Image-based Procedural Modeling of Facades

Pascal Müller* ETH Zürich
Gang Zeng* ETH Zürich
Peter Wonka† Arizona State University
Luc Van Gool* ETH Zürich / K.U. Leuven

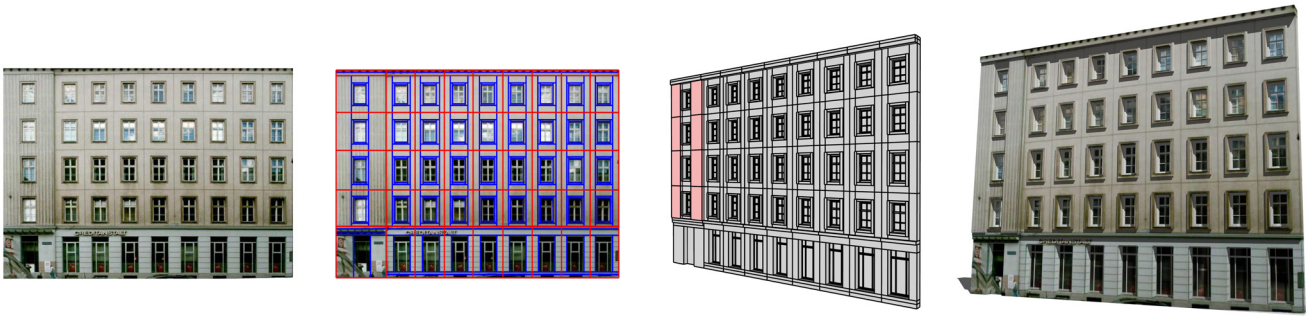

Figure 1: This paper describes how single facade textures of arbitrary resolution can be converted to semantic 3D models of high visual quality. Left: rectified facade image as input. Middle left: facade automatically subdivided and encoded as shape tree. Middle right: resulting polygonal model. Right: rendering of final reconstruction including shadows and reflections enabled by semantic information.

Abstract

This paper describes algorithms to automatically derive 3D models of high visual quality from single facade images of arbitrary resolutions. We combine the procedural modeling pipeline of shape grammars with image analysis to derive a meaningful hierarchical facade subdivision. Our system gives rise to three exciting applications: urban reconstruction based on low resolution oblique aerial imagery, reconstruction of facades based on higher resolution ground-based imagery, and the automatic derivation of shape grammar rules from facade images to build a rule base for procedural modeling technology.

CR Categories: F.4.2 [Mathematical Logic and Formal Languages]: Grammars and Other Rewriting Systems I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism I.6.3 [Simulation and Modeling]: Applications

Keywords: Procedural Modeling, Image-based Modeling, Urban Reconstruction, Design Computation, Architecture

1 Introduction

This paper addresses the following problem: given a single image of a building facade as input, how can we automatically compute a 3D geometric model that (1) looks like a plausible interpretation of the input image, (2) has much higher resolution and visual quality than the input image, and (3) includes a semantic interpretation. The proposed method is designed to work with input textures of arbitrary resolution. First, our method is able to enhance simple textured building mass models constructed from aerial images (see Fig. 2). While current models allow for stunning images from a bird's eye view, the resolution of geometry and textures is not sufficient for important applications where a user navigates near ground level. For example, a virtual training application for emergency response will require interaction with a detailed model of high visual quality and realism including semantic information for meaningful simulation. Other applications in the entertainment industry, urban planning, visual impact analysis, driving simulation, and military simulation have similar requirements. The second type of data we consider are single images from ground-based photographs (an example is given in Fig. 1). In this context modeling based on single facade images provides a simple and robust modeling framework. For example, a modeler could use imagery from the internet or a commercial supplier to quickly create an interesting urban model.

Our research problem is related to computer graphics and computer vision. While computer graphics techniques fulfill the quality criteria of most applications, the predominant method of large-scale reconstruction is to invest several man years of labor. Therefore, recent techniques in computer graphics are interested in efficient large-scale modeling [Parish and Müller 2001; Müller et al. 2006], but these techniques do not allow to establish a very close resemblance to a real environment. In computer vision there is a wide range of strategies for urban reconstruction, but there is only very limited work in the area that is helpful to our task: reconstruction from single facade images. Firstly, this is an ill-conditioned problem. Secondly, the driving applications are only just ramping up.

In our approach, we mimic the procedural modeling pipeline from computer graphics to subdivide a facade texture in a top-down manner into elements such as floors, tiles, windows, and doors. We use image analysis to ensure a meaningful subdivision corresponding to the input image. The major contributions of this paper are as follows: (1) We introduce the usage of mutual information to extract the high-level facade structure by detecting repetitions. (2) We propose a novel method to derive a top-down subdivision scheme by combining computer vision techniques and procedural modeling methods based on architectural knowledge. (3) We automatically infer shape grammar rule sets from complex images.

*e-mail: {pmueller|zengg|vangool}@vision.ee.ethz.ch
†e-mail: peter.wonka@asu.edu

Figure 2: The method can be applied to enhance state-of-the-art oblique aerial imagery. Left: close-up view of original texture (note the blur due to the low resolution). Right: our reconstruction.

1.1 Related Work

In the 70's, Stiny introduced the seminal idea of shape grammars [Stiny 1975] as a formal approach to architectural design. Shape grammars were successfully used for the construction and analysis of architectural design [Stiny and Mitchell 1978; Koning and Eizenberg 1981; Flemming 1987; Duarte 2002]. The direct application of shape grammars in computer graphics is intrinsically complex. The strategy of recent work in computer graphics was to simplify the geometric rules [Stiny 1982], but to extend the derivation mechanisms [Parish and Müller 2001; Wonka et al. 2003; Marvie et al. 2005; Müller et al. 2006]. Shape grammars could be complemented by cellular textures [Legakis et al. 2001] to generate brick layouts and generative mesh modeling [Havemann 2005] to generate facade ornaments. Many aspects of procedural architectural modeling in computer graphics are inspired by concepts introduced by L-systems [Prusinkiewicz and Lindenmayer 1991], such as geometry sensitive rules [Prusinkiewicz et al. 1994], the incorporation of computer simulation [Měch and Prusinkiewicz 1996], and artistic high-level control [Prusinkiewicz et al. 2001] .

In computer graphics, several techniques for matching shapes to 3D point clouds exist. Related to our method is the approach presented by [Ramamoorthi and Arvo 1999]. They recover generative models from 3D range data by automatically selecting the best suited hierarchy of shape operators and adjusting their parameters. In computer vision, a similar variety of algorithms for matching high-order structures to images exist. For example, Han and Zhu [2005] proposed a two step method that first detects rectangles in a perspective image (bottom-up) and then merges the rectangles with an attributed graph grammar (top-down). In contrast, our facade-specific interpretations include top-down influences from the very start.

Urban reconstruction algorithms make use of a wide variety of input data, for example: ground-based facade images [Debevec et al. 1996; Jepson et al. 1996; Dick et al. 2001; Wang et al. 2002; Lee et al. 2002; REALVIZ 2007], interactive editing using aerial images [Ribarsky et al. 2002], aerial images combined with ground-based panorama images [Wang et al. 2006], ground-based laser scans combined with aerial images [Früh and Zakhor 2001], ground-based and airborne laser scans [Früh and Zakhor 2003], ground-based laser scans combined with facade images [Karner et al. 2001], and LIDAR, aerial images, and ground-based images [Hu et al. 2006]. In practice, several systems still resort to semi-automatic methods, e.g. [Lee and Nevatia 2003; Takase et al. 2003; Bekins and Aliaga 2005]. Generally, in these systems, a user is assisted by computer vision methods [Debevec et al. 1996] during modeling. Most automatic processes need to rely on simplifications, such as considering windows as dark rectangles [Alegre and Dellaert 2004; Brenner and Ripperda 2006] or limiting the appearance of facade elements to pre-specified types, even when leaving some freedom in the values of their parameters [Dick et al. 2004]. The problem is simplified if 3D data is available as depth displacements between elements (e.g. windows vs. walls) yield a strong, additional cue for their segmentation [Schindler and Bauer 2003; Dick et al. 2004; Brenner and Ripperda 2006]. In [Lee and Nevatia 2004], a single image approach for window detection is presented that, similar to [Schindler and Bauer 2003], fully relies on the detection and analysis of edges. This is a notoriously fragile operation when taken on its own, and comes at the price of only being able to deal with windows placed in otherwise homogeneous facades. Our method is less prone to such low-level fragility - as we exploit high-order symmetries early on - and is able to detect additional facade structures like ledges and window sills.

1.2 Overview

The proposed solution consists of four parts organized as *stages* in a pipeline. This pipeline transforms a single image into a textured 3D model including the semantic structure as a shape tree. We use a top-down hierarchical subdivision analogous to splitting rules in procedural facade modeling [Wonka et al. 2003; Bekins and Aliaga 2005; Müller et al. 2006] (see Fig. 3).

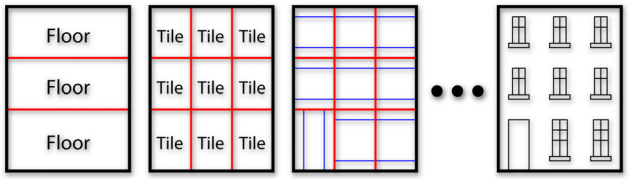

Figure 3: Our system computes a hierarchical subdivision of facades. This subdivision scheme was successfully employed in the procedural modeling literature by various authors.

The input images can stem from ground-based or aerial imagery. In practice, urban reconstruction is done by semi-automatic methods based on [Debevec et al. 1996]. Most of these implementations (e.g. [REALVIZ 2007]) can automatically extract various types of texture atlases. Thus, rectified facade textures including corresponding real-world sizes (from the reconstructed model) can be easily extracted from such photogrammetric urban models. However, there are various public tools for rectification for cases where such rectification is still needed. We used an automatic rectification tool of our own (described in appendix A). In the following we give a short description of the individual stages and fill in details in later sections:

Facade structure detection: The input is a single rectified facade image. In section 2 we describe an algorithm to automatically subdivide the facade image into floors and tiles by using mutual information. A *tile* is an important concept from procedural modeling that denotes an architectural element such as a window or door including the surrounding wall (depicted in Fig. 3).

Tile refinement: The input to this stage is a facade image subdivided into tiles including clustering information describing which groups of tiles are similar. This stage (see section 3) segments individual tiles into smaller rectangles. We make use of the translational symmetry in order to achieve a robust subdivision. This stage draws from split grammars [Wonka et al. 2003].

Element recognition: The stage described in section 4 matches smaller image rectangles from the previous stage with 3D objects from a library of architectural elements. Finally we output a 3D textured model including the semantic structure as a shape tree.

Editing and Shape Grammar Rule Extraction: Semantic facade interpretation can be used for various editing operations, including the extraction of shape grammar rules from the previously derived shape tree (see section 5).

2 Determination of Facade Structure

The goal of this stage is to detect the general structure in a facade and to subdivide it accordingly. The input is a single image and the output a subdivision into floors and tiles. Additionally, we compute symmetry information so that we know for each pixel the location of corresponding pixels in symmetric tiles. Fig. 4 shows an example subdivision computed by this algorithm. Please note that while this example is fairly symmetric, it is already challenging due to complex shadowing and different window appearances.

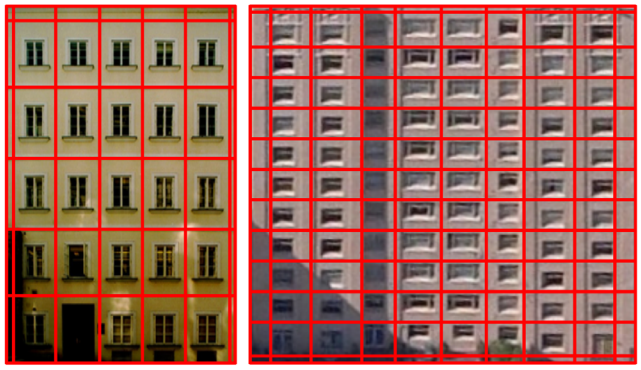

Figure 4: In the first stage of the process, the facade is automatically subdivided into tiles (illustrated as red lines). Moreover, similar tile appearances are detected and registered in groups.

The algorithm has three steps. Firstly, we detect similar image regions using mutual information (MI). Secondly, we create a data structure called *Irreducible Facade* (IF) that allows us to encode information about the symmetries that govern the floors and tiles. Thirdly, we analyze the IF to find the optimal, further subdivision of the tiles. Key to this algorithm is the detection of translational symmetry in step 2 before we compute splitting lines in step 3. This not only improves the robustness of the algorithm, but also guarantees that similar elements are split at corresponding positions.

2.1 Mutual Information

In probability theory and information theory, the MI of two random variables is a quantity that measures the mutual dependence of the two variables. It quantifies the Kullback-Leibler distance [Kullback 1959] between the joint distribution, $P(A=a,B=b)$, and the product of their marginal distributions, $P(A=a)$ and $P(B=b)$, that is

$$MI(A,B) = \sum_{a,b} P(a,b) \log \frac{P(a,b)}{P(a) \cdot P(b)}, \qquad (1)$$

where A and B are two random variables. MI was proposed as a similarity measure on image intensities for 3D rigid registration in medical imaging by Wells et al. [1996]. It does not assume any simple or one-to-one relationship between the intensities. This gives MI the flexibility that we seek.

In order to utilize MI to measure the similarity between image regions $\mathscr{R}_1$ and $\mathscr{R}_2$, the intensity values of the corresponding position pairs are considered. The joint and marginal intensity distributions are taken to be the normalized joint and marginal histograms of those pairs. The MI-based similarity $MI(I(\mathscr{R}_1),I(\mathscr{R}_2))$ measures the statistical dependence between intensities at corresponding positions in regions $\mathscr{R}_1$ and $\mathscr{R}_2$. Accordingly, $I(\mathscr{R}_1)$ and $I(\mathscr{R}_2)$ are the intensities at corresponding image locations in $\mathscr{R}_1$ and $\mathscr{R}_2$. Next we describe how MI is used to find similar image regions.

2.2 Symmetry Detection

In this step we use MI to find similar floors and tiles in the image. In the vertical direction, we expect translational symmetry of floors, even though the bottom and top floors often differ. In the horizontal direction, floors often exhibit an ordered sequence of repeated patterns or 'tiles'. Our algorithm searches first for symmetry in the vertical and then in the horizontal direction. In the following, we will describe the solution for the vertical direction. The solution for the horizontal direction is very similar and we will only indicate the minor differences.

Let $\mathscr{R}_{y,h}$ denote the rectangular image region with a lower left corner of $(0,y)$ and upper right corner of $(imagewidth, y+h)$. For the repetition detection in the vertical direction we need to analyze the similarity between $\mathscr{R}_{y_1,h}$ and $\mathscr{R}_{y_2,h}$ for arbitrary values of y_1, y_2 and h. These three parameters span a 3D search space, which is too big to be explored exhaustively, given the time MI takes to compute.

The problem is simplified by only analyzing adjacent regions $\mathscr{R}_{y,h}$ and $\mathscr{R}_{y-h,h}$. This, of course, restricts the order that can be found. On the other hand, the vast majority of cases correspond to such contiguous, periodic tiling. The similarity between two adjacent regions with height h is computed by:

$$S(y,h) = MI(I(\mathscr{R}_{y,h}), I(\mathscr{R}_{y-h,h})). \qquad (2)$$

We use an exhaustive search strategy to compute $S(y,h)$ for all positions y, and a range of parameters for h. The range specifies reasonable architectural floor heights. We use $3m < h < 5.5m$ for all examples in the paper and video (in the horizontal direction there is a larger variety of tile sizes, ranging from $0.5m$ to $9m$). The search yields the best symmetry value $S_{max}(y) = \max_h S(y,h)$ for each horizontal image line and the corresponding value $h_{max}(y) = \arg\max_h S(y,h)$ that indicates the height resulting in the best symmetry value (see Fig. 5). Please note that the values for $h_{max}(y)$ are fairly constant, but that peaks in $S_{max}(y)$ do not correspond to natural splitting lines in the image or floor divisions for that matter.

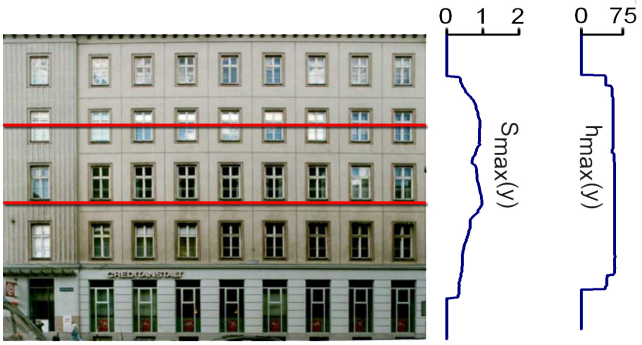

Figure 5: The symmetry detection of a facade image in the vertical direction. For each line in the facade image we show the best symmetry value and the corresponding height. The red lines are where $S_{max}(y)$ reaches a local maximum.

2.3 The Irreducible Facade

The IF is a data structure that encodes the facade's symmetry based on the computation from the previous section. The IF is a kind of collage $IF(x,y)$ that stores a list of pixels instead of a single pixel at each location (x,y), i.e. a collage of stacks of original, similar image fragments. The IF summarizes the facade, with pixels at symmetric positions now collapsed into the stack pixel lists. The idea is that the IF exhibits no further symmetries, hence the name 'irreducible'. Fig. 6 right shows the IF for the facade in Fig. 5. If no

symmetry can be detected in the original image, then it will be its own IF. The concept of a miniature image that captures all essential information has been presented also by [Jojic et al. 2003]. Their epitomes encode local texture but do not preserve geometric properties (e.g. lines are not mapped to lines). Hence epitomes are not meaningful images themselves, making them inappropriate here.

The IF is computed as follows: (1) $IF(x,y)$ is initialized to be identical to the facade image. (2) We iteratively select the postition $y = \arg\max(S_{max}(y))$ and shrink the image by shifting the region $\mathscr{R}_{y-hmax(y),hmax(y)}$ over the region $\mathscr{R}_{y,hmax(y)}$. The pixels of the region on top are added to the pixel list of the region under it. In practice we store indices to the original image, so that the operation becomes reversible. We compute a new value $S_{max}(y)$ by setting it to the minimum of the two overlapping positions. This ensures stable clustering. Fig. 6, left, shows the result of stacking up similar floors for the input image of Fig. 5. The collage consists of three such floor stacks and a top cornice. Then a similar step removes the horizontal repetitions within the floors, resulting in the right part of the figure. (3) The algorithm terminates when no more symmetry can be found i.e. no value $S_{max}(y)$ exceeds the threshold $0.75 * \tau_{max}$, where τ_{max} is the best similarity score. Fig. 6 shows a result. Next we pass the IF to the next step and find splitting lines in the image.

Figure 6: Left: the facade from Fig. 5 after removing the vertical symmetry. Right: further removing of the horizontal symmetry yields the *Irreducible Facade*. Please note that we use the average pixel color for display purposes.

2.4 Structure Subdivision

Splitting lines subdivide the IF into 'floors', vertically, and 'tiles', horizontally within the floors. When the splitting lines are computed in the IF we implicitly know them for other symmetric elements and can expand them into a full facade subdivision.

After analyzing many facade images, the following strategy suggested itself: include horizontal splitting lines where vertical edges are rare and horizontal edges are dense, and vertical splitting lines in the opposite case. Relying on perfect edge extraction would render the process fragile, but we can nevertheless hope that traces of edges will be found at the tile centers where frames of windows or doors are expected to appear. The following two functions are used to signal the presence of horizontal or vertical edges:

$$\begin{aligned} hor(x,y) &= \max\{(\frac{\partial I}{\partial y})^2 - \alpha|\nabla I|^2, 0\} \\ &= \max\{(1-\alpha)(\frac{\partial I}{\partial y})^2 - \alpha(\frac{\partial I}{\partial x})^2, 0\} \\ ver(x,y) &= \max\{(\frac{\partial I}{\partial x})^2 - \alpha|\nabla I|^2, 0\} \\ &= \max\{(1-\alpha)(\frac{\partial I}{\partial x})^2 - \alpha(\frac{\partial I}{\partial y})^2, 0\} \end{aligned} \quad (3)$$

where $\frac{\partial I}{\partial}$ is the partial differential operator and $\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ is the gradient operator. We used $\alpha = 0.9$ for all results in this paper. The preference for a horizontal splitting line at position y or a vertical one at position x is made dependent on the values of two interest functions:

$$\begin{aligned} Ver(y) &= (\sum_x ver(x,y)) * g_\sigma(y) - \beta(\sum_x hor(x,y)) * g_\sigma(y) \\ Hor(x) &= (\sum_y hor(x,y)) * g_\sigma(x) - \beta(\sum_y ver(x,y)) * g_\sigma(x) \end{aligned} \quad (4)$$

where $g_\sigma(\cdot) = \frac{1}{2\pi\sigma^2} e^{-\frac{|\cdot|^2}{2\sigma^2}}$ is the Gaussian kernel and $*$ is the (one-dimensional) convolution operator. The first and second terms encode the density of vertical and horizontal edges respectively (horizontal and vertical edges in the opposite case). β is a small constant parameter with value 0.1 for all examples. Furthermore, we use $\sigma = 1m$ for all examples. High values of Ver (or Hor) will accrue in regions with strong vertical (or horizontal) edges. Based on this interest function, we can extract a set of potential positions $\{y_i\}$ (or $\{x_i\}$) for the splitting line at the local minima. If we want to include a horizontal splitting line, $Ver(y)$ should go through a local minimum at its y position. As can be seen, it strongly penalizes any presence of vertical lines and if the horizontal line is chosen, it is a locally dominant feature. Finally, an exhaustive search is employed for the optimal combination of these potential positions $\{Y_i\} \subset \{y_i\}$ with the constraint of prior knowledge of the floor height (we restrict the floor height to be between $3m$ and $5.5m$):

$$\{Y_i\} = \arg\min_{\{\hat{y}_i\}} \frac{\sum_i Ver(\hat{y}_i)}{||\{\hat{y}_i\}||}, \text{ with } 3 < \Delta\hat{y}_i < 5.5, \{\hat{y}_i\} \subset \{y_i\} \quad (5)$$

where $||\cdot||$ denotes the number of elements in a set and $\Delta\hat{y}_i = \hat{y}_{i+1} - \hat{y}_i$. Similarly, the inclusion of vertical splitting lines follows from the optimization

$$\{X_i\} = \arg\min_{\{\hat{x}_i\}} \frac{\sum_i Hor(\hat{x}_i)}{||\{\hat{x}_i\}||}, \text{ with } 0.5 < \Delta\hat{x}_i < 9, \{\hat{x}_i\} \subset \{x_i\} \quad (6)$$

The result of this optimization is shown as red lines in Fig. 1, second from the left, which actually shows the subdivision on the IF, symmetrically unpacked onto the original facade.

3 Subdivision of Facade Tiles

At this stage we want to subdivide the detected tiles into smaller regions. We propose an algorithm which recursively selects the best splitting line in the region under consideration. See Fig. 7 for an example. This structure subdivision is a concept used in procedural modeling and will automatically create a hierarchy of elements. Such a hierarchy will be essential for further analysis, such as the generation of rules for a shape grammar.

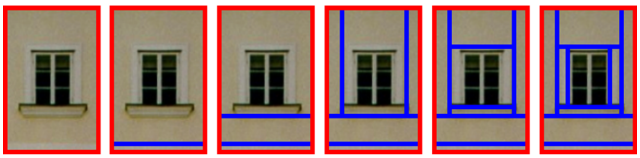

Figure 7: In the second stage of the process, the tiles are hierarchically subdivided (illustrated as blue lines). Each image represents one step of the subdivision.

Because individual tiles are noisy, the splitting algorithm exploits the knowledge about repetitions which is embedded in the IF. Fig. 8 left illustrates how noise makes the subdivision of individual tiles very unreliable. Therefore, the algorithm analyzes similar structures in other tiles to synchronize the derivation and in so doing, significantly improves the result (see Fig. 8 right). The algorithm works as follows:

```
initialize all tiles
while non-leaf and non-subdivided region left
  find best split candidate for each region
  synchronize splits of tiles within same group
  globally synchronize all split candidates
  subdivide regions into new regions
  mark non-subdivided regions as leafs
  mark small new regions as leafs
```

A region is a leaf node if it does not split any further or if it is smaller than a given minimum size τ_{size}, which has been set to 5 pixels for most examples (depending on sharpness). For airborne imagery this typically results in 1-3 splitting steps and for ground-based images we obtain about 2-5 splits. The splitting procedure consists of two main components: the selection of an optimal, local split and a global split synchronization.

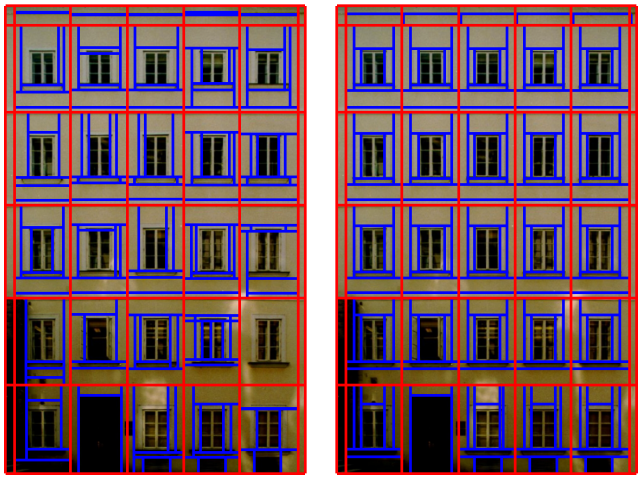

Figure 8: To make the splitting process more stable, we make use of the previously detected tile repetitions. Left: subdivided tiles based on per-tile local split detection. Right: result if global split synchronization is added.

3.1 Local Split Detection

The split detection algorithm aims at finding relevant edges closest to the boundary. The algorithm evaluates all splitting lines starting from the boundary to find the first suitable edge candidate moving inward from the left side, right side, top side, and the bottom side. We consider the following seven *split type* choices: (1) vertical dual split i.e. symmetric left and right split at once, (2) horizontal dual split, (3) left split, (4) right split, (5) top split, (6) bottom split, and (7) no split. Edges cross the entire region, which initially is a tile.

In order to assess the relevance of an edge, its strength needs to be compared against a threshold that takes account of the local noise level. This is quantified as the averaged gradient magnitude in a zone surrounding the boundary of the original tile. The algorithm of section 2 puts tile boundaries in wall regions and this value therefore indicates the typical edge response on walls. Edges are considered relevant only if their averaged gradient strength surpasses this wall response multiplied by a fixed factor τ_{edge}, in our experiments set to 4. The first relevant edge that is met is taken as a candidate, resulting in a maximum of four candidates (one for each side).

In a further selection step, we avoid edge intersections. Therefore, the edge-responding segments of the surviving edges (see e.g. the edge segment above the window in Fig. 9) are checked for crossing edges. There must be no transversal edge segment across a edge-responding segment of another detected edge.

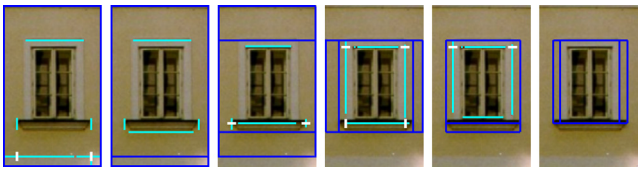

Figure 9: This figure shows how crossed edges prohibit certain split types and how the longest edges are selected. Cyan colored lines illustrate the edge-responding segments of the detected split candidates, and the blue lines display the winning splits. Split candidates that cross edge-responding segments of perpendicular split candidates are shown in white.

Among the remaining edge segments the longest candidate is chosen. If the runner-up is parallel and has a length of at least 80% of that of the winner, both are chosen and yield a dual split. This tends to save on the number of iterations needed and helps to further increase the effectiveness of the global split synchronization.

3.2 Global Split Synchronization

As can be expected the local split detection still suffers from image noise. We employ a solution to exploit the resulting imperfect symmetries based on the following ideas: (1) we can compare local solutions among a cluster of similar tiles and (2) we can compare local solutions among all tiles.

Figure 10: This derivation sequence illustrates the impact of the global split synchronization. Left: the algorithm without synchronization. Right: synchronization can eliminate almost all errors.

To synchronize split solutions within a cluster we select the most frequent split type. We then impose the most common split type onto all members in the group. Due to imperfections in the clustering, we have to carefully align the split locations. Obviously, we can only align vertical splits within columns of tiles and horizontal splits within rows of tiles. This may still yield imperfect symmetries between these columns and rows. The final step is to check for those imperfect symmetries. The algorithm is based on a simple idea that works very well in practice. There are only a limited set of element sizes on each facade. Therefore, we globally cluster all distances between parallel splitting lines and select cluster medians as final split values. After such clustering on a per-group basis, the process is repeated globally, using all the splitting lines extracted for the whole facade. Fig. 10 shows the impact of the global split synchronization step by step.

4 Matching 3D Elements

At the previous stage we obtained a subdivision of the facade image. Due to the hierarchical nature of the subdivision we can also store clustering information by storing groups of similar regions. Subdivision of facade tiles leads to a set of rectangular regions clustered into groups of similar regions. At this stage we want to match some of the architectural elements with 3D objects in a library. This is useful for the generation of high-quality geometric information and can provide some semantic interpretation. The solution has to fit the computer graphics modeling pipeline leading to two constraints: We need fast computation times and a general solution working for 3D models in a library. We believe that it would be difficult for professional modelers to create parametric templates or complex probabilistic models as in [Dick et al. 2004].

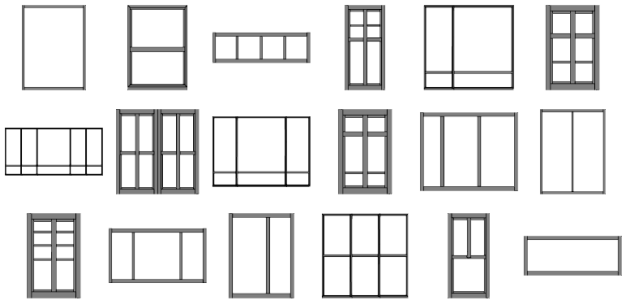

Figure 11: At this stage, we match the detected regions with architectural elements. The figure shows the 2D representations of an example set of 18 elements out of our 150 elements library.

In order to achieve our goal, we first create a 2D representation for each object $\{e_i\}$ in the element library. The 2D representations, denoted as $\{p(e_i)\}$, have been computed as fronto-parallel projections in Autodesk's Maya by using a simple MEL script that iterates over the element library (foreach element: load element, adjust camera, render image). Some examples can be found in Fig. 11. We can then compute the region type $T(\mathscr{R})$ for each rectangular image region $\mathscr{R}$ as follows:

$$T(\mathscr{R}) = \arg\max_i MI(I(p(e_i)), I(\mathscr{R})). \quad (7)$$

The equation above is direct and efficient. However, it may yield some mismatches due to noise and different appearances of the same element. Fortunately, the clustering information in the IF provides an additional constraint so that elements in the same cluster should belong to the same type. Thus, we can determine an element type for each cluster C as follows:

$$T(C) = \arg\max_i \sum_{\mathscr{R}\in C} MI(I(p(e_i)), I(\mathscr{R})). \quad (8)$$

Hence, the best-fitting element per cluster is selected. The intensities that are also kept in the IF, allow for a backprojection of the original textures. Furthermore, the objects in our library can contain shader information, e.g. material attributes like reflecting glass.

5 Editing and Rule Extraction

At this stage of the pipeline, the resulting facade interpretation is encoded as shape tree including elements, but does not contain depth information. Therefore, simple editing operations are required to set the depth of the facade elements. The user can select clusters of elements and adjust their depth interactively. Fig. 12 illustrates this editing process. The added depth information is stored in the shape tree.

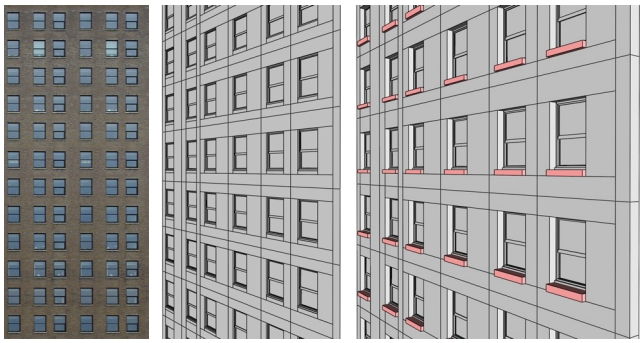

Figure 12: Adjusting depth. Left: input image. Middle: automatically extracted subdivision with matched elements. Right: the user can manually adjust the depth of selected region clusters.

In a next step, we can encode the computed subdivision (i.e. the shape tree) as shape grammar rules [Bekins and Aliaga 2005]. The generated rules contain the hierarchical information and correct dimensions. In addition, we can make use of the similarity information to encode the repetition of tiles. As example, we present the rule set for the facade in the teaser encoded as *CGA Shape* [Müller et al. 2006]. The rules for the facade structure (i.e. floors and tiles) are encoded as combination of subdivision and repeat split:

1: facade $\rightsquigarrow$
Subdiv(Y,5.4,1r,3.9,0.6){ floor1 | Repeat(Y,4){floor2 } | floor3 | top }
2: floor1 $\rightsquigarrow$ Subdiv(X,5.3,1r){ tile1 | Repeat(X,3.1){ tile2 } }
3: floor2 $\rightsquigarrow$ Subdiv(X,5.3,1r){ tile3 | Repeat(X,3.1){ tile4 } }
⋮

The first rule splits the facade into floors and the other rules split each floor into tiles. Rule 1 and 3 are illustrated in Fig. 13. The dimensions of the subdivision split operation *Subdiv* for non-repetitive shapes are in absolute values and the others in relative values according to their relative size. Due to the nature of CGA Shape, this ensures that the resulting rule set is size-independent and can later be used in a flexible way. If no repeating elements exist (either in horizontal or vertical direction), the dimensions of the first and last shape of the split are in absolute values, the others in relative values. The following rules encode the tiles:

6: tile1 $\rightsquigarrow$ Subdiv(X,1r,0.3){ region1B | region1T }
⋮
9: tile4 $\rightsquigarrow$ Subdiv(X,1r,1.9,1r){ region4L | region4C | region4R }
⋮

If we have a non-dual split type (section 3.1) as in rule 6, the subdivision split divides the tile into two shapes. Note that the dimension of the smaller shape is encoded as absolute value while the bigger shape has 'floating' dimensions i.e. given as relative value. For dual splits, as in rule 9, we decided to make the outer regions floating and the center region absolute sized. The split dimensions of multiple-defined shapes can be computed by averaging the split positions (illustrated as dashed lines in Fig. 13). Therefore, the split synchronization ensured the same hierarchy of split types within the same group (note that the texture can be averaged accordingly). The next splitting levels are encoded in a similar way:

14: region1B $\rightsquigarrow$ Subdiv(X,1r,0.3){ region1BB | region1BT }
15: region1T $\rightsquigarrow$ S(1r,1r,0.2) T(0,0,-0.2) I(wall)
⋮

If we arrive at a leaf shape as in rule 15, we set the depth of the scope, translate it accordingly and insert a wall shape (cube) or the matched object from the element library. We end up having a complete rule set that describes the segmented facade and can be applied to differently dimensioned facades as illustrated in Fig. 14. Also parts of the rule set could be included in other building designs via copy-paste. Note that this shape tree to rule set conversion could also be done in a depth-first manner (instead of breadth-first).

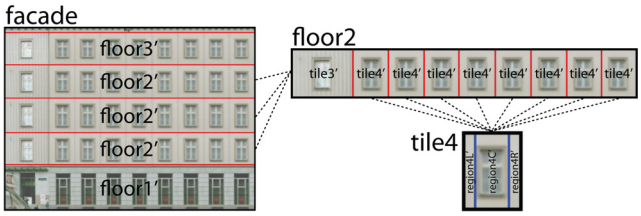

Figure 13: The extracted shape tree can be automatically converted into a CGA shape grammar rule set. The figure illustrates rule 1 (subdivision into floors), rule 3 (subdivision into tiles on middle floors), and rule 9 (subdivision of the corresponding tile).

Figure 14: The extracted rule set is size-independent and can be applied to copy-paste the design (or part of) onto other 3D models.

6 Results

We implemented our system in C++. The typical running time for the first stage, described in section 2, is a couple of minutes. For a 1600×1200 image the average time was 3 minutes. The typical running times of stages 2 and 3 was between 30 seconds and 90 seconds, where stage 2 requires about 90% of the computation time.

We conducted our first tests using ground-based images as input. One example can be seen in the teaser. The side by side comparisons in Fig. 15 show how even high resolution imagery can be visually enhanced with our method. To simulate airborne urban reconstruction, we generated 3D models out of aerial facade images from buildings in San Francisco. The input images, the computed

Figure 15: Ground-based examples. Left: input textures. Right: resulting 3D mesh with shadows and reflections applied.

subdivision and the resulting models are shown in Fig. 16 (a close-up of the first facade is depicted in Fig. 2). In total, the method was tested on 58 facades, which formed a representative set of downtown buildings. Of 16 low resolution aerial images 3 could not be handled fully automatically. Of the 42 ground-based images 10 were problematic, mostly because of extensive ornamentation, structural irregularities (rare cases like a mezzanine), low contrast, or large area mirror reflections (some skyscrapers act almost like perfect mirrors).

7 Discussion

Comparison to Previous Work: Most previous work assumes that multiple images of facades are available. However, this assumption is not valid in the applications we consider. Firstly, state-of-the-art aerial imagery does not have high enough resolution to allow 3D reconstruction of details. Secondly, we have to work from existing, single texture maps if the goal is to upgrade the quality of currently existing 3D city models. Nonetheless, it would be interesting to provide an option to allow for the incorporation of multiple images in our pipeline. Moreover, rather than taking a rather bottom-up approach as found with several automated techniques, where an early detection of facade elements tends to take prominence, the strategy expounded in this paper is strongly top-down. It exploits the far more generic repetitiveness of facade structures, not so much strong expectations about the appearance of their subparts.

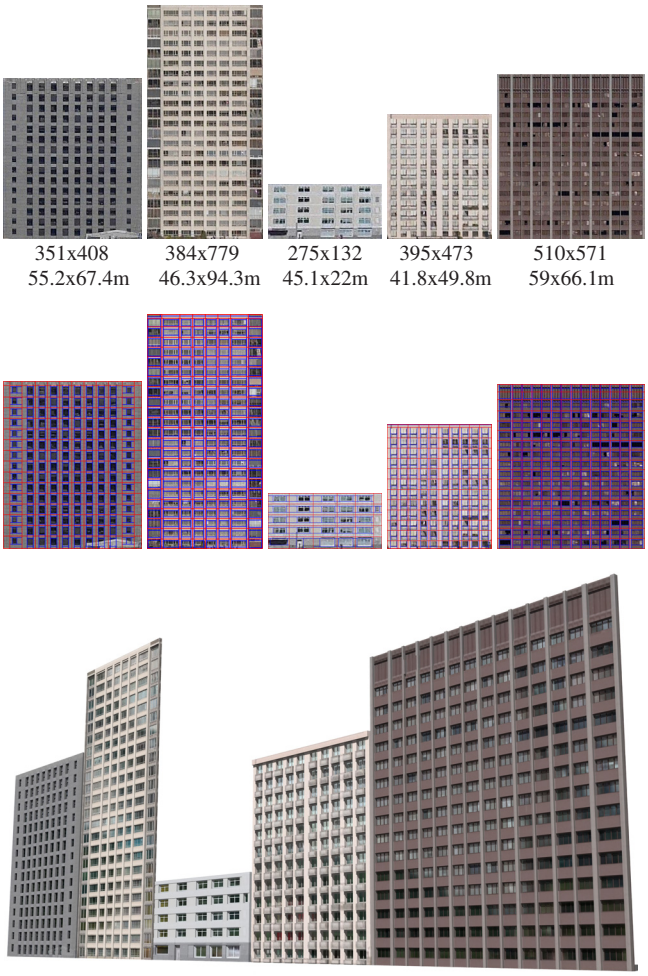

Figure 16: Airborne examples. Top: input facade images from oblique aerial imagery (with pixel resolutions and real-world dimensions). Middle: automatically extracted subdivision. Bottom: resulting 3D mesh with shadows and reflections applied.

Robustness and Limitations: A strength of our method is that it works well even for low resolution textures, a challenge that has not been tackled previously. Even though the approach is robust in general, there are smaller and larger errors depending on the quality of the input image and input image complexity. Fig. 17 illustrates typical failure cases. The main problems for the fully automatic processing are heavy image noise or small irregular elements (e.g. several irregularly placed air conditioners outside of the window boundaries). In these difficult cases MI might be unable to detect repetitions. Also ground floors of commercial buildings are often problematic for MI due to their non-repetitive structure. As a consequence, vertical symmetries may be left undetected (even if the floors above consist of the same tiles). Another problem is posed by windows with prominent, thick frames. If τ_{size} is smaller than the frame's thickness, it will be incorrectly detected as a split. The split of that cluster has then to be reversed in the user interface. Furthermore, our approach assumes an orthorectified image as input. Strongly protruding elements, such as balconies, violate this assumption and lead to incorrect tile subdivisions. To summarize, if our approach is applied on less repetitive architectural facades, we lose the structural support and run into the classic difficulties of edge detection i.e. the operations in section 3 will be less stable. Hence, we suggest using our technique only in urban areas with buildings of multiple stories.

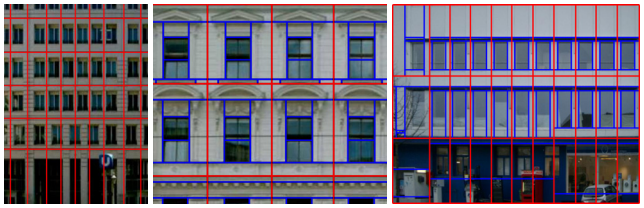

Figure 17: Failure cases. Left: The facade structure detection cannot handle asymmetric patterns like mezzanines or non-aligned tiles. Middle: The parameter τ_{size} has been adjusted to detect the window sills. As a consequence, τ_{size} is smaller than the thickness of the prominent window frames. Hence, the frame will be wrongly interpreted as a split and the user has to reverse the split manually. Right: Worst case scenario consisting of a blurry texture with low contrast, a chaotic ground floor disturbing the MI-based repetition detection, and image noise caused by vegetation (left).

Practical Modeling: The important aspect of our application is that we assist the user with the most challenging part of facade modeling: we can derive the exact dimensions, ratios and spacing of architectural elements automatically. In practice, commercial urban model providers use semi-automatic 3D modeling tools (to create crude building volumes and texture them). Hence, even if not working perfectly on all facades, our method is a useful add-on to accelerate such modeling tools. We implemented a graphical user interface where typical failure cases like a wrongly detected split type can be corrected quite easily. However, if the input imagery consists of downtown facades with repetitive tiles not much user interaction is required. Exceptional cases where a parameter tuning has to be done are range adjustments due to huge floors (see section 2.4), and adjustments of τ_{edge} due to low contrast images (see section 3.1).

Future Work: We believe that the following two cross-related problems are most promising for future contributions: First, the development of a robust and practical probabilistic framework that includes bottom-up and top-down propagation of knowledge as well as parametric templates (encoded with shape grammars). Second, a more systematic architectural approach to the extraction of shape grammar rules, e.g. mechanisms to automatically identify, apply and transform matching rule sets or styles from a given library.

Acknowledgments

The authors thank Simon Haegler and Andreas Ulmer for helping with the implementation, Filip Van den Borre for testing previous work, and the anonymous reviewers for their constructive comments on improving this paper. This project is supported in part by EC IST Network of Excellence EPOCH, EC IST Project CyberWalk, NSF-0643822 and NSF-0612269.

References

ALEGRE, F., AND DELLAERT, F. 2004. A probabilistic approach to the semantic interpretation of building facades. In *International Workshop on Vision Techniques Applied to the Rehabilitation of City Centres*.

BEKINS, D., AND ALIAGA, D. 2005. Build-by-number: Rearranging the real world to visualize novel architectural spaces. In *IEEE Visualization*.

BRENNER, C., AND RIPPERDA, N. 2006. Extraction of facades using rjMCMC and constraint equations. In *Photogrammetric Computer Vision*, 155–160.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of ACM SIGGRAPH 96*, ACM Press, H. Rushmeier, Ed., 11–20.

DICK, A., TORR, P., RUFFLE, S., AND CIPOLLA, R. 2001. Combining single view recognition and multiple view stereo for architectural scenes. In *ICCV*, IEEE Computer Society, Los Alamitos, CA, 268–274.

DICK, A. R., TORR, P. H. S., AND CIPOLLA, R. 2004. Modelling and interpretation of architecture from several images. *International Journal of Computer Vision 60*, 2, 111–134.

DUARTE, J. 2002. *Malagueira Grammar – towards a tool for customizing Alvaro Siza's mass houses at Malagueira*. PhD thesis, MIT School of Architecture and Planning.

FLEMMING, U. 1987. More than the sum of its parts: the grammar of queen anne houses. *Environment and Planning B 14*, 323–350.

FRÜH, C., AND ZAKHOR, A. 2001. 3D model generation for cities using aerial photographs and ground level laser scans. In *CVPR*, IEEE Computer Society, 31–38.

FRÜH, C., AND ZAKHOR, A. 2003. Constructing 3d city models by merging ground-based and airborne views. *Computer Graphics and Applications* (Nov.), 52–61.

HAN, F., AND ZHU, S.-C. 2005. Bottom-up/top-down image parsing by attribute graph grammar. In *ICCV*, IEEE Computer Society, Washington, DC, USA, 1778–1785.

HAVEMANN, S. 2005. *Generative Mesh Modeling*. PhD thesis, TU Braunschweig.

HU, J., YOU, S., AND NEUMANN, U. 2006. Integrating lidar, aerial image and ground images for complete urban building modeling. In *3DPVT*.

JEPSON, W., LIGGETT, R., AND FRIEDMAN, S. 1996. Virtual modeling of urban environments. *PRESENCE 5*, 1, 72–86.

JOJIC, N., FREY, B., AND KANNAN, A. 2003. Epitomic analysis of appearance and shape. In *ICCV*, IEEE Computer Society, 34–41.

KARNER, K., BAUER, J., KLAUS, A., LEBERL, F., AND GRABNER, M. 2001. Virtual habitat: Models of the urban outdoors. In *Third International Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Imaging*, 393–402.

KONING, H., AND EIZENBERG, J. 1981. The language of the prairie: Frank lloyd wrights prairie houses. *Environment and Planning B 8*, 295–323.

KULLBACK, S. 1959. *Information theory and statistics*. John Wiley and Sons., New York.

LEE, S. C., AND NEVATIA, R. 2003. Interactive 3D building modeling using a hierarchical representation. In *International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, IEEE Computer Society, 58–65.

LEE, S. C., AND NEVATIA, R. 2004. Extraction and integration of window in a 3D building model from ground view image. In *CVPR*, IEEE Computer Society, 113–120.

LEE, S. C., JUNG, S. K., AND NEVATIA, R. 2002. Automatic integration of facade textures into 3D building models with a projective geometry based line clustering. *Computer Graphics Forum 21*, 3 (Sept.), 511–519.

LEGAKIS, J., DORSEY, J., AND GORTLER, S. J. 2001. Feature-based cellular texturing for architectural models. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, E. Fiume, Ed., 309–316.

LIEBOWITZ, D., AND ZISSERMAN, A. 1998. Metric rectification for perspective images of planes. In *CVPR*, IEEE Computer Society, 482–488.

MARVIE, J.-E., PERRET, J., AND BOUATOUCH, K. 2005. The FL-system: a functional L-system for procedural geometric modeling. *The Visual Computer 21*, 5, 329–339.

MĚCH, R., AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *Proceedings of ACM SIGGRAPH 96*, ACM Press, H. Rushmeier, Ed., 397–410.

MÜLLER, P., WONKA, P., HAEGLER, S., ULMER, A., AND VAN GOOL, L. 2006. Procedural Modeling of Buildings. In *Proceedings of ACM SIGGRAPH 2006 / ACM Transactions on Graphics*, ACM Press, New York, NY, USA, vol. 25, 614–623.

PARISH, Y. I. H., AND MÜLLER, P. 2001. Procedural modeling of cities. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, E. Fiume, Ed., 301–308.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1991. *The Algorithmic Beauty of Plants*. Springer Verlag.

PRUSINKIEWICZ, P., JAMES, M., AND MĚCH, R. 1994. Synthetic topiary. In *Proceedings of ACM SIGGRAPH 94*, ACM Press, A. Glassner, Ed., 351–358.

PRUSINKIEWICZ, P., MÜNDERMANN, P., KARWOWSKI, R., AND LANE, B. 2001. The use of positional information in the modeling of plants. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, E. Fiume, Ed., 289–300.

RAMAMOORTHI, R., AND ARVO, J. 1999. Creating generative models from range images. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 195–204.

REALVIZ, 2007. Realviz ImageModeler V4.0 product information. http://www.realviz.com.

RIBARSKY, W., WASILEWSKI, T., AND FAUST, N. 2002. From urban terrain models to visible cities. *IEEE Computer Graphics & Applications 22*, 4, 231–238.

SCHINDLER, K., AND BAUER, J. 2003. A model-based method for building reconstruction. In *International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, IEEE Computer Society, 74–82.

STINY, G., AND MITCHELL, W. J. 1978. The palladian grammar. *Environment and Planning B 5*, 5–18.

STINY, G. 1975. *Pictorial and Formal Aspects of Shape and Shape Grammars*. Birkhauser Verlag, Basel.

STINY, G. 1982. Spatial relations and grammars. *Environment and Planning B 9*, 313–314.

TAKASE, Y., SHO, N., SONE, A., AND SHIMIYA, K. 2003. Automatic generation of 3d city models and related applications. In *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 113–120.

WANG, X., TOTARO, S., TAILLANDIER, F., HANSON, A., AND TELLER, S. 2002. Recovering facade texture and microstructure from real-world images. In *Proc. ISPRS Commission III Symposium on Photogrammetric Computer Vision*, 381–386.

WANG, L., YOU, S., AND NEUMANN, U. 2006. Large-scale urban modeling by combining ground level panoramic and aerial imagery. In *3DPVT*.

WELLS, W., VIOLA, P., ATSUMI, H., NAKAJIMA, S., AND KIKINIS, R., 1996. Multi-modal volume registration by maximization of mutual information.

WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. Instant architecture. *ACM Transactions on Graphics 22*, 3, 669–677.

A Image Rectification

To rectify facade images we implemented a variant of the algorithm presented by Liebowitz and Zisserman [1998]. First, we compute the gradient operator for each pixel in the image. The argument and magnitude of the resulting gradient vector indicate the orientation and reliability of a local edge respectively. Then, we apply the Hough linear transformation on these potential edges. Since lines are mapped into points in the Hough space, the reliable lines have strong corresponding points and a set of lines can be automatically extracted. Finally, two vanishing points are extracted by the RANSAC optimization based on these lines. The 2D projective transformation that transfers these two vanishing points into infinite points can finally be used to rectify the input image.