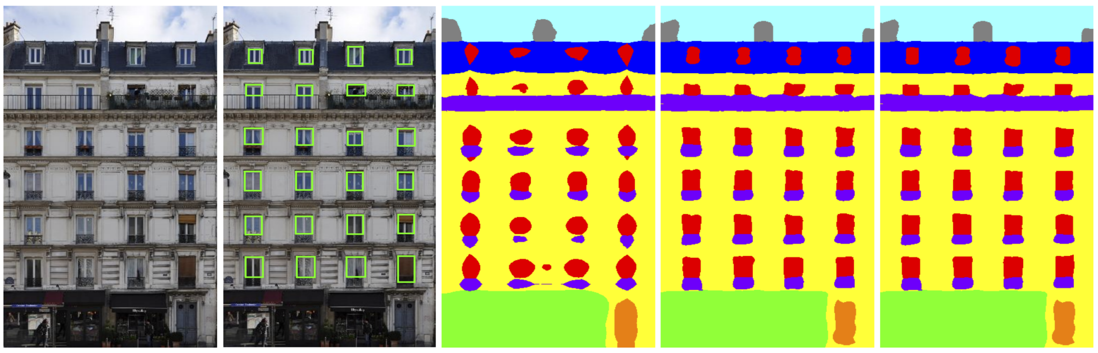


Figure 2.5: DeepFacade [Liu et al., 2020].

Another type of facade parsing is to subdivide the 2D facade images into different rectangle regions. Müller et al. [2007] model the facade grammars based on image analysis to derive a hierarchical facade subdivision. The facade images can be subdivided into floors and tiles using mutual information. A shape tree can be constructed based on element recognition. Finally, the shape grammar rule can be extracted from the shape tree for the 3D modelling (see Figure 2.7). Alhalawani et al. [2013] proposed a semi-automatic framework to recover a factored facade representation based on the repetition patterns and the deformation parameters. They extract the Canny edges of the facade images first. Then, identify potential candidate elements using Normalized Cross Correlation (NCC) based on the user-provided window template. They use an optimization-based method to select the candidate elements to make the layout regular and match the pre-provided template (see Figure 2.8). Koziński et al. [2015] proposed a method to parse facade using a linear binary programming, which can approximate global optimal segmentation without grammar samplings. The method is also inspired by the hierarchical image subdivision. Object detection can detect bounding boxes of the facade elements, so it has also been used a lot to get the initial layout of the facade elements in many approaches. Zhang et al. [2019] detected bounding boxes of windows and doors in facade texture images using Mask R-CNN to model the facade elements on the coarse model and generate LOD3 CityGML models. They changed the Mask R-CNN model framework to make it more suitable for the facade images. This method is close to what we have done in this project, however, they did not use the multi-view images and the facade images used are of low quality (Figure 2.6).

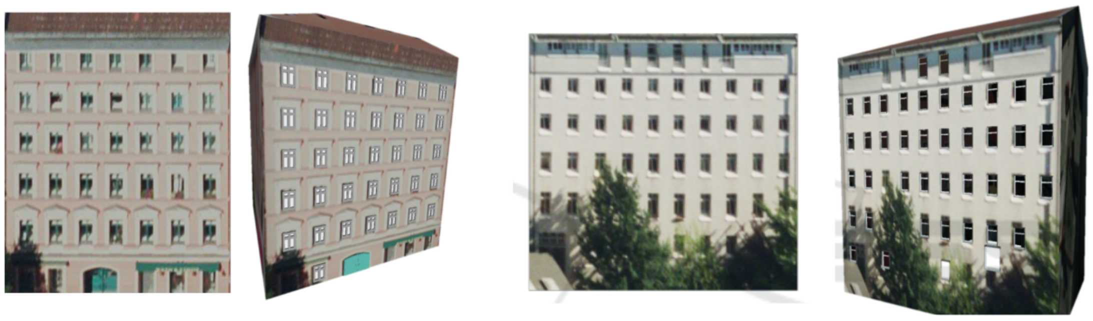


Figure 2.6: A data-driven approach for adding facade details to textured LoD2 CityGML models [Zhang et al., 2019]

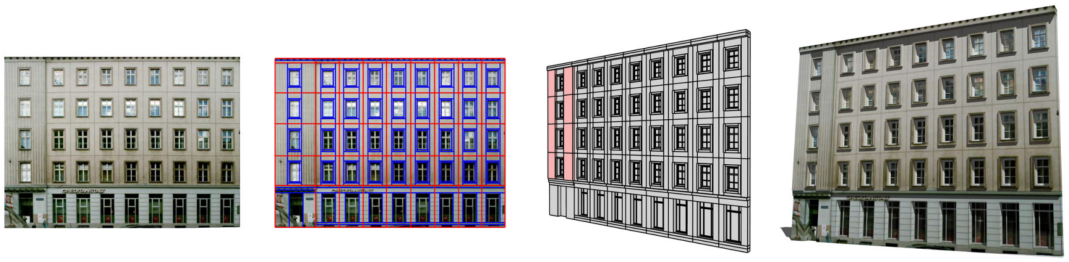


Figure 2.7: Image-based procedural modeling of facades [Müller et al., 2007]

Facade layout enhancement

Urban facades often contain variations because of allowed deformations of repeated elements. So the layout enhancement is also important when modeling the facade. In most facade subdivision tasks, the layout regularity is considered as a crucial concept. We also reviewed some in-depth study on the layout enhancement.

The layout enhancement problem can also be called regularization and beautification, which has been studied in different areas, e.g., object alignment, 3D shape symmetrization. Nan et al. [2011] model conjoining Gestalt rules for grouping and summarization of facade elements. Huang et al. [2014] combine patch-based image completion with translational symmetry detection to fill in the missing part of a planar structure. Jiang et al. [2016] use an optimization-based method to automat-

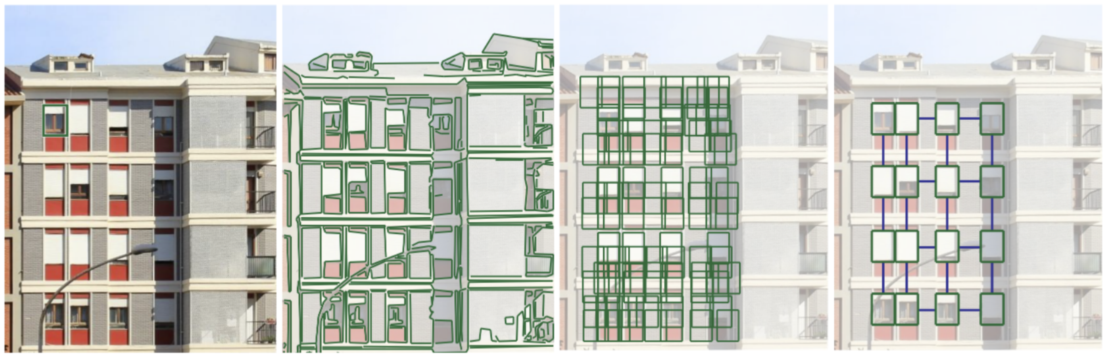


Figure 2.8: Interactive facades analysis and synthesis of semi-regular facades [Alhalawani et al., 2013]

ically detect constraints in the layout. And then formulate the layout regularization as a quadratic programming problem similar to previous work. The work can be used in many applications apart from facade layout enhancement, such as the slide design and poster design.

3 Methodology

3.1 Overview

In this chapter, we will provide details about the methodology adopted for this graduation project. An overview of the methodology is provided in Figure 3.1. Firstly, the data should be prepared and pre-processed to be applied as input for the reconstruction. The methodology can be divided into four steps: (1) coarse model reconstruction, (2) facade images selection and rectification, (3) facade elements detection and regularization, (4) facade elements addition. Every sections in this chapter will discuss each step. An example building will be selected to present the whole pipeline.

3.2 Geometry and motion recovery from images

The first step of the method is to prepare the data that will be used to apply the reconstruction. The input of our pipeline is a set of street view images. A 3D point cloud for these images can be extracted using SfM and MVS. At the same time, the camera parameters of every image can be recovered from SfM. Since we will reconstruct the coarse model of the building facade based on the point cloud, we need to manually remove the planes and outliers outside the facades roughly.

3.3 Coarse model reconstruction

In most situations, it is hard to take photos around the whole building during data collection, which means we will only get a few facades of the building from the images. We cannot obtain information about the roof from the street view images, so the point clouds will only contain walls of the building. As a result, we consider to reconstruct the facade faces first. Then, complete the building blocks as closed models based on the reconstructed faces.

Since we consider all the buildings as Manhattan-world buildings, all the facades should be rectangles. In this way, orthogonality and parallelism of the building structure can be enforced in the coarse model. There are two reconstructions situations: one-face and multi-face situations. It is relatively simple to find a rectangle representing the facade based on the point cloud for the one-face situation. For the multi-face situation, the main idea is to find the footprints of the facades and then extrude them as rectangles. All the points belonging to every facade will be projected as lines to a plane representing the ground, and then we can snap all the lines as a graph to reconstruct the footprint. The ground plane should not be considered as the horizontal plane since the point clouds extracted from SfM and MVS may be placed in 3D space at an arbitrary angle. Both the situations will start from the initialization, which contains plane fitting and outlier removal.

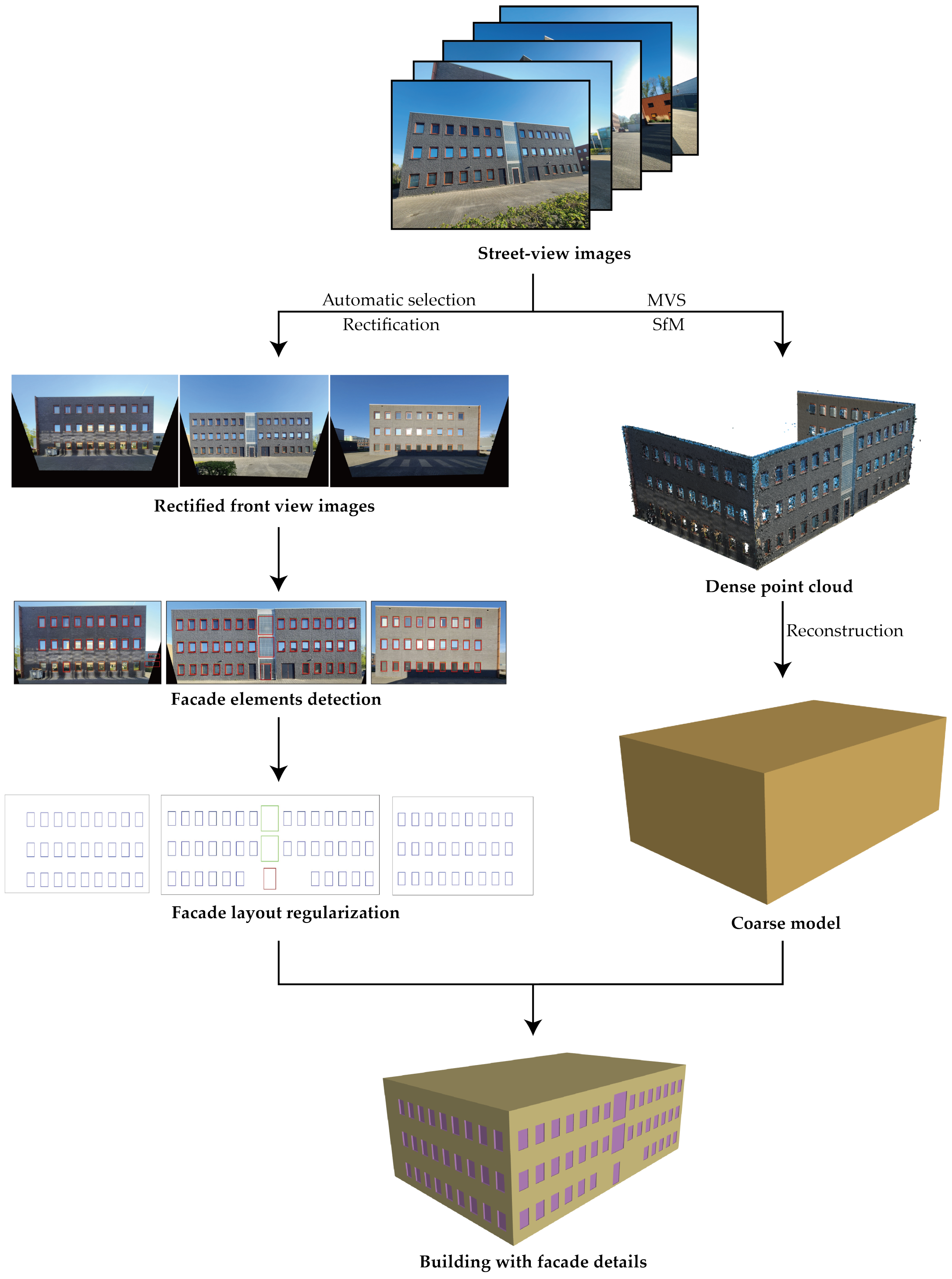

Street-view images
Automatic selection
Rectification
MVS
SfM
Rectified front view images
Dense point cloud
Reconstruction
Facade elements detection
Coarse model
Facade layout regularization
Building with facade details

Figure 3.1: An overview of the whole pipeline.

3.3.1 Initialization

In this step, we use a local RANSAC-based method [Schnabel et al., 2007] to segment the input point cloud into subsets, which contain the different sets of points lying approximately on a plane and a set of unclassified points. Before the RANSAC stage, a local Principal Component Analysis (PCA) [Ian T. Jolliffe, 2002] with fixed size neighbourhoods should be applied to approximate 3D normal vectors from the point positions. After the segmentation, a statistical outlier removal algorithm needs to be applied to every point segment. This algorithm can remove points that are further away from their neighbours compared to the average for the point cloud, which is suitable if we want to remain the main structure of the facades.

After obtaining different point segments of facades, we can fit planes on them. We apply a Least Median of Squares (LMS) estimator[Rousseeuw and Leroy, 1987] to find normal vectors of the facade planes, which can fit a plane to points that contain no more than 50% of outliers. Then, we can project all the corresponding points onto the plane for every fitted plane.

3.3.2 One-face situation

When the model has only one facade, we can simply convert the 3D coordinates of the points on the same plane to 2D. Then, the problem can be simplified as a 2D problem. In order to polygonize the on-plane points, we calculate the minimum area bounding rectangle of the points. Since we assume that every facade should be rectangle and we removed the outliers, this method can be very efficient and accurate for most facades. Figure 3.2 shows the pipeline for this situation.

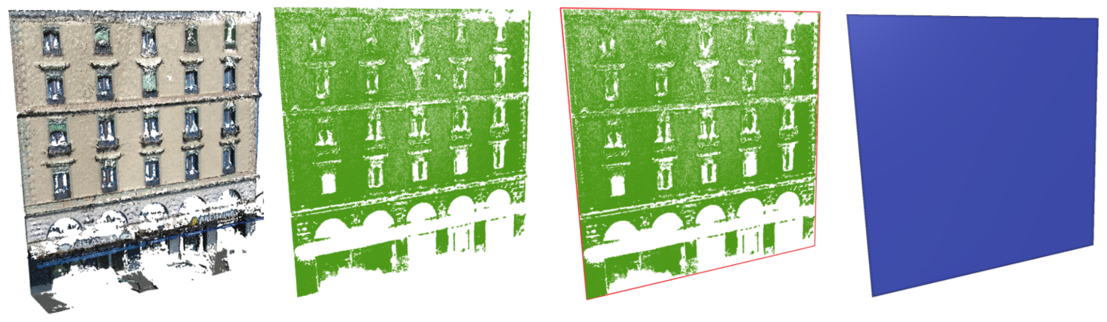


Figure 3.2: The pipeline of reconstructing one facade face. The initialization step is included (left, middle left). A 2D minimum area bounding box can be calculated for the projected points. The bounding box in 3D is drawn in red (middle right). The final facade polygon can fit the facade point cloud pretty well (right).

3.3.3 Multi-face situation

When the model has multiple faces, the main idea of the reconstruction is to extrude every edge of the building footprint as rectangles. Since we do not have information about the camera positions, we cannot do geo-registration for the model, so the gravity and horizontal direction of the model are not parallel to the axis. As a result, we cannot just project the points of every facade plane to the x-y plane to get the footprint. In this step, we apply a method to find the bottom plane of the building, and then we can project the points and reconstruct the footprint. Figure 3.3 shows an overview of this situation.

3.3.3.1 Find the bottom plane

Since the gravity and horizontal direction of the point cloud may not parallel to the axis, we should determine the bottom plane of the building in our experiments. First, every intersection line be-

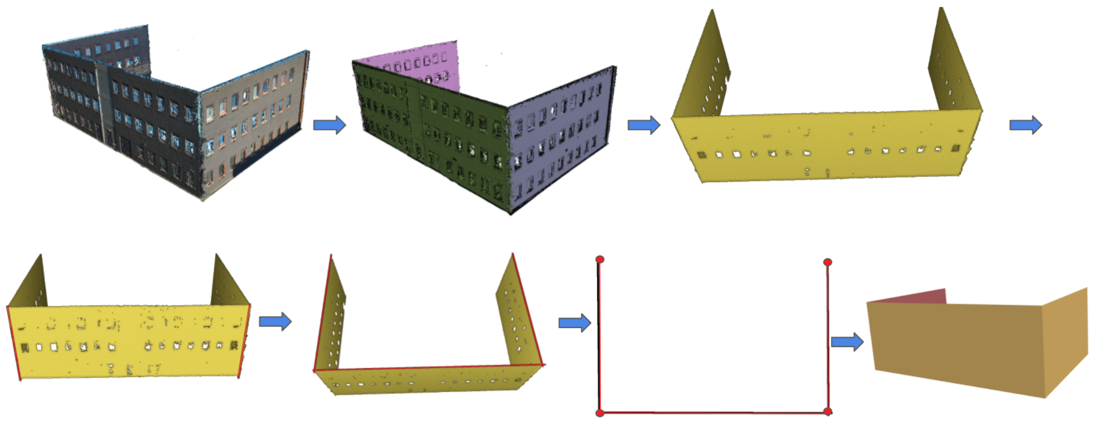


Figure 3.3: The pipeline of reconstructing connected multiple faces.

tween adjacent facades needs to be calculated, and this can be done based on every plane's normal vector. Then, for every intersection line, we can project all the points belonging to the adjacent planes to the line. The two endpoints of these lines can be simply found using the point coordinates. The two endpoints of the projected points will determine the length of the intersection line, and we choose the longest one as the normal vector of the bottom plane. The point with the lowest z coordinate should be the point on the bottom plane. We can determine the final bottom plane with a normal vector and a point on the plane. The length of the selected line should be the height of the building, which also means how much should we extrude from the footprint.

3.3.3.2 Construct the footprint graph

We project points belonging to every face to the bottom plane, and we will get a set of lines made of points. Then, we can find the two endpoints of different lines, determining every line segment, and it will result in a soup of unconnected line segments $L = \{l_1, l_2 ... l_n\}$. Every line segment is combined with two vertices. In order to construct the footprint graph based on the line soup, we need to find the intersection node between adjacent lines and make a linked list of the nodes. Figure 3.4 shows an illustration of this step.

First, for every vertex in each line segment, we compute the distance between them with the other vertices belonging to other ling segments. We need to record the corresponding line pairs for every calculated distance value. Then, for every vertex, we can find the point closest to it with the smallest distance value. If the value is smaller than a threshold, then we consider the two line segments corresponding to these two vertices to be adjacent. In our experiment, the threshold should be set as half of the length of the shortest line segment.

After getting the line segment pairs, the corresponding intersection vertex can be updated. If a vertex cannot be paired successfully, we know it is an endpoint for this unclosed footprint. All the intersection vertices and endpoints will be the nodes that should be connected in the footprint graph. We store all the node points in a list. If the footprint is unclosed, we will start from one endpoint and find the other node that should connect to it based on the relationships between vertices and line segments. If the footprint is closed, we can start from an arbitrary node in the list. Finally, the linked graph of nodes can be constructed.

The graph's direction is not determined, so we need to make it counterclockwise. We applied a formula to measure the winding of the graph, which is inspired by the *Green's Theorem*. Given the vertex list in the graph, we know the 2D coordinate (x_n, y_n) for the vertex v_n. The following metric can be calculated:

$$\sum_{n=1}^{N} (x_{n+1} - x_n)(y_{n+1} + y_n) \tag{3.1}$$

The graph has a clockwise winding if the value in formula 3.1 is positive and a counterclockwise winding otherwise.

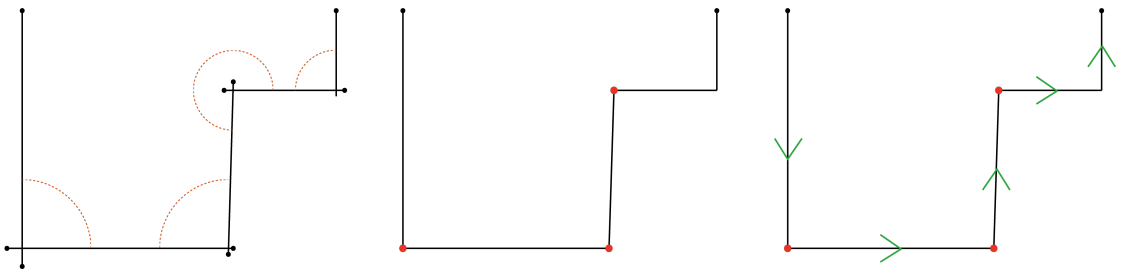


Figure 3.4: An example of the footprint graph construction. We start from the initial soup of line segment, and then find the adjacent pairs of the lines (left). The intersection node can be found for every line sement pair (middle). After connecting all the nodes, the graph can be constructed (right).

3.4 Facade image selection and rectification

The image used for the object detection should be chosen from all the input street view images automatically. And every facade face should correspond to a facade image which is chosen as the best image from the data. The best image means it contains the complete building facade and requires minimal transformation when rectified to the orthogonal frontal view image. So the best image should ideally be taken exactly in front of the facade's centre and include the whole facade face. The rectification is a crucial step for accurate facade elements detection. Unlike conventional methods that use image information, we used the camera parameters and the information of the 3D facade plane for the rectification.

3.4.1 Automatic image selection

The image selection is a crucial step in which we will connect the coarse model with the images. In other words, we will pair every face with the corresponding facade image to correctly add facade details to faces. We need some geometric elements to calculate the relationship between cameras and the facade face. The normal vector of the facade plane is written as v_0. For every facade, positions for a camera to take photos with an orthogonal frontal view should be located on a plane. This plane should be orthogonal to the facade face, parallel to the vertical edge of the facade, and passes the centroid of the facade face. We call it the *central orthogonal plane*. The normal vector of the *central orthogonal plane* should be a vector parallel to the horizontal edge of the plane, and we call it v_p. At the same time, we can make a vector which points from the facade face centroid to the camera centre, and this vector can be called v_c. Another vector we need is the forward vector of every camera v_f. Figure 3.5 illustrate the different vectors.

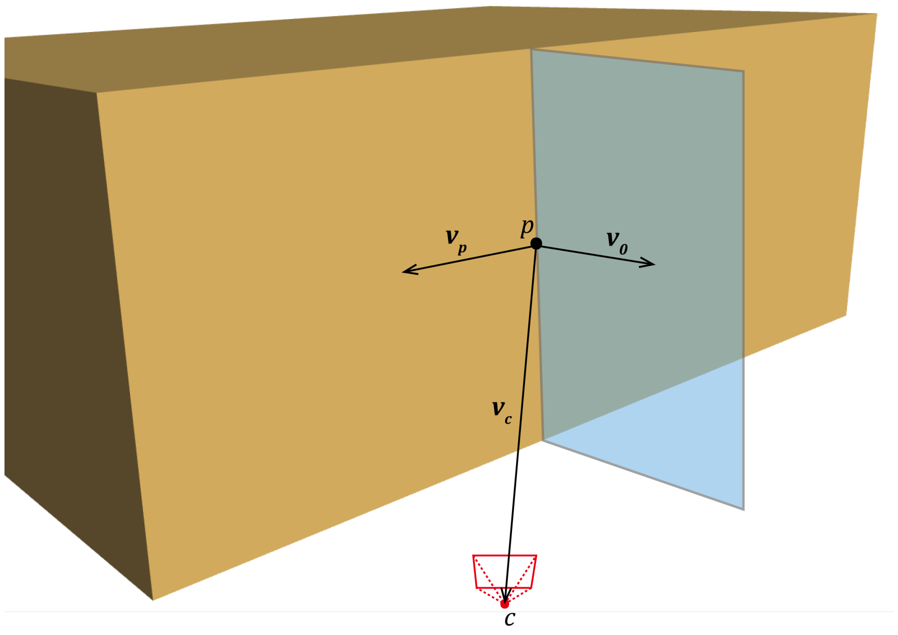


Figure 3.5: The vectors defined to calculate the relationship between cameras and the facade face. The blue plane is the central orthogonal plane of the facade face. p is the centroid of the face and c is the camera center.

We can also project the facade face to the image plane based on the camera parameters. The projected polygon can be written as $\boldsymbol{P}_f$, and the image rectangle is $\boldsymbol{P}_i$. Figure 3.6 illustrate the relationship between these two polygons. Based on these, we can determine three indicators to select the best facade image for every face:

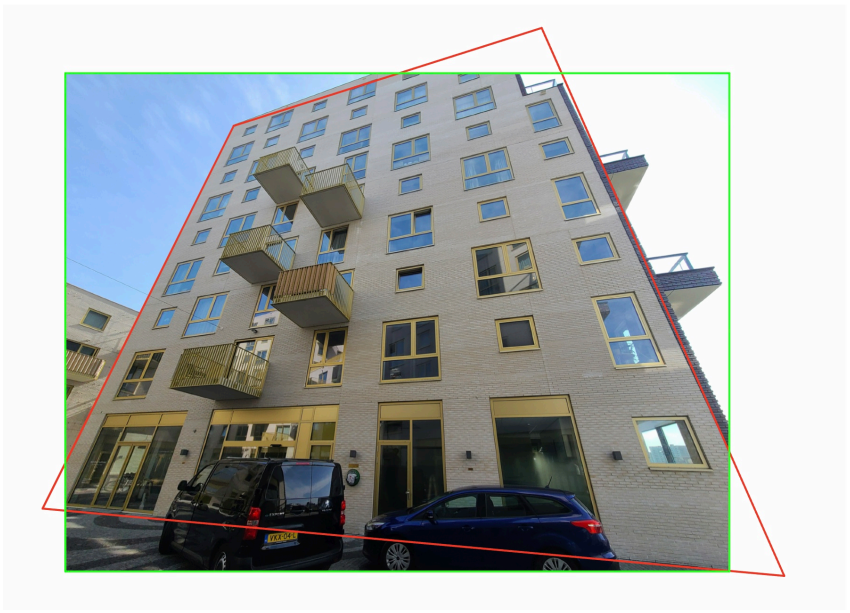


Figure 3.6: The projected facade polygon $\boldsymbol{P}_f$ (red) and the image rectangle $\boldsymbol{P}_i$ (green).

1) The angle θ_0 between $\boldsymbol{v}_f$ and $\boldsymbol{v}_0$, which can be used to identify if the camera is located in front of the facade. When the camera centre is in front of the facade, θ_0 should larger than 90°, and the dot product between $\boldsymbol{v}_f$ and $\boldsymbol{v}_0$ should smaller than 0.
2) The percentage of the intersection area between the projected facade polygon and the image bounding box to the facade polygon: $\boldsymbol{p}_{intersect} = \boldsymbol{area}(\boldsymbol{P}_i \bigcap \boldsymbol{P}_f) / \boldsymbol{area}(\boldsymbol{P}_f)$, which can be used to identify the completeness of the facade in the image.
3) The angle θ_1 between $\boldsymbol{v}_c$ and $\boldsymbol{v}_p$, which can be used to identify how close the camera is to the central orthogonal plane. When the camera centre is perfectly locate on the central orthogonal plane, θ_1 should equal to 90°, and the dot product between $\boldsymbol{v}_c$ and $\boldsymbol{v}_p$ should equal to 0.

The first two indicators can be used as two constraints when selecting the best image. For a given image, if θ_0 is smaller than 90° and the $\boldsymbol{p}_{intersect}$ is larger than 0, it's impossible for this image to be selected as the best image. For images that meet the two constraints, we can compare their θ_1 value. The image with the smallest θ_1 value should be selected as the best image.

3.4.2 Facade image rectification

After selecting the best facade image for every face, we need to rectify the image to make the facade face and facade elements rectangles, which is crucial for the next object detection step. It is helpful to make sure all the facade elements in the image are rectangles and maintain the correct aspect ratio. The rectification is based on the perspective transformation. Perspective transformation can be represented as the transformation of an arbitrary quadrangle(a system of four points) into another one. Figure 3.7 shows an example of the perspective transformation for a facade image. We need to specify the coordinates of quadrangle vertices in the source image and the coordinates of the corresponding quadrangle vertices in the destination image. The destination rectangle's aspect ratio should also be provided based on the facade face. Then the transformation matrix can be calculated according to the coordinates correspondence.

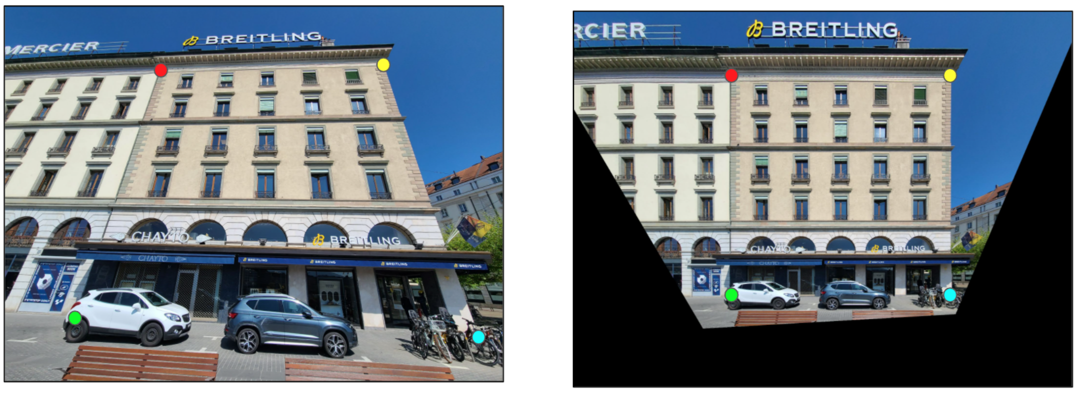


Figure 3.7: A facade image before(left) and after(right) the perspective transformation. The four circles represent the four corners of the facade face.

We have the camera parameters for every image, so the projected four corners of the facade face can be determined on the image. However, the four corners may locate outside the image in some cases. In order to make sure the corners of the facade locate inside the image, we add an offset for the image pixels, which means we enlarge the image to include all the four corners based on their projected coordinates.

3.5 Facade element detection and regularization

We assume there are three types of facade elements: window, balcony and door. All of them will be treated as rectangles when modelling. Treating them all as rectangles simplifies the problem and simultaneously facilitates the extraction of their location information and other geometric information. The rectangular facade elements can present a relatively accurate 3D structure during modelling through simple extrusion/intrusion operations. The traditional facade parsing method may only provide us with pixel-level results. Fitting rectangles in pixel-level segmentation results may cause more errors than the actual situation. The object detection method has been applied to many object detection and segmentation tasks and provides the position of a bounding box for each detected object in an image and matches the object to a class. In theory, the bounding boxes can match pretty well with the rectangular shape of most facade elements. In this research, we apply the Faster R-CNN object detection architecture to detect the bounding boxes of the facade elements. After the detection, we proposed a method that automatically regularizes the detected bounding boxes.

3.5.1 Facade element detection

We apply Faster R-CNN as the object detector in this project. R-CNN algorithm stands for the Region-based Convolutional Neural Networks. The original R-CNN was introduced in [Girshick et al., 2014], which applies a Selective Search Algorithm to extract the candidate region proposals. Fast R-CNN is proposed in [Girshick, 2015], which improves the R-CNN by introducing the Region of Interest pooling technique. Generation of region proposals through selective search in R-CNN and Fast R-CNN is time-consuming due to the calculation of similarity features among all input image regions. Faster R-CNN [Ren et al., 2015] eliminate the need for the selective search and instead lets the network learn the region proposals. It does so by introducing a Region Proposal Network (RPN). Features from the convolution will be shared with the RPN and the RPN can tell the network where to look. The output of an RPN is a set of rectangular object proposals, and it will be done with a fully convolutional network. As shown in the figure, Faster R-CNN consists of Deep Fully Convolutional Net-work (DFCN), RPN, ROI pooling, Fully Connected networks, Bounding Box Regressor and Classifier.

The training details of the Faster R-CNN in this project are provided in Section 4.2.2.

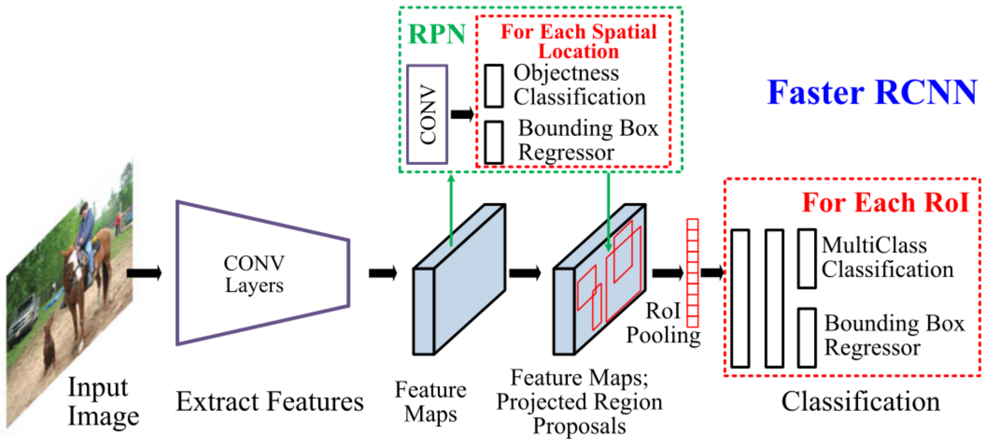


Figure 3.8: High-level framework of Faster R-CNN [Liu et al., 2018]

3.5.2 Facade element regularization

The detected facade element bounding boxes are not well aligned. However, the windows on the same wall normally have the same size for the same row or column and are well aligned in horizontal and vertical directions in real world. We hope to regularize the layout of the windows, doors and balconies to make the facade modelling result closer to the real situation after detecting the facade elements.

Given the initial result of the object detection, the layout consists of the elements bounding boxes, so the layout can be written as $\boldsymbol{L} = \{b_1, ...b_n\}$. Every box b_i is defined by its left corner coordinate(x_i, y_i) and the size (w_i, h_i) (Figure 3.9). The following calculations will all be based on these four attributes of the bounding boxes.

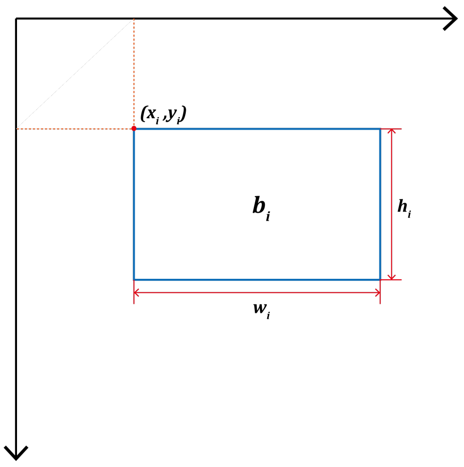


Figure 3.9: A facade element can be seen as a box.

For most facades, the layout of windows is regular or semi-regular, which means there should always be windows distributed at the same level and same column, and windows of different types should have the same size. The regularity of the facade layout can be expressed in a series of constraints. The constraint selection method is inspired by [Jiang et al., 2016]. We use two types of constraints, alignment and same-size constraints, to ensure the final layout is regular. Our final goal is to regularize the layout based on these constraints. The regularization can be done as an optimization problem to minimize the box location and size changes.

3.5.2.1 Constraint groups

Figure 3.10 illustrates the constraints used in the regularization. The detected boxes will be divided into different constraint groups. In order to accurately select the constraint groups, we use one threshold σ to control the size of the groups for both horizontal and vertical directions. The threshold value will multiply the average size of the elements to decide a distance value. For the vertical direction, it will multiply the width and height for the horizontal direction.

Horizontal and vertical alignment

The alignment constraint can guarantee that the same row or column boxes align with the same horizontal or vertical line. For example, the horizontal and vertical alignment between two boxes b_i and b_j can be formulated as $y_i/2 - y_j/2 = 0$ and $x_i/2 - x_j/2 = 0$.

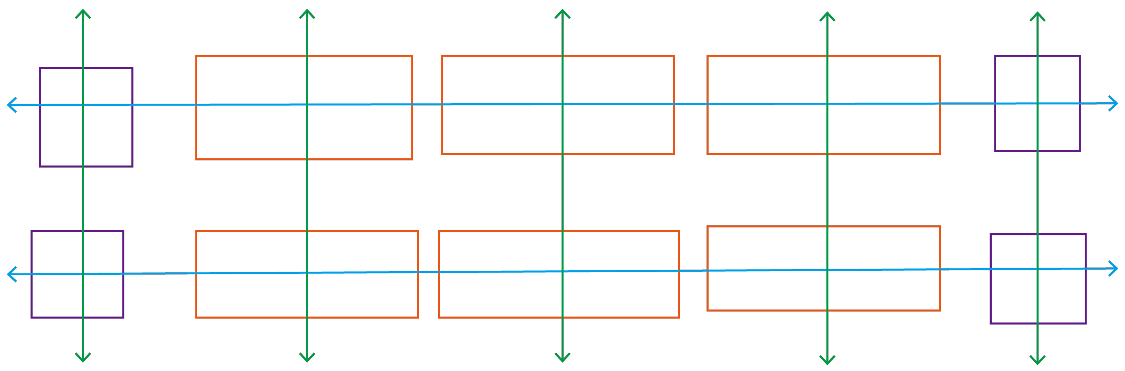


Figure 3.10: Constraints for the facade element layout. Boxes along the blue line should be horizontal aligned. Boxes along the green line should be vertical aligned. Boxes with the same color should have the same size.

To determine the same-alignment groups in the input layout, we must divide them based on their coordinates. For example, if we want to get the vertical alignment groups, we will sort all the boxes in the initial layout according to their centroid's x value (middle-X value). Then we can calculate the difference of the middle-X value of every adjacent pair of the boxes in this sorted list. If the difference is smaller than $\sigma_v(\sigma \times avg(width))$, this pair of boxes are belonging to the same vertical alignment group. This method can also be used to determine the horizontal and vertical grid lines if the layout of windows is regular.

Same size

A clustering method will be used as an unsupervised classification for the different types of windows. Since the unstable prediction result of the object detector, it is difficult for us to classify each box into the correct group accurately based on size without knowing the number of types. We choose to specify the number of window types before the clustering and apply the k-means method. The input variables for the k-means are the width and height of every box. The same-size constraints can be formulated as $w_i - w_j = 0$ and $h_i - h_j = 0$.

3.5.2.2 Fix the initial layout

Due to the variations of the facade elements and the limitation of the facade image quality, we may not detect all the facade elements. However, we can fix the detection result based on the regularity of the facade. In reality, most facade layouts are regular grids, and we can infer the structure of the grid based on the part of the layout. When most of the facade elements are located correctly, and we can observe a regular distribution of the layout, then we can fix the initial detection to make it a regular grid. Figure 3.11 illustrates the process of this step.

First, we can identify facade elements on the same row or column using the method in Section 3.5.2.1. Then the average coordinate for every row (y value) and column (x value) can be calculated. All the coordinate values for the row and column can be used as the gridlines. Every intersection point of the gridlines should be a centre point for one facade element. We can add facade elements at these positions if the added element does not intersect with the existing elements. We will record the average width and height for the elements on the same row, and the added element will have the same size as the elements on the same row.

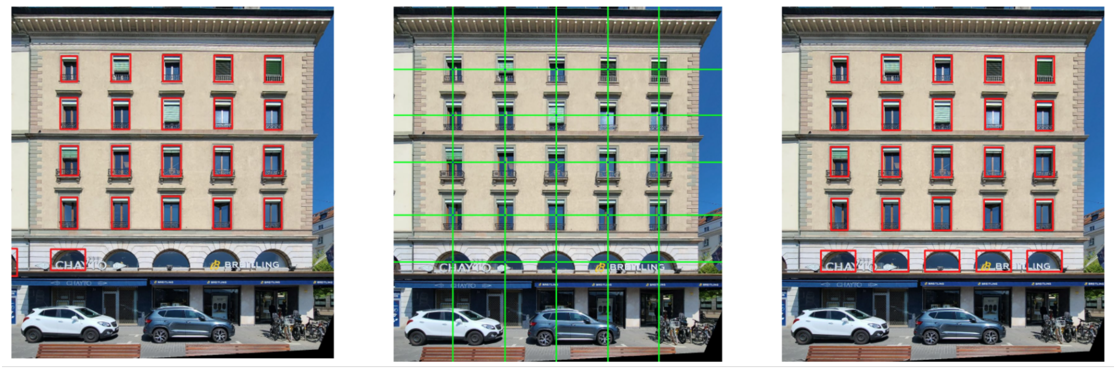


Figure 3.11: The detected facade layout is not completed (left). A grid can be constructed based on the existing elements (middle). The missing elements can be added, and the last row of this layout is complete (right).

3.5.2.3 Final regularization

After optimizing the initial layout, we will do the final layout regularization. The regularization problem can be addressed as a transformation, which changes the boxes' locations and sizes in the initial layout. We consider the energy term for differences in element locations as $\boldsymbol{E_l}$ and element sizes as $\boldsymbol{E_s}$. The regularized boxes will have their new locations (x_i^*, y_i^*) and new sizes (w_i^*, h_i^*). The $\boldsymbol{E_l}$ and $\boldsymbol{E_s}$ can be formulated as below:

$$\boldsymbol{E_l} = \sum_{i=1}^{n}(x_i^* + \frac{w_i^*}{2} - x_i - \frac{w_i}{2})^2 + (y_i^* + \frac{h_i^*}{2} - y_i - \frac{h_i}{2})^2 \tag{3.2}$$

$$\boldsymbol{E_s} = \sum_{i=1}^{n}(w_i^* - w_i)^2 + (h_i^* - h_i)^2 \tag{3.3}$$

We can get a set of linear equations based on the determined constraint groups as mentioned in Section 3.5.2.1. Every adjacent element pair in a constraint group will form a constraint pair corresponding to a linear equation to constrain the objective function. To limit the range of the location and size changes, we also add some other constraints for the (x_i, y_i) and (w_i, h_i). The changed x value and y value should be smaller than the extent of the facade face: $0 \leq x_i^* \leq w_f$ and $0 \leq y_i^* \leq h_f$. w_f and h_f) means the width and height of the facade face, which can be derived by project the facade face on the 2D image. To prevent the size change of the elements be too large, we also added an upper bound for the w_i^* and h_i^*: $0 \leq w_i^* \leq 1.5 \cdot w_i$ and $0 \leq h_i^* \leq 1.5 \cdot h_i$. The overall objective function to be minimized is:

$$\boldsymbol{w} \cdot \boldsymbol{E_l} + \boldsymbol{E_s} \tag{3.4}$$

where $\boldsymbol{w}$ is a weight to balance between the two terms $\boldsymbol{E_l}$ and $\boldsymbol{E_s}$ according to the user preferences. Finally, we can solve the quadratic programming problem defined in Equation 3.4 to get the regularized layout. Figure 3.12 shows an example of the regulariztion results.

3.6 Detailed facade reconstruction

We can add the facade elements to the facade face based on the regularised layout, which can be done using the camera parameters to back-project the boxes. Every box on the facade face will be

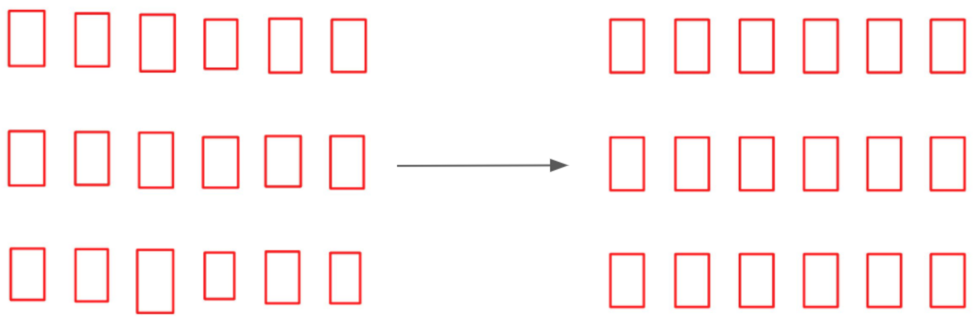


Figure 3.12: Original and regularized layout of 18 detected windows.

extruded(for balconies) or be intruded(for windows and doors) to make them 3D structures.

3.6.1 Back-projection

Given an image with camera parameters and the corresponding 3D facade, we can back-project 2D pixels in the image to the 3D facade. Let the intrinsic parameter matrix be $\boldsymbol{K}$, the rotation matrix be $\boldsymbol{R}$, and the translation matrix is $\boldsymbol{T}$. We can calculate the 3D coordinate of the camera centre P_c:

$$P_c = -\boldsymbol{R}^{-1} \cdot \boldsymbol{T} \tag{3.5}$$

Given a pixel coordinate(u, v) in the image, since we don't know the depth value(Z value) of this pixel, we can assume the Z value as 0, so the coordinate P'_{pixel} of this pixel in the camera coordinate system is:

$$P'_{pixel} = (\frac{(u - c_x) \cdot Z}{f_x}, \frac{(v - c_y) \cdot Z}{f_y}, Z), Z = 0 \tag{3.6}$$

where f_x, f_y are focal lengths in x and y direction; c_x, c_y are respective principal point offsets. Then, we can further convert the coordinates to the world coordinate system. We assume the 3D coordinate of this pixel is P_{pixel}:

$$P_{pixel} = \boldsymbol{R}(P'_{pixel} - \boldsymbol{T}) \tag{3.7}$$

With the 3D coordinate of the camera centreP_c and the pixelP_{pixel}, we can connect them to get a 3D line. The intersection point between the line and the facade face should be the back-projected point of this pixel(Figure 3.13).

3.6.2 Intrusion/Extrusion

The projected rectangles will be intruded or extruded at a certain depth on the facade. The depth information can be extracted from the dense point cloud where the facade elements are recessed a bit from the facade plane. We can filter out the points from the fused point cloud whose distance from the facade is less than a threshold. Coordinates of the filtered points will be converted based on the facade plane, which means the points on the plane will have the zero Z value. This is helpful to locate points within the bounding box of the facade elements. Then, we overlay the facade model on the filtered point cloud to measure the distance between the points and the plane for every facade element. The median value of the distances will be the depth for the corresponding element type.

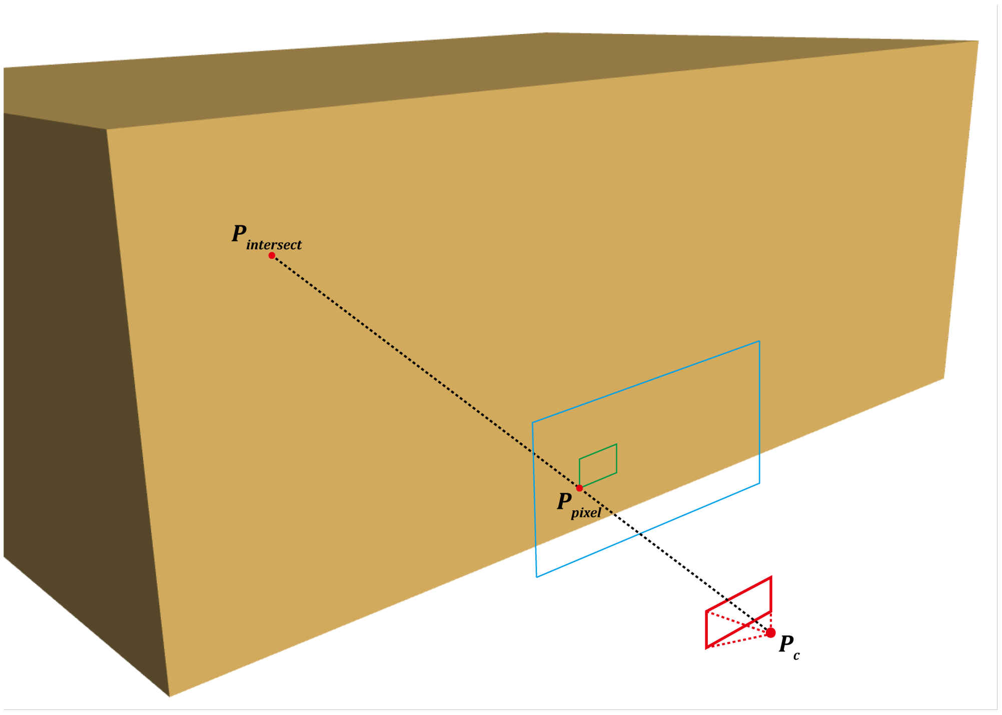


Figure 3.13: Back-projection of an image pixel.

We assume all the facade elements of the same type have the same depth. For example, we calculate the depth for each window element and then get the average depth value as the depth for all the window elements. The intrusion or extrusion will clip a hole on the facade face first and then add the 3D structure based on the depth.

We also allow the intersection between a window and a balcony. Figure 3.14 shows an example of this situation. When we want to model the a window and the balcony below it, we need to modify the size of the window to ensure the reconstructed model is watertight. It can also be used to express the occlusion relationship between the window and balcony. We can calculate the height at which the window is blocked by the balcony, and then the height of the window will be changed based on this value.

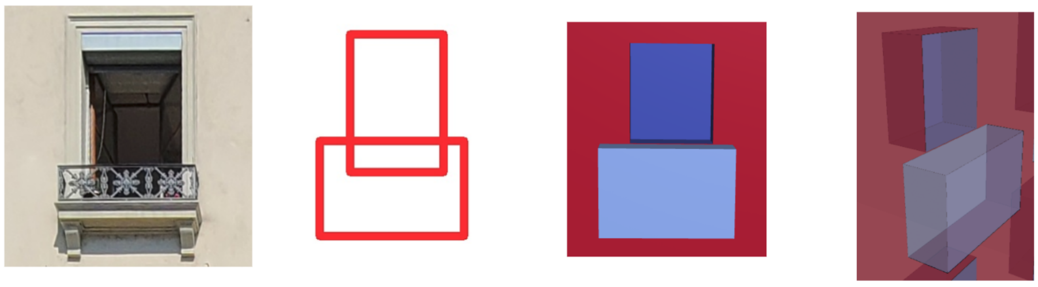


Figure 3.14: From left to right: A window and balcony in the image, the detected bounding boxes, the reconstruction results, side view of reconstruction results.

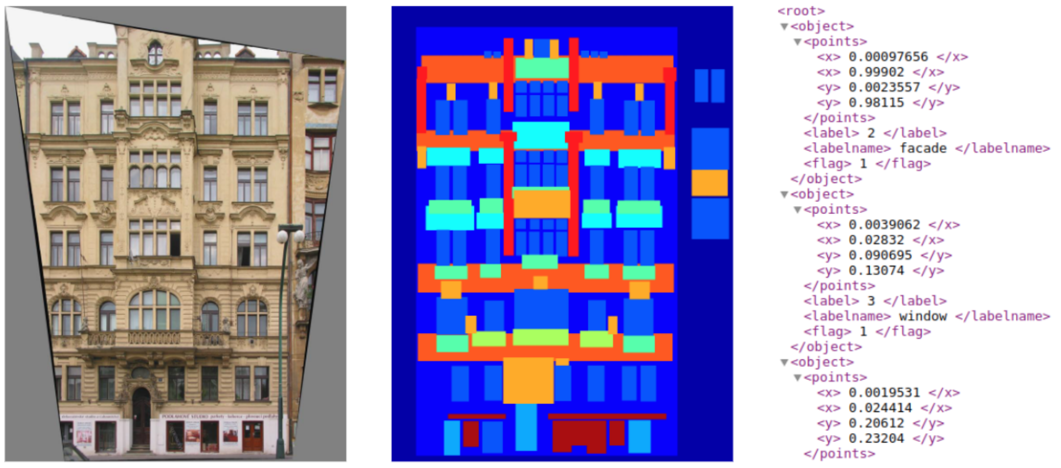
4 Implementation

This chapter introduces some implementation details of this project. Section 4.1 explains the dataset we used for the project and how they are prepared, including the SfM reconstruction and editing. Section 4.2 talks about some implementation details, including the programming specifications and some other important decisions we made during the experiment.

4.1 Datasets

The datasets used in this project can be divided into two parts: the training facade image data for the Faster R-CNN and the ground-level images for the model reconstruction.

We use the CMP facade database [Tyleček and Šára, 2013] as the training and evaluation data, including 378 images in the base dataset and 228 images in the extended dataset. It contains 12 facades classes, and we will only select windows, doors and balconies in our project. All the facade images are manually annotated and rectified. The facade images are taken from many different cities worldwide and contain various architectural styles. The base dataset is used for the training, and the extended dataset is used to evaluate the detection results. Figure 4.1 shows an example of the data.



```
<root>
▼<object>
  ▼<points>
     <x> 0.00097656 </x>
     <x> 0.99902 </x>
     <y> 0.0023557 </y>
     <y> 0.98115 </y>
   </points>
   <label> 2 </label>
   <labelname> facade </labelname>
   <flag> 1 </flag>
 </object>
▼<object>
  ▼<points>
     <x> 0.0039062 </x>
     <x> 0.02832 </x>
     <y> 0.090695 </y>
     <y> 0.13074 </y>
   </points>
   <label> 3 </label>
   <labelname> window </labelname>
   <flag> 1 </flag>
 </object>
▼<object>
  ▼<points>
     <x> 0.0019531 </x>
     <x> 0.024414 </x>
     <y> 0.20612 </y>
     <y> 0.23204 </y>
   </points>
```

Figure 4.1: An example of the CMP data, which contains the facade image (left), the annotated image rendered in PNG format (middle) and the annotations (right).

To conduct our whole pipeline, we take several sets of ground-level photos of buildings ourselves, with various building types. All photos were taken with the same mobile phone. The image sets may contain different numbers of facade faces, from 1 to 4. The largest image set includes 89 images, and the smallest one includes 9 images. The resolution for all the images is 4032×3024. The SfM and MVS process is conducted in COLMAP [Schönberger et al., 2016] [Schonberger and Frahm, 2016], a general-purpose SfM and MVS pipeline with a graphical interface. Since all the images are undistorted and are taken with the same camera, we choose the PINHOLE camera model and share the intrinsic parameters when recovering the camera parameters. We do not have geographic information about the camera positions, so we cannot do geo-registration for the extracted MVS point cloud. The gravity and horizontal direction of the point cloud are not parallel to the axis.

The MVS point clouds are edited in Mapple [Nan, 2021], a software for processing and visualizing point clouds and polygonal models. We remove planes outside the building and only remain the main facades in the point clouds.

4.2 Implementation details

4.2.1 Programming specifics

For the purpose of our project, the following hardware and software were used for implementation. All the programming was finished on a device with Ubuntu 20.04 having a NVIDIA GTX2060 graphic card. The main part of the methodology was implemented in C++, along with the Open Source Computer Vision Library (OpenCV)[1] for image processing, Eigen[2] for linear algebra calculation, Open3D[3] for the point cloud processing and Easy3D [Nan, 2021] for the 3D modelling. At the same time, the ETH3D Format Loader [Schöps et al., 2017] was used to load camera parameters from the COLMAP text format. The non-commercial GUROBI[4] solver was used to solve the optimization problems. The object detection algorithm was done in Python with Pytorch [Paszke et al., 2019], an optimized tensor library for deep learning. We also used Albumentations[5] to augment the training images.

Since we did not implement the object detection algorithm in C++, the facade detection step should be done outside the main program, so we divided the main program into two parts. The first part includes the coarse model reconstruction, the selection of front view images and rectification. We record the selected image names and corresponding camera parameters to be used in the second part. The second part will read the object detection result and then finish the regularization and back-projection.

4.2.2 Training specifications

In order to train an object detector for the facade elements, we use the pretrained Faster RCNN model with the ResNet50 FPN backbone provided by Pytorch. We train the model using 347 images and 31 for validation, and set the learning rate to 0.001 and momentum to 0.9. Due to the hardware limitation, we can only set the batch size as 1 in the experiment. The images and the corresponding bounding boxes will both be resized to 640 before feeding the images to the augmentations, which can shorten the training time. At the same time, in order to increase the diversity of the images, we augment the training images, including flipping, rotation and adding blurs.

[1] www.opencv.org
[2] www.eigen.tuxfamily.org
[3] www.open3d.org
[4] www.gurobi.org
[5] www.albumentations.ai

The evaluation of the object detection result is done using the CMP extended dataset. We use the normally used evaluation scores to measure the performance of the object detection results, including precision, recall and IoU. We calculate the scores separately for each facade element type. The result is shown in Section 5.1.

4.2.3 Detection result optimization

For different styles of buildings, the layout of facades is changeable. As a result, the detected facade elements may not perfectly subdivide the space without overlapping. In general, we only accept one type of overlapping: the overlapping between the windows and the balcony below them(Figure 4.2). Meanwhile, we need to fix the other overlapping types in the detection result.

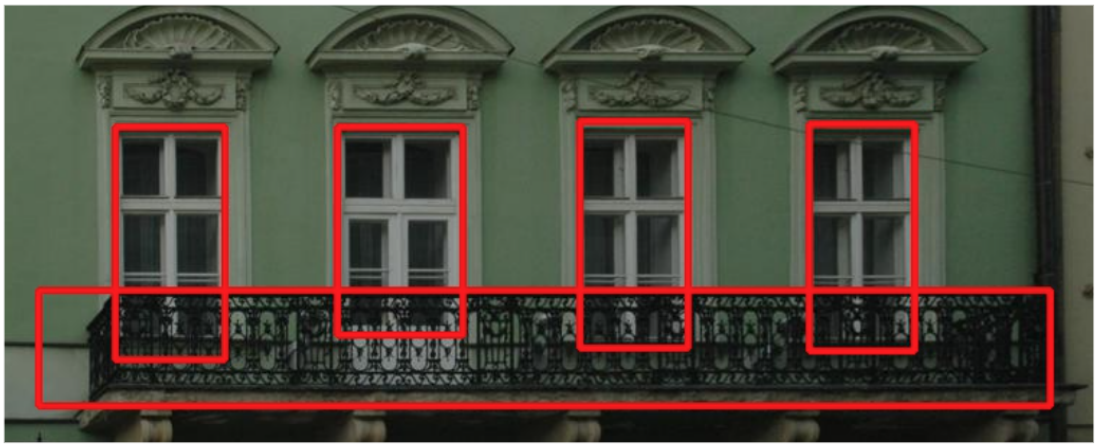


Figure 4.2: An example of the correct window-balcony relationship.

One simple type of overlapping is some windows located close together (Figure 4.3). It is also possible for some windows or balconies to be detected inside other bounding boxes of the same type. We can merge the elements for such a situation based on their common extent. Then, the overlapping windows or balconies can become one large element.

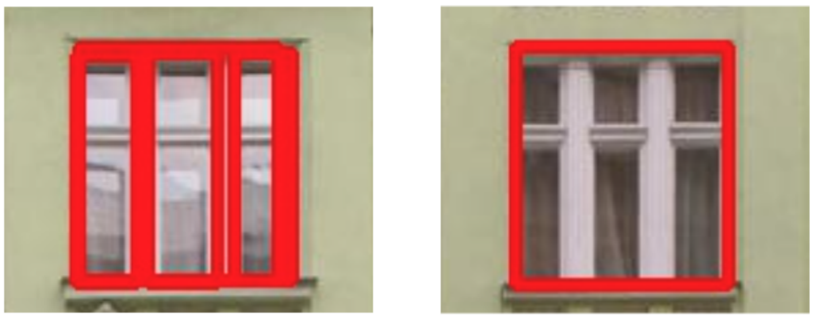


Figure 4.3: A window before and after solving the overlapping.

Another type of overlapping is between the balconies and windows. Since we allow them to form the state of the Figure 4.2, we need to identify if their relationship is not the correct situation. Then, we can compute their overlapping length horizontally or vertically. Finally, we can adjust the width or height of the balcony based on this length to separate them. One example is provided in Figure 5.1.

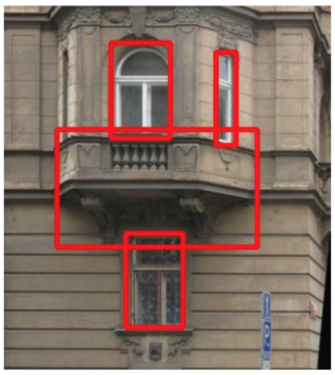


Figure 4.4: An example of incorrect window-balcony relationship, the balcony cut the window below it.

When we take photos of the facades, other buildings may be included in the image extent, and their facade elements may also be detected. The bounding boxes not belonging to the facade will influence the regularization of the layout. We choose to project the facade rectangle onto the 2D image and transform it based on the transformation matrix obtained during the rectification. Then the facade face can be used as a large bounding box to filter the facade elements not belonging to the current facade.

4.2.4 Parameter tuning

From the analysis of the methodology in the chapter3, users can tune the following basic parameters to control the quality of reconstruction:

1) The minimum support points for the detected planes in RANSAC. This is an important parameter for the RANSAC algorithms. It can decide if we can detect the main facade plane and neglect the other plane structures. In our experiment, this parameter should be fixed as 100000 when using the fused point cloud from MVS. All the points belonging to the wall can be extracted, and the points inside the window extents can be filtered out.

2) The distance threshold σ mentioned in Section 3.5.2.1. In our experiment, most regular or semi-regular facade layouts can be perfectly divided into constraint groups with a value of 0.3. We also fix this value for most of the facade layouts. However, there will also be some totally irregular facade layouts, for example, the facade in Figure 4.5. The object detection can perfectly detect all the windows, but when we still use the threshold 0.3, the regularization will ”optimize” windows that are not on the same row to the same row in the result. We set the threshold as 0.05 to limit the extent of searching the same column or row elements for such a situation.

3) The number of window types in the facade layout. This value will be used in the k-means algorithm when selecting the same-size constraint groups. At the same time, users also need to decide if the facade layout needs to be fixed by the method mentioned in section8. For every facade, the program will display the object detection result first and then ask the user the decisions for these two parameters before the back-projection.

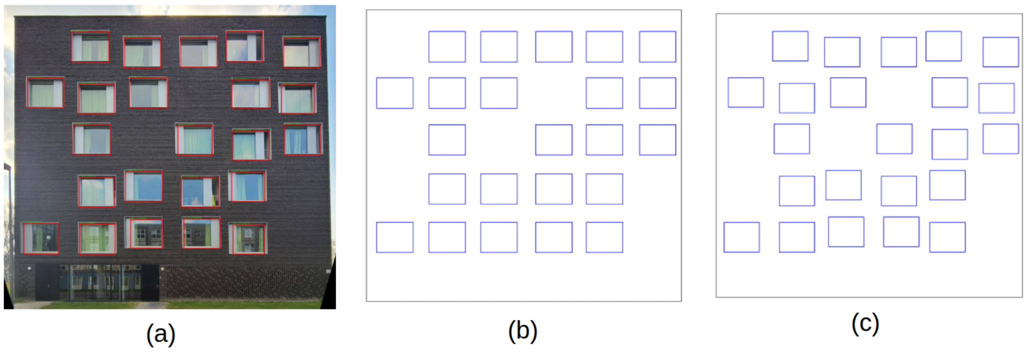


Figure 4.5: An example of the totally irregular facade. The original detection result(a). The regularization cannot handle this situation with a σ value of 0.3(b). The regularization result is improved by setting the σ value as 0.05(c).

4) The detection confidence threshold. This value will filter out the bounding boxes with a confidence lower than it. It ensures that the predicted bounding boxes have a certain minimum confidence score. In most cases, we set this value as 0.8 to remove the unreliable predictions. However, the value needs to be tuned to 0.9 or higher for some more complex scenes to ensure a cleaner layout.

5 Results and evaluation

This chapter reveals the reconstruction results of the proposed pipeline. Section 5.1 is about the object detection results evaluation. Section 5.2 shows the final reconstruction results and also gives some analysis based on the results.

5.1 Evaluation of the facade element detection results

5.1.1 Evaluation metrics

In the object detection context, basic concepts of the confusion matrix can be defined as:

- **True positive (TP)**: Total number of the correctly detected ground-truth bounding boxes.
- **False positive (FP)**: Total number of the wrongly detected ground-truth bounding boxes.
- **False negative (TN)**: Total number of the ground-truth bounding boxes which are not detected.

The true negative(TN) should not be applied in the object detection task since there will be an infinite number of bounding boxes that should not be detected in one image. Based on these definitions, we need to know when should a detected bounding box be classified as TP or FP. A common way is to use a threshold of Intersection Over Union (IOU). Given a detected bounding box and the corresponding ground-truth bounding box, the IOU can be defined as illustrated in Figure, which can evaluate the overlap between two bounding boxes.

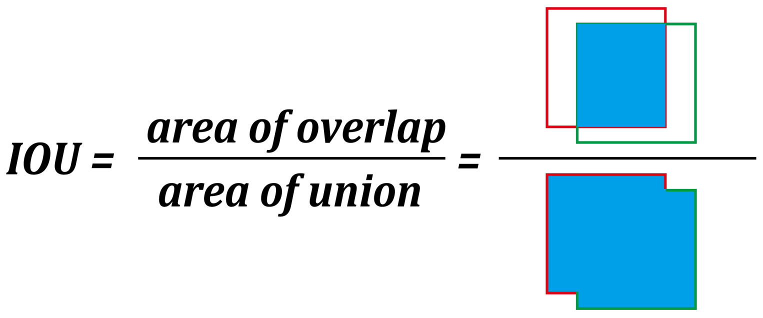


Figure 5.1: Intersection Over Uniou(IOU).

Then, we can set a threshold t to determine if a detected box is correct. If the IOU is larger than t, we know that this bounding box is correctly detected. Based on the definition of the TP, FP and TN, the commonly used assessment metrics precision P and recall R can be defined as:

$$P = \frac{TP}{TP + FP} = \frac{TP}{all\ detected\ boxes} \tag{5.1}$$

$$R = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truth\ boxes} \tag{5.2}$$

We also apply the F_1-score to measure the harmonic mean of the precision and recall:

$$F_1 = 2 \cdot \frac{P \times R}{P + R} \tag{5.3}$$

As mentioned in Section 4.2.4, we set the confidence threshold as 0.8 to filter the possible facade elements in the pipeline. This threshold will also be applied in the evaluation. In this project, the availability of the detection result depends largely on the level of the IOU, so we need to set a higher t value to ensure that the bounding boxes classified as TP have a larger overlapping area with the ground truth boxes. In the evaluation, the t value is set as 0.75.

5.1.2 Evaluation results

The evaluation is conducted on the extended dataset of the CMP facade dataset. First, we evaluate the detection results for all the facade images, including three types of facade elements: window, door and balcony. The evaluation result is shown in Table 5.1.

Class	P	R	F_1
window	0.823	0.816	0.819
balcony	0.662	0.691	0.676
door	0.516	0.487	0.501

Table 5.1: Performance of facade elements detection using all the test data

We can observe that the detector has a poor performance when detecting balconies and even poorer when detecting doors. This is due to the high diversity of these two structures and the limited number of them in the training data. There are only 1118 balcony instances and 398 door instances in the training data, while the number of windows is 12222. However, since windows are the most important elements that make up the facade layout, we can still conclude that the detector achieves a good result for the facade element detection.

In order to further evaluate the performance of the detector for specific facade types, we select three groups of images according to the facade styles and evaluate the window detection results only. **Group A** contains facade images with relatively simple window layouts and no wall decorations. Facades in **Group B** are from some classical buildings, with more decorations on the walls and prominent balconies. Facades in **Group C** contain many window-like structures on the wall, and it is difficult for humans to distinguish windows in these images. Figure 5.2 shows some examples for the three groups. The three groups have 13, 15, and 11 images respectively. The result of evalution for these three groups is listed in Table 5.2.

Group	P	R	F_1
A	0.915	0.872	0.893
B	0.878	0.744	0.805
C	0.869	0.457	0.599

Table 5.2: Performance of window detection using facade images from different groups.

The evaluation result shows that the detector performs much better when detecting windows on the clean and simple facades. Accuracy decreases when detecting the facades of complex classical

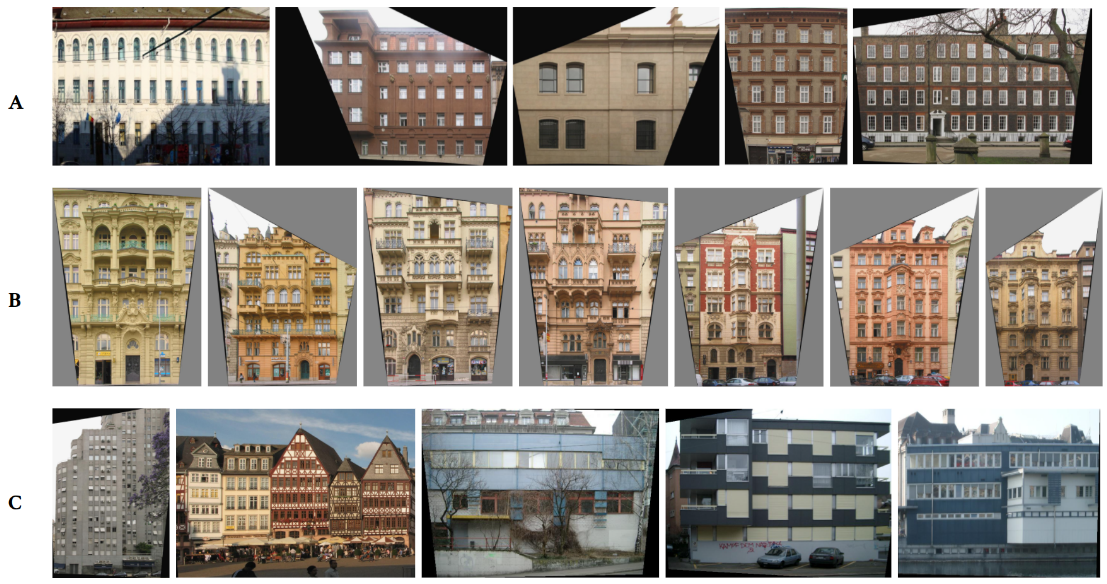


Figure 5.2: Some examples from the three facade image groups.

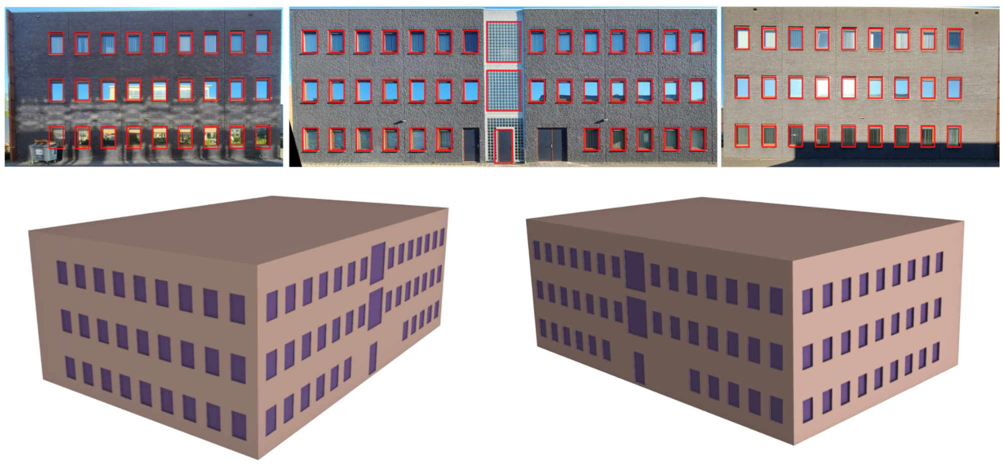
buildings, perhaps because there are too many balconies obscuring windows, but the result is still acceptable. However, the detection results are not satisfactory when detecting facades that are difficult for the human eye to recognize. The high recall value means that there are a large number of ground truth windows that are not detected. In general, the detector's performance is acceptable in most situations for window detection, especially for facades with clean walls and clear layouts.

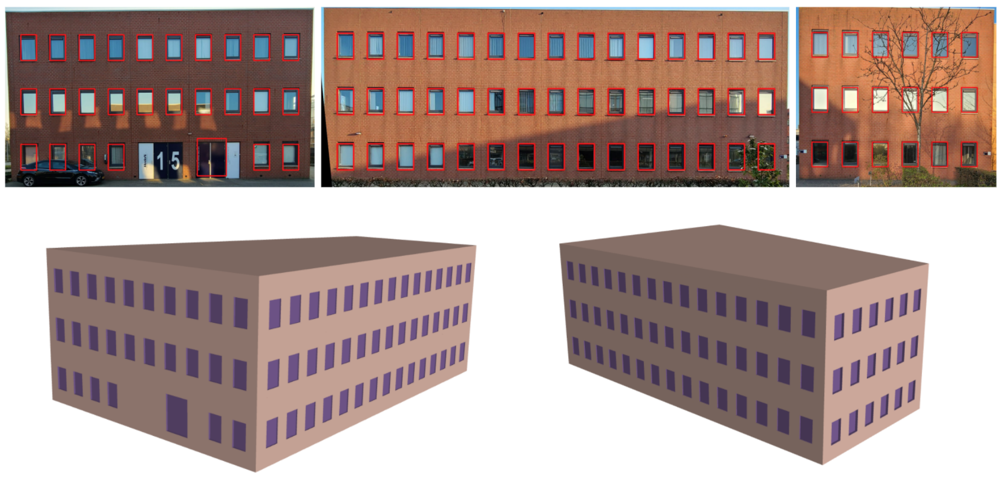
5.2 Reconstruction results

As we stated in Section 4.1, the pipeline's inputs are some street view images taken by myself. We chose 13 buildings which contain 18 facades of various architectural scenes, and they can be divided into 7 tasks. We applied these images and the corresponding MVS fused point clouds to generate the detail enhanced building models. The results of this project are some building blocks with facade details.

Figure 5.3 shows some reconstruction results of multi-face buildings. In this project, we can add facade details for every facet of a single building in one pipeline. In theory, if the user can provide the multi-view images and the corresponding 3D building model, we can reconstruct all the faces for the building. As can be seen in the result, the walls can match the facade images perfectly, and the distribution of the added facade elements also corresponds to the actual situation.

However, when we take street-view images, we are more likely to get images for only one face of the building. Figure 5.4 presents some results of the street-side facades. The facades in Figure 5.4(a) and Figure 5.4(b) are from some townhouses located on two different streets. For the facades in Figure 5.4(a), we take photos and extract point clouds for them separately, so the reconstructed models don't in the same coordinate system. For the case in Figure 5.4(b), we take photos of all the facades and extract point cloud in one SfM and MVS process. Then, we can divide the point cloud into different facade planes as the pipeline's inputs. The results can be well aligned and closer to the actual situation.


(a)


(b)

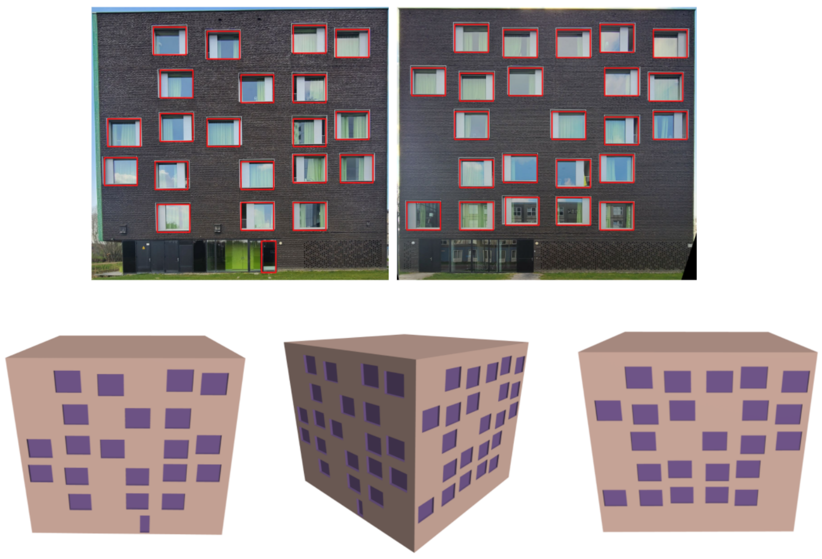
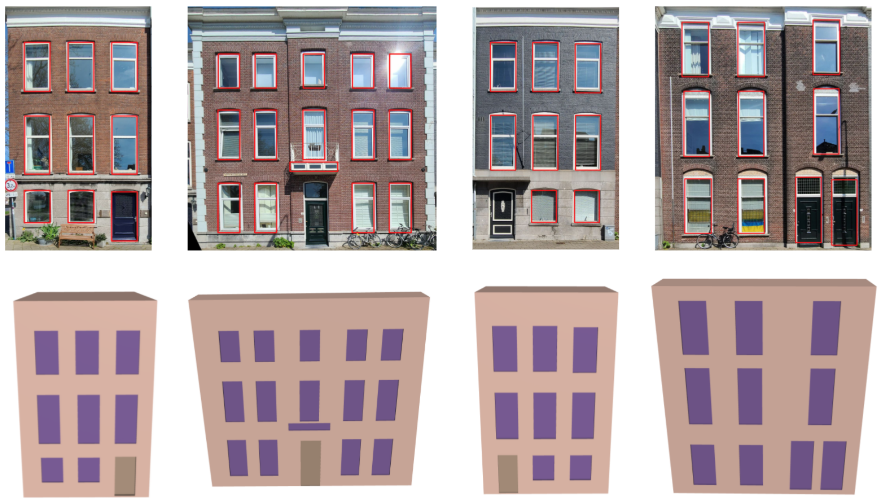
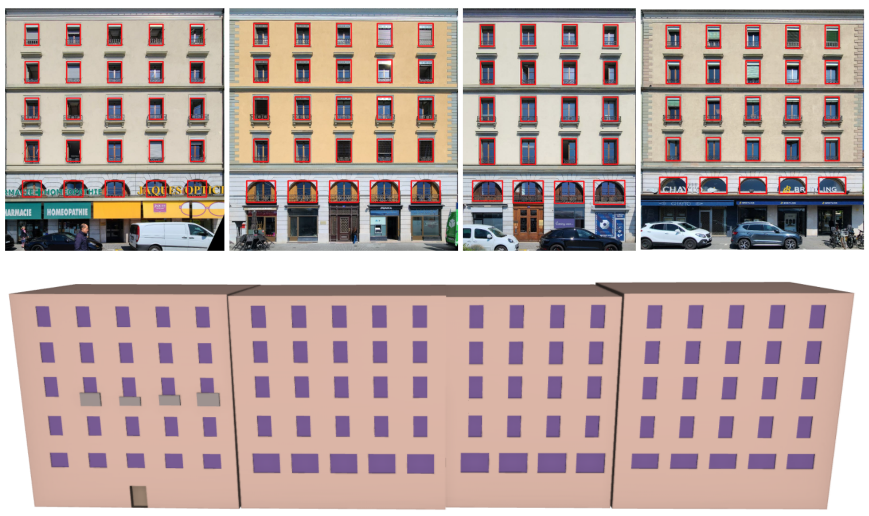

(c)

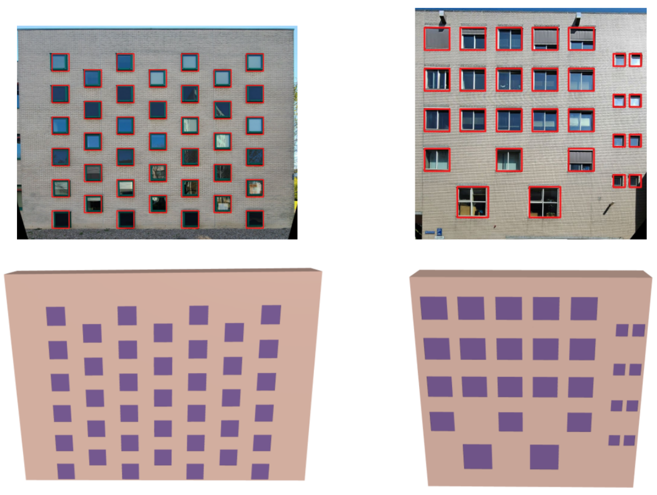
Figure 5.3: Detailed reconstruction results of multi-face buildings. Rectified images with the regularized facade elements (red boxes) are put on top of the 3D models. (a) $\sim$ (c) : task1, task2, task3.



(a)



(b)



(c)

Figure 5.4: Detailed reconstruction results of some street-side facades. (a) $\sim$ (c) : task4-1$\sim$task4-4 (from left to right), task5-1$\sim$task5-4 (from left to right), task6&task7.

As can be seen in these reconstruction results, our method of facade elements detection can locate all the window elements even though there are occlusions or reflections in the facade image. All the windows are well aligned, and the result can also reflect subtle differences in the size of different types of windows. However, most located doors are classified as windows, and there are still many doors not detected. In Figure5.4, there should be balconies on every facade, but we can only locate four of them in the leftmost model.

The total reconstruction time for each task is less than 50 seconds. The timing does not include the SfM and MVS processes. As can be seen from Table 5.3, the facade image selection and rectification step take the longest time. Since we need to traverse all the images for every facade face, the processing time increases when we input more images. The facade element detection and regularization take less than a second.

Task	f (#facade)	i (#image)	Execution time (sec)				
			Coarse model	Selection	Detection	Regularization	Total
task1	3	89	8.80	30.42	0.62	0.55	43.33
task2	3	69	7.20	28.27	0.52	0.63	38.85
task3	2	18	3.76	2.95	0.45	0.33	8.71
task4-1	1	13	0.89	2.32	0.35	0.22	4.35
task4-2	1	17	2.35	2.66	0.32	0.19	6.31
task4-3	1	13	1.22	2.07	0.31	0.22	4.55
task4-4	1	15	1.88	2.29	0.34	0.21	5.44
task5-1	1	64	2.15	2.57	0.31	0.16	6.35
task5-2	1		2.75	2.05	0.32	0.17	6.47
task5-3	1		1.68	2.66	0.30	0.23	5.98
task5-4	1		1.78	2.16	0.29	0.28	5.79
task6	1	17	2.94	2.40	0.37	0.19	6.67
task7	1	16	1.50	2.40	0.28	0.20	5.01

Table 5.3: Summary of the runtime of every step in the pipeline.

6 Conclusions and future work

In this final chapter, the research questions of this graduation project are reviewed and answered based on the experiments and results. The contributions of this thesis are presented, along with the limitations of the proposed methodology. Finally, we discussed some future work and an outlook in the last section.

6.1 Research overview

This thesis presents a pipeline to reconstruct detail enhanced 3D building models based on the street view images. The main question of this thesis is:

> ***How can we bring more facade details into the image-based urban reconstruction pipeline?***

The image-based urban reconstruction pipeline will typically produce models with a low LOD. The main idea of our method is to extend these produced models to LOD3. Every wall face can be identified in the images based on the coarse model. Using camera parameters, we can then connect each wall face with one facade image corresponding to the wall. As a result, the located facade elements in the image can be added to the wall face to achieve facade detail enhancement. So the main problem is actually about what is our proposed facade parsing method. The sub-questions of this project should be answered first before we can fully answer the main question.

The sub-questions are addressed as follows:

1. **How can we detect facade elements from the facade images?**

   Our answer to this question is to use the state-of-the-art object detection algorithm. This thesis selects Faster R-CNN with the ResNet50 FPN backbone to detect windows, balconies, and doors in facade images. The bounding boxes resulting from the object detection can align pretty well with the facade element rectangles. Unlike the pixel-level facade parsing techniques, object detection can directly output geometric information of the facade elements. In addition, object detection is much more efficient than some other detection or subdivision algorithms, such as the template matching or hierarchical subdivision. We trained the model using 378 facade images from the previous work. The detection process for every facade image may take less than 0.4 seconds. It can be implemented for any facade type with different patterns, though the detection precision may differ for different facade types.

   We also evaluate the performance of the Faster R-CNN on facade images. The F1 score for window detection can reach 0.817 for the 228 testing images under an IOU threshold of 0.75. When the wall is clean and the facade layout is relatively simple, the F1 score can reach 0.893 for the windows, which means most of the windows can be detected. The detector performs worse on more complex facades, but the F1 score can still be higher than 0.8 for window detection. In conclusion, object detection can be served as a good start for facade parsing.

The object detector cannot always detect all the elements. The results may lose some facade elements when occlusions or reflections are in the facade image. In order to optimize the detection results, we apply a method to fix the results based on the global regularity of the facade layout. This method can work pretty fine, which can also be seen as a balance between the regularity rule and the data.

2. **How can we ensure the regularity of the facade elements' layout?**

   We assume there are two types of regularity for the facade layout: alignment and same-size. These two types of regularity ensure that the facade layout conforms to the real-world situation. Suppose the facade elements can be well aligned in horizontal and vertical directions and the elements of the same type have the same size. In that case, we think the facade layout has a reasonable regularity. To improve the regularity of the facade elements, we need to change the position of the detected facade elements and also their sizes. The problem can be seen as an optimization problem.

   We divide the facade elements into different constraint groups according to the requirement of regularity. The positions or sizes of the elements in the same group should conform to the established rules, which will be used as the constraint for the subjective function. After solving the optimization problem, the facade element will have the correct positions and sizes. As a result, the answer to this question can be concluded. We should adjust the elements' coordinates and sizes while maintaining the global layout as much as possible based on the object detection results.

After answering the sub-questions, we can also conclude the answer to the main question. However, other steps that make up the whole pipeline are also crucial apart from the facade parsing. We reconstruct the coarse model as the fundamental of detail enhancement. The front view facade images can be selected and rectified automatically based on the coarse model. Finally, we apply the back-projection to locate the facade elements in 3D. We also find a way to roughly calculate the facade elements' depth based on the fused point cloud. Combining all these steps will be our answer to the main research question.

6.2 Discussion

6.2.1 Contributions

In this graduation project, we bring more facade details to the coarse building models in the image-based reconstruction pipeline based on object detection. We take advantage of the multiview facade images, which allow us to overcome the occlusion problem when modeling the facade elements. The whole pipeline can be done automatically with a reasonable amount of user-defined parameters. At the same time, this method can be very efficient because of the use of the object detection algorithm. The project's backbone is actually Faster R-CNN network, and some of the ideas may not be the origin. However, we still believe that our implementation can contribute to current research. The prominent contributions of this work are:

- We design a whole pipeline that can automatically reconstruct building models with fine facade details from a set of images.
- We provide a LOD1 model reconstruction method from the unclosed building point cloud. The walls in the data do not need to be orthogonal to the ground.

- We propose a method to automatically select the best facade image for the corresponding 3D wall face in a set of street view images around the building.

- We design a facade parsing method using object detection and optimization-based regularization.

6.2.2 Limitations

In addition to the contributions of our method as mentioned in Section 6.2.1, there are also some limitations, which we will discuss in this section. Some limitations may not result from the design of the methodology but are related to the complexity of the facade elements detection problem. Due to time constraints, our work cannot be tested on more types of buildings, so there may be other situations where it does not apply. Therefore, the limitations we proposed are just some problems that we can summarize so far. With this information, we can propose some ideas to improve the methodology in future work.

- We assume that at least one photo contains all the facade elements on the wall. We only use one image from the image set to detect the elements for every facade. If the selected image cannot contain the whole face of the facade, our method will not model the complete facade layout. This limitation means we can only reconstruct buildings with relatively small facades. At the same time, this limitation will also set a strict requirement for the input image set. In most cases, taking photos to contain the whole facade exactly can be challenging.

- Doors and balconies are poorly detected. The training difficulty is very high since the types of doors and balconies have high intraclass variation. The training data we used contains many different facade styles, so the training results for the doors and balconies are not satisfying. We can consider relabeling the datasets based on the material or color of the doors and balconies, which means differentiating them more precisely. However, this will significantly increase the complexity of this problem, and it can still be challenging to distinguish windows and doors in some situations.

- The reconstruction result is largely dependent on the unpredictable object detection result. Even though we can fix the detection results to a certain extent based on the regularity of the layout, the accuracy of the results is still not guaranteed. The Faster R-CNN framework is not designed for facade elements only, and the detection performance may be terrible in some situations. We can apply the texture or color information of the facade image along with the detection results to optimize the detection performance. However, processing the image may be time-consuming, affecting the method's efficiency. One possible way of improving is to change the framework of the Faster R-CNN, for example, to combine the pixel-level segmentation results with the detection results, and the loss functions should also be modified.

- Differences in facade layout between different building types are not taken into account. This limitation is also related to the post-processing of the object detection results. We only define some simple principles to regularize the facade layout. However, the different building facade types may have different regularity. For example, for the Haussmannian-style buildings in Paris, the last row of the facade layout will always be a door, and every window will be above a balcony. We may also observe the symmetry or co-occurrence of the facade elements in some facade styles. If we can adjust the principles for different facade types, the results can be more reliable.

6.3 Future work

Based on the limitations mentioned in section1 and some further research, we would like to discuss some future work in this section. These future works will improve the current method and also allow the method to extend to other applications.

- **Enhance the Faster R-CNN for facade elements detection.** Another fully connected layer can be integrated into the classification layer of the original R-CNN. This new fully connected layer can generate a pixel-level binary mask based on the image or the depth map, which can be used to adjust the classification and position of the bounding boxes.
- **Investigate the facade styles and their layout principles.** Building facade styles are varied and can be changed in different cities and eras. There is still no comprehensive and specific research on building facade styles. If we can identify the main facade style of the building that will be reconstructed, the reliability of the facade parsing will be greatly increased.
- **Apply this method to large-scale city modeling.** In theory, if the camera parameters and the coarse building model are provided, we can reconstruct the LOD3 model based on the facade images. As a result, extending this method to large-scale city modeling is possible. The large-scale LOD1 or LOD2 model can be obtained easily using many existing methods. The extent of the 3D facade face can be identified on the UAV images since we have the camera parameters(Figure 6.1). We can easily rectify the image based on the projected polygon. However, the problem is distinguishing the facade planes of different buildings. In our methods, we assume different facades are different faces, but the facade faces may be connected in the large-scale models. In addition, the quality of UAV images may also be lower than the street view images. The detection results may also be affected by the image quality.

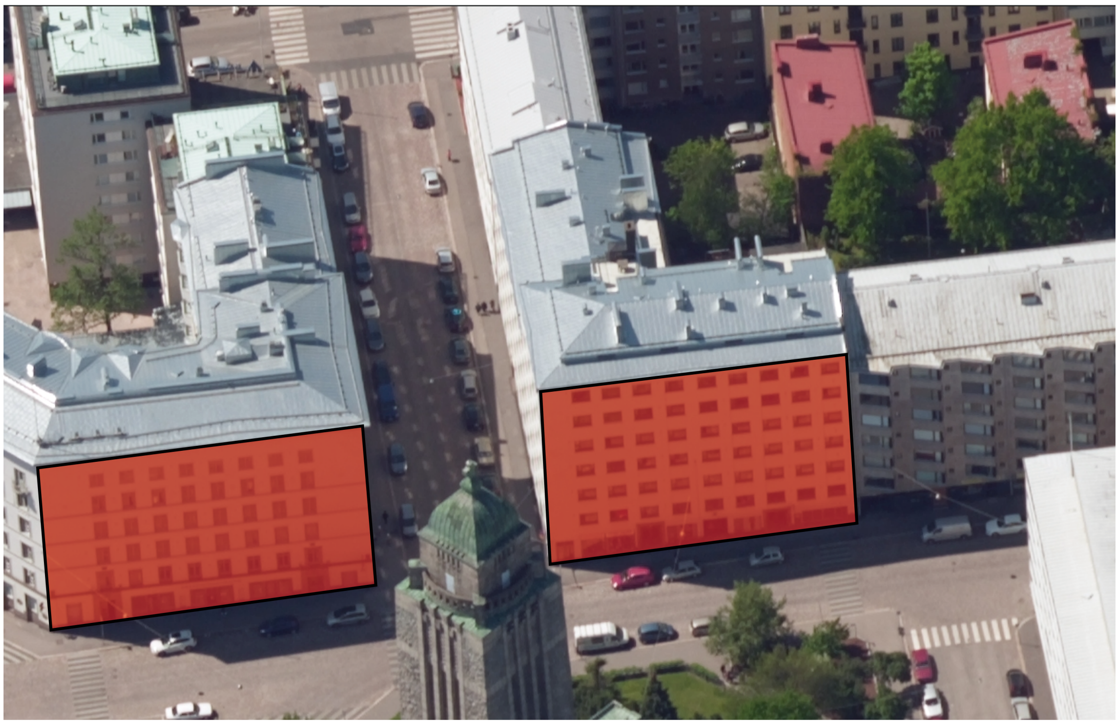


Figure 6.1: The 3D facades can be projected to the UAV images.

Bibliography

Alhalawani, S., Yang, Y. L., Liu, H., and Mitra, N. J. (2013). Interactive Facades Analysis and Synthesis of Semi-Regular Facades. *Computer Graphics Forum*, 32(2pt2):215–224.

Arikan, M., Schwarzler, M., Flory, S., Wimmer, M., and Maierhofer, S. (2013). O-snap: Optimization-based snapping for modeling architecture. *ACM Transactions on Graphics*, 32(1).

Biljecki, F., Ledoux, H., and Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25–37.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information 2015, Vol. 4, Pages 2842-2889*, 4(4):2842–2889.

Cohen, A., Schwing, A. G., and Pollefeys, M. (2014). Efficient Structured Parsing of Facades Using Dynamic Programming.

Coughlan, J. M. and Yuille, A. L. (1999). Manhattan World: Compass direction from a single image by Bayesian inference. *Proceedings of the IEEE International Conference on Computer Vision*, 2:941–947.

Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, ICCV2015:1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587.

Holzmann, T., Oswald, M. R., Pollefeys, M., Fraundorfer, F., and Bischof, H. (2017). Plane-based surface regularization for urban 3d construction.

Huang, J. B., Kang, S. B., Ahuja, N., and Kopf, J. (2014). Image completion using planar structure guidance. *ACM Transactions on Graphics*, 33(4).

Ian T. Jolliffe (2002). Principal Component Analysis.

Jiang, H., Nan, L., Yan, D. M., Dong, W., Zhang, X., and Wonka, P. (2016). Automatic Constraint Detection for 2D Layout Regularization. *IEEE Transactions on Visualization and Computer Graphics*, 22(8):1933–1944.

Jonsson, M. (2016). Make it Flat : Detection and Correction of Planar Regions in Triangle Meshes.

Kelly, T., Femiani, J., Wonka, P., and Mitra, N. J. (2017). BigSUR: Large-scale structured urban reconstruction. *ACM Transactions on Graphics*, 36(6).

Kolbe, T. H. (2009). Representing and exchanging 3D city models with CityGML. *Lecture Notes in Geoinformation and Cartography*, pages 15–31.

Koziński, M., Obozinski, G., and Marlet, R. (2015). Beyond Procedural Facade Parsing: Bidirectional Alignment via Linear Programming. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9006:79–94.

Li, M., Nan, L., Smith, N., and Wonka, P. (2016). Reconstructing building mass models from UAV images. *Computers & Graphics*, 54:84–93.

Lin, H., Gao, J., Zhou, Y., Lu, G., Ye, M., Zhang, C., Liu, L., and Yang, R. (2013). Semantic decomposition and reconstruction of residential scenes from LiDAR data. *ACM Transactions on Graphics*, 32(4).

Liu, H., Xu, Y., Zhang, J., Zhu, J., Li, Y., and Hoi, S. C. (2020). DeepFacade: A Deep Learning Approach to Facade Parsing with Symmetric Loss. *IEEE Transactions on Multimedia*, 22(12):3153–3165.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. (2018). Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, 128(2):261–318.

Martinović, A., Mathias, M., Weissenberg, J., and Van Gool, L. (2012). A Three-Layered Approach to Facade Parsing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7578 LNCS(PART 7):416–429.

Müller, P., Zeng, G., Wonka, P., and Van Gool, L. (2007). Image-based procedural modeling of facades. *ACM Transactions on Graphics*, 26(3).

Nan, L. (2021). Easy3D: a lightweight, easy-to-use, and efficient C++ library for processing and rendering 3D data. *Journal of Open Source Software*, 6(64):3255.

Nan, L., Jiang, C., Ghanem, B., and Wonka, P. (2015). Template Assembly for Detailed Urban Reconstruction. *Computer Graphics Forum*, 34(2):217–228.

Nan, L., Sharf, A., Zhang, H., Cohen-Or, D., and Chen, B. (2010). SmartBoxes for interactive urban reconstruction. page 1.

Nan, L. and Wonka, P. (2017). PolyFit: Polygonal Surface Reconstruction From Point Clouds.

Nan, L., Xie, K., Chen, B., Sharf, A., Wong, T. T., Deussen, O., and Cohen-Or, D. (2011). Conjoining Gestalt Rules for Abstraction of Architectural Drawings. *ACM Transactions on Graphics*, 30(6):1–10.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.

Rousseeuw, P. J. and Leroy, A. M. (1987). Robust Regression and Outlier Detection.

Schmitz, M. and Mayer, H. (2016). A convolutional network for semantic facade segmentation and interpretation. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 41:709–715.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226.

Schonberger, J. L. and Frahm, J. M. (2016). Structure-from-Motion Revisited. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:4104–4113.

Schönberger, J. L., Zheng, E., Frahm, J. M., and Pollefeys, M. (2016). Pixelwise view selection for unstructured multi-view stereo. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9907 LNCS:501–518.

Schöps, T., Schönberger, J. L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., and Geiger, A. (2017). A multi-view stereo benchmark with high-resolution images and multi-camera videos. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:2538–2547.

Tyleček, R. and Šára, R. (2013). Spatial Pattern Templates for Recognition of Objects with Regular Structure. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8142 LNCS:364–374.

Verdie, Y., Lafarge, F., and Alliez, P. (2015). LOD Generation for Urban Scenes. *ACM Transactions on Graphics*, 34(3):30.

Wu, C., Agarwal, S., Curless, B., and Seitz, S. M. (2012). Schematic surface reconstruction. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1498–1505.

Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E., and Quan, L. (2008). Image-based façade modeling. *ACM SIGGRAPH Asia 2008 Papers, SIGGRAPH Asia'08*.

Xiao, J., Fang, T., Zhao, P., Quan, L., and Lhuillier, M. (2009). Image-based Street-side City Modeling. *ACM Transactions on Graphics*, 28(5):1–12.

Zhang, X., Lippoldt, F., Chen, K., Johan, H., and Erdt, M. (2019). A data-driven approach for adding facade details to textured LoD2 CityGML Models. *VISIGRAPP 2019 - Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 1:294–301.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class scrbook. The main font is Palatino.