Repetition-based Interactive Façade Modeling

Thesis by
Sawsan Nabeel AlHalawani, Bachelor of Science in
Computer Science

Submitted in Partial Fulfillment of the Requirements for the
degree of
Masters of Science

King Abdullah University of Science and Technology
Mathematical and Computer Sciences and Engineering
Computer Science

Thuwal, Makkah Province, Kingdom of Saudi Arabia
July, 2012
The thesis of Sawsan Nabeel AlHalawani is approved by the examination committee.

Committee Chairperson: Niloy J. Mitra
Committee Member: Markus Hadwiger
Committee Member: Peter Wonka
ABSTRACT

Repetition-based Interactive Façade Modeling

Sawsan Nabeel AlHalawani

Modeling and reconstruction of urban environments has gained researchers attention throughout the past few years. It spreads in a variety of directions across multiple disciplines such as image processing, computer graphics and computer vision as well as in architecture, geoscience and remote sensing. Having a virtual world of our real cities is very attractive in various directions such as entertainment, engineering, governments among many others.

In this thesis, we address the problem of processing a single façade image to acquire useful information that can be utilized to manipulate the façade and generate variations of façade images which can be later used for buildings’ texturing. Typical façade structures exhibit a rectilinear distribution where in windows and other elements are organized in a grid of horizontal and vertical repetitions of similar patterns. In the first part of this thesis, we propose an efficient algorithm that exploits information obtained from a single image to identify the distribution grid of the dominant elements i.e. windows. This detection method is initially assisted with the user marking the dominant window followed by an automatic process for identifying its repeated instances which are used to define the structure grid. Given the distribution grid, we allow the user to interactively manipulate the façade by adding, deleting,
resizing or repositioning the windows in order to generate new façade structures. Having the utility for the interactive façade is very valuable to create façade variations and generate new textures for building models. Ultimately, there is a wide range of interesting possibilities of interactions to be explored.
Acknowledgments

In the name of Allah, the Most Gracious and the Most Merciful

All praises to Allah for the strengths and his blessings in completing this thesis. I would like to take this opportunity and express my sincere gratitude to all the people who were beside me throughout the coursework of this research. Without their support, I would not have been possibly here today.

First and foremost, my utmost gratitude to my supervisor, Prof. Niloy Mitra, for all of what he has offered me. I deeply thank him for believing in me and giving me the chance to be lucky enough and be one of his students. Despite the long distance and time difference, he has always been patient and willing to generously support me with his invaluable guidance, feedback, advice and encouragement. His passion and enthusiasm have inspired me to enjoy researching in this field. My appreciation for him is much more than what I can express in a few lines.

My thanks extend to Yongliang Yang for his valuable help throughout my work. He has never hesitated to answer my queries, discuss new ideas and help me fix coding issues. I also thank him for providing me with the framework upon which I built this thesis work. My appreciation also extends to Zheng Youyi for helping me with the optimization problem and Suhib Alsisan for the valuable discussions. I owe gratitude to Mohamed Ibrahim for his help at different times and the exchange of useful ideas.

I am thankful for having the privilege to learn from a great group of faculty members and researchers at the Geometric Modeling and Scientific Visualization Center.
(GMSV). My special thanks to Prof. Helmut Pottmann for continuously asking about my work progress. I also thank Dongming Yan for his kind support all the time. I have also been blessed with a cheerful and friendly group of fellow students at GMSV. They have been always there willing to help.

Not to forget my greatest blessing of having my dearest friend Sara Shaheen who has always been by my side. I thank her for being there to share the joy and laughs at the sweetest moments and her valuable advice at the hardest times. I would like to also thank a few of my friends: Amal Abolhasan, Imene boudellioua, Doha Hamza, among many others for having fun, coffee breaks, activities and making life at KAUST an enjoyable experience. My special sincere thanks to my roommate Basmah Altaf for all the wonderful moments we spend at home and for her kind wishes and positive thoughts that boost me up.

My deepest gratitude goes to my source of inspiration and strength, to my parents who were beside me all the times. I thank them for their prayers and faith, for tirelessly supporting me throughout my studies, for encouraging me to pursue my dreams and look for the better and for everything they have provided me with until I have reached this place. Words can never be enough to express my gratitude for them. I am also grateful to my brothers Abdullah and Muhammad and my sisters Abeer and Yasmeen for their support and cheerful talks, for bearing with me and listening to me during tough times and for helping me to keep faith in myself.

My gratitude also extends to many other friends and family members. Thank you all. Nothing was possible without your great help and support.

Gratitude is the fairest blossom which springs from the soul. –Henry Ward Beecher
dedicated to my parents,

Jalilah and Nabeel
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>4</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>6</td>
</tr>
<tr>
<td>List of Illustrations</td>
<td>11</td>
</tr>
<tr>
<td>List ofTables</td>
<td>16</td>
</tr>
<tr>
<td>I  Introduction</td>
<td>17</td>
</tr>
<tr>
<td>I.1 Motivation</td>
<td>19</td>
</tr>
<tr>
<td>I.2 Problem Statement</td>
<td>20</td>
</tr>
<tr>
<td>I.3 Contribution</td>
<td>21</td>
</tr>
<tr>
<td>II Related Work</td>
<td>23</td>
</tr>
<tr>
<td>II.1 Segmentation</td>
<td>23</td>
</tr>
<tr>
<td>II.1.1 Morphological segmentation</td>
<td>24</td>
</tr>
<tr>
<td>II.1.2 Symmetry and Repetitive Patterns Detection</td>
<td>25</td>
</tr>
<tr>
<td>II.1.3 Higher-order knowledge based segmentation</td>
<td>27</td>
</tr>
<tr>
<td>II.2 Facade Modeling</td>
<td>28</td>
</tr>
<tr>
<td>II.3 Texture Generation</td>
<td>31</td>
</tr>
<tr>
<td>III Façade Partitioning</td>
<td>34</td>
</tr>
<tr>
<td>III.1 Preprocessing</td>
<td>35</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>III.2 Structure Discovery</td>
<td>35</td>
</tr>
<tr>
<td>III.2.1 Repeated windows detection</td>
<td>36</td>
</tr>
<tr>
<td>III.2.2 Regulating detected windows</td>
<td>40</td>
</tr>
<tr>
<td>III.3 Interactive refinement</td>
<td>46</td>
</tr>
<tr>
<td>III.4 Sky and Ground Detection</td>
<td>46</td>
</tr>
<tr>
<td>III.4.1 Sky Detection</td>
<td>47</td>
</tr>
<tr>
<td>III.4.2 Ground Detection</td>
<td>50</td>
</tr>
<tr>
<td>IV Interactive Façade</td>
<td>51</td>
</tr>
<tr>
<td>IV.1 Image Decomposition</td>
<td>52</td>
</tr>
<tr>
<td>IV.2 Grid Edits</td>
<td>53</td>
</tr>
<tr>
<td>IV.2.1 Static resize</td>
<td>54</td>
</tr>
<tr>
<td>IV.2.2 Edit selected windows</td>
<td>55</td>
</tr>
<tr>
<td>V Results and Discussion</td>
<td>56</td>
</tr>
<tr>
<td>V.1 Evaluation</td>
<td>56</td>
</tr>
<tr>
<td>V.2 Visual Comparison</td>
<td>61</td>
</tr>
<tr>
<td>V.3 Limitations</td>
<td>64</td>
</tr>
<tr>
<td>V.4 User Interaction</td>
<td>64</td>
</tr>
<tr>
<td>V.5 Textured Models</td>
<td>65</td>
</tr>
<tr>
<td>VI Conclusion and Future Work</td>
<td>68</td>
</tr>
<tr>
<td>VI.1 Summary</td>
<td>68</td>
</tr>
<tr>
<td>VI.2 Future Research Work</td>
<td>69</td>
</tr>
<tr>
<td>References</td>
<td>70</td>
</tr>
</tbody>
</table>
List of Illustrations

I.1 An overview of urban reconstruction approaches. The methods are roughly grouped based on their outcome. Figure courtesy of [1] . . . 18

I.2 The various input data types which use imagery or LiDAR scans acquired from ground or air level. Figure courtesy of [1] . . . . . . . . . 19

I.3 Thesis outline. Façade images are first collected. Then, the façade is partitioned to identify the repeated windows. Later, the user can interactively edit the façade. Finally, we use projective texturing to map the newly generated images on a simple building model. . . . . 22

II.1 An example of detecting windows outline based on the edges by creating histograms for the horizontally and vertically projected edges and identify the peaks at which the windows are located. (a) The horizontal projection of the edges (b) The vertical projection of the edges (c) The extracted windows at the intersections of the peaks of both projections histograms. Figures courtesy of [2] . . . . . . . . . . . . . . . . 24

II.2 An example of detected regular patterns in 3D architectural scans. Figure courtesy of [3] . . . . . . . . . . . . . . . . . . . . . . . . . . 26

II.3 Detected repeated elements under perspective skew in the original images. Figure courtesy of [4] . . . . . . . . . . . . . . . . . . . . . . . . . . 26
II.4 Annotated façade images with their input images and the colored line strokes used to train the GMM (Gaussian Mixture Model) used for the annotation. Figure courtesy of [5].

II.5 Comparison of the results of the automatic method of [6] (left, 409 shapes, excluding windows matched from a template library) to the interactive method of [7] (right, 1,878 shapes). Figure courtesy of [7].

II.6 An example of a detailed model generated with the multi-modal method on synthetic data. By performing a virtual scan of the building using ray-casting sampling technique, an accurate depth layers is computed (middle). Moreover, the repetitions are used to propagate the geometry among the missing data and generate a textured polygonal 3D façade model (right). Figure courtesy of [8].

II.7 An example showing the use of images in texture mapping on façade models. On the left is the façade image used for texturing, in the middle is the textured model and on the right is the block geometry. Figure courtesy of [9].

II.8 An image pair of the input façade and the result after removing the occluding objects by symmetry propagation. Figure courtesy of [10].

II.9 Texture synthesis results of an image of Copenhagen harbor by re-ordering strips of the source image. The new generated images represent paths in a graph describing the space of synthesizable images. The image marked with the pink square is the source image. Figure courtesy of [11].

III.1 Edge detection and processing to eliminate the non-axis aligned segments. (a) Facade image (b) Edge map (c) Edge map with axes aligned edges.

III.2 Structure discovery pipeline.
III.3 Score plots - Blue: NCC score, Red: Edges score, Green: (NCC+ Edges) and Yellow: the smoothed final score.

III.4 Clustering and regularization of the detected repeated elements by comparing their attributes. (a) Detected repetitive instances (b) Regulated grid with regulated instances.

III.5 Visualizing the confidence weights with circles; the center of each circle represents the top left corner of a potential window and its radius corresponds to the confidence weight. Blue rectangle is the template $T$; white circles represent the potential windows positions, yellow and green circles represent the projections of the white circles onto the X and Y axes, respectively.

III.6 The splitting lines determined by the optimization. Each intersection of these lines denote a position of a window.

III.7 An example on the sky and ground detection. (a) The façade rectified image (b) An image specifying the detected regions; the blue area denotes the sky and green area denotes the ground.

III.8 Sky region detection. (a) Original image (b) Extracted blue channel (c) Detected edge map (d) Closed boundaries (e) Sky region specified in white.

IV.1 A set of generated models by retargeting existing models even when they have irregularities. The input models, shown on the left, were modeled by an artist after real-world buildings. Figure courtesy of [12]

IV.2 Image decomposition into background and foreground layers.
IV.3 Some examples of the results of the possible interactive edit operations.

(a) The original rectified façade image with the detected grid (b) The building is resized and new columns were added (c) Reposition the columns and resize the middle one (d) Delete individual windows (e) Decrease the width and some columns were deleted automatically

V.1 Some results of the façade partitioning algorithm. We show the user selection of the region of interest for the grid and the dominant window (left) as well as the detected grid (right). The red lines indicate the splitting lines whose intersections specify the locations of the repeated windows shown in yellow.

V.2 Robustness to illumination variations. We show the detected grid where the red lines indicate the splitting lines whose intersections specify the locations of the repeated windows shown in yellow.

V.3 Robustness to occlusions. We show the detected grid where the red lines indicate the splitting lines whose intersections specify the locations of the repeated windows shown in yellow.

V.4 An example showing the robustness of our detection algorithm against both illumination changes and occlusions when they both exist in the façade image.


V.8 Identifying the grid mainly depends on locating the building floors as a first step. (a) The user selection of the grid region and the dominant window (b) The horizontal lines represent the top side of each detected floor. There are two wrong detected floors; the green line which will be discarded by the optimization due to the regularity term and the orange line which is an inaccurate location for this floor (c) The final detected grid

V.9 An example of using the images generated during the interactive session for texture projection on a simple building model.
List of Tables

V.1 The running time of the repetition detection algorithm on a set of images is an order of a few seconds. However, an average of 85\% of this time is taken by calling the MATLAB code. . . . . . . . . . 57
Chapter I

Introduction

A great deal of efforts has been made with the aim of modeling and reconstructing the appearance of our real world. One particular example of these efforts has been made in modeling architectural scenes. This has very useful and attractive applications such as exploring different places and various cities unrestricted to fences, distances or jetlag. From an economical point of view, there are enormous benefits of being able to quickly generate high quality digital worlds in the growing market of virtual worlds.

One of the main and most important consumers of the virtual worlds is the entertainment industry. Virtual cities and urban models have been a key feature of recent computer games and movies. Moreover, digital maps industry has become available in many everyday usage items such as mobile phones, cars and desktop computers. Examples of applications that provide aerial view include Google Earth, Microsoft Virtual Earth and Microsoft Bing Maps. Others provide ground level view such as Google Street-view and Microsoft Live Street-Side.

Other applications of two-dimensional and three-dimensional simulations include cyber tourism, city planning, security training, emergency management and disaster control. Virtual worlds also help to plan and manage traffic and public transporta-
Figure I.1: An overview of urban reconstruction approaches. The methods are roughly grouped based on their outcome. Figure courtesy of [1]

tion as well as telecommunication. Additionally, urban modeling is very suitable for archaeological research. The data can be acquired directly from building remains and be combined with the knowledge from a variety of sources such as maps, books and paintings. The generated models can be of a great value for education and planning purposes.

To achieve the aforementioned goals, many researchers in multiple fields exert a great deal of efforts to provide tools that support efficient generation of highly detailed urban models. Researchers follow different approaches and aim to produce a wide range of various outcome to serve multiple purposes. These approaches are summarized in Figure [1]

There are various data types that can be used as input sources for urban reconstruction methods. They vary among imagery and 3D scans. These input types were discussed in the STAR report [1] as shown in Figure [2]. Imagery is the most common data type as it is easy to obtain, store and exchange. Moreover, there are various techniques that facilitate image processing which yield a wide range of useful information from images. More specifically, ground level imagery has been the most
Figure I.2: The various input data types which use imagery or LiDAR scans acquired from ground or air level. Figure courtesy of [11].

obvious data source with the huge amount of images which could be uploaded and shared through the Internet. On the other hand, aerial and satellite images were restricted for the professional usage for many years. Recently, they have become more available with the advances of Web-mapping projects such as Google Maps. Another input data source is LiDAR data. These scans are generated by projecting laser light onto surfaces and capturing the reflected backscattered light. The structure of the scanned model is determined with the time-of-flight principle [13]. Many urban reconstruction algorithms utilize LiDAR scans in its both forms; ground and aerial level. In addition, many techniques incorporate imagery and LiDAR scans to harvest the advantages of using both. Typically, images are easy to acquire and of high resolution but view dependent and inherently lack depth information. LiDAR scans are intuitively easier for the reconstruction algorithms as they are already in the 3D form but incomplete, noisy, sparse and difficult to acquire. Combining both data types produces a complementary nature of the input data.

I.1 Motivation

The goal of urban modeling spans a wide range of specialities such as entertainment, city planning, education and navigation among many others. Urban reconstruction
is a complex problem that requires a diverse set of tasks along the reconstruction pipeline such as data acquisition, data analysis, geometry extraction and modeling as well as texture generation. Various methods were introduced and devised to approach these tasks and produce high quality results. Nonetheless, some of these methods are computationally expensive especially those that depend on huge optimization problems.

Alongside, urban reconstruction methods encounter many challenges. Most computer based approaches aim at providing full automatic solutions which turns out to be hard to achieve. Moreover, it is common that the quality of automatically recovered models is quite low and cannot be used in the industries that require high quality standards. Therefore, many approaches incorporate the user to provide the necessary guidance to the algorithm which also has the problem of poor scalability. Systems that require a lot of user intervention do not scale well with huge amounts of data. Moreover, other problems are related to the data and their acquisition. Typically, urban environments are organized into narrow streets and buildings are occluded with various obstructions. This produces noisy and incomplete data. In addition, imagery has other problems such as glass and mirror surfaces as well as different shadow and lighting conditions. These challenges were discussed in more details in the STAR report [1].

1.2 Problem Statement

In this thesis, we seek to provide solutions that address problems such as façade image processing and editing using a single image as our input data. We aim at providing an easy to use tool that has a balance between the automation and interaction by involving the user at the beginning to guide the automatic algorithm with simple cues and towards the end to refine the results. The main focus of this thesis is to answer
the question of how to process the façade image efficiently and then manipulate the façade in order to generate new textures for building models. This research aims to fulfill humans curiosity for exploring variations without paying financial costs induced by prototyping, for instance. We formulate the problem as a set of research questions which split the subject into steps as follows:

- How can we analyze and partition a single façade image in order to identify its structural grid such that the repeated windows are determined?
- How can we generate new façade structures with simple, smart and intuitive user interaction operations?

I.3 Contribution

The main goal of this thesis is to process a single façade image and formulate an understanding of its façade structure. The proposed method assumes that the façade exhibits a rectilinear nature that can be defined as a regular grid. We collect the input images either from the Internet, from the *Ecole Centrale Paris Facades Database* [14] or by capturing photographs of buildings with a handheld camera. The first part of this thesis focuses on partitioning a façade image by exploiting the repeated windows to identify the structural grid as discussed in chapter III. We propose an algorithm that efficiently detects the repeated windows even those with different configurations such as open or close windows. Furthermore, our algorithm is robust to extreme illumination changes and occlusions. In the second part of the thesis presented in chapter IV, we show how to utilize the detected grid in order to support smart edit operations for façade manipulation. We aim at providing the user with an easy to use tool that supports intuitive edit operations of the image in order to generate new façade variations. In chapter V, we present some results which we obtained with our
Figure I.3: Thesis outline. Façade images are first collected. Then, the façade is partitioned to identify the repeated windows. Later, the user can interactively edit the façade. Finally, we use projective texturing to map the newly generated images on a simple building model.

algorithm as well as some examples of using the newly generated images to texture building models.

The main contribution of this thesis is a framework that supports the following (see Figure I.3):

• Partition a façade image and identify its structural grid.

• Interactively manipulate the façade in order to generate new texture images with façade variations.
Chapter II

Related Work

In this chapter, we provide an overview of the work done in the field of urban and facade modeling and reconstruction. During the past years, this topic has gained the attention of many researchers from different fields such as computer graphics, computer vision and machine learning. This enriched the literature with a good amount of work. This topic is still a very active research topic and is going through a good progress. In the following sections, we discuss some of the main methods and strategies that have been explored.

II.1 Segmentation

Under this category lies the methods that propose algorithms for detecting the structure of facade data whether images or laser scans. Most of these techniques involve processing of the façade data with classical techniques such as edge [15], corner [16] or feature [17] detection. Using these methods gives information that supports local segmentation which is usually not reliable for the detection of general façade structure. Therefore, many new methods involve global consideration of the façade structure such as symmetry and repetitive patterns detection. Moreover, other methods tend to combine low-level processing techniques with high level techniques such as unsu-
supervised clustering \cite{18} or machine learning \cite{19}. Some of the work that follows these approaches is discussed next.

II.1.1 Morphological segmentation

This section addresses the methods that segment the facade based on local properties. One of the proposed methods in this area is the work of Lee and Nevatia \cite{2} who propose to project the edges among a façade image horizontally and vertically to get information about the edges distribution (see Figure II.1). They assume that these projections have peaks at the locations of the windows. This defines an outlines of the windows which is later refined against the image edges and used to approximate the subdivision of the facade.

![Figure II.1: An example of detecting windows outline based on the edges by creating histograms for the horizontally and vertically projected edges and identify the peaks at which the windows are located. (a) The horizontal projection of the edges (b) The vertical projection of the edges (c) The extracted windows at the intersections of the peaks of both projections histograms. Figures courtesy of \cite{2}]

Tsai et al. \cite{20} proposed a method that uses morphological operations to generate texture and map it to 3D buildings models. They use a sequence of video frames to extract overlapping images by identifying the corners as interest points to register the images and integrate them to produce complete texture images. Occluding objects
were identified, e.g. occluding vegetation was measured by calculating a greenness index (GI). Then, these unwanted objects were removed and the images were completed by mirroring neighboring textures.

II.1.2 Symmetry and Repetitive Patterns Detection

Symmetry is a topic that has inspired researchers in many areas. Buildings and architectural scenes exhibit a great deal of symmetry and regular patterns. Recent approaches tend to focus on detecting the symmetries in order to partition the façade and define its structure.

An early work on detecting symmetry in images was done by Liu et al. [21] who proposed a symmetry based method that performs paper cut patterns on images and then create new synthesised images with different symmetry groups. Another method for detecting affine symmetry groups in 2D images was introduced by Loy and Elklundh [22] who group feature points to detect and characterize their underlying symmetries. Similarly, there is a good amount of work for detecting symmetries in 3D data such as the work proposed by Mitra et al. [23] that process geometric models and extract a compact representation of their Euclidean symmetries. This was followed by an algorithm [24] that symmetrizes the geometry of 2D and 3D shapes as well as methods to discover regularities in 3D scans [3, 25]. An example of these detection results is shown in Figure II.2.

Another algorithm in this class of work was proposed by Wu et al. [4] which detects the repeated elements under perspective skew at which they rectify the image by finding the elements’ salient boundaries as shown in Figure II.3. In their follow up work [26], they used this detected grid of symmetry to reconstruct dense 3D structure.

Nan et al. [27] proposed an interactive tool to discover underlying architectural structures such as repetitive patterns with a moderate user interaction over a noisy point cloud. The user initializes the boxes which are smart as their sizes and locations
Figure II.2: An example of detected regular patterns in 3D architectural scans. Figure courtesy of [3]

Figure II.3: Detected repeated elements under perspective skew in the original images. Figure courtesy of [4]
are automatically adjusted based on an optimization between a data-fitting term and a contextual term. Another method that detects repetitive patterns along the axes in orthogonal façade images was proposed by Musialski et al. \cite{musialski}.

Ceylan et al. \cite{ceylan} introduced a method for image-based 3D buildings reconstruction that relies on symmetry priors. Their extracted symmetry patterns can be used for interactive and intuitive model manipulations. Going a step further beyond the detection of the repeated patterns, Alsisan and Mitra \cite{alsisan} introduced a method to encode different configurations of repeated facade structural elements by using a Markov Random Field (MRF) optimization to label these elements.

Recently, Shen et al. \cite{shen} presented an automatic method to discover high-level facade structures in images and point clouds. Their main observation about these structures is that they are formed by concatenated and/or interlaced grids. This method has the advantage of adaptive specification of the parameters such as the splitting direction and the number and location of splitting planes.

In this thesis, we propose a method that exploits the repetitive pattern of the façade windows in order to identify the structural grid in a rectified façade image.

**II.1.3 Higher-order knowledge based segmentation**

In this section, we discuss the methods that are driven by hierarchical rule based segmentation techniques. They subdivide the façade into smaller parts which are arranged based on hierarchical context free grammar rules.

This approach was followed by Muller et al. \cite{muller} who subdivide a simple regular facade into floors and tiles in a synchronized manner. This reduces the façade into a set of irreducible elements. The result is then turned into procedural rules by using a template grammar to describe the floor pattern. Another example in this category was proposed by Teboul et al. \cite{teboul} who create perceptual interpretation of buildings by performing supervised learning using shape grammar priors and random walks on
these learned models. Later, they extended their work to use recursive binary split grammar and reinforcement learning to annotate the facades (as shown in Figure II.4) into elements such as windows, doors, roofs, etc using training data.

II.2 Facade Modeling

In the seminal work ofDebevec et al. [33], they introduced an interactive method that combines image-based modeling with geometry-based modeling. They use line features in images and 3D polyhedral primitives to reconstruct the model with view-dependent texture mapping. The user fits line segments on the image, specify their correspondence on the generated model and may choose to constrain some properties of the blocks. When the number of images increases, the manual process becomes tedious to specify the features and their correspondences.

Automatic creation of 3D models of high visual quality from a single rectified facade image was done by Muller et al. [6]. They presented an algorithm that combines procedural modeling of shape grammars with image analysis to derive a meaningful hierarchical facade subdivision. They encode the symmetry in the form
Figure II.5: Comparison of the results of the automatic method of \[6\] (left, 409 shapes, excluding windows matched from a template library) to the interactive method of \[7\] (right, 1,878 shapes). Figure courtesy of \[7\]

of an irreducible facade. The grammar rules are defined automatically by exploiting the edges and mutual information to identify a repetitive pattern on a regular grid and cut the facade image into floors and tiles. Their detection is first done in the horizontal direction and then in the vertical direction. The result is then turned into procedural rules by using a template grammar to describe the floor patterns. This technique is very robust for analysis and detection of repetitive patterns. However, this method does not maintain vertical coherence and relies on strong assumptions of regularity, orthogonality and rectilinear grid of facade structure. Therefore, this algorithm fails to handle large differences due to reflected surfaces or multiple repetitive patterns. Moreover, the depth is manually assigned which makes it difficult for large scale modeling. Later, Musialski et al. \[7\] proposed an interactive framework for detailed modeling of building façades from images which is based on coherent based operations. It exploits the partial symmetries across the façade and mixes manual interaction with automatic splitting and grouping operations. This produces façade models with high level of details. A comparison of the results of these two approaches is shown in Figure II.5.

Image based modeling was proposed by Xiao et al. \[34\] in their semi-automatic
method that uses images captured along the streets and relies on structure from motion to obtain camera positions and initial point cloud for modeling. They proposed a systematic and automatic decomposition approach for both analysis and reconstruction. A building facade is considered as a flat rectangular grid or a developable surface where details are also rectangular elements on top of it. A rectified facade image is decomposed along detected edges using a top-down recursive subdivision to obtain a Directed Acyclic Graph of the facade. This decomposes the facade into rectilinear patches which are augmented with depth values obtained from the structure from motion. Then, it is followed by a bottom-up merging that maintains bilateral symmetry to handle repetitive patterns. Further refinements are allowed by the user such as to control the decomposition or depth assignment. This approach relies on the initial boundary specified for the facade which is probably assigned by the user. Moreover, as the decomposition relies on the image edges, it might produce over segmented partitioning especially with additional details such as balconies. In their follow-up work [9], they replaced the need for the interactive strokes by an automatic multi-view image-based approach to generate 3D models.

It is common that the models generated with automatic methods are not very accurate. Usually, in order to create models with the desired level of details, the user needs to refine the models which can be a tedious task. Therefore, giving the user some control at an early stage can be of a great advantage. An example of an interactive modeling system [35] which combines the user drawn outlines on the image with the multi-view information to automatically reconstruct the 3D model. In the automatic context, Ning et al. [36] proposed a hierarchical facade reconstruction framework by combining facade structures, detailed windows prorogation, hierarchical model consolidation and contextual semantic representations.

Combining different types of data such as images and 3D scans can be very useful as they are complementary to each other. Li et al. [8] decompose the facade images
Figure II.6: An example of a detailed model generated with the multi-modal method on synthetic data. By performing a virtual scan of the building using ray-casting sampling technique, an accurate depth layers is computed (middle). Moreover, the repetitions are used to propagate the geometry among the missing data and generate a textured polygonal 3D facade model (right). Figure courtesy of [8].

into planar fragments and diffuse depth information from the corresponding 3D data. Following this method enables repetition detection which helps to propagate geometry, remove outliers and enhance the 3D model. This produces detailed, layered and textured models as shown in Figure II.6.

II.3 Texture Generation

The term texture generation means mostly synthesizing one or more images to create a new façade image which can be later used for texturing purposes. High quality façade imagery is a vital element for realistic representation of urban environments. Editing and retargeting an image is much easier and less expensive than manipulating the underlying geometry. Therefore, retargeting the façade images and projecting them on a simpler models can be of a great value to efficiently render models especially in complicated scenes within games and virtual environments. This can be addresses by projective texturing from perspective photographs. Many interactive systems allow the use of projective geometry such as ”Façade” [33] which projects textures on the reconstructed models. Others use a collection of images [35, 34, 9] or video frames [20]
Some researchers focus on creating a cleaner facade by removing shadows and foreign objects, such as the example shown in Figure II.8. This is mainly achieved by exploiting the symmetrical patterns within the facade [20, 10]. Musialski et al. [37] introduced a framework that generates ortho-rectified images from a set of photographs taken by a hand-held camera. They remove the occluders by exploiting the multi-view information.

Another direction considers resizing the facade images [38] by summarizing the translational symmetries which enables not only pixels manipulations but manipulations of semantic cells in the lattice. Lefebvre et al. [11] proposed a synthesis method that utilizes a shortest path problem in a graph describing the space of images while allowing the user to control the repetitions as well as specifying positional constraints.
Figure II.9: Texture synthesis results of an image of Copenhagen harbor by reordering strips of the source image. The new generated images represent paths in a graph describing the space of synthesizable images. The image marked with the pink square is the source image. Figure courtesy of [11]

An example of their results is shown in Figure II.9. In this thesis, we present a new interactive method for manipulating the façade image to allow the user to edit the grid structure as well as the individual windows and later use the generated image to texture a simple 3D building model.
Chapter III

Façade Partitioning

Understanding the façade and partitioning it into its structural elements is an important factor in many applications and has many usages such as façade planning, designing and modeling. In this chapter, we propose an efficient and robust algorithm that addresses the problem of partitioning typical buildings façades with their windows organized in 2D grid patterns. Our approach exploits the rectilinear repetitive nature of this distribution to identify the positions of the windows within the façade. The algorithm initially relies on having a template window specified by the user. Then, it detects the repetitive instances of the template on an orthogonal image along the X and Y dimensions. The potential positions of the windows are later regulated to define a grid of axes-aligned splitting lines which delineate the façade into horizontal and vertical repetitive instances. Each intersection of the splitting lines indicates a position of a repeated window. Once the grid is identified and the windows are located, the user has the control to interactively refine the results. The following sections discuss the algorithm in details.
III.1 Preprocessing

The facade image is first rectified by extracting the vanishing lines [4]. Then, we perform image-space edge detection using the open-source EdgeLink [39] which is based on Canny edge operator [15] to define the edge map and then form a collection of 2D line segments. The set of edges is further processed in order to eliminate the non-horizontal and non-vertical edges by computing the slope of each line segment and excluding those whose slope violates a threshold value (we use a threshold = 0.1). In addition, we remove all the line segments that are shorter than a minimum length which is set to 10 pixels in our implementation, see Figure III.1.

![Image](image.png)

(a) Facade image (b) Edge map (c) Edge map with axes aligned edges

Figure III.1: Edge detection and processing to eliminate the non-axis aligned segments. (a) Facade image (b) Edge map (c) Edge map with axes aligned edges

III.2 Structure Discovery

There is a good collection of available techniques in the literature that propose a variety of methods to discover the grid defining the structure of the facade elements which could be interactive, automatic or a combination of both. We approach the structure discovery problem with a semi-automatic method assisted initially by the user. The main goal of this stage is to identify the general structure of the facade in the image. We target facades that exhibit a natural rectilinear structure in the horizontal and vertical directions. The input to this stage is a single rectified image
together with the line segments information that were processed in the pre-processing stage to include only the segments along the horizontal and vertical directions. The input is processed to produce a set of splitting lines defining the facade grid as tiles of horizontal and vertical translational symmetries.

The algorithm is subdivided into multiple stages as shown in the pipeline in Figure III.2. As an initial step, the user marks the region of interest (ROI) for the grid boundary and a window in the facade as to identify the main element and guide the automatic detection of similar instances across the selected ROI of the rectified image. This is followed by an automatic process for defining the grid structure. Given the selected window, the potential building floors are located by detecting its vertical repetitions within the grid boundary. Then, for each detected vertical repetition, its repeated elements are determined along the horizontal direction. This produces a set of irregular elements that identify the locations of many possible repeated windows. After that, the set of elements is regulated to define the main grid structure defined by a set of horizontal and vertical splitting lines. Moreover, we give the user the control to refine the resulting grid.

III.2.1 Repeated windows detection

Given a selected stencil, we find the other repeated stencils along one direction whether horizontally or vertically by sliding a stencil of the same size as the template along that direction and compute a confidence score value for each stencil. The sliding can be with a varying step size. We use a step size of one pixel to ensure that the complete image was scanned. The basic idea for computing the score depends on two main components which are the image and the edges terms. The image term quantifies the similarity between the template stencil and the sliding one with respect to the pixels intensity. The edges term measures the similarity between the two stencils with respect to the detected edges along the horizontal and vertical directions.
Figure III.2: Structure discovery pipeline.
By maximizing the score value, we can identify the potential locations of the matched repeated windows. We compute the score as follows:

$$w_i = w_{image} + w_{edges} \quad (\text{III.1})$$

*Image term* ($w_{image}$): This term measures the similarity between the sliding window and the template window which is computed by evaluating the Normalized Cross Correlation (NCC) \cite{40}. This correlation quantifies the similarity between two signals and is a common technique for feature matching. High NCC score implies a good match between the two signals.

Let $t$ be the template stencil that we want to find its repeated instances and $f$ be the sliding stencil, both having the same size. We compute the NCC score to evaluate the similarity between the two image stencils as shown in equation III.2.

Notice that we use RGB images; however, we convert them into gray scale images and compute their NCC score. One advantage of using the Normalized Cross Correlation is that it is invariant to changes in images such as those due to different lighting and shadowing conditions across the image which is achieved by normalizing the image into unit length.

$$w_{image} = \frac{\sum_{x,y}[f(x,y) - f'][t(x,y) - t']}{\left\{\sum_{x,y}[f(x,y) - f']^2 \sum_{x,y}[t(x,y) - t']^2\right\}^{0.5}} \quad (\text{III.2})$$

where $f'$ is the mean value of the sliding stencil and $t'$ is the mean value of the template.

However, using NCC alone does not give an accurate measure for matching the template (see Figure III.3-blue plot). Therefore, we incorporate edges details into the scores as described in the following section.

*Edges term* ($w_{edges}$): In order to enhance the reliability of the detection, we add another similarity measure that is based on the detected edges in the image. We use
the preprocessed line segments, i.e. the horizontal and vertical edges. Let $L' = \{l'_j\}$ denote the set of horizontal and vertical line segments in the template stencil and let $L' = \{l'_j\}$ be the line segments of the sliding stencil. For each line $l' \in L'$ we select all parallel lines $l^t \in L'$ with slope difference $\leq \varepsilon$ using a suitable threshold $\varepsilon$ and with $dist(l', l'^t) \leq \delta$ where the distance is computed as follows:

$$dist(l', l'^t) = \begin{cases} 
\text{difference in y coordinate,} & \text{if horizontal lines;} \\
\text{difference in x coordinate,} & \text{if vertical lines.}
\end{cases}$$

(III.3)

The score is defined as the number of matched parallel lines which lie within a suitable distance threshold $\delta$. High score values mean that the stencils have more edges that match the template well as shown in Figure III.3-red plot. After that, the $w_{image}$ and the $w_{edges}$ are normalized by their maximum values. Then, they are combined to form the final score for each stencil as shown in Equation III.1.

By looking at the plot in Figure III.3 we find that the repeated windows occur at the local maximums of the scores. However, the scores are not smooth and detecting the local maximum yields overlapping windows. Therefore, we smooth the scores by applying Gaussian filter five times which is of size 15 and sigma = 3.5. The yellow plot in Figure III.3 represents the smoothed score. The positions of the repeated elements are determined by evaluating the first and second derivatives. The extremes are located at the points whose first derivatives evaluate to zero. The first derivative of the scores $w_i$ is computed as:

$$w'(x) = \frac{w(x + \Delta x) - w(x)}{\Delta x}$$

(III.4)

To identify the local maximum positions, we compute the second derivative as following:
\[ w''(x) = \frac{w(x + \Delta x) - 2w(x) + w(x - \Delta x)}{\Delta x^2} \]  

(III.5)

Then, we select those whose second derivative evaluates to a value \( \leq 0.001 \).

We apply the above algorithm for detecting the vertical repetitions of the user specified stencil to automatically specify the number and positions of the floors as buildings are organized into floors. Then, we identify the horizontal similar windows in each floor. The outcome at this stage is a set of stencils specifying the positions of the candidate repeated windows. However, these windows might be irregular or detected in the wrong places. Therefore, we regulate the detected elements as described in the following section.

### III.2.2 Regulating detected windows

Given the set of detected potential windows, we aim at regulating these windows to define a grid that specifies their positions. We first attempted this problem by regu-
lating the windows with a simple and trivial windows’ attributes comparison to identify the positions of the main splitting lines. However, this produces many outliers. Therefore, we regulate the windows by targeting the problem with an optimization formulation. Both methods for regulating the windows are discussed next.

**Comparing windows attributes**

The main goal of this step is to eliminate the outlier windows and group the valid windows into rows and columns. This is first done by considering the windows attributes and comparing them in order to define the positions of the splitting lines and the general façade structure grid (see Figure III.4). The details are as follows:

- The rows are already defined by detecting the vertical repetitions of the user selected window. To remove any outlier, the rows that have horizontal repetitions less than the average number of repetitions in all the rows are discarded.

- The columns are identified by clustering the windows that have x coordinate within 20 pixels offset. Then, clusters which have a number of elements less than the average vertical repetitions are discarded.
Figure III.4: Clustering and regularization of the detected repeated elements by comparing their attributes. (a) Detected repetitive instances (b) Regulated grid with regulated instances.

**Optimization**

The other more efficient approach for defining the grid is by addressing the problem as an optimization formulation. We have a set of positions corresponding to potential top left corners for the windows such that each position is associated with a confidence value computed at the detection step as shown in Equation III.1. In order to visualize these points, we draw a circle at each position with a radius indicating the confidence at the corresponding position. Due to the nature of the problem which is to identify a rectangular 2D grid pattern that describes the windows positions, we can attempt the problem in the X and Y dimensions independently. As an initial step, we project all the points onto the X and Y axes in order to define two independent sets of points which are handled independently as shown in Figure III.5. Let us denote the set of projected points as $P = \{p_1, p_2, p_3, ..., p_n\}$ with their associated confidence weights $W = \{w_1, w_2, w_3, ..., w_n\}$. Our goal is to find a set of points $C = \{c_1, c_2, c_3, ..., c_k\}$ that best explains the points in $P$. We assume that we know the number of splitting lines $k$ which is four in our implementation.
Figure III.5: Visualizing the confidence weights with circles; the center of each circle represents the top left corner of a potential window and its radius corresponds to the confidence weight. Blue rectangle is the template $T$; white circles represent the potential windows positions, yellow and green circles represent the projections of the white circles onto the X and Y axes, respectively.
For simplicity, we start by exhaustively defining the potential sets \( S = \{S_1, S_2, S_3, \ldots\} \) such that each set contains \( k \) points i.e. the original template position \( T \) and \( k - 1 \) other points from \( P \). A set of points is considered to be valid only if the stencils at the corresponding positions do not overlap. In the future, we plan to select these sets by following a better approach such as a Markov Random Field (MRF) [41] formulation. In order to identify the set \( S \) that best describes the grid, we compute an energy function for each set \( S_i \). After that, we locate the set with the minimum energy value whose points correspond to the optimal positions of the splitting lines. We formulate the problem as follows:

\[
\min_{S_i \in S} E(S_i) = E_{\text{data}}(S_i) + \lambda E_{\text{reg}}(S_i) \tag{III.6}
\]

where \( \lambda \) determines the relative contributions among the data and regularity terms, which is taken as 0.1 in our implementation. The following sections describe the elements of the energy function in details.

Data term \( (E_{\text{data}}) \): This term measures the confidence of a given set \( S_i \) as well as how well it covers the complete set of points \( P \). We compute this term as following:

\[
E_{\text{data}}(S_i) = \sum_{i=1}^{k} \frac{1}{w_i \ast g_i} \tag{III.7}
\]

As shown in Equation [III.7], the data term involves two components. The first element \( w_i \) measures the extent of repetitions which is the confidence weight computed at the detection step as shown in Equation [III.1]. The other element \( g_i \) measures how well the points of the set \( S_i \) agree with the other points \( p_j \in P \) by estimating Gaussian falloff as follows:

\[
g_i(S_i) = \sum_{j=1}^{n} e^{-d_j^2 / 2s^2} \tag{III.8}
\]

where \( d_j \) is the distance between the points \( p_i \) and \( p_j \) and \( s \) represents the width
Figure III.6: The splitting lines determined by the optimization. Each intersection of these lines denote a position of a window.

or height of the template $T$ depending on the dimension of the optimization.

Regularity term ($E_{reg}$): A typical facade has a uniform distribution of distances among its neighboring windows. Therefore, we introduce this term to measure the regularity of the points $p_i$ among the set $S_i$. We find the spacing distribution among $p_i \in S_i$ as $\{d_1, d_2, d_3, ..., d_{k-1}\}$. Then, we measure the regularity by computing the standard deviation of the distribution as follows:

$$E_{reg}(S_i) = \sigma = \sqrt{\frac{\sum_{j=1}^{k} (d_j - \mu)^2}{k}} \quad (III.9)$$

where $d_j$ represents the spacing and $\mu$ is the average of spacing distances.

In order to find the splitting lines that specify the grid containing the windows, we minimize the energy given in Equation [III.6]. This identifies the set $S_i$ whose points
specify the positions of the splitting lines which are shown in Figure III.6.

### III.3 Interactive refinement

After the grid of elements is identified and displayed, the user can manually refine the repeated elements as follows:

- **Single-select:** Clicking on a window selects it for further refinement.

- **Multi-select:** Clicking on a horizontal or vertical splitting line selects all the windows associated with this splitting line.

- **Deselecting element(s):** Clicking somewhere else on the image deselects the selected element(s).

- **Remove:** The user can delete the selected element(s) by pressing the Delete key.

- **Horizontal reposition:** The user can drag the selected element(s) to a new position horizontally. For group repositioning, the user drags the splitting line to the new position.

- **Resize:** As the user hovers the mouse over the boundary of an element, the mouse cursor changes indicating an edge which could be dragged to resize the element. Resizing is possible in the horizontal and vertical directions. Resizing is also possible for a single or multiple selected elements.

### III.4 Sky and Ground Detection

Some images contain regions other than the facade such as the sky and the ground. Therefore, we provide a way for detecting the sky and the ground regions as shown
in Figure III.7. This is helpful to enhance the efficiency of the detection as to reduce the search space of the repeated elements. This is discussed in details in the following sections.

![Figure III.7: An example on the sky and ground detection. (a) The façade rectified image (b) An image specifying the detected regions; the blue area denotes the sky and green area denotes the ground](image)

### III.4.1 Sky Detection

We identify the sky region by following the work of Laungrungthip et al. [42]. Their general approach involves determining the edges of the sky region using pixel intensity gradients. Given a facade image, the steps involved in detecting the sky region are as follows:

1. **Extract a color channel:** There are many factors that affect the sky detection such as the varying weather conditions including bright sky with or without clouds and overcast sky with white or gray clouds. The aim of this step is to choose the channel that increases the contrast between the sky and the rest of the image and to convert the 24 bits of colour to 8 bits of colour as to prepare the image for edge detection. According to Laungrungthip et al. [42], the blue channel increases the contrast between the sky and the non-sky regions as well as decreases the contrast between the blue sky and the scattered clouds. Therefore, it was used for sky detection (see Figure III.8-b).

2. **Determine the boundary lines in the image:** The goal of this step is to
detect the edges in an image as to identify the boundary lines between the sky and the non-sky regions. There is a variety of methods for detecting the edges such as the Sobel, Canny, Prewitt, Robert and Laplacian of Gaussian edge detector methods [42]. The Canny operator [43], which is the most widely used operator, is used in the implementation (see Figure III.8-c). The Canny operator uses multiple steps to identify edges in an image. Initially, the image is smoothed with a Gaussian filter to eliminate the noise and remove the unwanted details. Then, it measures the change in image intensity at the edges by computing the edge gradient strength. After that, it evaluates the direction of the edges and apply non-maximum suppression to trace along the edges and suppress any pixel values that lie outside of an edge. This produces thin lines in the output image. Finally, hysteresis thresholding is used to eliminate the breaking up of edges. Two thresholds are used which are a high and a low threshold. Edge pixels are those whose value in the image is greater than T1. Then, any pixels that are connected to this edge pixel and that have a value greater than T2 are also selected as edge pixels. The output of the edge detection may also contain edges around clouds in the sky. Therefore, the thresholds should be adjusted to avoid such an effect. According to the implementation [12], we use a lower threshold equals to 50 and an upper threshold equals to 200.

3. Close the gaps in the identified boundaries: Using Canny edge operator produces an edge map that can be used to distinguish the sky and non-sky regions. However, it may leave some gaps in the boundary edges causing the sky area to spill through these gaps. To overcome such a problem, the morphological closing operator [44] is used on the binary edge map. The morphology closing operation performs a dilation followed by an erosion using the same structuring element, which is a uniform intensity pixel array (e.g. 3 × 3 or 5 × 5). For both operations, the middle pixel of the structuring element is placed over each pixel
in the image and the value of each structure element pixel is compared against the pixel it overlays. If all the pixels have the same value, the image pixel is set to 1 for the erosion algorithm and 0 for the dilation operation. In order to fill large gaps, a larger structure element is required. According to [42], we apply the morphological closing algorithm with a structure element of size equals to 3 (see Figure III.8-d).

4. **Identify the sky region:** Given the boundary lines that separate the sky from the non-sky regions, we can identify all the pixels that belong to the closed region of the sky using FloodFill algorithm [45] by placing seed points at the top left, middle and right corners of the image (see Figure III.8-e).

![Figure III.8: Sky region detection. (a) Original image (b) Extracted blue channel (c) Detected edge map (d) Closed boundaries (e) Sky region specified in white](image-url)
III.4.2 Ground Detection

By following the same technique used for detecting the sky, detecting the ground was implemented. The only difference was using the gray image instead of the blue channel. However, the results were not robust due to the various factors that affect the ground such as objects on the ground or shading. We plan to enhance this detection in the future.
Chapter IV

Interactive Façade

The need for generating variations of architectural facade models arises in many contexts and can be used in many applications within the industrial, educational and entertainment fields. Some of these applications include the development of virtual environments and computer games as well as in urban and architecture planning and training. An example of a work that addresses this problem is the work of Lin et al. [12] who proposed an interactive algorithm to retarget existing 3D models while preserving their general overall structure.

Images can be projected into simple 3D models as texture materials and convey a great deal of information about the nature of the facade structure. Therefore, manipulating the façade image can be used to generate a new image with a different façade structure. Editing the images is easier than handling the 3D models and can produce reasonable results. The newly generated images can be used for texture mapping and produce new effects for rendering architectural scenes. In this thesis, we present an interactive method which allows a range of intuitive image editing possibilities to manipulate the facade structure. We aim to provide a tool that helps in creating façade variations efficiently.
Figure IV.1: A set of generated models by retargeting existing models even when they have irregularities. The input models, shown on the left, were modeled by an artist after real-world buildings. Figure courtesy of [12]

**IV.1 Image Decomposition**

At this stage, we use the detected grid information to create the image layers that facilitate manipulating the grid. We decompose the rectified image into background and foreground layers as shown in Figure IV.2. The background layer is created by removing the regions of the grid windows from the image and then performing image completion [46]. The foreground layer is created by copying the detected grid windows into this layer. Later, the facade is interactively modified by manipulating the foreground layer.

In order to create the foreground layer, we keep two instances of the grid which we call the fixed and interactive grids. The fixed grid is a copy of the finalized grid after the user finishes the interactive refinements on the automatically detected grid. This copy is kept in order to provide the images for the windows within the facade and is not modified during the interaction session. The interactive grid is another copy of
the grid that is initialized with the same finalized grid. As the user manipulates the facade image, the changes are reflected on the interactive grid producing the newly modified facade. In the following section, we discuss the possible interactions that can be used to edit the facade structure.

### IV.2 Grid Edits

This section describes the possible operations that we support to perform on the facade to change its structure. As mentioned earlier, the facade is manipulated by handling the foreground layer of the image which contains the windows, thus, applying the changes on the interactive grid only. We allow the user to perform a range of simple and smart edit operations which are discussed in details as follows. Moreover, some examples of the results generated during the edit session are shown in Figure IV.3
Figure IV.3: Some examples of the results of the possible interactive edit operations. (a) The original rectified façade image with the detected grid (b) The building is resized and new columns were added (c) Reposition the columns and resize the middle one (d) Delete individual windows (e) Decrease the width and some columns were deleted automatically

IV.2.1 Static resize

Given the façade image and the bounding rectangle of the structural grid of the façade, we allow the user to change the dimensions of the grid by dragging a boundary edge until achieving the desired dimensions. This provides the option to adaptively alter the number of grid rows and columns by exploiting the regularity of the grid. In this thesis implementation, we allow these manipulations on the right vertical edge, thus, handling the columns within the grid. As the user drags a bounding edge, we consider the grid width and determine whether to add or delete a column. If the new width is enough to accommodate a new column, we compute its location based
on the regularity of the other columns and append it to the grid. We also add new windows to this column and copy their images based on the fixed grid. Similarly, if a bounding edge reaches a column and hits its windows, the column and its windows are deleted. These addition and deletion operations on windows are reflected on the interactive grid such that the fixed grid is untouched.

**IV.2.2 Edit selected windows**

The user can select a single window by clicking on its region or select multiple windows by clicking on their splitting line. Then, the user can manipulate these selected window(s) with a set of edit operations which are as follows:

- **Remove**: The user can delete the selected window(s) by pressing the Delete key.

- **Horizontal reposition**: The user can drag the selected element(s) to a new position horizontally. To reposition a set of multiple windows, the user drags the splitting line to the new position.

- **Resize**: In order to resize the selected window(s), the user drags a boundary edge until the desired size is achieved.
Chapter V

Results and Discussion

In this chapter, we discuss the results of our proposed algorithm with an emphasis on its strengths as well as its limitations. Furthermore, we discuss the proposed user interaction as well as the usage of the generated images for texture projection.

V.1 Evaluation

We implemented our algorithm using a mixture of C++ and MATLAB code running on an Intel Xeon X5680 @ 3.33 GHz (2 processors), 48 GB RAM and Windows 7 64-bit computer. The initial C++ framework upon which we build our work is provided by Yongliang Yang. We use EdgeLink [39] for edges detection, OpenCV [47] for basic image processing operations and PatchMatch [46] for image completion. The main work was involved in the first part of the thesis which is the detection of the repeated windows. Its running time is an order of a few seconds that is proportional to the grid ROI size specified by the user. In our implementation, we use an un-optimized MATLAB code and we invoke it through MATLAB engine in the main C++ program. This increases the needed running time by the algorithm as shown in Table V.1. We test our algorithm on a set of images that we either capture with a handheld camera, acquire from the Internet or use some images from the Ecole Centrale Paris Facades
Table V.1: The running time of the repetition detection algorithm on a set of images is an order of a few seconds. However, an average of 85% of this time is taken by calling the MATLAB code.

<table>
<thead>
<tr>
<th>Image No.</th>
<th>Repetition Detection Time</th>
<th>MATLAB Time</th>
<th>Ratio (MATLAB / rep. detection time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.029</td>
<td>7.858</td>
<td>87%</td>
</tr>
<tr>
<td>2</td>
<td>13.395</td>
<td>11.988</td>
<td>89%</td>
</tr>
<tr>
<td>3</td>
<td>1.177</td>
<td>0.834</td>
<td>70%</td>
</tr>
<tr>
<td>4</td>
<td>10.809</td>
<td>9.409</td>
<td>87%</td>
</tr>
<tr>
<td>5</td>
<td>3.245</td>
<td>2.726</td>
<td>84%</td>
</tr>
<tr>
<td>6</td>
<td>11.681</td>
<td>10.049</td>
<td>86%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>8.223</strong></td>
<td><strong>7.144</strong></td>
<td><strong>84%</strong></td>
</tr>
</tbody>
</table>

*Database [14].*

Our detection algorithm is able to detect repeated windows with different configurations i.e. open or close windows as shown in Figure V.1. We also tested our algorithm on images with extreme illumination changes. As we demonstrate in Figure V.2, the algorithm is robust against lighting changes. Furthermore, our algorithm is able to detect repeated windows which are obscured by various occluding objects as shown in Figure V.3. Combining both factors, our algorithm showed robustness against both illumination changes and occlusions within the same image as shown in Figure V.4. The main strength of our algorithm is due to the fact that we base our detection on two factors; the color and the edges of the compared image stencils. Thus, when one of them does not convey enough information, the other one complements it producing efficient results.
Figure V.1: Some results of the façade partitioning algorithm. We show the user selection of the region of interest for the grid and the dominant window (left) as well as the detected grid (right). The red lines indicate the splitting lines whose intersections specify the locations of the repeated windows shown in yellow.
Figure V.2: Robustness to illumination variations. We show the detected grid where the red lines indicate the splitting lines whose intersections specify the locations of the repeated windows shown in yellow.
Figure V.3: Robustness to occlusions. We show the detected grid where the red lines indicate the splitting lines whose intersections specify the locations of the repeated windows shown in yellow.
V.2 Visual Comparison

We compare the results of our partitioning algorithm with the method suggested by Teboul et al. [5]. For this purpose, we ran some of our test images using GraPeS system which implements façade parsing using reinforcement learning principles. Figures V.5, V.6 and V.7 present the results that we obtained by using GraPeS system. However, the results that we presented for our partitioning algorithm consider only a sub-region of these images since our implementation for the optimization assumes a fixed number of rows and columns. We plan to enhance the optimization and avoid this constraint in the future.
Figure V.5: Running GraPeS [5] on some of our test images.

Figure V.6: Running GraPeS [5] on some of our test images.
Figure V.7: Running GraPeS [5] on some of our test images.
V.3 Limitations

The repetition detection algorithm works by first detecting the vertical repeated instances of the user specified window, then, locating their horizontal repetitions as was discussed in section III.2. The algorithm fails to identify the structure grid in the case of its failure to locate the correct vertical instances of the dominant window which determines the building floors. An example of this case is provided in Figure V.8. This may occur when the user selects a window within a non-representative column such as a partially occluded column. In Figure V.4, if the user selects the dominant window from the first or second column, the algorithm fails to detect the correct grid.

![Figure V.8](image)

Figure V.8: Identifying the grid mainly depends on locating the building floors as a first step. (a) The user selection of the grid region and the dominant window (b) The horizontal lines represent the top side of each detected floor. There are two wrong detected floors; the green line which will be discarded by the optimization due to the regularity term and the orange line which is an inaccurate location for this floor (c) The final detected grid

V.4 User Interaction

We target causal users to provide them with an efficient method to generate new façade images. We provide a set of intuitive and easy to use operations. First, the
user is involved in selecting the dominant window of the façade. After the automatic
detection identifies the grid structure based on the selected window, the user has the
control to refine the results. When the user is satisfied with the detected grid, he/she
indicates finishing the detection step and the interactive mode starts. Moreover, the
user can save the detected grid and load it on the next session in order to avoid
recomputing it.

Once the grid is laid on top of the image, the user needs only to select a window
or multiple windows and perform the desired operations whether deleting, resizing
or horizontal repositioning of the selected window(s). Furthermore, the user can
manipulate the global structure by resizing the main grid area and columns of windows
are added or deleted automatically. All of this can be done with a few mouse clicks
as presented in chapter IV. Alongside, we visualize the effect in 3D by rendering a
simple 3D building model and texturing its front faces with the generated image and
reflect the interactive changes at run time as we discuss in the following section.

V.5 Textured Models

One of the main addressed problems in this thesis is to generate new façade images
that can be used for texture projection. We demonstrate the usage of the outcome
of our method on a simple building model. If we only consider the building and its
windows without the other details, we can create a building using cubes by rendering
a cube for the main building structure and a cube for each of the detected windows.
Then, we project the generated image on the front faces of these cubes. We assign
an initial depth to each of these elements. Besides, we allow the user to interactively
modify the depth of the windows by changing the value of a slider. Furthermore, as
the user interactively edits the façade image to add, delete, resize or reposition the
windows as presented in chapter IV the changes are directly reflected on the textured
model as shown in Figure V.9. This provides an easy to use, quick and efficient tool to generate new façades without dealing with the 3D geometry.
Figure V.9: An example of using the images generated during the interactive session for texture projection on a simple building model.
Chapter VI

Conclusion and Future Work

In this chapter, we conclude the thesis and provide a summary about the work and main contributions as well as discuss some thoughts about possible extensions and future research.

VI.1 Summary

In this thesis, we investigated the problem of processing a façade image to identify its structure which is utilized to manipulate the façade and generate new façade images. We proposed an interactive system for smart, fast and efficient façade editing. Our tool is unique in that the façade manipulation is achieved in 2D image level without going through the complications of the 3D geometry computations. The user interactively edits the façade image and, in turn, the changes are reflected on a simple 3D model which produces a variety of possible façades to explore.

In the image analysis phase, we propose an algorithm to discover the structural grid that defines the distribution of the windows among the façade based on an initial user selection of the dominant window. Then, we exploit these repetitions to provide a set of intuitive edit operations to generate a variety of façade structures mimicking modifications desired by designers and engineers among many others. Finally, we
project the newly generated images on simple 3D building models to visualize the benefits of generating new texture images.

VI.2 Future Research Work

Urban reconstruction is a complex problem that has been addressed from various perspectives by many researchers. Although this topic is going under a good progress, there are plenty of open questions waiting for answers. Furthermore, as some problems approach their solutions, they open up new directions for exploration. Our research work follows the same rhythm.

In our method, we assume that the façade has a regular rectilinear pattern composed of repeated elements. Thus, the first natural enhancement would be to investigate handling the irregularities within the façade. This opens a wide range of cases such as having balconies and different windows’ types as well as having a mixture of grids and elements’ organizations. Furthermore, it is interesting to consider more detailed subdivision of the façade as some users require to handle elements other than the windows.

With respect to the interaction mode, there are plenty of extensions that can be explored. We plan to add more edit operations to support more actions. We currently handle resizing the building from one side only, we plan to allow resizing from the other sides and adjust the façade structure accordingly. Another interesting resizing’ feature would be to dynamically adjust the positions of the windows as the building is being resized. Moreover, generating a meaningful mixture of multiple façades would be a good application for this work. Another direction is to consider image enhancement operations such as removing outliers as well as efficient handling of other façade elements such as roof and doors. We also plan to extend the work to manipulate street side images of city blocks such as the image shown in Figure II.9.


