

**Semi-automatic Modeling of Facades based on
image references**

by

Alexander Dolotov, B.Sc.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Interactive Entertainment Technology

University of Dublin, Trinity College

09 2014

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Alexander Dolotov

September 2, 2014

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Alexander Dolotov

September 2, 2014

Acknowledgments

I would like to thank my supervisor Dr. John Dingliana for his patient guidance in the completion of this projection and his very valuable input. I'd also like to thank him for his encouragement and advices he have provided throughout my time as his student. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly.

ALEXANDER DOLOTOV

University of Dublin, Trinity College
09 2014

Semi-automatic Modeling of Facades based on image references

Alexander Dolotov

University of Dublin, Trinity College, 2014

Supervisor: John Dingliana

3D building reconstruction is an emerging field in image processing and computer vision that aims to create 3D representation of various objects, scenes from image sets and etc. One aspect that is still lacking, however, is a way to reconstruct high quality facade elements such as windows and other domain related details. These are usually not noticed when seen from a frontal viewport or are used in a background models, where high details are not essential. But when the reconstructed model is viewed from certain angles or realistic look is the main requirement, the lack of details leap out. Therefore the aim of this dissertation is to reconstruct a 3D model with refined details based on facade image references. The application has been developed that combines an automatic detection algorithms and user interactive experience to create a 3d model based on the rectified facade images. The application pipeline consists of four stages: high-level structure extraction based on vertical and horizontal edges profiling, windows

detection, low-level window elements extraction based on Line Segment using Weighted Mean Shift and user interactive tools, and the last stage is responsible for creating a 3D model of the facade.

Contents

Acknowledgments	iv
Abstract	v
List of Tables	x
List of Figures	xi
Chapter 1 Introduction	1
1.1 Motivation	2
1.2 Objectives	3
1.3 Overview	3
Chapter 2 State of the Art	4
2.1 Determination of Facade Structure	5
2.1.1 Facade tiling	5
2.1.2 Detection of Repeated Structures	6
2.1.3 Image rectification	6
2.1.4 Profile projections	7
2.1.5 Cascade classifiers	7
2.1.6 Mutual Information	8
2.1.7 Shape Grammar	8
2.1.8 Image Segmentation	9
2.2 Window elements extraction	9
2.2.1 Edge and Line Detection	10
2.2.2 3D Element Matching	10

2.2.3	Machine Learning	11
Chapter 3	Design	12
3.1	Windows Detection	12
3.2	Window Elements Detection	13
3.3	Interactive Modeling	14
3.4	User Interface	15
Chapter 4	Implementation	17
4.1	Window Detection	17
4.1.1	Assumptions	17
4.1.2	Edges and window alignment	18
4.1.3	Facade hierarchy	23
4.2	Window elements extraction	24
4.2.1	Assumption	24
4.2.2	Line Segment detection using Wighted Mean Shift	24
4.2.3	Lines filtering	25
4.2.4	Clustering lines and geometry extraction	26
4.2.5	Regions Tree Hierarchy	28
4.3	Interactive Modeling	29
4.3.1	Windows grouping	29
4.3.2	Horizontal and Vertical line splits	30
4.3.3	Coherent regions selection	32
4.4	User Interface	32
Chapter 5	Evaluation	35
5.1	Extracting high-level facade structure	35
5.2	Detection of high-quality facade elements	35
5.3	User assisted modeling tools	36
5.4	Interactive modeling environment	37
5.5	Results	37
5.6	Limitations	39

Chapter 6	Conclusions and future works	40
6.1	Summary	40
6.2	Future works	40
6.3	Conclusion	41
	Appendices	42
	Bibliography	46

List of Tables

5.1	Data Units, Sources, and Dates	38
-----	--	----

List of Figures

3.1	The comparison of LSWMS(Left) and Hough transform(Right)[22] . . .	14
3.2	Window groups dialog(Left Image) and The right mouse click menu dialog(Right Image)	16
4.1	Original Image(Left), Gaussian filter(Middle), Bilateral filter(Right). Kernel size 7x7	18
4.2	Edge map(Top image) and the Hough transform(Bottom image)	19
4.3	Horizontal and Vertical profile projections	20
4.4	Histogram projection has been smoothed with average filter	21
4.5	Otsu thresholding(Left) Threshold Histogram projection(Right)	22
4.6	Smooth thresholding histograms(Left) and extracted approximate window locations	22
4.7	Detected connected contour components(Left) and contours approximations(Right)	23
4.8	Using edge and threshold profile projection, the approximate location of the windows is found.	24
4.9	Detected line segments using LSWMS	25
4.10	Filtering out unnecessary lines using window border	26
4.11	Extracted window geometry	27
4.12	Window geometry detection by LSWMS, followed by computing intersections points(marked in red) and regions formulation.	28
4.13	Regions Tree Hierarchy	29
4.14	Overview of window groups	30

4.15	Window H and V splits. Left image illustrates detected lines by LSWMS and the right image shows window geometry after couple of V and H line splits were added	31
4.16	Region H and V splits. Left image illustrates detected lines by LSWMS. Center image shows window geometry after window and region splits were applied. The right image represents flexibility and division depth of the region split.	31
4.17	Coherent selection. Left image shows selected regions for one window member. Right image illustrates the results after coherent selection was applied. Red regions on both images indicate initial selection.	32
4.18	The Facade Modeling Application GUI.	34
5.1	The comparison between implemented application(left) and 3Ds Max(Right)	38
1	Test 1.	42
2	Test 2.	42
3	Test 3.	43
4	Test 4.	43
5	Test 5.	44
6	The facade structure was not detected correctly, as some of the windows are overlapping due to incorrect contour detection.	44
7	Overlapping window and not all of the window were detected.	45
8	First floor windows were not detected due to additional frame gradient, causing the algorithm set seconds floor as the first one.	45

Chapter 1

Introduction

Three-dimensional reconstruction of cities or different range of architectural structures using image references is a topic of significant intellectual and commercial interest, which finds its use in many applications ranging from autonomous robotics to industrial vision and consumer digital entertainment. It is a very complex and non-trivial task that has involved intensive study since the early days of computer vision and does not have a general solution to the overall problem.

Manual modeling and control over the data is one of the common and popular approaches to represent architectural structure, but this method results in poor scalability, as the human interaction does not scale well with huge amounts of input data. In the last few years things have changed, with general purpose computers having become much more prevalent, cheaper and significantly more powerful. This advancement of new technologies has allowed researchers to investigate new modeling and reconstruction approaches. However the complexity of input data types, levels of detail, amount of automation and other classifications do not allow us to fully rely on automatic approaches due to the resulting loss of quality.

A vast majority of methods have been introduced that tend to simplify and improve urban modeling. The most popular one is reconstruction using point clouds that have been acquired from LiDAR cameras. Facade reconstruction using images is of particular importance in the domain of modeling urban areas. Various concepts for facade decomposition, detection of symmetry and repetitive elements are created with the appearance of Facade modeling. The most preferred method of creating an archi-

tectural structure today is procedural modeling, which tends to generate architectures of building or cities according to specific rules. Even though each of the listed methods has been quite deeply investigated, there is still space to research new methods that allow more accurate and as a result better quality of the details.

1.1 Motivation

Automatic urban reconstruction has been actively researched over the past ten years and good contributions have been made within the computer vision community. One of the still challenging tasks is the recognition and reconstruction of facade details like windows and balconies. These are considered as key elements of realistic representations of building facades. The appearance of reconstructed facade geometry gains a lot of realism from the displacement of windows, doors and other domain related detail. Not considering those, would result in a flat facade, which looks strange and unrealistic from certain angles.

Because of this a variety of approaches to urban reconstruction and facade elements recognition have been developed. The raw building geometry is constructed either using a point cloud, which in turn was built by terrestrial laser range finders or reconstructed from satellite, airborne or ground based images. Facade details like windows are mostly extracted from images and then superimposed on the coarse building models. However, fully automatic methods are still unable to reliably address this problem fully. The rich geometrical details of architectural buildings makes it extremely difficult to develop a method that works for every urban environment without requiring a fair amount of user input.

Nowadays projects that have a tight relation with geolocation, historical data or a movie production, involve using the approaches of sufficient accuracy and quality. Majority of methods that are available today are not providing a complete solution pack that can address general problems. One of the challenges is to achieve an automated reconstruction as much as possible, but the downside of which is the quality and in addition there is no unique solution to the fundamental problems. On the other side, manual modeling addresses issues related to automatic modeling, however poor scalability and too much time consumption make this approach less and less preferred as the technology moves forward. Some of the recent ones employ compromised solutions

that can cast the problem in such a way that both the user and the machine can focus on tasks which are easy to solve for each of them. However complex user interaction and time consuming methods cannot deliver satisfying results.

1.2 Objectives

The main goal of this project is to implement a flexible solution that provides a combination of manual and automatic set of tools to create a facade model out of 2d image in relatively small amount of time. To achieve this task, a number of objectives must be met:

- Extract high-level facade structure
- Provide a method to correctly determine high-quality architectural elements.
- Offer a set of tools to assist the user during the modeling process.
- Implement an interactive modeling environment.

The dissertation also involves the study of current facade related challenges and unsolved problems. The main focus of this thesis is to study a facade extraction key elements and creating a high quality level of architectural details.

1.3 Overview

Chapter 2 contains a review of the current state of the art in facade including an introduction to the available methods that are used to simplify and automate different stages of reconstruction. Chapter 3 covers the major design decisions made during this project such as facade structure extraction and window details detection. In addition this chapter discusses all necessary design features related to interactive modeling pipeline. Chapter 4 discusses and outlines chosen algorithms and techniques to achieve established objectives. Chapter 5 evaluates the modeling process and final results. Moreover an informal comparison between industrial modeling package 3Ds Max and the current project tool was made. Chapter 6 concludes the results and gives the possible future works.

Chapter 2

State of the Art

The problem of facade image analysis for reconstruction purposes has been tackled by many researchers in the last ten years. Many different approaches for extraction of facade structure, facade elements and facade geometry have been proposed. This section structures the literature of related work in order to give a good overview of all the different techniques that have been applied or created to solve problems in the class of recovering facade details, window detection and structure elements extraction.

A comprehensive literatures of urban reconstruction has been reviewed in [1], outlines sophisticated methods and techniques used nowadays in the modeling area. The research works reviewed in this paper enclose four research communities such as: computer graphics, computer vision, photogrammetry and remote sensing. Reconstructing 3d space out of 2D is an exciting area of research that benefits the applications in the following industrial sectors: entertainment, mapping for mobile devices, Urban planning, training and simulations. However most of the developed city reconstruction application are showing a greater demand for automation to minimize work amount while processing big chunks of data. The following discussion of the state of the art attempts to judge the degree of automation of the proposed solutions, which is not always easy because many authors do not provide exhaustive details regarding this matter.

In the next few sections, the problem solution strategies are categorized by their general approach and briefly explained. Most of the available methods, approach the facade reconstruction as an image segmentation problem, others define it as a feature

detection challenge. Then again, some of the proposed solutions use both approaches in different steps of their processing pipelines.

2.1 Determination of Facade Structure

An important role is dedicated to facade decomposition, which is a major part in the reconstruction process. There are many methods and algorithms that can extract architectural structure, geometry and other important building elements. Facade reconstruction can be interpreted as a segmentation or feature detection[1] problem, but each of these involves applying a set of methods and their combinations, either it is a classical image processing problem such as corner, edge and feature detection[2] or using a global methods like shape grammars in order to integrate a top-down model of the facade[8]. There are many ways to describe facade structure.

2.1.1 Facade tiling

One of the facade reconstruction methods is based on the Monte Carlo sampling approach[6]. The idea behind this technique is to partition facade image into tiles, where each tile contains separated window. The method handles only certain types of facades, where horizontal and vertical repetitions of similar pattern occur. Most of the facade structures exploit the nature of these repetitions, making this method very robust with such input data. The algorithm begins searching for a dominant repetitive patterns that appear in the facade. This operation is quite expensive, therefore image pyramid is used, where facade image is subsequently scaled using a factor of 0.5.

Down-sampling operation creates levels of the same facade image with different resolutions. The next stage involves swiping a window with a size 15x15 pixels across all pyramid levels and compare how similar pattern is. However this method is quite expensive as it compares every pair of different locations. Therefore a common approach was introduced to deal with high-dimensional search spaces. Rather than checking every possible location, Monte Carlo algorithm takes a statistical probe at a number of random positions. The proposed technique can handle facades with partial occlusion. The method showed its reliability and efficiency.

2.1.2 Detection of Repeated Structures

Facade textures are often obscured by foreground objects (vegetation, advertising signs, wires, cars, people and other buildings). Occluded region can be segmented and reconstructed by comparing the repetitive parts of the facade[11] [12]. Textures derived from aerial photography, often have a low resolution and poor quality of the bottom floors. If the facade information can be partitioned into floors, then it's possible transfer the texture information from the upper floors to lower ones[8].

Discovered facade elements such as windows or balconies, can be replaced by a more detailed three-dimensional models to enhance the overall quality. Knowledge of the repetitive parts of the texture and the corresponding elements of the geometry can be used for file compression of three-dimensional models and textures, which is important for Internet applications and geolocation services.

The pattern repeatability and regularity are the best tips in the interpretation of facades. The Classical solution to extract the regularity of the image of a flat object without perspective distortion is the analysis of the peaks of the image autocorrelation. This is only suitable for the case, when there is a strong regularity, which occupies a large part of the image without significant occlusions.

2.1.3 Image rectification

Ground-based prospective view of the images is the most suitable for reconstructing facades, due to the high quality demand of the facade details[1]. However most of the facade textures are distorted. For best results, the facade image needs to be in the ortho-perspective view that can be achieved with 2D Homography. A homography is an invertible transformation from one projective plane to another which is characterized by mapping straight lines to straight lines[2]. It can be done either by manually setting four points defining a rectangular in a three-dimensional space, or using automated algorithms that find lines in the image and group them in correspondence with the points of similarity. There are also various public tools for rectification for cases where this is still needed.

2.1.4 Profile projections

The profile projection method facilitates the extraction of rectangles and alignment of windows using facade regularity. This approach utilizes the same concept as in [6], where a repeating patterns of the window along horizontal and vertical directions are considered. However implementation principles significantly differ. This semantic information is used to extract an approximate height of each floor and width of each window tile[4].

Once profiling is done, it is easy to formulate a grid, which separates windows from each other. The idea behind this technique is to project horizontal and vertical histograms using the edge map. Assuming that windows are vertically and horizontally aligned, the window edges at each row and column will be accumulated, forming a histogram bins. Each histogram contains peaks that can be smoothed out to correctly extract necessary information[3]. As the noise level of the image is usually high, the values are approximated and not used directly to build a grid. The technique facilitates facade grid extraction and gives the approximate position of windows. However strong gradient around window frames dramatically reduce accuracy of this method.

A better algorithm was developed based on the same projection principles [5], but provides a more precise approach, which is capable to deal with complex facades that have unusual ornaments and gradients. The projection methods for simple facades have strongest gradients located at the edges of the windows. However this is not true for complex facades, where strong responses can be at the arches, rims, signs and other places.

The idea behind this method is to partition the facade into blocks, where each block is labeled as "facade" or "window". To correctly label every block, the color histogram is used as an addition source of information. The size of the block, color and the gradient gives a sufficient information about making a labeling decision for each block. The big difference from the classical profile projection[4] is that this technique can handle facades under severe perspective distortion.

2.1.5 Cascade classifiers

High-level structure can also be extracted using machine learning as it was proposed in [7]. This method starts with an early image processing stage, which builds image

tiles based on the Haar wavelet representation. This information is further used for window detection by the cascade classifier. The classifier is trained with Adaboost decision tree of image tiles which is then fed into a cascaded classifier for the task of window detection. Once the classifier is ready to detect windows it is directly applied to the preprocessed image data. The output is a list of coordinates that form a bounding box of hypothesized window related subimages. The experimental results demonstrate that system is capable to detect at least a few windows per building to enable window classification for building recognition.

2.1.6 Mutual Information

Mutual Information(MI) is another technique that has proven its reliability. MI of two random variables describing the amount of information contained in one random variable with respect to another. Mutual information is defined through the entropy and conditional entropy of two random variables. It is used to find a similarity measure on image intensities. It has been successfully implemented using a concept of irreducible facade, where a Mutual information is used to measure similarity between image regions. In addition it takes into account intensity values of the corresponding position pairs. In the second stage of their detection pipeline MI is applied to locate similarity between floors and different tiles. The vertical direction is used to exploit symmetry of floors and horizontal to extract a sequence of similar tiles. MI has been slightly extended in [10], which incorporates user interaction to target all repetitive elements found on the image. From the other side if approach is applied on less repetitive architectural facades, it loses the structural support and runs into the classic difficulties of edge detection i.e. The operations of finding symmetry will be less stable, which makes this technique only applicable for urban facades.

2.1.7 Shape Grammar

A more complex approach involves to use of a shape grammar rules. This consists of modeling architectural styles using a set of basic shapes and a set of parametric rules, that correspond to increasing levels of detail. This approach is able to model elaborate and varying architectural styles, using a tree representation of various depth and complexity. The choice of the rules for grammar (vertical or horizontal divisions)

and their parameters (coordinate divisions) can be formulated using random Markov chains, where mutual information or edges within and between fragments of the candidate will be used as a metric[8]. Metric information is used to reduce the parameter space. The resulting framework can produce precise 3D models from single view. It also can deal with lack of texture fragments, presence of occlusions and specular reflections, while maintaining the ability to cope with very complex architectural styles. Promising results demonstrate good potential for further development[14]. However specifications of the rules, may be limited in the realism of resulting models and their variations. Furthermore, it is very difficult to define the appropriate rules to generate existing buildings exactly. Moreover rule-based approach creates additional complexity if the user interaction needs to be involved, which makes it hard to apply this method in real-time[8].

2.1.8 Image Segmentation

The Image segmentation approach is used to split overall image view into different groups. Early attempts were using filtering, splitting heuristics and others[15], but it did not provide sufficient reliability, especially for detecting complex facades. A better techniques were proposed in[16] that are used to improve processing speed and visual recognition. A more comprehensive approach uses a combination of shape grammars, supervised classification and random walks[19]. They have proposed a novel modular approach that segments the facade image using procedural grammar. The shape grammar is constrained towards fixed tree topology, that gives an opportunity to recognize different architectural topologies. The machine learning techniques establish the connection between grammar semantics and the image itself. The proposed method demonstrates very high results and is able to deal with complex facade structure with partial occlusions.

2.2 Window elements extraction

Apart from general structure extraction, it's also important to identify window panes and other window relevant elements. As the noise level is high around window area, applying a classic technique usually does not provide a good results and therefore

some non standard methods are required.

2.2.1 Edge and Line Detection

When facade structure is discovered, it is necessary to move towards the modeling step, which involves finding key points to create the final model. These are found using a combination of Canny edge detection[Canny edge detection, 1986] and Hough transform [Duda and Hart, 1972] to recover the lines. To achieve better results and improve the robustness. The direction of the Hough transform is constrained to only horizontal and vertical direction, which happens in most architectural facades. The detected lines will form a grid that contains many non overlapping short line segments by taking intersections of Hough lines as endpoints. These line segments are now the hypothesis to partition the facade. The Hough transformation is good for structure discovery since it can extract the hidden global information from the facade and align line segments to the facade structure.

2.2.2 3D Element Matching

As was proposed in [8] a subdivision of each tile is applied first in order to localize window. The algorithm tends to find a local split and subdivide the tile as much as possible. A subdivision of facade tiles leads to a set of rectangular regions clustered into groups of similar regions. Next step is to identify acquired information by comparing it with a database of architectural objects to achieve high quality geometrical information. Although the proposed approach shows good results, it still tends to fail in complex building types that do not comply with established rule set. In addition if database lacks of particular pattern the actual window tile can be identified incorrectly. An interactive method was proposed in [10] to extract window elements using horizontal and vertical splits. Given a shape, a manual split is generated by the mouse click at the appropriate position. This method is under control of the user and if an error occurs, it can be easily corrected. This technique ensures that there will be overlapping elements. Its intuitive use and easy to maintain tree data structure allows the user with little interactive effort extract very detailed facade elements. However there is too much clicking involves when manual split is used. Automatic split is also available, but it is suitable for breaking high-level repetitive elements. The accuracy of the automatic

split is affected by the image quality, hence manual split will be used more frequently. In addition this technique involves the user to spend a lot of time on thinking how facade should be subdivided using manual split.

2.2.3 Machine Learning

Some approaches combine template matching with machine learning[7]. However this method needs to train its classifier by sorting and feeding thousands of templates in order to achieve high targeting rate. The method shows quite a high detection rate in some cases and there is a high probability that all elements will be detected if image is rectified. As the image gets more distorted, detection level decreases. Another approach, which is discussed in [18], assumes that all windows are rectangles and identifies strictly based on axis-aligned rectangular pixel configurations in a Markov Random Field. The problem is defined in terms of pixel labeling. A simple image model is considered to easily learn from exemplar images. A Probability distribution of the RGB pixel color is learned for each label separately. In this case the results are much better, but heavy image noise or small irregular elements within window area or some of the unusual forms of window panes can cause this method to fail in some cases.

Chapter 3

Design

This chapter outlines the major design decisions made during this project. The Solution proposed here consists of a three staged pipeline. Facade related issues usually start with windows detection and grid formulation from a single image. The grid of horizontal and vertical lines allows us to separate each window within each tile for further elements analysis. Lastly user interactivity will help to achieve higher quality results and correct existing work.

Three-dimensional model of urban buildings can be divided into two parts: the facade and roof. Roofs can be modeled with aerial images or aerial Lidar data using hand tools or by automatically fitting the parametric models of roofs into available data. Facade modeling is a more difficult task, but buildings generally have a structure and regularity. These semantic knowledge can help in solving many problems in modeling, processing and interpretation of the facade image.

3.1 Windows Detection

Window detection is a complex problem that usually depends on the architectural structure, and whether the architecture is symmetrical or asymmetrical[1]. Some facades include various types of windows, which adds additional complexity to the detection algorithm. Reliability and precision of the methods should have a good targeting rate as the pipeline stages that follows after, depend on the detected results like it was mentioned in [8] and [10].

Ideally window detection functionality should be completely automated and not require any user input, however in practice this is not possible, because the input data vary in types and it is hard to predict the output. Therefore there should be a manual approach to fill the gap in the detection procedure to avoid unnecessary results. The main emphasis will be on investigating a histogram profile projection as this method has a less implementation complexity in comparison to Mutual Information algorithm and is not performance and time consuming as the machine learning algorithms. The output is expected to be an approximated solution, due to the fact that most of the images contain a lot of noise.

The profile projection is used in combination with the Hough transform, which is a widely used method for extracting line segments. The main idea is to consider straight line characteristics not as image points $(x1, y1)$, $(x2, y2)$, but in terms of the straight line formula $y = mx+b$, where m is a slope parameter and b is intercept parameter[3]. Using window positions the proper grid can be constructed, which will separate each window and hence breaking the whole facade into tiles, which are processed further.

3.2 Window Elements Detection

Unfortunately facade reconstruction approaches do not provide many solutions to extract window details, such as pane, frame and other relevant elements, due to high noise level in the localized window area. The most common one is to use a database matching, which was described in [8], although such method outputs high quality geometry it is hard to build up a complete database that will contain all window types.

Machine learning consumes too much time to train and detect architecture elements[18]. In addition failure can be dramatically higher, because of the training data. Good results were achieved by plugging in interactive methods proposed in [10] using simple vertical and horizontal line splits in relevant places where it is necessary to form geometry. Too much clicking adds slight disadvantage to this approach as it was mentioned before that user interaction does not scale well, when huge amounts of data is processed. Therefore this method is going to be slightly improved with automated support using Line Segment detection using Weighted Mean-Shift(LSWMS). As the name suggests it is used to detect lines within each window. The big advantage of it over the Hough transform is that it does not need tuning parameters, and will still provide comparable

results for most images[20].

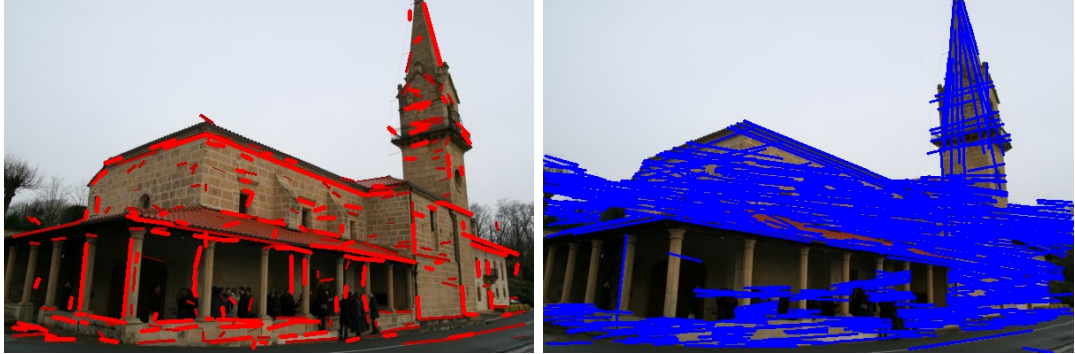


Figure 3.1: The comparison of LSWMS(Left) and Hough transform(Right)[22]

Figure 3.1 illustrates both methods in action and it is obvious that LSWMS provide much cleaner results in comparison to the Hough transform. However in some cases LSWMS does not detect necessary lines and in this case Hough transform needs to be applied. The whole point of LWSMS is to detect line splits in appropriate locations and then break windows into regions that can be edited and grouped together. Actual lines that were found by the algorithm represent the border between regions and will be limited by the window height and width.

3.3 Interactive Modeling

Relying on full automation will not give accurate results, as the modern algorithms and techniques do not provide a general solution to the wide range of different input data types, where manual modeling addresses above problem, but consumes too much time. Finding the balance between Automatic and manual modeling will result in a better quality model. In terms of user interaction, the most necessary tools/options should be provided to achieve better accuracy. The following interactive functionality is going to be added:

- Add and adjust existing geometry
- Group Windows and window regions
- Synchronize editing

- Allow results to be rebuilt
- Add horizontal and vertical splits within window

The above functionality needs to be properly displayed on the screen and the main focus to provide intuitive usage. Facade grid and all lines including window segments can be adjusted with a simple mouse movement. Interactive features that allow users to align windows along x and y axis and correct window height and width to match neighbor windows. In addition windows that are grouped according to their size, will have an option to be synchronized and edited in real time by propagating equal changes. Each stage of the pipeline whether it is window detection or window elements extraction has an option to be regenerated based on the previous stage information.

As LSWMS does not always detect line segments, there is a high rate that some detail will be missed. In this case, there is an option that can fulfill window with simple horizontal or vertical line splits. All regions that were established up to this moment based on the line segments are going to be regenerated as soon as the new line segment is added. There are windows that contain quite complex structures and therefore cannot be represented by lines that spread from the top of the window to its bottom. Therefore additional splits were designed, which are affecting only selected regions. It provides a much more flexibility to create a more complex window elements.

3.4 User Interface

The scope of the project is to use a combination of the automatic and manual techniques, the user interface should provide necessary GUI elements to address this matter. As it is hard to combine geometry creation on top of 2D image and rendering a 3D model at the same time, the decision was made to separate windows into an editor and renderer, where each will be responsible for certain functionality. Rather than displaying both windows at the same time, a tab control can be introduced to switch between dialog windows[10]. The important part of this application is the Canny filter. However it requires tweaking input parameters, which usually involves recompiling the project. Therefore the edge map needs be displayed to correctly in real-time. This allows user to adjust input values appropriately to acquire necessary Hough lines as well as the connected contour components for detecting window location.

The editor context provides a full mouse interaction(e.g. zoom, rotation, transformation, dragging and etc..) to control scene workflow without any difficulties. The facade modeling process is represented as a pipeline stages, where each button is responsible for each stage. Each stage has to be applied in sequence and does not require additional skills or knowledge about 3D modeling.

Most of the scene workflow involves using the mouse only, but the grouping operation requires to use a keyboard combination. CTRL + MOUSE CLICK is used to select the whole window. The right mouse click outlines a menu, as it is illustrated in Figure 3.2 where the user can select necessary options. Windows that has been selected can be grouped to speed up a modeling process and use "Sync Group". Synchronized editing allows freshly made changes to be propagated to all objects that are grouped together. ALT + MOUSE CLICK allows to select regions and group them later using similar manner like with windows.

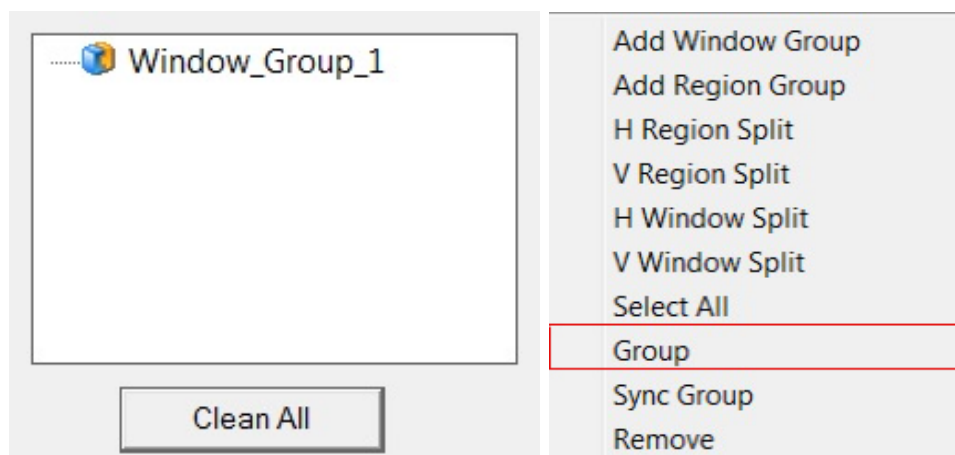


Figure 3.2: Window groups dialog(Left Image) and The right mouse click menu dialog(Right Image)

Chapter 4

Implementation

This chapter describes the implementation details of the facade modeling application based upon the design stages laid out in the previous chapter. All the implementation steps will follow the same order as it was specified in the production pipeline.

4.1 Window Detection

In most cases facade scene appears to be distorted, which complicates further analysis and architecture elements detection. As there are many facade textures available that are ortho-rectified, we can assume that our input data will be aligned accordingly. Orthogonal view allows us to exploit alignment properties of the windows. Assuming that most of the windows are aligned vertically and horizontally, we can exploit a concept of dividing an image onto window and non window regions. This approach is based on a horizontal and vertical histogram profiling of the Hough lines. Using this method allows us to classify and combine regions, that will form windows.

4.1.1 Assumptions

To clarify the situation in a more depth, we assume that windows have orthogonal sides and are aligned equally. In particular window rows and columns that share the same height and width. Although most of the facade images exploit a concept of symmetry it does not mean that all windows will have the same positions and alignment, but that does not violate our assumption or window detection rate.

4.1.2 Edges and window alignment

The Canny Edge Detection algorithm is the most popular of its kind, but like any edge detection method it is susceptible to noise in the image. To clear some noise in the image we are applying a Bilateral filter with 3x3 kernel size. The main reason why we chose Bilateral filter is because our main goal during this step is to extract edges and Bilateral filter preserve edges while dissolving the noise. It considers the difference in intensity between neighboring pixels, unlike Gaussian filter, which smooth away all the edges. The following figure shows the difference between Gaussian and Bilateral filters.



Figure 4.1: Original Image(Left), Gaussian filter(Middle), Bilateral filter(Right). Kernel size 7x7

Next step is to apply the Canny filter with a Sobel 3x3 kernel, however extracting edge map does not give a sufficient information about facade structure, therefore a Hough transform is applied to detect all possible lines. Feeding in the facade image will give us all possible lines that are surrounding window frame. Using an edge map, groups of horizontal and vertical Hough lines are extracted. Figure 4.2 illustrates described stages.

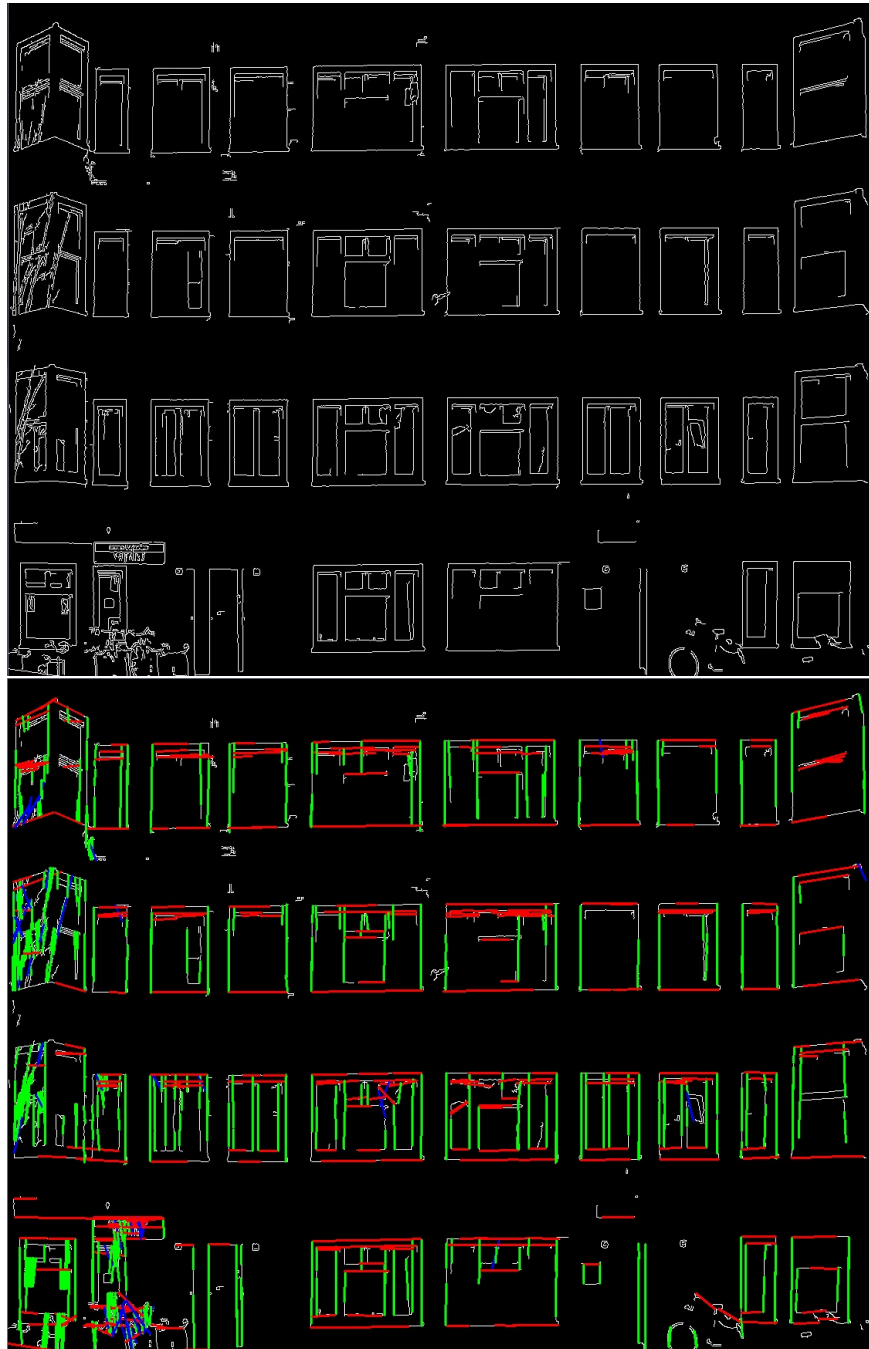


Figure 4.2: Edge map(Top image) and the Hough transform(Bottom image)

From the above figure, horizontal lines are marked in red and verticals are green. This has been achieved by controlling allowed angle. All horizontal lines have an angle

(θ), where $[\theta \leq 15$ and $\theta \geq -15]$ degrees. The vertical group has a (θ), where $[\theta \leq 105$ and $\theta \geq 75]$ degrees. The fact that some facade images cannot be 100% correctly ortho-rectified, there still will be details that are distorted or skewed. Therefore angle threshold has been applied to target all possible lines in horizontal and vertical directions.

Using the strategy described above, it is easier to get histogram projections for each group, because lines at each row and column can be accumulated to appropriate bins. Figure 4.3 illustrates Horizontal and Vertical histogram projections.

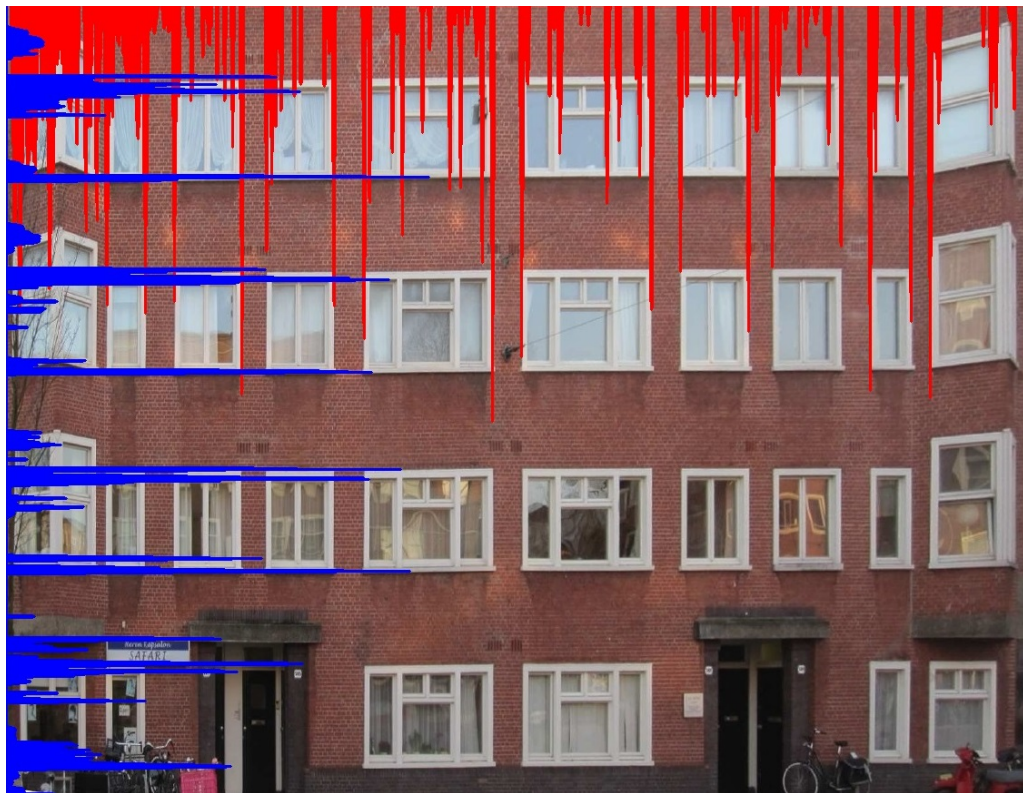


Figure 4.3: Horizontal and Vertical profile projections

The peaks are located at the positions where Hough lines have a clean start and end(.i.e where peaks start to grow and returns to the same place). This gives us a useful information, as the peaks are highly correlated with a border of the windows. However the results are quite messy, due to the fact that peaks represent accumulated edge at each row and column. Therefore a smoothing average filter is applied to get

the right number of peaks.



Figure 4.4: Histogram projection has been smoothed with average filter

Figure 4.4 demonstrates smooth projection profile with peaks that are located at the average positions of the window edges. Even that does not tell us where the actual windows are. Therefore additional image information is extracted. Using the same profile principle, we are projecting histograms using Otsu thresholding operation. Opening and Closing Morphological transformations have been applied to clean binary image from noise and to fill small holes. Figure 4.5 demonstrates thresholding map after Opening and Closing methods have been applied.

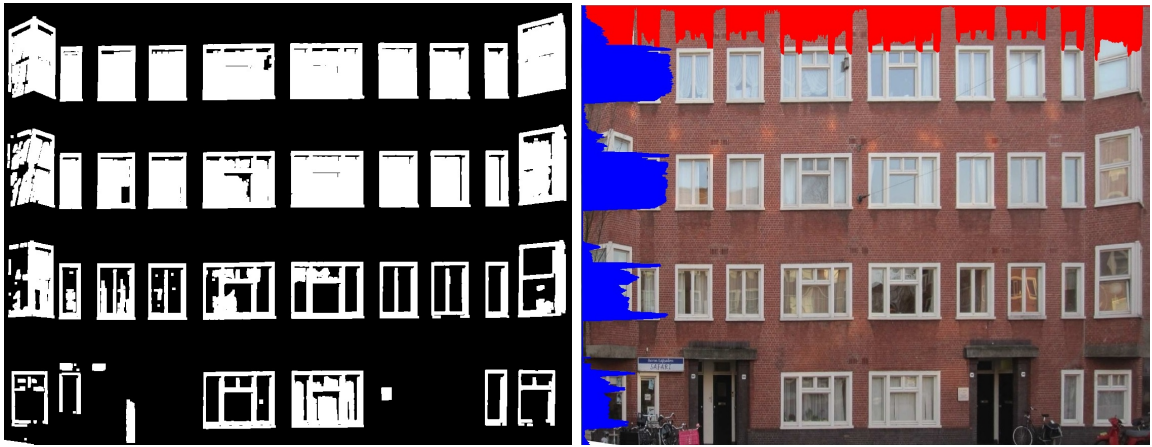


Figure 4.5: Otsu thresholding(Left) Threshold Histogram projection(Right)

This time, the peaks of the profile projection are located at places where windows occur most likely(Figure 4.5). As with Hough line profile projection, this also needs to be smoothed(Figure 4.6).



Figure 4.6: Smooth thresholding histograms(Left) and extracted approximate window locations

The combination of the Edge and Threshold profile projections, gives an approximate location of the windows(Figure 4.7). The extracted information can be used to create a high-level facade structure, that split all windows into separate tiles.

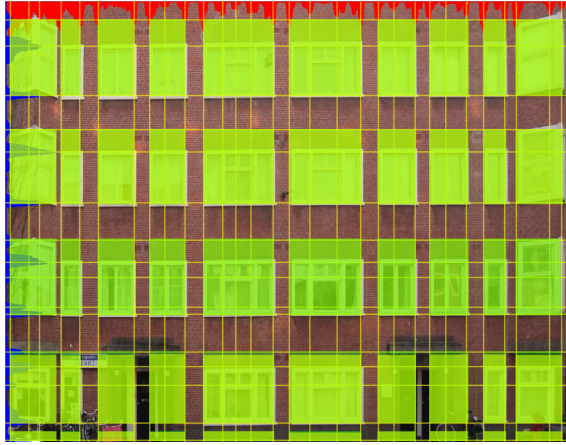


Figure 4.7: Detected connected contour components(Left) and contours approximations(Right)

4.1.3 Facade hierarchy

The grid is created based on the extracted information from profile projections. To create a grid and break windows into tiles, the hierarchy is created first, where the main idea is to keep each floor separate from others. The algorithm starts with top left location. Assuming that all found locations are equally aligned, it takes min(can also be max) position and looks for the closest min along x. Once it has been found, it stores current window into array and continues to search the next closest location, until nothing can be found. When it reaches the end, we assume that there are no areas left, hence this floor is complete and the new floor is added to the hierarchy and the process is repeated over until nothing can be located. Using window hierarchy, we first break facade structure into floors, where an average line between floors is calculated. The same principle applies to average line between windows for each level. At this stage facade structure has been extracted and tile grid is create, which separates approximate window locations. The next step involves using closed contour detector to locate window frame, as we cannot rely on approximate area, which was acquired using image histogram profiling. The left image of the Figure below illustrates extracted window contours and the right image is the approximation of the contour area.

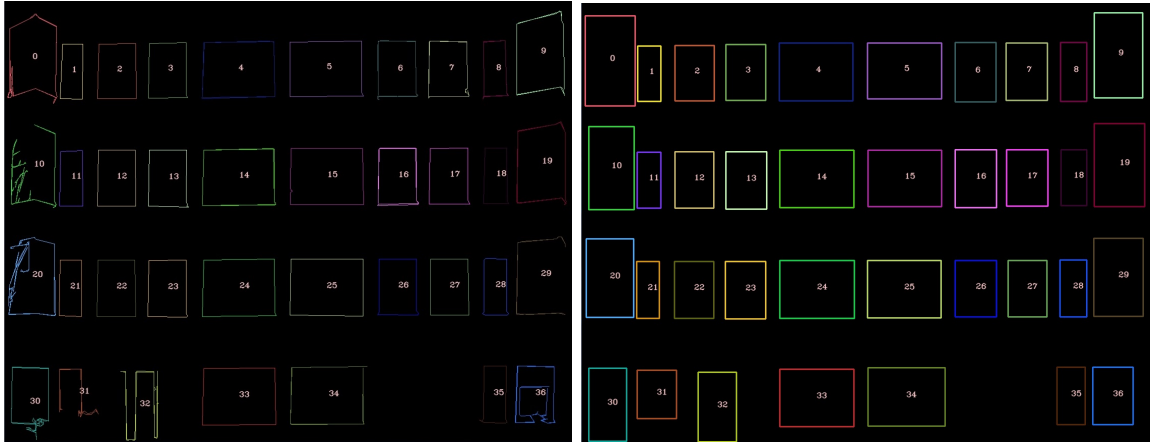


Figure 4.8: Using edge and threshold profile projection, the approximate location of the windows is found.

4.2 Window elements extraction

4.2.1 Assumption

At this stage we assume that all windows were detected correctly and grid was proportionally displayed between non window regions, avoiding overlap of windows with each other.

4.2.2 Line Segment detection using Wighted Mean Shift

Initial strategy was to use a Hough transform, that is available in OpenCV as HoughLinesP function, however it is highly susceptible to noise and involves to tune input parameters, which makes this technique quite unreliable and difficult to tune with different type of windows. Therefore a Line Segment detection using Weighted Mean Shift(LSWMS) solution was adopted, which tends to provide more persistent results in comparison to the Hough transform. It is based on a random sampling strategy combined with growing procedures based on the approximation to a straight line between two given points using the Bresenham algorithm and refinement using Mean-Shift. The following figure shows LSWMS in action, keeping only good line segments. Unlike Hough lines the LSWMS implementation[22] does not require to

adjust the parameters for each specific window case and it works the same for any kind of images. The comparison example of both methods can be seen in Figure 3.1.

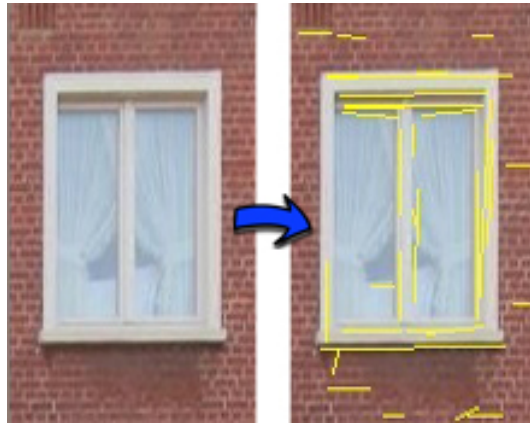


Figure 4.9: Detected line segments using LSWMS

4.2.3 Lines filtering

Although LSWMS gives a cleaner line representation, but there are still a lot of false positives and majority of lines that are not going to be used in the formation of windows geometry. For this reason a filtering operation needs to be applied to remove unnecessary lines. Using frame around each window that was detected in the previous stage, we can omit outside lines and keep only the lines that are located within the window frame. The figure below illustrates border around the window, which serves as a filtering mechanism.

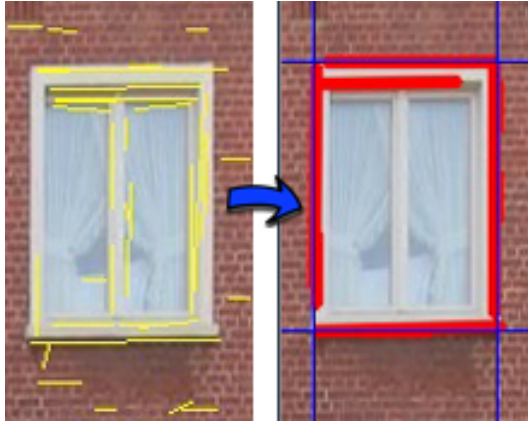


Figure 4.10: Filtering out unnecessary lines using window border

This allows us to deal only with the lines that represent window geometry. Same approach like with Hough lines is used to split lines into vertical and horizontal groups by controlling the allowed angles.

4.2.4 Clustering lines and geometry extraction

Even though some of the lines were correctly classified, it is still not enough to accurately extract window elements. Therefore a decision was made to create a clusters of average lines that will represent window pane. A trivial algorithm was developed to store lines that are close to each other in a separate cluster. Each cluster maintains an average position for all aggregated lines. The algorithm works as follows:

Data: Vertical and Horizontal lines

Result: Clustered Vertical and Horizontal lines

create vertical cluster using first line;

flag = false;

while *not end of the lines* **do**

while *not end of clusters* **do**

if $| \text{cluster avrg pos} - \text{line pos} | < \text{threshold}$ **then**

 push line into cluster;

 update average;

 flag = true;

end

end

if *!flag* **then**

 add new cluster;

 update cluster average;

end

end

The algorithm aggregates lines separately for vertical and horizontal directions, where x position is evaluated for verticals and y is for horizontals. Threshold value in pixels specifies the distance between the current line and closest neighbor. Each time, when a new line is added to the cluster the average position is updated to maintain approximate mean value for all stored lines in the cluster. The results of this stage can be observed in the figure below.

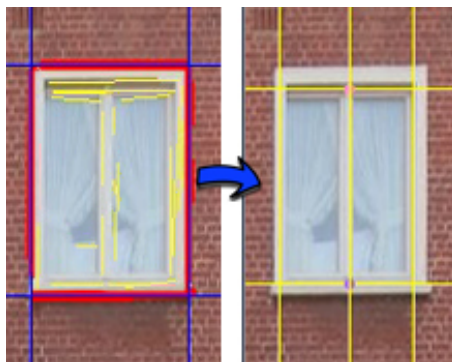


Figure 4.11: Extracted window geometry

4.2.5 Regions Tree Hierarchy

Using extracted lines we can combine these with window frame lines to form regions. A tree structure was chosen to provide a quick window access and easier region management. To form the region tree, all possible intersections between lines are calculated using determinants[23].

$$(P_x, P_y) = \left(\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \right. \quad (4.1)$$

$$\left. \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right) \quad (4.2)$$

where P represents an intersection point. The First line is defined by two points (x_1, y_1) and (x_2, y_2) , and the second line being defined by two other points (x_3, y_3) and (x_4, y_4) accordingly. The next stage involves sorting all intersections, so that each column has an equal number of rows. As the number of rows in each column will be of equal size, we can represent (i, j) as min and $(j + 1, i + 1)$ as max to form a rectangle. Iterating through all of the points, thus forming necessary regions that build up a window.

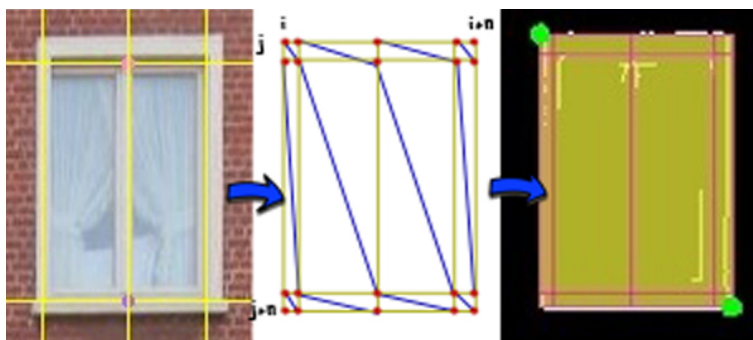


Figure 4.12: Window geometry detection by LSWMS, followed by computing intersections points (marked in red) and regions formulation.

The above figure illustrates the process of regions formulation using intersection points between detected lines.

Each region is stored as a tree node, thus forming different levels of hierarchy. The first level always represents the regions that have been formed by LSWMS. If window

split is added, it causes a tree hierarchy to be rebuild in order to consider new regions. However when the region is divided using a region split, a new level is formed to represent subregions of the current region. The region split always splits the region into two subparts. The figure below shows difference between window split and region split.

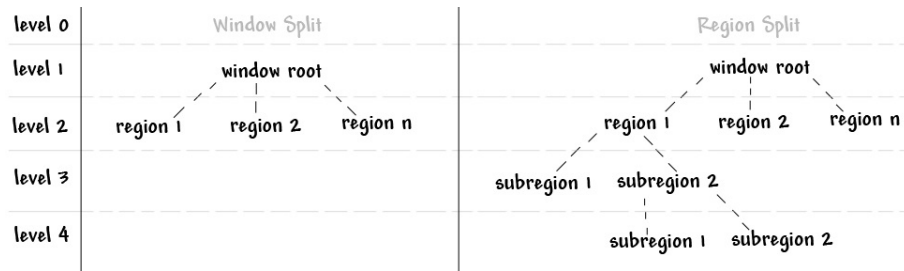


Figure 4.13: Regions Tree Hierarchy

4.3 Interactive Modeling

As it was discussed before, it is not a good idea to rely completely on fully automatic methods, as the quality of these may fail dramatically due to the occlusion or distortion in the input data. Therefore to compensate this issue an interactive pipeline has been introduced to make changes on the fly to achieve a more desirable results. Mainly it involves the user to correct facade geometry that does not correspond with the actual image, using intuitive editing mechanism.

4.3.1 Windows grouping

To avoid multiple editing of the repeated elements, a grouping option is introduced. The main goal is to group similar windows into one set, allowing freshly made changes to one of the members to be equally propagated to all objects in the group.



Figure 4.14: Overview of window groups

The above figure demonstrates how windows are manually grouped according to their architectural structure. As the windows were detected and sorted previously, we can easily group windows according to their indexes. To implement such strategy, an array of groups is created. A separate group is a temporary memory space that holds selected windows. Basically whenever the user selects/deselects particular window, it is instantly add to or removes an index from temporary storage. When all necessary window members are selected, the results are stored as a separate group of window indexes into an array of groups, thus maintaining an easy to access window structure.

4.3.2 Horizontal and Vertical line splits

There are two types of splits that extend tree hierarchy. First one is the window split, where a horizontal or vertical lines are added to the window geometry across its width or height. The figure below illustrates window split.

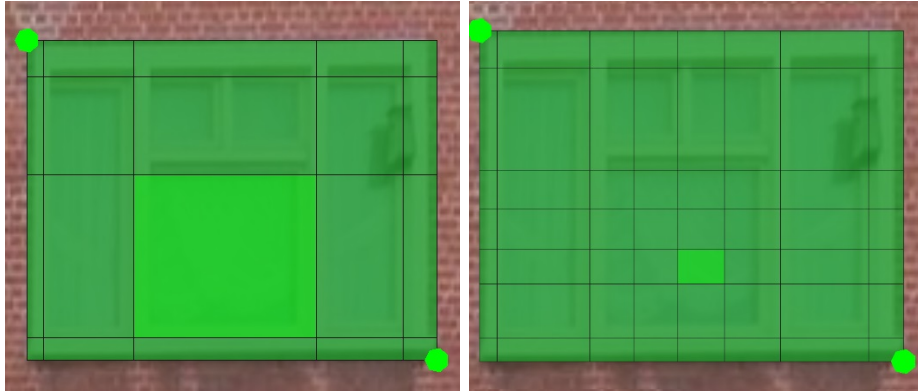


Figure 4.15: Window H and V splits. Left image illustrates detected lines by LSWMS and the right image shows window geometry after couple of V and H line splits were added

Window split is usually used when general details need to be fulfilled or lines that were missed by LSWMS. The second type is the region split, which is applied only to internal regions and do not divide the whole window geometry like window split does. This type of split intends to provide a more detailed look, where window split should not be applied.

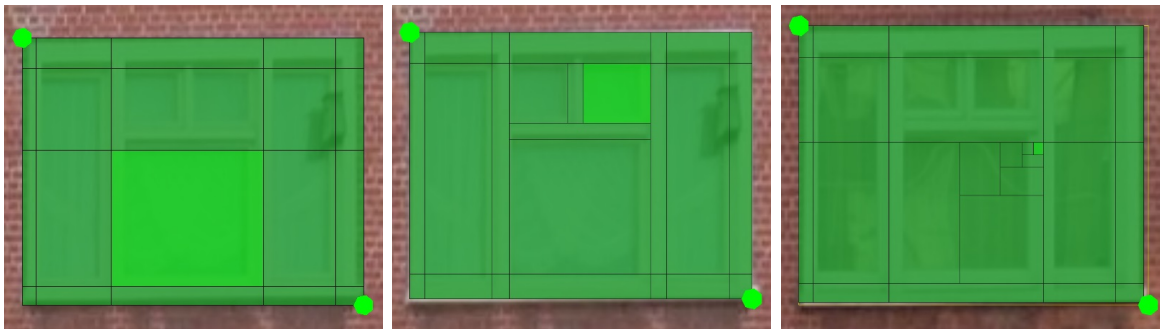


Figure 4.16: Region H and V splits. Left image illustrates detected lines by LSWMS. Center image shows window geometry after window and region splits were applied. The right image represents flexibility and division depth of the region split.

4.3.3 Coherent regions selection

A coherent selection was introduced to avoid repetitive selection of regions that are grouped. The main idea behind this feature is to allow the user to select only necessary regions from each window group and allow the selection to be propagated to the rest of the group members. Due to the fact that every window member in each group will have an equal geometry representation, it is possible to avail of this information to provide a coherent selection. Each window region has a unique index. Every window member in the group share the same index for each region. When the coherent selection is applied, it takes selected region index and iterates over the other member regions by selecting required indexes. The figure below shows a coherent selection in action.

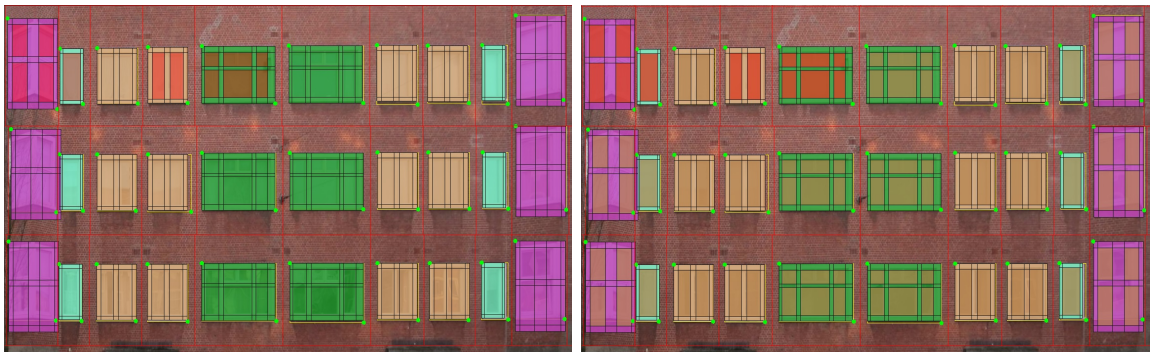


Figure 4.17: Coherent selection. Left image shows selected regions for one window member. Right image illustrates the results after coherent selection was applied. Red regions on both images indicate initial selection.

After a coherent selection has been applied, regions can be grouped together. All grouped regions will be assigned with a depth value in the final rendering.

4.4 User Interface

The Graphical user interface has been implemented in a very easy to use manner. It provides a series of menus and parameters that can be altered to view the result in real-time. WinApi was a primary choice to represent graphical components on the Windows OS, as it offers the reliable framework, which is compatible with OpenGL 3.3+. The usage of external GUI library gives a flexibility for the user, as it does

not require project recompilation when some parameters are changed. To represent an interactive modeling environment a simple prototype was implemented that has an integrated editor and renderer context in one window application. Most of the interactive controls, such as buttons, checkboxes, track bars and etc, were created to support editor workflow.

All the primitives such as lines, rectangles and dots with their hovering and dragging features are rendered using OpenGL on top of the loaded image. Threshold, edge maps and other computer vision related aspects are calculated using OpenCV library, but the results are passed to the OpenGL pipeline for rendering purposes. The image is loaded into a buffer using OpenCV, which is overtaken by OpenGL to render it as a texture on top of the plane. The image plane is automatically resized to correspond with image proportions. The first context is occupied by the editor, where the user can control facade modeling process by adjusting and adding the necessary details to the existing geometry. The major changes are made within the editor window. As the facade editing involves process only 2d components, the camera is locked to orthographic projection.

The second window context represents a pure renderer, which is only responsible for displaying the actual 3d model, based on the geometry that was built into the editor. The control over the scene is implemented using a mouse to rotate around or zoom 3d model. To improve scene look, additional shaders, such as Blinn-Phong, Variance Shadow Mapping and simple shading were implemented. The tab control allows user to switch between renderer and editor scenes. Grouped elements of the facade are displayed in the separate dialog window as a tree structure. The Figure below demonstrates a graphical user interface of the facade modeling environment.

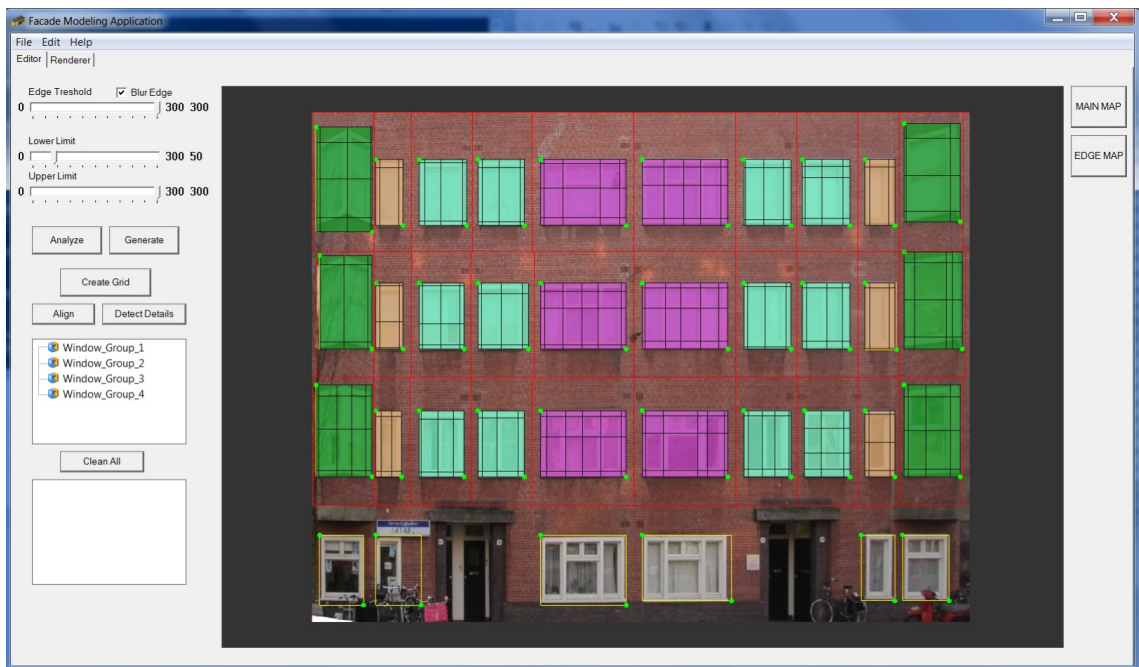


Figure 4.18: The Facade Modeling Application GUI.

Chapter 5

Evaluation

This chapter concludes the development of this project application by discussing the objectives that were defined in Chapter 1. Each objective is evaluated to determine whether or not it has been attained. In addition, this chapter outlines the results of informal comparison between developed application and industry package.

5.1 Extracting high-level facade structure

The implementation of this stage somewhat satisfies the objective of extracting high-level facade structure, but the reliability of the methods that have been chosen is not good enough to apply it as a general solution. Strong occlusion, low resolution and noise level are still limiting factors that prevent the facade structure to be correctly identified. Though the use of the Mutual Information algorithm that has been mentioned in chapter 2 would improve detection rate, but due to the time constraint of this thesis, it was not implemented in this project.

5.2 Detection of high-quality facade elements

This objective involved to extract architectural elements, such as windows, doors and other domain specific details. To simplify extraction of the facade structure, the ground level of each building was not considered due to the fact that it contains unusual details, such shops, cars, pedestrians and other objects that occlude the ground floor

view. Since doors are usually located on the ground floor, they are automatically ignored by the algorithm. So, the task was narrowed down to correctly determine window details, such as pane and glass, as they are the main elements of the window.

The implementation shows promising results and it is capable to process all types of windows with a rectangular shape. However inefficient and incomplete implementation makes the appearance of false positives at some stages. Even with false results algorithm still allows the user to choose the most appropriate representation of the window pane and fill missing details using interactive set of tools. The provided technique is based on LSWMS, but it sometimes does not identify obvious line segments, where Hough transform does, but it is quite susceptible to noise. The solution could exploit the potential of combining the Hough and LSWSM to improve the line segments detection rate. In addition, selecting a better clustering algorithm, would increase the quality level.

5.3 User assisted modeling tools

Different tools have been implemented to assist modeling process. Some of them like creating a grid and aligning windows along x and y axis, do not require much emphasis from the user side, other than clicking particular button. The other ones require user involvement to achieve desirable results. These are window size correction, structure grid adjustment, windows and regions grouping. Although it is still possible to create a 3d model of the facade just relying on existing functionality, however there are a lot of issues that should be addressed.

Taking into account modeling process, the application provides a function to remove the false positive window, but no function to draw a window in the proper location. There is also no option to remove incorrectly detected lines or false positive lines in the window tile, however this can be quickly implemented. When the user aggregates similar windows, sometimes it involves grouping up to 30-40 windows by clicking on each. This is quite a tedious job to do, so there should be implemented a mechanism to select similar windows at once according to their geometry or size. The selection of windows just by using their size or area, might introduce a lot of false positive selections. In this case the usage of the Mutual Information algorithm would be handy to speed up window selection.

The current version of the application is not implemented in real-time. In order to see applied changes, the user has to generate results. In addition to the real-time problem there is no functionality to control depth of grouped elements. The depth value is predefined and cannot be adjusted. Another option that must be included is the UNDO-function that allows the user to step back in case of incorrect decisions. The list of the new functionalities can be extended even more, however to create a 3D model, it is enough to use current interactive tools.

5.4 Interactive modeling environment

During the development stage, most of the attention was paid towards developing an interactive editor with an automatic support, running on a background. The implementation of the environment satisfies the objective in terms of functionality, but not from the design point of view. The user can interact with scene by moving the camera position, zooming and adjusting necessary components of the facade.

The renderer was created exclusively to display 3D model with some shaders applied for more realism. It does not provide any interactive features, other than zooming and rotating camera around the model. However initial goal was to achieve a real-time workflow in the renderer window too. The idea was to adjust the depth of the grouped elements on the fly, as well as the proportions of the 3d model itself.

5.5 Results

Conducting a full experiment, where users with a limited knowledge about the facade reconstruction are trying to create a 3d model using our application was beyond the scope and time available to this project. However an interesting statistic was retrieved during the informal comparison of this application and industry standard package: 3Ds Max. The comparison results that demonstrate how much it would take to create a simple facade model using both tools are outlined in the table below.

Taking into account the approximate time that was spent on creating a 3d model in 3Ds Max, is the approximate time +/- 1 min that would be spent by a the general content creator, who has an experience with 3D applications. As the architectural




Facade				
3Ds Max	~7:50 min	~9:40 min	~12:30 min	
Our tool	~0:50 min	~1:05 min	~1:30 min	

Table 5.1: Data Units, Sources, and Dates

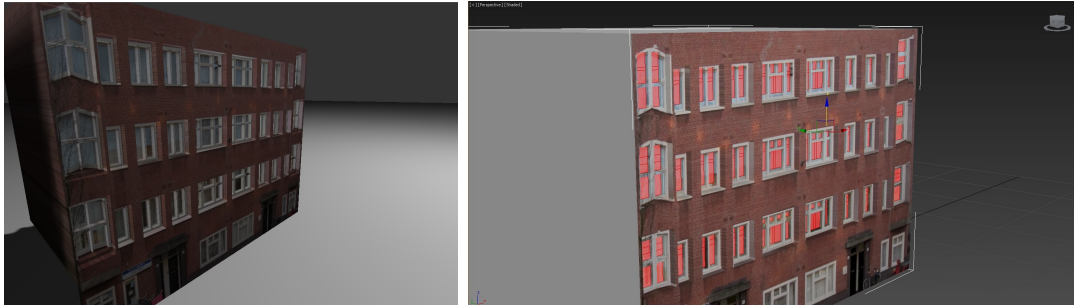


Figure 5.1: The comparison between implemented application(left) and 3Ds Max(Right)

facade becomes more complex, the more time it takes to do it in 3Ds Max and relatively slight increase in time using our tool. The modeling time using our approach can be decreased even more, by optimizing our solution and introducing new methods that might potentially speed up the modeling process.

Figure 5.2 illustrates the visual comparison of 3d models that were created using our solution and 3Ds Max. Considering the fact that professional modeling package provides a broader range of tools, it usually requires a good understanding of the modeling principles and huge amount of time to get used to work with software of such type. However tools that are designed to implement specific tasks are much easier to understand and it takes much less time to deliver the final model, where the quality will be negligible in comparison to industrial software.

5.6 Limitations

There are a number of limiting factors that prevent the acquisition of facade structure, which in turn makes impossible to process architectural elements further. The first one and the most important is the symmetry of the facade image. In order to correctly determine structure, input data should contain equal number of window rows per each column. The second limitation or a group of limiting factors is usually associated with input data quality, such as:

- Strong image gradient and occlusions.
- Too low resolution.
- Ambiguous regions (e.g Blocky facade structures).
- Image artifacts and reflections.

These influences on the automatic detection methods that are in turn preventing the subsequent stages from successful completion. Relying on the fact that most of the facade structures have rectangular windows, the design of the window detection algorithm was initially aimed towards processing only rectangular shape, however other shapes were also considered, but could not be achieved due to the time constraint. Different facade images have been processed. Figures that illustrate successful and failure cases can be found in the Appendix.

Chapter 6

Conclusions and future works

6.1 Summary

In this thesis we annotated semi-automatic approach of the facade modeling using a single image.

The tool that has been developed, encompasses a combination of computer vision and graphics area to provide a practical solution of the facade reconstruction. The main goal of this dissertation was to create a detailed facade look using a novel technique. Under the assumption that the facade images utilize the symmetry properties. We have applied window detection methods using histogram profiling for vertical and horizontal directions based on the edge map and Hough lines. The important aspect of our application it to assist the user in a challenging part of the facade modeling. Using interactive approach such as correcting geometry, allow changes to be synchronized between group members and flexible editing mechanisms, make the modeling process much easier and quicker. Furthermore the new method of extracting window details was introduced to allow the user in no time create a detailed look of the widows.

6.2 Future works

Despite the fact that the methods used in this thesis allowed to reach good results for the facade, there are still a lot of problems related with very complex facade images, such as Jungle facade or Gothic style buildings. To target this niche of architectural

structures a more robust and systematic approach needs to be involved. A good candidate is a Mutual Information algorithm that was discussed in Chapter 2, gives a much robust representation of architectural structure.

The most promising contributions should be dedicated towards establishing a good set of solutions, that are not affected by occlusions, noise level, distortions or other facade structural features such as asymmetric alignment. The automatic approaches are still not 100% reliable. We believe that developing a robust and practical framework also needs to be contributed. Implementing a flexible editing mechanism that allows the user to adjust geometry on the fly is more preferable. The majority of the facade images share the same property of the symmetry and structure, for this reason shape grammar with encoded parametric templates can be integrated to speed up modeling process.

6.3 Conclusion

In this project we retrieved very important semantics of the facade modeling: a full 3d reconstruction of the facade and window details extraction. Although a valuable start towards human-like facade interpretation was made, it is still can be considered as a drop in the ocean in comparison to what humans perceive. With the advancement of new algorithms in the computer vision, the current facade problems will become more achievable and integration of interactive tools will be decreased to a minimum.

Appendix

Figures below demonstrate facade images that were successfully reconstructed into 3D models.

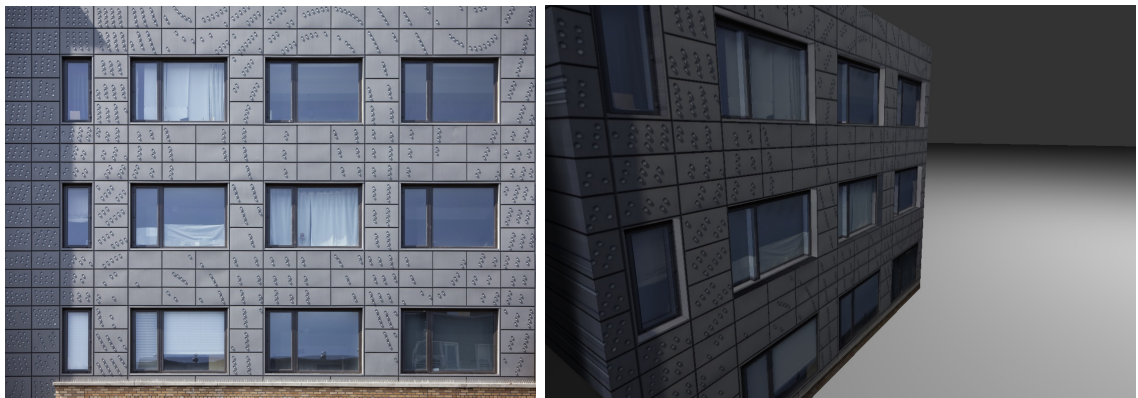


Figure 1: Test 1.



Figure 2: Test 2.



Figure 3: Test 3.

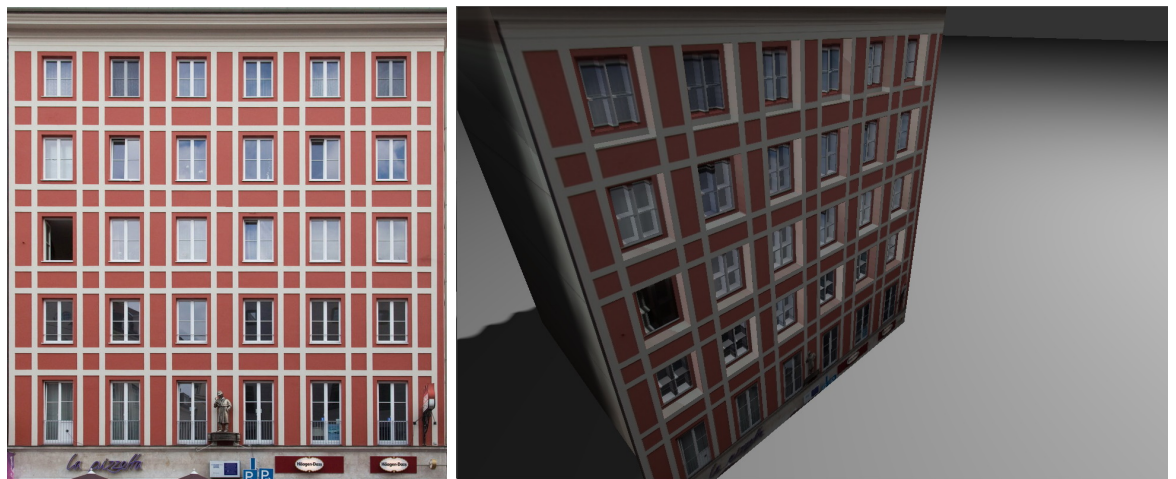


Figure 4: Test 4.

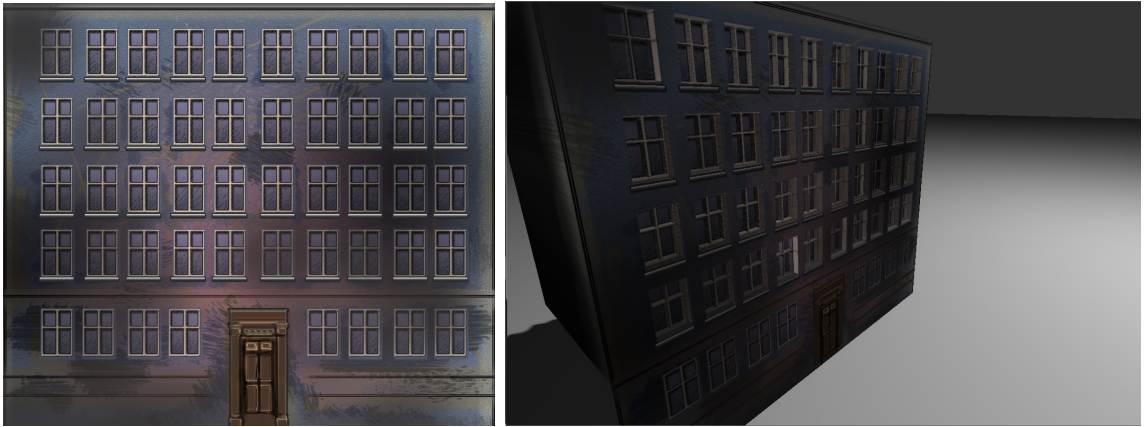


Figure 5: Test 5.

Figures below demonstrate failed test due to a image quality reasons.

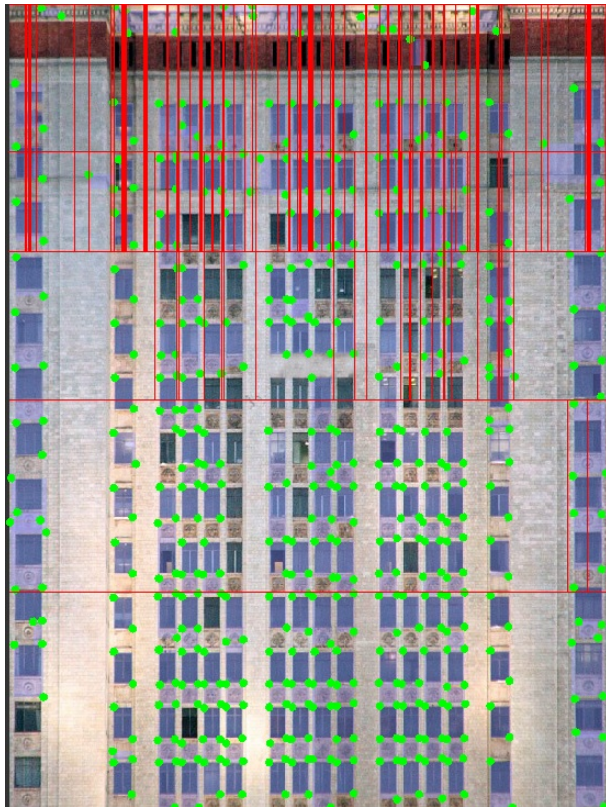


Figure 6: The facade structure was not detected correctly, as some of the windows are overlapping due to incorrect contour detection.



Figure 7: Overlapping window and not all of the window were detected.



Figure 8: First floor windows were not detected due to additional frame gradient, causing the algorithm set seconds floor as the first one.

Bibliography

- [1] P.Musialski, P.Wonka, D.G.Aliaga, M.Wimmer, L.van Gool, W. Purgathofer, A Survey of Urban Reconstruction, *Computer Graphics Forum*, 2013.
- [2] Georgios Vouzounaras, Petros Daras, Michael G. Strintzis, Automatic generation of 3D outdoor and indoor building scenes from a single image, *Springer*, 2011.
- [3] Tjerk Kostelijk, Semantic annotation of urban scenes: Skyline and window detection, *Msc Thesis*, 2012.
- [4] Wiraj Kulkarni, Rohan Nagesh, Hong Wu, Window Detection in Frontal Facades, *University of California, Berkeley*, 2009.
- [5] Michal Recky, Franz Leberl, Window Detection in Complex Facades, *Visual Information Processing (EUVIP)*, 2010.
- [6] Przemyslaw Musialski, Meinrad Recheis, Stefan Maierhofer, Peter Wonka, Werner Purgathofer, Tiling of Ortho-Rectified Facade Images, *SCCG*, 2010
- [7] Haider Ali, Christin Seifert, Nitin Jindal, Lucas Palletta, Gerhard Paar, Window Detection in Facades, *Vienna University of Technology*.
- [8] Pascal Muller, Gang Zeng, Peter Wonka, Luc Van Gool, Image-based Procedural Modeling of Facades, *ACM SIGGRAPH*, 2007.
- [9] MUSIALSKI P, RECHEIS M, MAIERHOFER S, WONKA P, PURGATHOFER W, Tiling of ortho-rectified facade images, *In 26th Spring Conference on Computer Graphics - SCCG 10, ACM Press*, (New York, USA, May 2010).

- [10] Przemyslaw Musialski, Michael Wimmer, Peter Wonka, Interactive Coherence-Based Facade Modeling, *EUROGRAPHICS*, 2012.
- [11] Korah T., Rasmussen C., Analysis of Building Textures for Reconstructing Partially Occluded Facades, *ECCV*, 2008.
- [12] Konushin .V, Vezhnevets V., Automatic building texture completion, *Proc. of GraphiCon2007*, M.: MSU.
- [13] Wenzel S., Drauschke M., Forstner W., Detection of Repeated Structures in Facade Images, *Pattern Recognition and Image Analysis*, 2008.
- [14] Panagiotis Koutsourakis, Loic Simon, Olivier Teboul, Single View Reconstruction Using Shape Grammars for Urban Environments, *Computer Vision, 2009 IEEE 12th International Conference*.
- [15] Jorge Hernandez, Beatriz Marcotegui, Segmentation of Facade Images using Ultimate Opening, *31me journee ISS France*, 2008.
- [16] Dengxin Dai, Mukta Prasad, Gerhard Schmitt, and Luc Van Gool, Learning Domain Knowledge for Facade Labelling, *Computer Vision ECCV*, 2012
- [17] Sung Chun Lee, Ram Nevatia, Extraction and Integration of Window in a 3D Building Model from Ground View images, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004
- [18] Jan Cech and Radim Sara, Windowpane Detection based on Maximum A posteriori Probability Labeling, *IWCIA*, 2008.
- [19] Olivier Teboul, Loic Simon, Panagiotis Koutsourakis, Nikos Paragios, Segmentation of Building Facades Using Procedural Shape Priors, *Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [20] Marcos Nieto, Carlos Cuevas, Luis Salgado, Narciso Garcia, Line segment detection using weighted mean shift procedures on a 2D slice sampling strategy, *Springer*, 2011.

- [21] Angelo Martinovic, Markus Mathias, JulienWeissenberg, Luc Van Gool, A Three-Layered Approach to Facade Parsing, *ECCV12 Proceedings of the 12th European conference on Computer Vision - Volume Part VII*.
- [22] Marcos Nieto, Line segment detection OpenCV C++, *Marcos Nieto Blog*, 2012.
- [23] , Line-line intersections, *Wikipedia Article*, can be accessed via: *http : //en.wikipedia.org/wiki/Line – line,ntersection*, last time accessed on: 28.08.2014.