Application of a Formal Grammar to Facade Reconstruction in Semiautomatic and Automatic Environments

Nora Ripperda and Claus Brenner
Institute of Cartography and Geoinformatics, Leibniz Universität Hannover

ABSTRACT

3d city models are used in a huge number of applications today. They are applicable in the area of urban planning and city development, tourism and marketing and as well for navigation. All these applications need a 3d city model of a large area. And in these days the desire for actuality and a high degree of details is rising. Due to this the modelling of buildings as block models as before is not sufficient any more. There is a need for facade reconstruction methods that model windows and other facade elements in more detail. To acquire the huge demand of models, automatic methods are needed.

In this paper we show the importance of the use of structure information for the reconstruction process. With an example we demonstrate the problems of reconstructions methods which work without structure information. Then we present how to use a grammar to integrate structure in the process. Thereafter we present a manual modelling tool which is based on our facade grammar. And finally an automatic facade reconstruction method based on reversible jump Markov Chain Monte Carlo (rjMCMC) is shown.

1. INTRODUCTION

An overview of the extraction of man-made objects from sensor data is given in (Baltsavias, 2004). Especially for the modelling of 3D buildings, numerous approaches apply sensor specific extraction procedures and barely use structure information for the reconstruction.

But modelling structure can be very helpful for the extraction process. A fixed set of structural patterns allows to span a certain subspace of all possible object patterns, thus forms the model required to interpret the scene. The structure is also important for downstream usability of the data, especially for the automatic derivation of coarser levels of detail from detailed models. If the model admits generalization, one model of a building is sufficient for many applications

Our aim is to derive a structural description of a facade from range and image data automatically. In this paper we first present a simple method to extract windows from range data and show the problems arising in reconstruction without any structural information. Then we present a way to integrate structure information in the reconstruction process. Therefore we develop a grammar which describes facades (Ripperda, 2008). To model facades manually according to this grammar and to evaluate the facade grammar a modelling tool is developed. Subsequently an automatic grammar based reconstruction method is presented. For this method reversible jump Markov Chain Monte Carlo is used.

2. RELATED WORK

Facade and building reconstruction methods divide in the two groups of top-down and bottom-up approaches. Bottom-up methods start with the data and build up a model. Examples are (Becker and Haala, 2007) and (Pu, 2007). Whereas the top-down approaches design a model first and in some cases (Wonka et al.,2003) create building models completely without data.

The model can be given by a grammar. Lindenmayer systems (Prusinkiewicz and Lindenmayer, 1990) are used for modelling plants and also for modelling streets and buildings (Parish and Müller, 2001; Marvie et al., 2005). For the facade reconstruction they are not appropriate because facades and buildings in general differ in structure from plants and streets. They don't grow in free space and modelling is more a partition of space than a growth-like process. For this reason, other types of grammars have been proposed for architectural objects. Stiny and Gips (1972) introduced shape grammars which operate on shapes directly. The rules replace patterns at a point marked by a special symbol. Grammars are also used in architecture (Mitchell, 1990) but the derivation is usually done manually, which is why the grammars are not readily applicable for automatic modelling tools.

Alegre and Dallaert (2004) use a stochastic context free attribute grammar to reconstruct facades from image data by applying horizontal and vertical cuts. Van Gool et al. (2007) discuss different facade reconstruction algorithms and show the use of repetitions in the structure for the reconstruction with shape grammars.

Wonka et al. (2003) developed a method that uses a combination of two grammars. The split grammar builds a large set of rules, which divide the building into parts and the control grammar guides the propagation and distribution of attributes. This method allows the automatic reconstruction of different kinds of buildings using one rule set.

Reversible jump Markov Chain Monte Carlo (rjMCMC) is a stochastic method which is also used for building reconstruction. Dick et al. (2004) introduce a method which generates building models from measured data, i.e. several images based on rjMCMC.

The rjMCMC algorithm is used for other applications e.g. detection of road marks (Tournaire and Lafarge, 2007) as well. In general rjMCMC is a top-down-approach, but Tu et al. (2005) integrated generative and discriminative methods and used a data driven MCMC (DDMCMC) for image parsing. In (Ripperda and Brenner, 2007) a data driven extension of an rjMCMC based facade reconstruction method is presented.

3. FAÇADE RECONSTRUCTION WITHOUT STRUCTURE INFORMATION

In this section we show results from Warneke (2008) to discuss facade reconstruction without structure information. The range data used for this work is acquired by the mobile scanning system street mapper (Kremer and Hunter, 2007). In Warneke (2008) two methods are developed. Both use the information that the point cloud contains fewer points where windows are located. The first approach generates a raster and works with morphological operations. The second triangulates the point cloud and defines large triangles as window triangles (like Pu et al., 2007).

The first approach calculates a binary raster image of the point cloud. First the facade plane is calculated and then all points belonging to the facade are projected to this plane. According to these points a raster is filled with ones where points are present and with zeros where no points can be found in the facade. The further process works on the binary raster image. First the morphological closing operator was used to reduce noise. After that the method searches for facade elements. These are supposed to be rectangular in this work. So rectangles are set to the connected regions containing

zeroes in the raster. The intermediate result contains many rectangles which are too large or too small to be windows or doors. To eliminate these false rectangles finally a filter with thresholds for a minimum and maximum window size is applied. Fig. 1 (left) shows the extracted windows.

The second approach is based on the Delaunay triangulation. First the point cloud of the facade is triangulated. The triangles in window regions have at least two long edges because of fewer points in these areas. So a filter over the sum of the three edge lengths can determine triangles belonging to windows. Connected triangles are aggregated to window regions and a bounding rectangle gives the model of a window. Results of this method are shown in Fig. 1 (right).

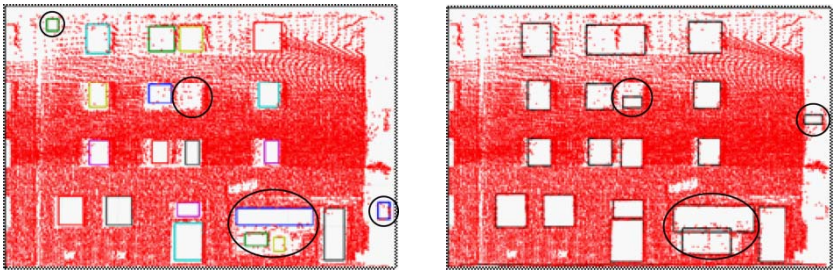

Figure 1: Results of the raster approach (left) and the triangulation approach (right).

Both methods extract most of the windows. While the first approach tends to model somewhat undersized windows most extracted windows with the second method fit well in size and position. But the results show that a window is missing in the reconstruction of the first method. In both results some windows are too small and in the lower right corner is a window which is modelled by three respectively two windows and some windows are aggregated to one rectangle. Also the reconstructed windows are not aligned as the real ones usually are.

All these reconstruction errors can be meliorated by using structure information of facades. In most cases windows are arranged in a grid and the rows and columns are aligned. The windows in a grid have the same size or at least there exists two or three window forms. Besides the grid we often have symmetries or repetitions in facades which can support the reconstruction. In the next section we present a formal grammar that includes the structure information about facades. With this facade grammar we improve the reconstruction process.

4. FAÇADE GRAMMAR

To describe the structure of facades we use a formal grammar. First we explain the concept of grammars in general and then we introduce a facade grammar.

A formal grammar G is a 4-tuple $G=(T,N,S,P)$. The terminal symbols T and the non terminal symbols N build the alphabet of the grammar. S is the start symbol, a non terminal symbol from which all derivations start. And P is a set of production rules. Grammars can be divided in classes of the Chomsky hierarchy depending on the form of the production rules. We use a context-free grammar, which implies that P contains rules of the form $N \rightarrow (T \cup N)+$. This means that a non terminal symbol on the left side can be replaced with a number of terminal and non terminal symbols. All words that can be derived from S with rules from P build the language $L(G)$ of the grammar G.

We want our facade grammar to comprise the structure of facades. We use the matter of fact that facades often have a regular structure. Windows are arranged in a grid and the windows in a row or column have the same size. Looking at the whole facade there often occur symmetries or repetitions. And if there are parts which differ in structure they can often be subdivided in ground floor and upper parts.

For the facade reconstruction we define a grammar *GF*. The language *L(GF)* of *GF* contains all possible facade models (for details see Ripperda, 2008). The facade reconstruction is a derivation process and the model of the facade is to be developed further in each step. Therefore each rule splits the part of the facade corresponding to the left side symbol in a variable number of facade parts corresponding to the right side symbols. So the derivation process is a partitioning process of the facade. The start symbol *S* is an empty facade symbolised by *Facade*. This is subdivided in further derivation steps.

The grammar contains two different kinds of splits. The first is a split in multiple symbols. It is caused by differences in the facade structure and each part is modelled individually in the next steps. A change in structure often occurs in ground floor and upper floors.

The other kind of split is a split in similar regions modelled by one symbol. This is caused by similarities or repetitions. If a facade is symmetric or contains repetitions, the repeated pattern needs to be stored and modelled further only once. Additional information like number of repetitions completes the model.

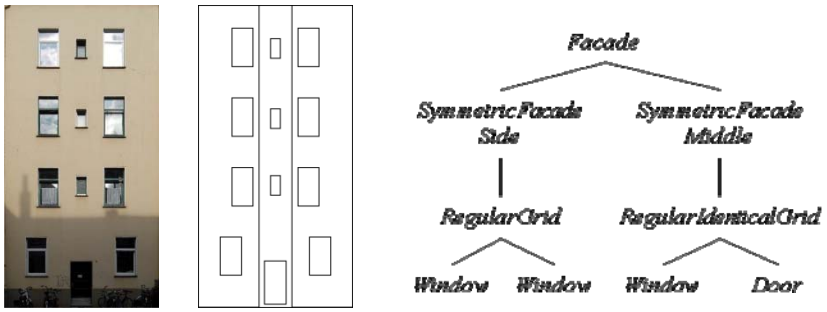

Figure 2: Example facade (left), partition according to the grammar rules (centre) and the corresponding derivation tree (right).

Fig. 2 illustrates an example of a facade reconstruction. It shows a small symmetric facade. First it is divided in *SymmetricFacadeSide* und *SymmetricFacadeMiddle* where only one side of the *SymmetricFacadeSide* is modelled further. *SymmetricFacadeSide* is replaced by *RegularGrid* and *SymmetricFacadeMiddle* by *RegularIdenticalGrid*, each with four rows and one column. On the *RegularGrid* the rule *RegularGrid→Window Window* is applied. Two windows are used because the part contains different windows. The *RegularIdenticalGrid* in the centre is derived at *Window Door*. Here we have identical *Window*s but an additional *Door*.

The model is described by a parameter vector θ which contains the derivation tree and the attributes of the symbols. E.g. the parameter vector of the configuration in Fig. 2 is represented by the hierarchic structure

θ=*Facade*(0,0,*w*,*h*,(
SymmetricFacadeSide(
RegularGrid(1,4,p_x,p_y,d_x,d_y,
Window(*w*,*h*,p_x,p_y), *Window*(*w*,*h*,p_x,p_y))),
SymmetricFacadeMiddle(*w*,
RegularIdenticalGrid(1,4,p_x,p_y,d_x,d_y,
Window(*w*,*h*,p_x,p_y),*Door*(*w*,*h*,p_x,p_y)))))),

where w and h are the width and height of the elements and p_x and p_y give the position and d_x and d_y the distances in x and y direction.

The grammar contains 45 rules. We don't want to list all rules here, but describe the function of the main groups of rules. The first rule is *Facade*→*FacadeRow PartFacade* it divides the ground floor and the upper parts of a facade if there are differences in structure. This is commonly occurring with shop windows in the ground floor. And if the structure in the *FacadeRow* is very irregular this can be replaced by several *FacadeElement*s which can contain distinct terminal symbols.

To model repetitions symbols like *Facade*, *PartFacade* etc. are replaced by *RepeatedFacade*. The repeated part is modelled once and additionally the number of repetitions is stored. The same can be done with symmetric facade. Here are the two possibilities to model exclusively the *SymmetricFacadeSide* or to model additionally a *SymmetricFacadeMiddle* if the centre part differs from the rest.

A very important group of rules are the grid rules. Different symbols can be replaced with a grid. Whereas there are different kinds of grids. We distinguish between regular and irregular grids. This means the spacing of the grid cells is constant or not. And the second criterion is the number of possible children. Identical grids are supposed to have the same windows and other grids can contain different windows. This results in the four grid types *Grid*, *RegularGrid*, *IdenticalGrid* and *RegularIdenticalGrid*.

The last important rule group contains rules which lead to terminal symbols. *FacadeElement*s and all kind of grids can be replaced with terminal elements which are *Window*s and *Door*s. Some elements can be replaced with other terminal symbols like *Shop* or *Doorway*.

5. MANUAL MODELING AND EVALUATION OF THE GRAMMAR

In this section we introduce the manual modelling tool *FacadeModeler* (Uden, 2008). It is a tool for manual modelling and evaluation of the grammar. The user can apply the grammar rules to a facade image using a graphical user interface to model a facade manually. Fig. 3 shows the graphical user interface of the *FacadeModeler*. The programme shows the user all applicable grammar rules for the given situation, other rules are greyed out. The user can apply a rule by pushing one of the enabled buttons in the upper left. Each rule needs a different number of parameters like the number of rows and columns in a grid. These can be chosen in the specification part at the bottom of the window. Positions and sizes of facade elements can be set by moving graphical objects over the facade image controlled by mouse interaction. In the example, *SymmetricFacadeSide* and *SymmetricFacadeMiddle* are modelled. Vertical dotted lines can be moved in the area of *SymmetricFacadeSide* to define the position of window columns. Here, the logic of the grammar symbols is integrated and a move in one part of the *SymmetricFacadeSide* causes a mirrored action in the symmetric part. Synchronously to the manual modelling the derivation tree is build. It is displayed dynamically in the bottom left corner. The resulting image and the derivation tree can be saved after modelling.

For this programme the rules are combined to groups of rules. For example the three rules *Facade → RepeatedFacade*, *PartFacade → RepeatedFacade* and *SymmetricFacadeSide → RepeatedFacade* are combined to one group because for the user the different left sides are not important. The effect of the rule group is always the same and the related button is activated if the left side is matching. Furthermore sequential rules are combined. After deriving a Grid the user has to establish facade elements at the grid points. Therefore rules of the kind *Grid→Window* or rules with other facade elements on the right side must be applied. In the *FacadeModeler* these sequential rules are combined. The user chooses a grid rule and gives all additional settings in the specification part or by mouse interaction. This grouping leads to a number of eight rules the user can choose.

A test with different users shows that people have different ideas to model facades with the given grammar rules. Fig. 4 shows three different possible models of an example facade. Some users model the facade with a differing ground floor and a symmetric upper part where the bricked windows in the middle are not modelled. Others use a symmetric facade with a part in the centre and a grid of windows. And a third possibility is grid of windows and a door. It shows that there is no unique solution for one facade. So we have to find a measure to decide which model is the best. This will be introduced in the following after a general introduction to rjMCMC.

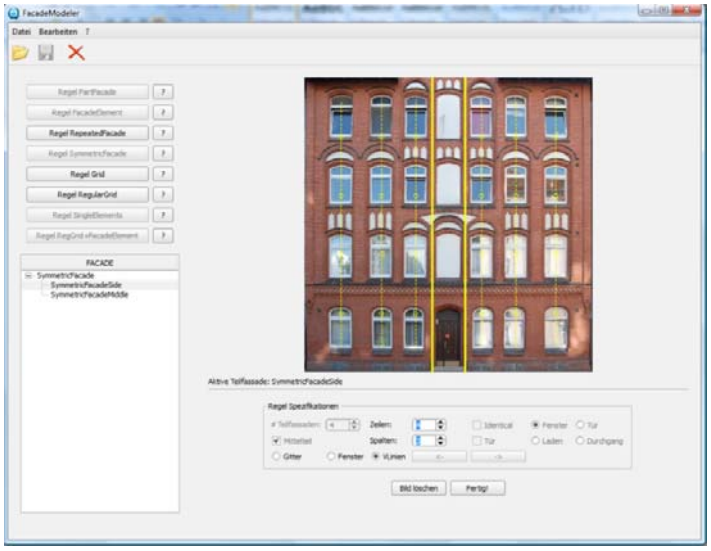

Figure 3: Graphical user interface of the tool *FacadeModeler*.

***Figure 4*:** Different models of an example facade.

6. AUTOMATIC FAÇADE RECONSTRUCTION USING rjMCMC

For the automatic reconstruction we use the stochastic process rjMCMC. With this process we can determine the model θ (see sec. 4) with the highest probability $p(\theta|D_S D_I)$ under given data. The data we use are range data D_S and an orthophoto of the facade D_I. The vector θ encodes the derivation tree of the current state and additional attributes.

The probability $p(\theta|D_S D_I)$ is unknown and to sample from an unknown distribution we need Markov Chain Monte Carlo (MCMC) methods. A Markov Chain is used to simulate a random walk in the space of facade models θ. The transition probabilities from one model to another are given in a transition kernel J. The probability to propose a move from θ_{t-1} to θ_t is given by $J(\theta_t|\theta_{t-1})$. When a new model is proposed the acceptance probability decides whether the change is accepted. This acceptance probability is chosen in a way that the process converges to the target distribution $p(\theta|D_S D_I)$.

The basic MCMC is designed for parameter vectors with constant size. But in our case the dimension of θ changes during the process. Because of that we use rjMCMC which allows changes in the dimension of θ (so called jumps). The probability of a change in dimension is integrated to the transition kernel.

For the rjMCMC process with target distribution $p(\theta|D_S D_I)$ we have to define a transition kernel $J(\theta_t|\theta_{t-1})$ and the acceptance probability α. For more details on the transition kernel see (Ripperda, 2008). The acceptance probability α is given below.

$$\alpha = \min\left\{1, \frac{p(\theta_t \mid D_S D_I) \cdot J(\theta_{t-1} \mid \theta_t)}{p(\theta_{t-1} \mid D_S D_I) \cdot J(\theta_t \mid \theta_{t-1})}\right\}$$

The transition kernel $J(\theta_t|\theta_{t-1})$ is given but the acceptance probability depends also on the unknown distribution $p(\theta|D_S D_I)$. According to Bayes' Theorem $p(\theta|D_S D_I)$ is proportional to the product of likelihood and prior of the facade $p(D_S D_I|\theta_t) \cdot p(\theta_t)$.

To determine the likelihood $p(\theta|D_S D_I)$ we use the minimum description length (MDL) concept introduced by Rissanen (1978). This is often used for hypothesis selection; it selects the hypothesis with the smallest sum of the description of the hypothesis and the description of the data in respect to the hypothesis.

For our reconstruction process the MDL has two benefits. First it gives us a method to compute the likelihood and second this method takes the complexity of the model into account. The second point is important because if the model complexity is ignored simple facades could be modelled by superfluous complex structures. A facade with a simple regular grid is not to be modelled as symmetric or repeated facade which was a possible model even for a simple facade.

For the model complexity the number of symbols and their parameters to describe the model is needed. This can be taken simply from the parameter vector θ. The complexity $c(T)$ of a derivation tree T is given by

$$c(T) = |\, Nodes(T) \,| \cdot \log_2 N_{Sym} + \sum_{N \in Nodes(T)} pb_N$$

where $Nodes(T)$ are the nodes of the tree T, N_{Sym} the number of symbols in the grammar and pb_N the number of bits used for all parameters of symbol N.

With the number complexity *c(T)* a rank order of the models is defined. Fig. 5 shows some example models in their rank order from low on the left top corner to high on the right bottom corner. The given complexities are relative to the most complex possible model.

The complexity of the facade models grow in each derivation step. The start symbol *Facade* is least complex of all. The number of symbols affects the complexity but still more the number of parameters belonging to a symbol. For example a *RegularIdenticalGrid* needs the values for the number of rows and columns, the grid position and the distance between rows and columns. A *RegularGrid* may contain different window types and therefore it needs an additional map which assigns each grid cell a window type. The fourth and fifth example in fig. 5 would have the same complexity if the fifth facade would contain a second *RegularIdenticalGrid* instead of the *RegularGrid*. But the *RegularGrid* needs the window map which says which window type is placed at which grid position. This map increases the complexity from 0.37 to 0.49.

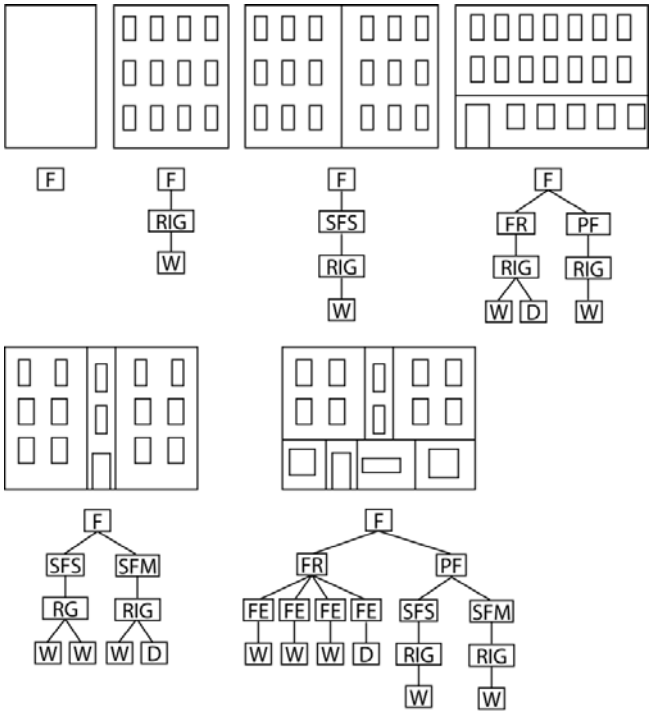

***Figure 5*:** Facade models and related derivation trees in order of complexity (complexity relative to the most complex possible model from left to right: 0.04, 0.15, 0.16, 0.37, 0.49, and 0.72).

The second important part is the matching of the facade model and the data. In MDL the length of the description of the data using a certain model is used. In a pre-processing step we calculate cluster images of the colour and range image. Because of different colour and different depth of window, door and wall areas they belong to different clusters. And parts of the facade with different colour which often indicate different structure in these areas also belong to different clusters.

The facade model given by the derivation tree also divides the facade in different parts. The scoring function evaluates the difference between the facade model and the cluster image. Therefore it counts the number of pixels differing from the main cluster in the discussed area.

Fig. 6 illustrates the scoring method. Input data are a facade model on the left hand side and a cluster image on the right hand side. The facade model in this example defines a window area with a two times two grid of windows and a wall area. For both areas the main cluster, which is the cluster with the most pixels, is determined. In the window area the blue cluster is the main cluster and in the wall area the yellow one. Now for each area the number of pixels that do not belong to the main cluster is counted. In fig. 6 the differing pixels of the window area are marked in bright red and the differing pixels in the wall area in dark red. The sum of differing pixels in all areas divided by the total number of pixels is the scoring function for this part of the facade.

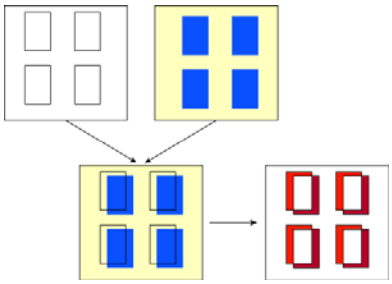

***Figure 6*:** Determination of differing pixels from facade model and cluster image.

In more complex cases the scoring function traverses the derivation tree. And if a symbol splits in different symbols, the score of all subtrees is calculated and summarised. As an example we use the facade on the right hand side of the upper row in fig. 5. The derivation tree is traversed from the start symbol *Facade*. This splits in *FacadeRow* and *PartFacade*. For both subtrees beginning with *FacadeRow* and *PartFacade* the score is calculated separately according to the sample given above. If one or both subtrees consist only of one symbol the method would be the same. The main cluster of the whole area is determined and subsequently the differing pixels are counted.

The scoring function *s(T)* combines the complexity *c(T)* and the difference of data and model *diffPixel(T)*. The sum of both values is normalized by the maximal values *maxComp* and *numPixel* and inverted.

$$s(T) = 1 - \frac{c(T) + diffPixel(T) \cdot \log_2(numCluster)}{\max Comp + numPixel \cdot \log_2(numCluster)}$$

7. RESULTS

For the reconstruction we use a point cloud and an orthophoto of a facade. We show the point cloud of the example facades and visualize the reconstruction result in the orthophoto.

Fig. 7 shows the reconstruction of a symmetric facade. It consists of a *SymmetricFacadeSide* and a *SymmetricFacadeMiddle*. Both parts contain an *IdenticalGrid*. In the outer part a grid of *Windows* and in the centre part a grid of *Windows* and a *Door*.

Fig. 8 shows the second reconstruction. The resulting model is an *IdenticalGrid* with *Windows* and a *Door*. The distances of rows are constant and the distances of columns are varying. Most windows are modelled correctly. But the second column is a little bit displaced to the right. And the

windows in the upper corners are modelled wrong. In the data are large windows subdivided in three parts, which cover the area of two windows in the other parts of the facade. Possible models would be one large window or three small ones. These constellations are not provided for in grammar. A grid contains the same number of windows in each row respectively column.

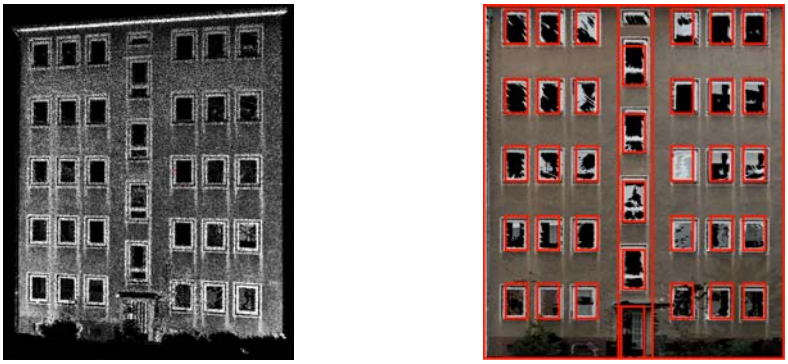

***Figure 7*:** Point cloud and reconstruction result of a symmetric facade.

Figure 8: Point cloud of a facade and the reconstruction result visualized in the orthopoto.

8. CONCLUSION

In this paper we present methods for the facade reconstruction. First we show an approach, which doesn't take structural information into account. We mention the disadvantages of this strategy and introduce a facade grammar to integrate the structure of facades in the process. The *FacadeModeler* allows a manual reconstruction to test the grammar rules. It shows that a large part of facades can be described by the grammar. Even though there are also some facade structures which are not contained in the grammar. But the grammar could be extended for these cases.

For the automatic reconstruction we present an rjMCMC approach, which build the derivation of the model without user interaction. For the rating of proposed models we introduce an MDL based scoring function, which gives us a unitary function to score all derivation trees.

ACKNOWLEDGEMENTS

This work was done within in the scope of the junior research group 'Automatic methods for the fusion, reduction and consistent combination of complex, heterogeneous geoinformation', funded by the VolkswagenStiftung, Germany.

BIBLIOGRAPHY

Alegre, F. and Dallaert, F. (2004): A probabilistic approach to the semantic interpretation of building facades. In: International Workshop on Vision Techniques Applied to the Rehabilitation of City Centers.

Baltsavias, E. P. (2004): Object extraction and revision by image analysis using existing geodata and knowledge: current status and steps towards operational systems. ISPRS Journal of Photogrammetry and Remote Sensing 58, pp. 129–151.

Becker, S. and Haala, N. (2007): Refinement of building facades by integrated processing of lidar and image data. In: Photogrammetric Image Analysis 2007.

Dick, A., Torr, P., Cipolla, R. and Ribarsky, W. (2004): Modelling and interpretation of architecture from several images. International Journal of Computer Vision 60(2), pp. 111–134.

Kremer, J. and Hunter, G. (2007): Performance of the Streetmapper Mobile LIDAR Mapping System in "Real World" Projects. In: Photogrammetric Week 2007, S. 215 - 225

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), pp. 329 – 339.

Mitchell, W. J. (1990): The Logic of Architecture: Design, Computation, and Cognition. Cambridge, Mass.: The MIT Press.

Parish, Y. and Müller, P. (2001): Procedural modeling of cities. In: E. Fiume (ed.), ACM SIGGRAPH, ACM Press.

Prusinkiewicz, P. and Lindenmayer, A. (1990): The algorithmic beauty of plants. New York, NY: Springer.

Pu, S. and Vosselman, G. (2007): Extracting Windows from Terrestrial Laser Scanning. ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007, pp. 320-325.

Ripperda, N. and Brenner, C. (2007): Data Driven Rule Proposal for Grammar Based Facade Reconstruction. In: Photogrammetric Image Analysis 2007, vol. 36, no. 3/W49A, pp. 1-6.

Ripperda, N. (2008): Grammar Based Facade Reconstruction using RjMCMC. Photogrammetrie Fernerkundung Geoinformation (PFG), vol. 2, pp. 83-92.

Rissanen, J. (1978): Modeling by the shortest data description. Automatica-J.IFAC 14, pp. 465-471.

Stiny, G. and Gips, J. (1972): Shape Grammars and the Generative Specification of Painting and Sculpture. Auerbach, Philadelphia, pp. 125–135.

Tournaire, O., Paparoditis, N. and Lafarge, F. (2007): Rectangular road marking detection with marked point processes. In: Photogrammetric Image Analysis 2007, vol. 36, no. 3/W49A, pp. 149-154.

Tu, Z., Chen, X., Yuille, A. and Zhu, S. (2005): Image parsing: Unifying segmentation, detection, and recognition. International Journal of Computer Vision 63(2), pp. 113–140.

Uden, M (2008): Entwicklung einer Benutzeroberfläche zur manuellenFassadenmodellierung mittels Grammatiken. Bachelor thesis.

Van Gool, L., Zeng, G., van den Borre, F. and Müller, P. (2007): Towards mass-produced building models. In: Photogrammetric Image Analysis 2007.

Warneke, A (2008): Extraktion von Fassadenmerkmalen aus Streetmapper-Daten. Diploma thesis.

Wonka, P., Wimmer, M., Sillion, F. and Ribarsky, W. (2003): Instant architecture. ACM Transaction on Graphics 22(3), pp. 669–677.