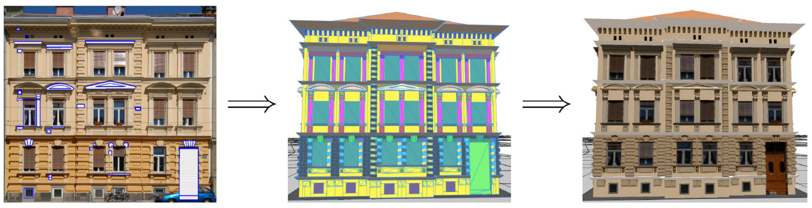
CITYFIT: HIGH-QUALITY URBAN RECONSTRUCTIONS BY FITTING SHAPE GRAMMARS TO IMAGES AND DERIVED TEXTURED POINT CLOUDS

Bernhard Hohmann, Ulrich Krispel, Sven Havemann and Dieter Fellner

Institute of Computer Graphics and Knowledge Visualization (CGV)
Graz University of Technology
Inffeldgasse 16c/II, 8010 Graz, Austria
{b.hohmann, u.krispel, s.havemann, d.fellner}@cgv.tugraz.at
http://www.cgv.tugraz.at/cityfit



KEY WORDS: LIDAR, point cloud, urban, reconstruction, shape grammar, fitting

ABSTRACT:

Many approaches for automatic 3D city reconstruction exist, but they are still missing an important feature: detailed facades. The goal of the CityFit project is to reconstruct the facades of 80% of the buildings in the city of Graz fully automatically. The challenge is to establish a complete workflow, ranging from acquisition of images and LIDAR data over 2D/3D feature detection and recognition to the generation of lean polygonal facade models. The desired detail level is to represent all significant facade elements larger than 50 cm by explicit polygonal geometry. All geometry shall also carry descriptive annotations (semantic enrichment). This paper presents an outline of the workflow, important design decisions, and the current state of the project. First results were obtained by case studies of facade analysis followed by manual reconstruction. This gave important hints how to structure grammars for automatic reconstruction.

1 INTRODUCTION

In recent years GIS services such as Microsoft's Virtual Earth or Google Earth have become incredibly popular. They show a strong trend towards 3D as both services begin to offer height fields and buildings in addition to aerial images. However, 3D models are currently provided only for a few major metropolises, and not even for those the whole city area is covered. High-quality reconstruction apparently still involves some manual interaction ("human in the loop"). A scalable solution can only be achieved with a fully automated workflow. This has become operational in 2008. However, the state-of-the-art for automatically generated models from aerial survey data are basically just extruded ground polygons with roofs. Facades are typically textured with aerial images and exhibit artifacts due to the oblique viewing angle. Detailed facade models, which are indispensable for road-side walkthroughs and for car navigation systems, are still a missing feature.

CityFit is a 3-year project that started in 2008. It is a collaboration of CGV with the Institute for Computer Graphics and Vision (ICG), also TU Graz, and Microsoft Photogrammetry Graz. The ambitious goal is to establish a fully automatic workflow for the generation of detailed facade models with explicit geometry for all distinctive facade features. The system will be evaluated on the city of Graz, where many complex building styles coexist. Especially the highly decorated neo-classical facades in the downtown area are challenging (Fig.2). The goal is to reconstruct at least 80% of the facades automatically while the remaining 20% are expected to exhibit untypical or singular style features that are difficult to recognize automatically (solitaire buildings).

The input data for CityFit consist of highly redundant road side photographs and LIDAR scans acquired by Microsoft Photogrammetry. The LIDAR 3D point cloud is not sufficient for direct facade reconstruction. It allows, however, to derive the main orientation and the rough structure of the facade very reliably. For examining the preprocessing results a point cloud viewer was developed (section 3). The LIDAR data is used together with information extracted from the road side photographs. The combination of 2D and 3D allows to obtain true orthophotos by estimating the principal facade plane. An image-based segmentation yields the facades of individual buildings, on which sophisticated feature detection is performed to determine architectural elements. The image processing tasks are performed by ICG (see, e.g., (Fraundorfer et al., 2006)). However, the recognition should of course produce information that is of architectural significance. In or-

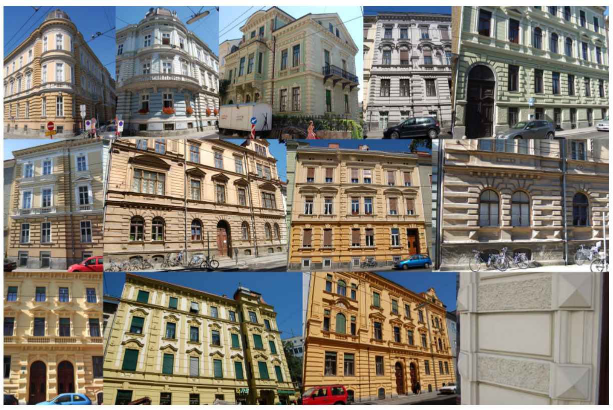


Figure 2: Challenging facades in the downtown of Graz, of the *Gründerzeit* style (1850-1900) with repetitive ornaments

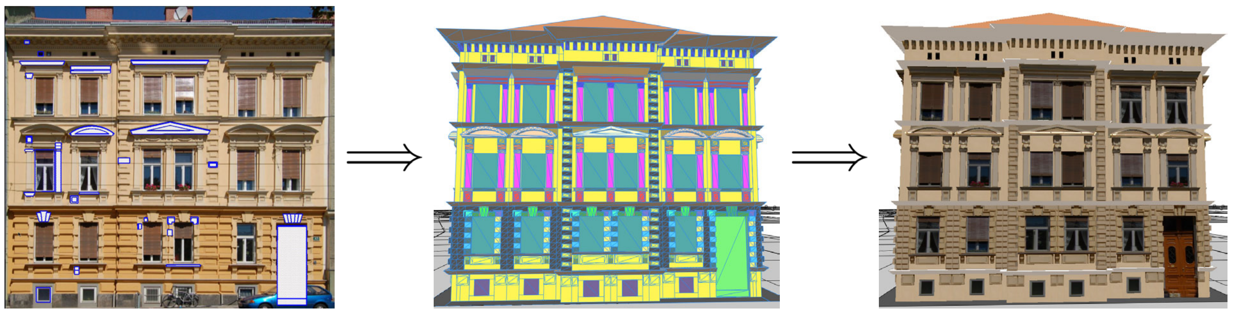

(a) (b) (c)

Figure 1: (a) Orthorectified photo of a neo-classical facade in Graz with terminal symbols, (b) Shape grammar representation in the CityEngine (Procedural Inc., 2008) with CGA Shape (Müller et al., 2006), (c) Shape grammar representation with applied textures.

der to define which recognition results are required for facade reconstruction, several case studies were carried out to analyze representative facades. The result is a set of labels for the main architectural elements (section 4, see also Fig. 1(b)).

The reconstruction of a complete city yields huge amounts of data. Consequently a compact data representation is essential. The goal is to produce lean polygonal models, which nevertheless exhibit enough geometric detail. Ideally the geometric representation is *scalable*, allowing coarse-to-fine shape refinement. Shape grammars can provide this and are, furthermore, very effective in exploiting the regular structures in facades. For each individual facade a shape grammar needs to be generated that describes its particular structure. Each grammar symbol is annotated with the labels from the recognition (semantic enrichtment) and corresponds to one or more geometric elements. The shape grammar system of CityFit is based on the main concepts of CGA Shape (Müller et al., 2006), but it uses the Generative Modeling Language (GML) (Havemann, 2005) as grammar description language. GML is a very simple, stack-based programming language that was developed for automatic code generation. So grammars in GML syntax are easier to generate than, e.g., grammars in CGA-Shape, the grammar description language of the CityEngine from (Procedural Inc., 2008). Existing shape grammars are based on simple boxes ('scopes'). To capture the more complex facades in Graz the grammar concept will be extended to a generalization of boxes, i.e., convex polyhedra (section 5).

The project will gradually shift from manual processing to semi-automatic and finally to automatic processing. Our paper describes the current state of the project and sketches the next steps in order to achieve higher degrees of automatization.

2 RELATED WORK

Automated 3D city reconstruction is a very active field of research. It was boosted by the public demand for complete 3D city models, triggered by the activities of Google and Microsoft. The automatic generation of extruded ground polygons from aerial photographs with roof landscapes of high accuracy is considered a solved task. The workflow of a system that is already operational is described in (Zebedin et al., 2006). Still missing, however, are detailed facade models. While planar textured facades are sufficient for the most part of many American cities, they are definitely insufficient for European cities which typically comprise buildings from many different periods.

Shape grammars have only recently become a standard tool for urban reconstruction. They were originally introduced by (Stiny and Gips, 1972). For a long time they were used in architecture only for theoretical research purposes but not as practical design tools. A new boom in the field was triggered by the introduction of split grammars (Wonka et al., 2003) which eventually matured to a commercial software, the CityEngine (Procedural Inc., 2008). This system is primarily designed for the automatic generation of large scale city models. It provides a scripting interface suitable for advanced users who are able to code a grammar using the CGA-Shape language presented in (Müller et al., 2006). A very interesting GUI-based approach for creating shape interactively was presented in (Lipp et al., 2008). This is an important improvement over manual editing of grammar rules, which is a tedious and rather abstract task. Another promising approach for grammar-based shape modeling was recently presented by (Finkenzeller, 2008). However, it also requires manual interaction and is not targeted at automatic facade generation.

An interesting approach for automatic facade reconstruction with CGA Shape using orthorectified photos is presented in (Müller et al., 2007). It introduces the concept of an *irreducible facade* which is identified using *mutual information* (MI), a stochastic method. It describes a facade using only six different symbols which works very well with highly regular, repetitive facades. In Graz, however, we have identified in a typical neo-classical facade 30 different symbols that were used repetitively (see section 4 and Fig. 1(b)). A further enhancement presented in (Van Gool et al., 2007) allows the system to deal also with strong perspective distortion. An important topic for automatic facade reconstruction is window detection. They typically appear in a sequential fashion which facilitates the detection as explained, e.g., in (Wenzel and Förstner, 2008). Another interesting approach for window detection using *Implicit Shape Models* (ISM) is (Reznik and Mayer, 2007). Apparently it is limited to windows, however.

A method which makes use of additional 3D information (terrestrial LIDAR) is presented in (Ripperda, 2008b). Several solutions for detecting windows and structures in facades are proposed, splits in the image are detected using auto-correlation. The reconstruction is based on the *reversible jump Markov Chain Monte Carlo* (rjMCMC) method and shape grammars. A corresponding facade analysis can be found in (Ripperda, 2008a). However, only simple structures are detected from single photos.

Impressive results for automatic online 3D city reconstruction were published in (Cornelis et al., 2006). It uses a stereo camera rig and perform object detection to filter out cars. However, the focus is on real-time reconstruction producing only coarse models. Finally, (Dörschlag et al., 2007) use exclusively 3D data to generate building models. Their method employs the minimum description length (MDL) principle that allows model selection with automatic complexity control. This mechanism could be useful for fitting parametric models to detected objects, but the paper provides no experimental results.

3 PREPROCESSING AND DETECTION

The raw data consist of streams of geo-registered street-side photographs, taken from a car. Currently multiple cameras looking in different directions and having an overlap between consecutive capture events are used. In addition to the images, a LIDAR laser range scanner is used to acquire a coarse 3D point cloud. All measurements are time-stamped to allow a precise registration of the point cloud to the images.

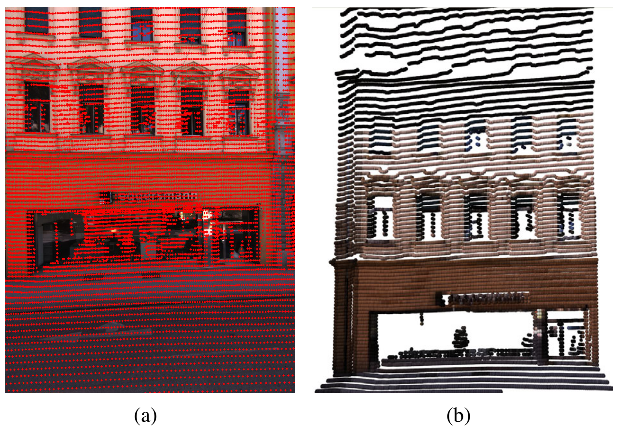


Figure 3: (a) LIDAR data projected into roadside photos, (b) Textured point cloud.

We have implemented a fast point cloud renderer to visualize the LIDAR data. To reduce loading time, the LIDAR data is converted from a text based representation into a binary format. In order to efficiently search within the data, it needs to be organized. Due to the spatial distribution (measurements on the ground plane) a regular two dimensional grid is sufficient. The grid partitions the space in quadratic columns with infinite height (z-axis) and a side length of 4×4 meters. In an offline preprocessing step the LIDAR points are colored and sorted into this grid. The color of a point is determined by selecting a suitable photograph where the point is visible (using the timestamp) and back-projecting the point into the image, cf. Fig. 3(a) and 3(b). The cells of the grid are organized in a quadtree structure, so that empty cells do not have to be stored. The point coordinates are stored in one big array, grouped by the grid cells. Inside one cell, the points are sorted by their z-component. The leaves of the quadtree store an offset into this array.

A fundamental design decision was to store all geometric data in 16.16 fixed-point integer representation. This yields a uniform resolution of about $1/65$ mm over 65 km. In contrast to using floating-point coordinates, the geometric accuracy does not depend on the distance from the origin. The mantissa of a 32 bit IEEE float has only 23 bit, which means that our accuracy of 2^{-16} meters can only be achieved in a distance of less than $2^{23-16} = 2^7 = 128$ meters from the origin. In a distance of 4.2 km from the origin, the float grid has a sparse spacing of only about 0.5 mm. Our algorithms and computations may use higher precision than 32 bit internally, but in the end the results are fit into the 16.16 grid. This allows an accurate estimation of the precision that we require for numerical algorithms. Another advantage is that we can test points for equality, i.e., the (x, y, z) coordinates are actually a unique point ID with 96 bit. Alternatively, we can use a distance-independent epsilon radius (e.g., $2^{-13} \approx 0.12$ mm) for merging close points reliably.

A special feature of the point cloud viewer is that it can visualize the points as spheres (e.g., with a radius of $10cm$), which allows us to visually relate reconstructed geometry to the input data. To achieve interactive frame rates the spheres are rendered as depth sprites (Gortler et al., 1998), which leave correct depth information in the frame buffer (cf. Fig. 4(d)). Using fast quadtree traversal techniques adapted from (Frisken and Perry, 2002) and (Amanatides and Woo, 1987) we can perform a ray-intersection with the spheres in the point cloud, which allows interactive navigation and point selection on our test dataset consisting of 5.9 million points (cf. Fig. 7).

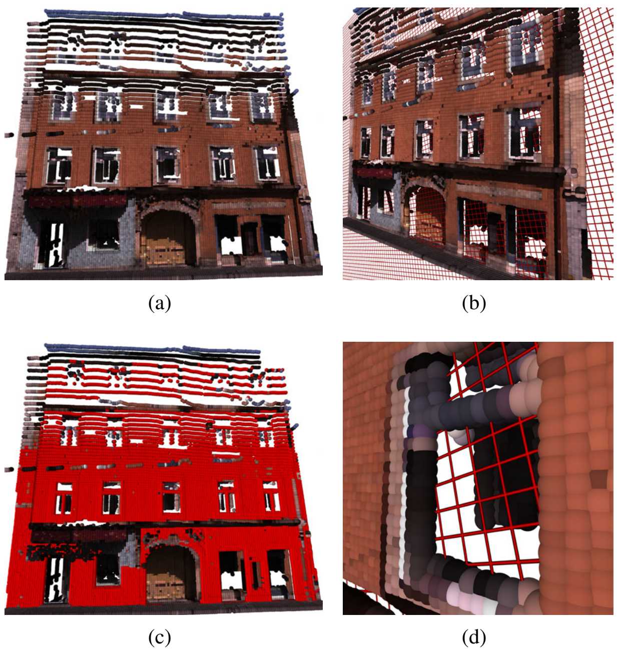


Figure 4: Retrieving information from 3D data: using RANSAC, plane structures can be obtained (a) facade point cloud (c) points voting for a plane (b) combined rendering of point cloud and the principal plane (d) points are visualized as spheres.

Currently facades are segmented manually (cf. Fig. 5), an automatic facade segmentation is under development. Image based feature detection is done by a *partial shape fragment matcher* which is used to detect facade elements (e.g. windows, arches and even smaller decorative elements) cf. Fig. 6. The results will be published in a paper which is currently under review.

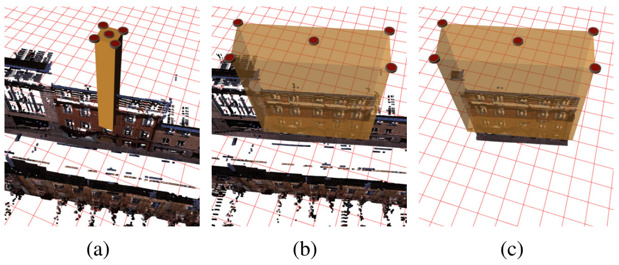


Figure 5: Manual point cloud segmentation. The segmentation is performed by selecting a starting point (a) and dragging the red knobs to define the segment boundaries (b). The points outside the yellow column are discarded (c).

From feature detection to grammar synthesis. The next step in the project is to derive a grammar from the detected features. The system will proceed as described in the following. The image based feature detection results can be seen as a segmentation for grammar symbols on the images. It proceeds in a bottom-up fashion by identifying individual elements. This segmentation is not sufficient since it misses depth information (e.g. the depth of a balcony). This information can be retrieved from the point cloud by using it to compute a depth map for the orthofotos. Performing RANSAC plane fitting on the point cloud yields a segmentation for the depth map: RANSAC proceeds by removing the largest plane first (principal facade plane, cf. Fig. 4). Then smaller and smaller planes are identified (note that these planes do not have to be parallel to the principal plane). The combination of both segmentations is used to generate a grammar representation for a facade: The same z-value is a strong clue for elements to carry the same grammar symbol. Then the resulting grammar is matched against a set of grammar templates. These templates will be obtained by the facade analysis described in section 4. Each template represents a specific group of similarly structured facades. Once a grammar template has been chosen as a hypothesis it can also be used to guide the feature detection. In that sense, a grammar template can be understood as a hierarchically structured shape prior.

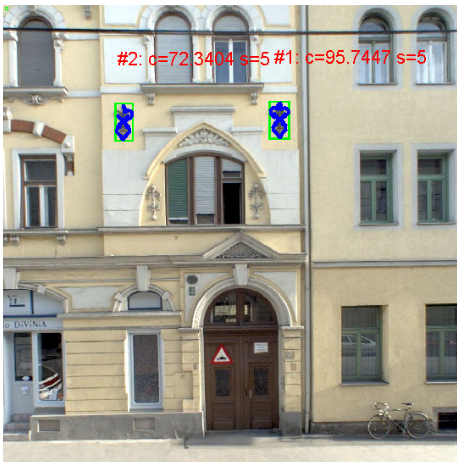


Figure 6: Appearance based detection of facade elements using a partial shape fragment matching method, detected shapes are shown in blue.

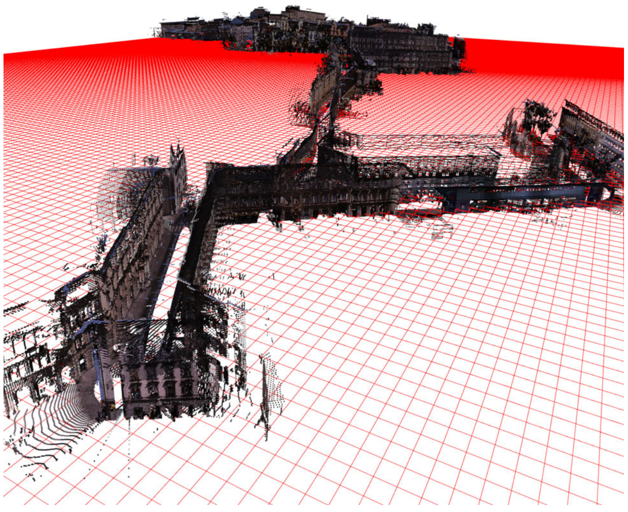


Figure 7: Textured point cloud of a test drive in Graz

4 FACADE ANALYSIS

Architectural knowledge is important for identifying the significant building blocks of the facades. The long history of the city of Graz requires our algorithms to cope with a great variety of different building styles - essentially the whole development of European architecture has left its traces in the city. Since Graz has grown very much in the 19th century, especially the highly decorated neo-classical ("Gründerzeit") facades pose particular challenges. We have therefore started by manually identifying the building blocks of some selected facades. This resulted, for example, in 30 elements that were repeatedly used in the facade in Fig. 1(a). This particular facade was reconstructed as a case study using a state-of-the-art software package, the CityEngine (Procedural Inc., 2008), cf. Fig. 1(b) and 1(c). A detailed view of the resalit of the reconstructed facade is shown in Fig. 10.

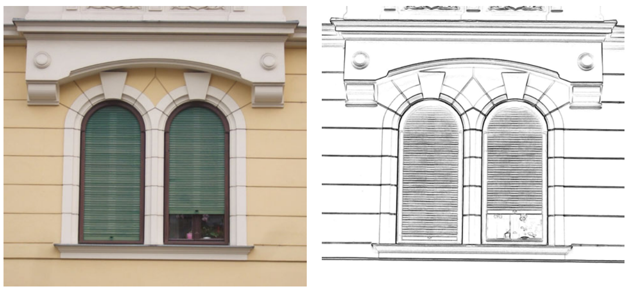


Figure 8: Structure of an architectonic element.

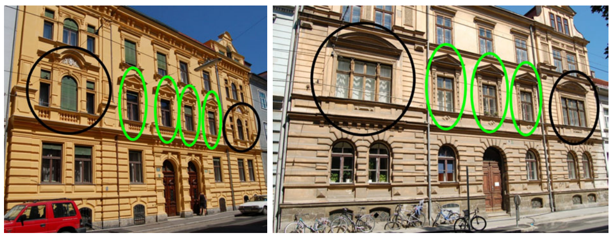


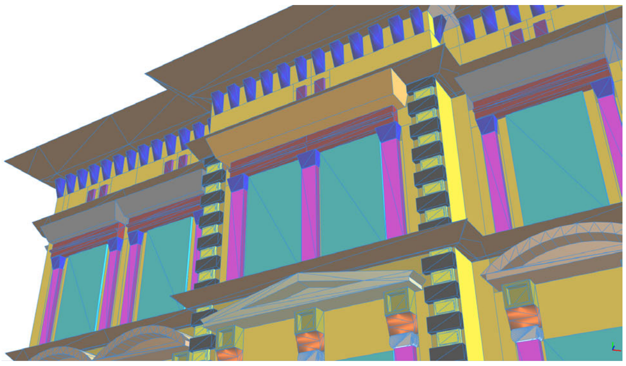
Figure 9: Different facades with a similar structure

During the manual reconstruction of the facade in Fig. 1(a), several problems of shape grammars have been identified. In particular, there are four common problems which are depicted in Fig. 11 and described in more detail in the following paragraphs.

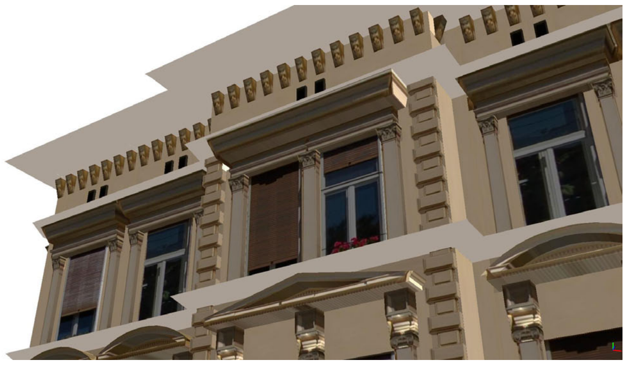
Overlapping Elements: When a subelement of a scopes extends into one or more adjacent scopes, additional splits have to be performed, cf. Fig. 11(a). This happens very often, for example, window sills are usually slightly wider than the window. Thus, not just the scope which contains the window has to be split, but also the adjacent scopes.

Split Order: This is probably the most obvious problem when reconstructing facades. There is always the question which way to split first, horizontally or vertically, i.e., floors or window axis, see Fig. 11(c). In most approaches presented in section 2, facades are first split horizontally into floors. However, facades are usually structured in a vertically symmetrical manner. This suggests that it is better to first split the facade vertically along symmetry axis or into superordinate structures. Resalits or oriels are such superordinate structures, as they are stepping out of the facade (which can be easily detected by exploiting LIDAR data). The problem is getting even more serious, when main ledges are intersecting resalits. In this case it would be desired to represent the ledge as well as the resalit with one element. In Fig. 10, the resalit was ranked prior to the main ledge. Finally, it is desired to put repetitive structures into one scope to be able to distribute their elements evenly over the scope. For example, the modillions (blue elements supporting the cornice) in Fig. 10.

Unknown Neighbors: As shape grammars are structured in a hierarchical manner, they can be interpreted as tree, cf. Fig. 11(b). Thus, adjacent elements in the facade do not have information



(a)



(b)

Figure 10: (a) Detailed view of the resalit from the reconstruction in Fig. 1, (b) With textured terminals.

about each other. This problem is strongly related to the split order problem. When a ledge has to be divided into several parts, due to a prior-ranked resalit for example, information such as the depth of these parts, has to be propagated from the top of the tree, down to every terminal node (cf. Fig. 10).

Exceptional Elements: Many facades are highly regular, e.g. dwellings from the sixties. Thus, they can be represented by very simple rules, i.e, one rule splitting the facade into floors, and one rule splitting the floors into window tiles. If there are, however, exceptional elements in these regular structures, such as blind windows, the grammar has to be changed in every superordinate level, cf. Fig. 11(d). In (Lipp et al., 2008), instance locators are introduced to tackle this problem.

Another case study was performed on facades in the Jakoministrasse in Graz. A very common setup is a center structure, which sometimes but not necessarily is a resalit, with adjacent left and right structures. Very often left and right structures are symmetric above the first floor. Usually the first floor is different from the upper floors. First, because in a typical city the first floor often accommodates shops, window displays or public services. Second, because throughout history it was always common to put emphasis on the first floor in any kind of building. Therefore, the facade of the first floor is often higher than the other floors and sometimes also includes a lower mezzanine. This emphasis on the first floor also has a practical reason, to give passages to the court sufficient clearance (Breitling, 1982). The upper part of the facade is usually split into main ledges and floors. Similar findings were also reported in a quantitative facade analysis in (Ripperda, 2008a).

Further findings, resulting from an examination of architectural literature (Breitling, 1982), can be useful for an automatic determination of the building period or style of facades. First, the

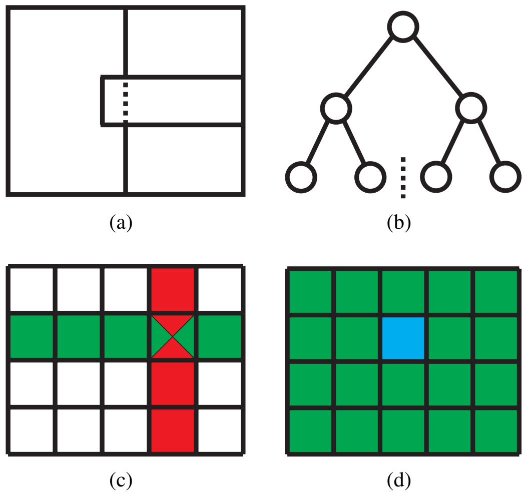


Figure 11: Common problems of shape grammars: (a) An element ranging over a higher order split line. (b) In the hierarchical structure of a grammar, neighboring grandchildren do not know of each other. (c) Very often vertical and horizontal structures intersect each other, making it difficult to decide which way to split first. (d) If one element differs in a grid of similar elements, this can effect several rules in the grammar.

relation of hole (i.e., windows) to wall. In the sixties steel skeleton frame buildings became popular in Graz. This made it possible to spend a higher proportion of the facade for windows, as the steel frame instead of the walls, is supporting the building. Thus, a high proportion of windows is evidence for a modern building. Second, the window division is another feature which changed over time. Old windows are usually divided into more and smaller window patches, typically six, whereas new windows are sometimes not divided at all. Finding out about such a detail might also be an indication for a specific building period. However, sometimes old windows in historic facades are replaced by new modern windows, which not only damage the appearance of the facade, but also could be missleading in such an automated determination of the building period.

One of the promising avenues for further research in our project is to use the grammar as *structural shape prior*, as pointed out in section 3. As mentioned before facades often consist of a center structure and symmetric left and right parts, which would be a typical template to be found in Graz, see e.g. Fig. 9. Another frequent type is a completely regular sixties style dwelling. In this case every window can be represented by the same symbol, with the door as the only different element in the facade. A Graz specific set of templates needs to be developed.

5 SEMANTIC FACADE REPRESENTATION

A shape grammar is conceptually simple, as it consists of terminal and non-terminal symbols and a set of replacement rules. Non-terminals are basically boxes with a name tag. Terminal symbols consist of 3D geometry that, when replacing a non-terminal box, is made to fit into this box. Replacement rules determine how a non-terminal box with a given name tag is split into smaller boxes carrying other tags. Therefore, shape grammars are also called split grammars. There are two basic split rules: split a box along an axis into k equally wide smaller boxes (repeat), or into as many equal boxes as possible larger than s_{min} (subdivide). Sizes can also be relative so that, e.g., subdivide(X,1r,2r,2r,1r) splits a box along the x-axis into four boxes, the inner ones twice as wide as the outer ones.

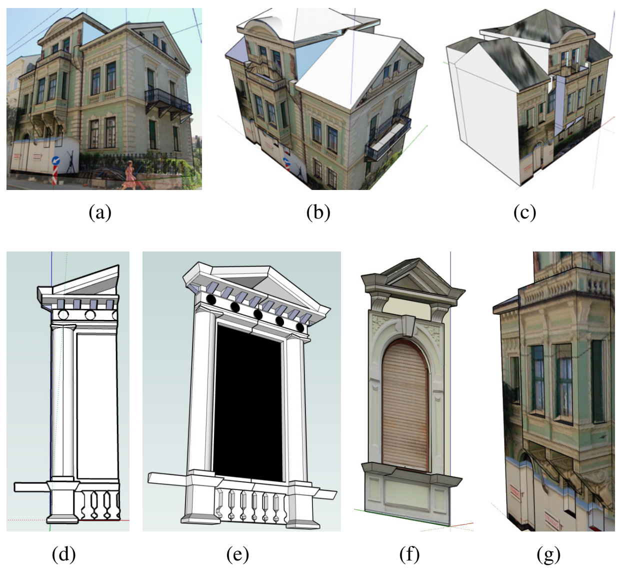


Figure 12: Direct photo-based approaches (here Google SketchUp) are great for manual reconstruction. The experience from our experiments was that the potential for automatization is only small, as it is difficult to determine the appropriate edges in the image to snap to.

CityFit uses the same concept, but with two variations:

- the grammar description language is GML, and
- it uses convex polyhedra instead of rectangular boxes.

Shape grammars in GML. The Generative Modeling Language (GML) (Havemann, 2005) is a simple stack-based programming language with a syntax very similar to Adobe PostScript. A GML program is merely a stream of individual tokens that are executed one after another. A token is either a piece of data, in which case it is put on a stack, or it is an operator. This can either be a built-in operator or a function, which is again another stream of tokens. So GML functions are in fact just *executable arrays*). When a built-in operator is executed, it takes its input values from the stack, processes them, and puts the output again on the stack. The processing can have side effects, i.e., manipulate internal data structures, for example create a 3D box.

GML was developed as general, extensible shape description language. Its first shape representation were *combined B-reps* (cB-rep), a conventional half-edge data structure with a boolean flag per edge to switch between *sharp* and *smooth* edges. Faces with a smooth edge are rendered as Catmull/Clark subdivision surfaces. GML provides mesh operators which can be used for creating meshes procedurally, e.g., different window types and even whole buildings (Fig. 17) with a single connected control mesh. The drawback of meshes is that they lack a hierarchical structure, which is predominant in facades.

Since grammar interpreters use a stack internally to keep track of the replacement rules being applied, GML should also be suitable for describing grammars. In fact, executing a GML function can simply be understood as replacing one symbol (token) by a sequence of symbols (tokens): In the example in Fig. 13, the replacement sequence is:

```
C  →  E D E
   →  G F G  F B F  G F G
   →  A A A  A B A  A A A
      A B A    B    A B A
      A A A  A B A  A A A
```

This is turned into a shape grammar by providing each replacement with a direction in space, and an extent. The split operator

```
ShapeGrammar.Tools.init
/A  { terminal−box } def
/B  { terminal−void } def
/C  { [ 0.1 −1 0.1 ] /X split  E D E  } def
/D  { [ 0.1 −1 0.1 ] /Y split  F B F  } def
/E  { [ 0.1 −1 0.1 ] /Y split  G F G } def
/F  { [ 0.1 −1 0.1 ] /Z split  A B A  } def
/G  { [ 0.1 −1 0.1 ] /Z split  A A A  } def
scope (0,0,−2) move (2,1,1) scale C
finish
```

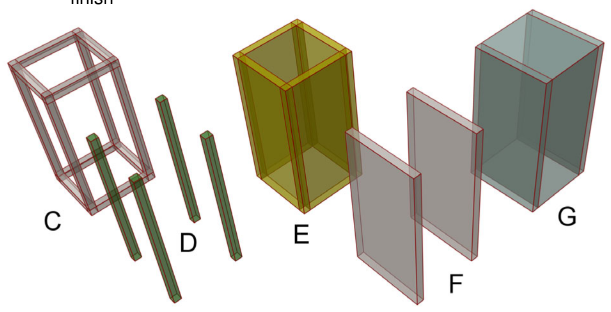


Figure 13: Shape grammars in GML. The main operation is the recursive box split operator. It expects on the stack a box ("scope"), an array of absolute (> 0) and relative (< 0) distances, and a split direction. The sub-scopes are again put on the stack to be processed by further splits, or consumed by terminal symbols. Grammar rules naturally correspond to GML functions.

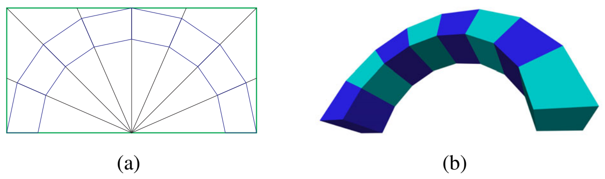


Figure 14: Creating an arch out of a box: (a) The arch (blue) is obtained by splitting the upper grammar level (green) into convex parts. (b) 3D model obtained by evaluating the parametric description.

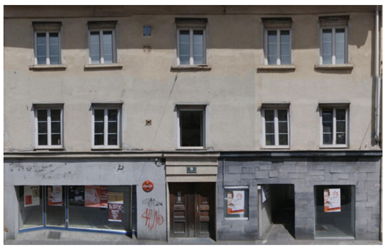
(rule) therefore not only expects a symbol (scope token) but also a direction and an array of extents. By elaborating this principle the GML code in Fig. 16 was obtained that produces the facade shown in Fig. 15.

In this example, a facade in Jakoministrasse in Graz, the first floor is no higher than the other floors, but it still differs significantly as it accommodates shops. The facade consists of a center part and symmetrical left and right parts. They are represented by two different rules RightFacade and LeftFacade because of the strip of wall at the left and right edges of the facade. The only difference in the code is indicated by the blue frame. This code fragment also shows the parametrization of the facades: The number of windows in a floor can be changed simply by changing a number in one of the code pieces with a green frame. – A more elaborate example, our university building, is shown in Fig. 18. This demonstrates that the identical approach is also feasible for representing the interior of buildings, not just facades.

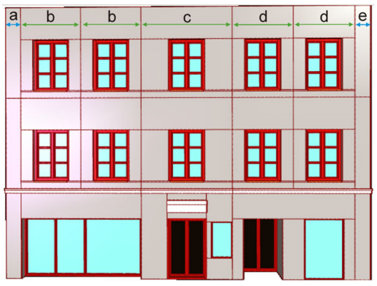
Convex polyhedra. It is difficult and error-prone to model non-rectangular geometry (e.g. pediments) with boxes, as our experiments with the CityEngine have shown. Architectural features such as an arch can be much more easily approximated with convex polyhedra (Fig. 14). Note that this corresponds to architectural practice, as walls are built from stones, most of which are actually convex polyhedra. A convex polyhedron is the intersection of half-spaces. A half-space is defined by a directed infinite plane that divides three-space into "interior" and "exterior".

```
1.2 !window-width
16  !wall-colour
/RightFacade {
   [2.85 0.15 3.0 3.0] /Y split-r
     FirstFloorRight
     0.15 3 Ledge
d    2 :wall-colour  0 0.5 Floor { :window-width :wall-colour -0.625 -0.125 0.04 WindowTileSill Window WindowSill } repeat
     2 :wall-colour  0 0.5 Floor { :window-width :wall-colour -0.625 -0.125 0.04 WindowTileSill Window WindowSill } repeat
} def                  e
/CenterFacade {
   [2.85 0.15 3.0 3.0] /Y split-r
     FirstFloorCenter
     0.15 3 Ledge
c    1 :wall-colour  0 0 Floor { :window-width :wall-colour -0.625 -0.125 0.04 WindowTileSill Window WindowSill } repeat
     1 :wall-colour  0 0 Floor { :window-width :wall-colour -0.625 -0.125 0.04 WindowTileSill Window WindowSill } repeat
  } def
/LeftFacade {
   [2.85 0.15 3.0 3.0] /Y split-r
     FirstFloorLeft
     0.15 3 Ledge
b    2 :wall-colour  0.5 0 Floor { :window-width :wall-colour -0.625 -0.125 0.04 WindowTileSill Window WindowSill } repeat
     2 :wall-colour  0.5 0 Floor { :window-width :wall-colour -0.625 -0.125 0.04 WindowTileSill Window WindowSill } repeat
} def                a
```

Figure 16: GML shape grammar code for the facade in Fig. 15 with labels for the different dimensions shown in the reconstruction.



(a)



(b)

Figure 15: (a) A facade in Jakoministrasse in Graz. The photo is distorted because it is obtained from multiple road side photos. (b) Facade reconstruction using a GML shape grammar. The labels correspond to the code in Fig. 16

Given a set of half-spaces, the points on the interior of *all* planes form a convex polyhedron. Intuitively this can be understood as chopping away with a knife parts from, e.g., a piece of cheese.

In the current phase of the project we are extending GML. The basis is one operator to create a directed plane from three points (in 16.16 coordinates), and another that creates a convex polyhedron from an array of planes:

```
p1:P3 p2:P3 p3:P3  cp−plane  →   e:Plane
[ Plane ]          cp−shell  →   s:CPolyhedron
```

The challenge is to define a set of operators for convex polyhedra that are also convenient for describing architectural elements.

6 CONCLUSION AND FUTURE WORK

This paper has described the main goals of the CityFit project, as well as the approaches that were chosen in order to reach these goals. To summarize, the key features and design decisions are:

- All raw data are transformed to 16.16 fix-point coordinates to guarantee uniformly a high resolution of about $1/65$ mm
- The workflow is based on 2D image streams and 3D LIDAR data that are registered
- Architectural knowledge is incorporated by using shape grammar templates obtained by facade analysis and classification
- The shape grammar for a facade is synthesized by combining three sources of information:
  - Feature detection on the orthophoto
  - Segmentation of plane regions on the depth images
  - Shape grammar template as structured shape prior
- Shape grammars and grammar templates are encoded in GML
- The shape grammar operates on convex polyhedra rather than rectangular boxes.

While these foundations are fixed and defined now, we are currently entering the most challenging phase of the project. During 2009 we will have to prove that the facade classification scheme is indeed sufficiently general to encode 80% of the facades in Graz, and that the grammar templates produced this way are suitable for guiding the detection.

The immediate next step is to create a facade classification tool. It will allow to browse through all the facades that have been acquired so far, to select a facade, and then to browse through the library of available facade classes. The shape grammar template of the selected class is then matched to the raw data of the selected facade. If successful, the result is a shape grammar that produces a faithful but lean polygonal reconstruction of the facade. In case no appropriate facade class can be found, the tool shall allow defining a new one, i.e. to define a new shape grammar template. A big challenge will be to define a suitable format for shape grammar templates. So far we have only gained experience with creating grammars, but not with creating grammar templates, i.e., meta-grammars, or grammar generators. For this next step we will have to rely on the fact that grammar code in GML can be generated automatically.

ACKNOWLEDGEMENTS

CityFit is funded by the FIT-IT program of the Austrian Research Promotion Agency (FFG) under Grant No. 815971/14472, and the Doctoral Program *Confluence of Vision and Graphics* sponsored by the Austrian Science Fund FWF. The authors would also like to thank their project partners Microsoft and ICG, especially Hayko Riemenschneider for providing us with Fig. 6.

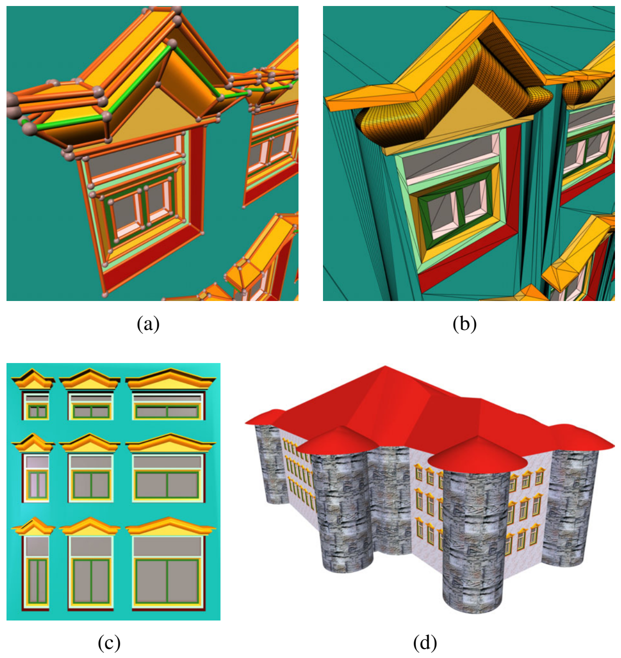


Figure 17: Combined B-rep can represent interesting freeform shapes. Unfortunately, half-edges have proven to be too fragile as references when building more complex facades. Therefore, shape grammars were chosen.

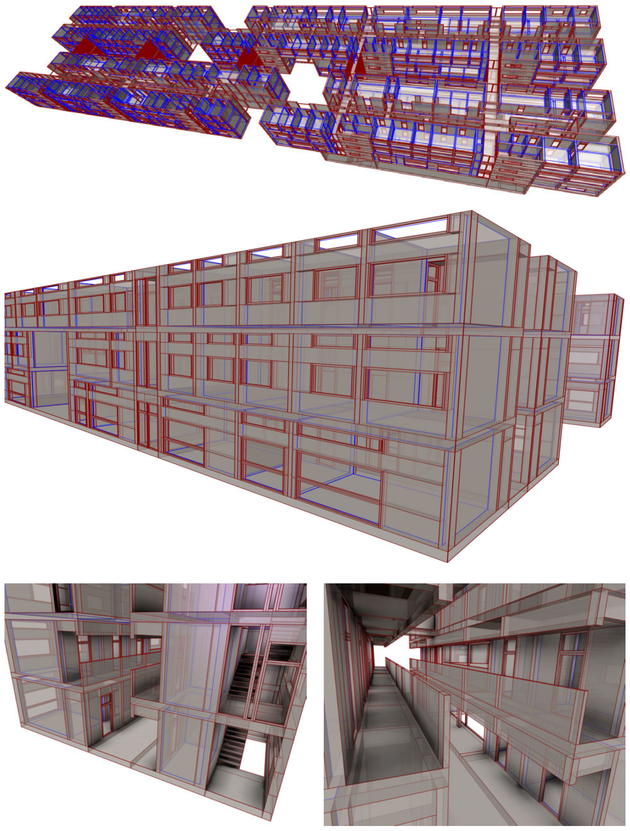


Figure 18: Reconstruction of our university building with the shape grammar based on GML

REFERENCES

Amanatides, J. and Woo, A., 1987. A fast voxel traversal algorithm for ray tracing. In: In Eurographics 87, pp. 3–10. 3

Breitling, P., 1982. In der Altstadt Leben. Leopold Stocker Verlag, Graz, Austria. 5

Cornelis, N., Leibe, B., Cornelis, K. and Gool, L. V., 2006. 3d city modeling using cognitive loops. In: 3DPVT '06: Proceedings of the Third Intern. Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06), IEEE Computer Society, Washington, DC, USA, pp. 9–16. 2

Dörschlag, D., Gröger, G. and Plümer, L., 2007. Sementically enhanced prototypes for building reconstruction. In: PIA07, pp. 111 – 116. 2

Finkenzeller, D., 2008. Detailed building facades. IEEE Computer Graphics and Applications 28(3), pp. 58–66. 2

Fraundorfer, F., Schindler, K. and Bischof, H., 2006. Piecewise planar scene reconstruction from sparse correspondence. Image Vision Computing 24(4), pp. 395–406. 1

Frisken, S. F. and Perry, R. N., 2002. Simple and efficient traversal methods for quadtrees and octrees. Journal of Graphics Tools 7, pp. 2002. 3

Gortler, S. J., Cohen, M. F. and wei He, L., 1998. Layered depth images. In: In Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 231–242. 3

Havemann, S., 2005. Generative Mesh Modeling. PhD thesis, Technical University Braunschweig. 2, 6

Lipp, M., Wonka, P. and Wimmer, M., 2008. Interactive visual editing of grammars for procedural architecture. In: Proc. ACM SIGGRAPH 2008, pp. 102:1–10. 2, 5

Müller, P., Wonka, P., Haegler, S., Ulmer, A. and Gool, L. V., 2006. Procedural modeling of buildings. In: ACM SIGGRAPH, Vol. 25. 2

Müller, P., Zeng, G., Wonka, P. and Gool, L. V., 2007. Image-based procedural modeling of facades. In: ACM SIGGRAPH, Vol. 26. 2

Procedural Inc., 2008. CityEngine. 2, 4

Reznik, S. and Mayer, H., 2007. Implicit shape models, model selection, and plane sweeping for 3d facade interpretation. In: PIA07, pp. 173–178. 2

Ripperda, N., 2008a. Determination of facade attributes for facade reconstruction. In: Intern. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, pp. 285–290. 2, 5

Ripperda, N., 2008b. Grammar based facade reconstruction using rjmcmc. In: Photogrammetrie Fernerkundung Geoinformation (PFG), pp. 83–92. 2

Stiny, G. and Gips, J., 1972. Shape grammars and the generative specification of painting and sculpture. In: The Best Computer Papers of 1971, Auerbach, p. 125135. 2

Van Gool, L., Zeng, G., Van Den Borre, F. and MÏler, P., 2007. Towards mass-produced building models. In: PIA07, pp. 209–220. 2

Wenzel, S. and Förstner, W., 2008. Semi-supervised incremental learning of hierarchical appearance models. In: Intern. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Beijing, China, pp. 399 – 404. 2

Wonka, P., Wimmer, M., Sillion, F. and Ribarsky, W., 2003. Instant architecture. Proc. SIGGRAPH 2003 pp. 669 – 677. 2

Zebedin, L., Klaus, A., Gruber Geymayer, B. and Karner, K., 2006. Towards 3d map generation from digital aerial images. ISPRS Journal of Photogrammetry and Remote Sensing 60(6), pp. 413–427. 2