A Framework for the Generalization of 3D City Models

Richard Guercke and Claus Brenner
Institute of Cartography and Geoinformatics, Leibniz University Hanover, Germany
Appelstr. 9A, 30167 Hannover
{richard.guercke, claus.brenner}@ikg.uni-hannover.de

ABSTRACT

With a growing availability of detailed 3D city models, an increased demand for the generalization of these models is going to arise. In order to build a useful system for the generalization of highly detailed 3D city models, it is necessary to give the user the opportunity to define the relevance of features on a semantic basis. This requires a high level of semantic information that is often not present in existing models and extremely flexible generalization strategies that allow different algorithms with different parameters to be used for the same model. In order to meet these requirements, we propose a framework for the generalization of 3D city models that supports the definition of custom feature hierarchies and generalization algorithms and offers predefined default features and generalization modules that can be used where the application allows it.

INTRODUCTION

At the current state, most 3D city models are created for special tasks like noise emission simulations or for navigation systems. Therefore, there has been little demand for generalization. With a growing availability of highly detailed 3D city models, the demand for the generalization of such models is going to increase when the models are used for purposes beyond those they were created for.

The generalization of 2D models for visualization (cartographic generalization) has been researched extensively, and there are several different tools for automatic cartographic generalization as well as a set of generally accepted generalization operators. The successful generalization of 3D city models, however, requires a higher degree of semantic information to be present in the model and more specialized (up to task-specific) generalization procedures.

In this paper, a framework for the generalization of 3D city models is presented. Within this framework, default generalization tools for the most common features like buildings, roofs etc. are provided. It is also possible to replace standard features with application-specific models and to define custom generalization strategies for different (constellations of) features.

Because the required semantic information is frequently not given explicitly in existing models, the extraction of such information is often done within the generalization procedure – in many cases implicitly. Due to this fact, many generalization approaches consist in greater parts of feature detection. Therefore, we propose a stricter distinction between feature extraction and generalization in order to be able to define more transparent generalization algorithms and reuse existing feature extraction algorithms. To make this possible, generalization algorithms are required to explicitly state within their interfaces what kind of features they operate on and – if these features are not part of the default set of features – how they can be extracted from a data set.

RELATED WORK

While there has been a lot of research concerning the generalization of 2D models (cartographic generalization), the generalization of 3D city models has not received the same level of attention yet. In the CityGML (Kolbe et al., 2005) specification, four distinct levels of detail (LoD) are defined and several approaches concerning the automatic derivation of less detailed models from models with a higher LoD have been presented. The focus this paper, however, is on continuous generalization of city models.

Döllner and Buchholz (2005) introduce the concept of Continuous-Level-of-Quality buildings that allows the user to model buildings with custom granularity according to the task at hand. They do, however, not provide concepts for the automatic generalization of such models. The concepts for generalization introduced in Buchholz (2006) are mostly concerned with visualization issues, especially the treatment of textures.

In his PhD thesis, Lal (2005) addresses the necessity of a stronger separation of the processes of feature extraction and generalization; his focus is on feature recognition and aggregation. Thiemann and Sester (2004) derive a CSG representation by cutting off smaller parts of a building and treating them as additions. In the approach of Kada (2007), a building complex is first divided into its wings (cells) using the main lines of its footprint. The borders of these cells are then used to form the walls of the generalized model. For the generalization of the roof shapes, the feature detection is done explicitly by instancing roof shape primitives for each cell and selecting the best-fitting one.

3D CITY MODELS AND THEIR GENERALIZATION

Features as Semantic Entities

Features represent entities of the real world and are the central part of the model. For the scope of this paper, the term *semantics* refers to a feature's type and application-specific data.

In the 2-dimensional case, it is often possible to use purely geometric algorithms for the generalization of a shape in the real world; streets and canals may, for example, be simplified by similar algorithms for line simplification. The semantics of the objects are often introduced by using different algorithms or parameter sets for different feature types; only in rare cases is semantic information used in the algorithms themselves. In the 3-dimensional case, however, the semantics are often of vital importance for the generalization of a feature because there are several constraints that depend strongly on a feature's semantics: A wall surface should, for example, stand (more or less) perpendicular to the ground while a roof surface should not. If the process of generalization is intended to transform a valid model into another less detailed but also valid model, it is therefore necessary to take the semantics of a feature into account in the algorithm itself.

The problem of this approach is that it requires specific algorithms for each feature type and potentially even different algorithms for features of the same type with different parameters. This problem can be alleviated by defining parameterized generalization algorithms and using geometric simplification methods in appropriate cases. A system for the generalization of 3D city models should, however, offer the possibility to introduce specific, application-dependent generalization operators for all features.

Most modeling systems (such as the one underlying the CityGML data format) use a hierarchical approach with different layers to represent the features in the real world. The different layers in such a model represent different domains. In the case of CityGML these are, for example, water bodies, buildings, traffic objects and vegetation.

The hierarchies within the domains usually indicate “part-of” relations. A roof object, for example, is usually part of a building object and will therefore usually appear as a child object of a building feature.

In order to make the model extensible, the concept of inheritance is often introduced as well: This way, it is possible to introduce new types of features that can replace features of a known type in the hierarchy; for example, a “gabled roof” feature type could be introduced as a specialization of the “roof” feature type.

Parametric and Explicit Models

An important criterion for the distinction of 3D city modeling systems is whether they use explicit or parametric representations of the underlying geometry. In explicit models, the geometry is stored with the feature, for example as a set of polygons that represents the roof surfaces of (a part of) a building. In a parametric model, the geometry is implicitly given through the parameters of the feature: The shape of a symmetric gabled roof, for example, is given by its eaves and ridge heights in combination with the width and depth of the building on which it is built.

Explicit models have the advantage that the geometry of the model can be reconstructed from the model without knowledge of the semantics. This means that the model can be visualized and interpreted in many respects using only a limited set of concepts like geometric primitives; visualization and purely geometric calculations can be done independent of the semantics. For this reason, the explicit approach is used especially in formats for the exchange of models. The CityGML model, for example, uses the explicit GML format to represent geometry. The disadvantage of this approach is that in many cases, semantic information has to be stored redundantly. In the CityGML format, for example, it is possible to explicitly label a roof as a gabled roof; the geometry associated with the roof may, however, form an entirely different shape.

In a parametric model, the features are represented by instantiating feature classes. The values of special data fields (parameters) determine the geometry of the object. The advantage of this approach is that a wide range of internal constraints can be assured implicitly and (for example generalization) algorithms can work on a higher level of abstraction. Additionally, parametric representations are often considerably more compact than explicit ones because the type information and some parameters can define highly complex models. For these reasons, the default model of the generalization framework uses a parametric modeling approach. In concept, however, the framework is able to work with both geometry representations. The problem with parametric representations is that in order to read a data set, the underlying model’s semantics have to be known. If necessary semantic information is contained in explicit geometric data, it has to be extracted from that data.

Generalization and Feature Extraction

Generalization and feature extraction are closely related: Most generalization algorithms require semantic information that is usually not explicitly given in a model. For this reason, heuristics are defined to deal with the lack of relevant information or it is extracted from the model for the purpose of the algorithm.

A common case are models in which buildings are given as a set of surfaces in which roof and wall surfaces are not labeled. If a generalization algorithm uses such information, it must be extracted from the model. Thiemann and Sester (2004), for example, consider smaller parts of a building that appear on its outside to be additions and therefore geometrically less relevant than the basic shape. The consequence of this observation is to cut off all parts of a feature with an extent of less than the target resolution. As in many other generalization algorithms, a major part of the effort is directed towards the extraction of the features instead of the generalization itself. For this reason, it is sensible to

separate the feature extraction and generalization steps explicitly. This way, it is possible to reuse both the feature extraction and generalization algorithms in different combinations.

An additional improvement of the separation of the steps is the fact that existing feature extraction solution could be used. Milde (2008) and Ripperda (2008), for example, present projects concerned with the extraction of detailed roof and façade structures from mostly geometric data. Models provided by these approaches contain a high level of semantic information and are therefore promising for semantics-based generalization.

THE UNDERLYING CITY MODEL

Features

In the framework presented in this paper, the central modeling element is the concept of features. For the modeling of the default features, a parametric approach was chosen because it is often easier to define generalization operators in terms of characteristic parameters and parametric models can ensure many constraints implicitly. Additionally, explicit models are usually less flexible and occupy more space in memory.

These features are structured in trees that represent containment hierarchies: For example, a roof as part of a building that is part of a building block which is a part of a quarter of a city which is part of a district (and so on). The hierarchy is extensible in all directions: Features representing finer details like bricks can be introduced as well as features that model large structures as whole countries. It is also possible to define application-specific features that can be used instead of or in addition to the features in the default hierarchy. The children of these features may be of types from the default hierarchy. This makes it possible to introduce application-specific feature classes without having to re-implement existing ones.

As in a scene graph, individual transformations can be defined for each feature in the hierarchy. Using these transformations, local coordinates can be used in the modeling of the features. An additional advantage of these transformations is that they can be used to alleviate the problem of locally different distortions that arise from the transformations used in most coordinate systems.

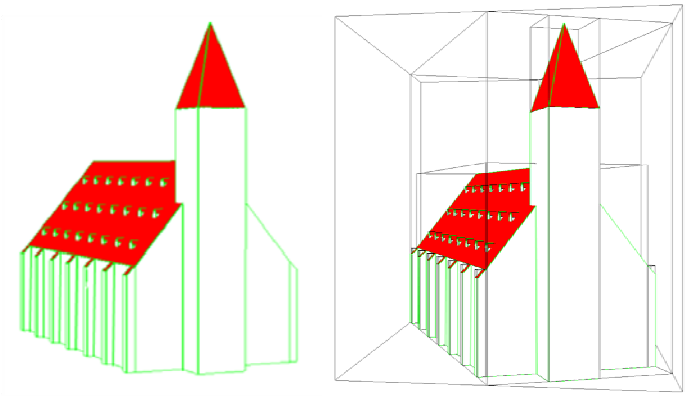

Figure 1: Model of a church with (right) and without (left) the bounding boxes of its parts.

Because the feature hierarchy represents containment relations, it can be also used as a search structure similar to the R-Tree. This is made possible by defining bounding boxes (or domains of more suitable shape) for all features (in their local coordinates) that are updated automatically when

new child features are attached. This can be extremely helpful, especially if geographic queries on large models have to be evaluated. Figure 1 shows a model of a simple church with and without the bounding boxes of the features from which it was built.

Due to the modular structure of the model, it is also possible to define specific feature models that use explicit modeling of geometry. If one wants to use, for example, the approach of Kada (2007) to generalize building complexes for which no parametric model could be found, it is possible to define a feature type like ComplexBuilding that can be inserted in the feature tree in places where buildings may appear. A generalization module that is able to handle models that contain such complex buildings has to provide a method that is called when such a building is encountered in the generalization process. Such a method may, for example, implement the approach of Kada (2007) for the generalization of the building model.

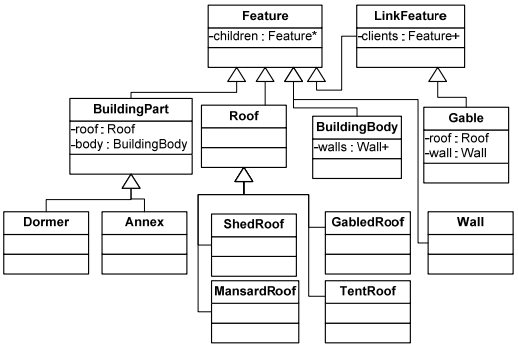

Figure 2: Feature classes in the prototype.

Figure 2 shows a UML diagram of the feature classes used in the first prototype. The attributes in the *Feature* and *LinkFeature* classes are properties: In the *BuildingPart* class, for example, the *roof* and *body* attributes are also part of the *children* list, and the *roof* and *wall* of a *Gable* are its *clients*.

Group Features

Group features represent groups of similar features, often arranged in a pattern. Such features are interesting in the context of generalization in two respects: They provide a means to reduce the amount of memory that is needed to store a feature and they are necessary for generalization operators like typification.

The reduction of memory used to describe a constellation of features is possible because only one default feature, the pattern the features are arranged in and the deviations of features from the pattern or shape of the default feature have to be stored.

For the purpose of generalization, group features are of special importance because they are necessary for generalization operators like typification in which a mostly regular pattern of similar features is replaced by a similar pattern (usually the same) with less – usually enlarged – features. This generalization operator makes it possible to keep the characteristic pattern of features even if the target resolution does not allow the individual features to be retained.

The most important group features in the context of generalization are features grouped in regular (matrix-like) patterns. These are especially suitable for typification because the reduction of the number of features is easily done by reducing the number of features in the different dimensions. By using the same ratio of reduction for all dimensions, it is also possible to achieve a homogenous degree of reduction in all parts of the region covered by the feature group.

In order to be able to extract group features from a data set in which they are not labeled, it is necessary to define strategies to determine how a constellation of more or less similar features can be allocated to different groups of features in such a way that the resulting model is best suited for generalization. An approach to detect group features from laser scanning data is, for example, introduced in Bokeloh (2009).

Link Features

Link features are features that represent relations between different features. Additionally, they may also represent tangible objects in the real world. A gable, for example, is a link between a roof and the walls below. Such a gable can, for example, be represented as a feature that is attached to a wall feature as an addition. When in the course of a generalization process the roof changes its shape, the gable is also changed and the affected wall can be notified automatically through the gable feature.

Such links can, however, also be of a more abstract nature. In the model of the church in Figure 1, a feature may be introduced that indicates that both sides of the church are symmetric or that all windows should have the same size. Such features would not necessarily be part of the general feature hierarchy. They offer, however, valuable information that can be used in the generalization process.

Additionally, link features can also be used to model intersections of features. This can be useful to model features located on the intersection of other features or complex intersections of (possibly) multiple features like many intersecting roofs or a terrain intersection curve (TIC).

Creating Geometry

Because a parametric model is used for the standard features, information about the geometry of the features is not present directly in the model. This makes it possible to develop different ways of deriving geometric representations of a feature.

For the process of generalization, information about the exact geometry of features is necessary in many cases but often it is not needed directly because custom generalization algorithms can adjust the parameters of a model to a target scale without using a geometric representation of the features.

If an explicit representation of a feature's geometry is needed, it has to be reconstructed from the parameters of the feature. This is done in a separate component of the system in which different modules are defined for the reconstruction of the geometry of each feature. This way, it is possible to adjust the generation of the geometry to the task at hand. If, for example, the thickness of a wall is given in the parametric model but some algorithm for which the geometry is intended works with walls that are represented as single surfaces, it is possible to create the geometry accordingly to return a single surface.

THE GENERALIZATION COMPONENT

Generalization of Parameterized Models

As indicated in the preceding chapters, the generalization of a parameterized model requires individual generalization modules for all feature classes. Additionally, scaling methods should be available because they are required for operations like emphasis and typification.

Generalization operations can, in principle, be defined for any constellation of features. They transform these features into a new set of features that is valid at the target scale. Usually, such algorithms are defined for small constellations in the feature hierarchy, for example for a building part with a certain roof shape.

The generalization module for a feature class basically provides a function with a signature get_generalization(Feature+, Resoultion):Feature* that returns a new (possibly empty) set of features that is a generalized version of the input at the given resolution. This means that in principle, the module is responsible for the generalization of all features below the given one(s) in the feature tree. In most cases, the different modules will only cover a depth of about one to three levels in the feature tree and let the central control unit decide how to deal with sub-features that are outside its scope.

This usually leads to a recursive approach: A typification module for group features, for example, may first request generalized versions of the features it contains. This process may again trigger different generalization modules. The process stops when a leaf feature is reached – in this context, a leaf feature is a feature for which the generalization returns an empty set. Once the generalized features are collected, the typification algorithm determines how many features fit into its domain and chooses the parameters of the pattern accordingly. A possible criterion for the scaling of the features is to require the diameter of their bounding box to be greater than the resolution.

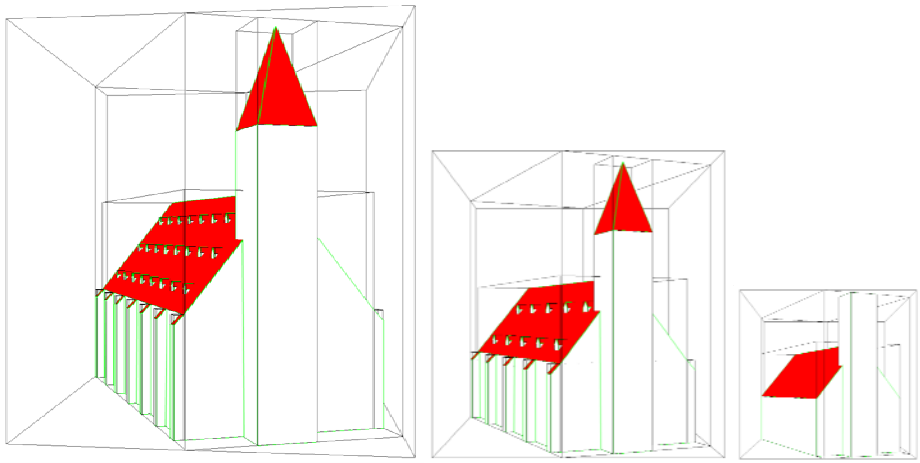

Figure 3: Generalization sequence for the church model.

An example of a generic generalization operator is the typification operator for group features: If the target resolution does not allow the individual features in the pattern to be retained, the features are emphasized in such a way that they can be represented at the target scale. If the emphasized features cause conflicts, the number of individual features is reduced while the general pattern is retained. In Figure 3, the church model from Figure 1 is displayed at different levels of detail. The second model was derived using typification: The three rows of eight dormers each were replaced by two rows with five dormers; the seven support poles on the sides were replaced by five.

Coordination: A Modular Automaton

The most complex problem with the generalization of city models is how conflicts are resolved. Conflicts arise when either different generalization modules can be chosen in a given situation or if a generalization operation produces result feature sets that violate constraints (for example overlapping features).

While, in principle, the current generalization module is responsible for the resolution of conflicts within its scope, it is useful to have a global instance to which requests for the generalization of feature outside the algorithm's scope can be directed.

For the resolution of conflicts, different models can be chosen. For the first prototype, a simple rule-based approach is being implemented. Later solutions can use blackboard techniques, randomized optimization strategies or analogies from physics like spring models or weak primitives.

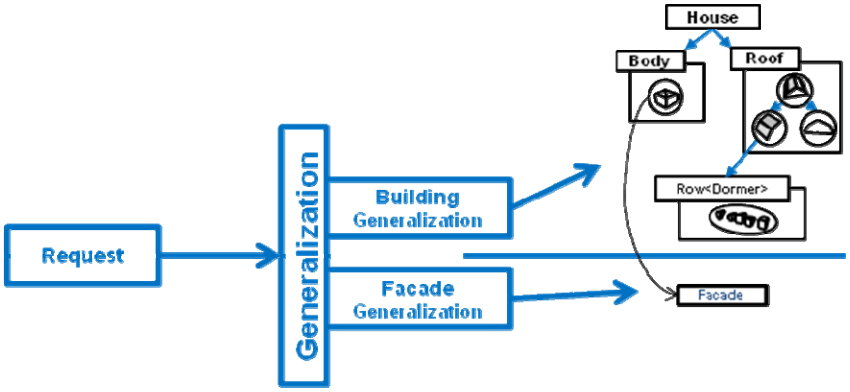

Figure 4: Modularized Generalization.

Figure 4 shows a possible delegation of responsibility among different generalization modules: The generalization of the building delegates the generalization of details on a wall to a specialized façade generalization module. The central generalization control module can be used by the building generalization to find the appropriate module for the generalization of facades. This way, the building generalization does not have to know all features that may appear in its scope but only those it needs.

CONCLUSION AND OUTLOOK

In this paper, a framework for the generalization of 3D city models has been proposed. Within this framework, custom feature types and generalization algorithms can be defined. In a first prototype, a small hierarchy for the modeling of buildings has been developed together with modules for generalization.

Further work is needed to enhance the prototype of the framework in different respects: The feature model is going to be extended, a control module with different conflict resolution strategies for the generalization model has to be implemented, and algorithms will be developed for the generalization of different feature constellations. The prototype is also going to be extended by the possibility to generate data sets with non-uniform user-defined resolutions.

ACKNOWLEDGEMENT

This work was funded by the German Ministry of Education and Science (BMBF) in the context of the GDI Grid project.

BIBLIOGRAPHY

M. Bokeloh, A. Berner, M. Wand, H.-P. Seidel, A. Schilling, 2009: Symmetry Detection Using Line Features, Computer Graphics Forum (Proceedings of Eurographics), to appear

H. Buchholz, 2006: Real-time Visualization of 3D City Models, PhD thesis, Universität Potsdam

J. Döllner and H. Buchholz, 2005: Continuous Level-of-Detail Modeling of Buildings in 3D City Models. In GIS '05: Proceedings of the 13th annual ACM international workshop on Geographic information systems, ACMPress, New York, NY, USA, 173–181.

M. Kada, 2007: 3D Building Generalisation by Roof Simplification and Typification. In: Proceedings of the 23th International Cartographic Conference, Moscow, Russian Federation.

T. H. Kolbe, G. Gröger and L. Plümer, 2005. CityGML: Interoperable access to 3D city models. In: First International Symposium on Geo-Information for Disaster Management GI4DM.

J. Milde, Y. Zhang, C. Brenner, L. Plümer and M. Sester, 2008: Building Reconstruction Using a Structural Description Based on a Formal Grammar, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXVII, Beijing, China

J. Lal, 2005: Recognition of 3D Settlement Structure for Generalization, PhD thesis, Technische Universität München, 2005

N. Ripperda, 2008: Determination of Facade Attributes for Facade Reconstruction, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 37, no. B3a, p. 285-290

F. Thiemann and M. Sester, 2004**:** *Segmentation of Buildings for 3D-Generalisation*, Proceedings of the ICA Workshop on generalisation and multiple representation, Leicester, UK