A CONCEPT FOR ASSIGNMENT OF TEXTURES TO PARTIALLY OCCLUDED FACES OF 3D CITY MODELS STORED IN CITYGML

D. Iwaszczuk, U. Stilla

Photogrammetry and Remote Sensing, Technische Universitaet Muenchen (TUM) - (iwaszczuk, stilla)@bv.tum.de

Commission IV, WG IV/8

KEY WORDS: Texture Mapping, 3D Building Models, Topology, CityGML

ABSTRACT: Automatic texture mapping is an important task in enrichment of the common 3D city models. A significant part of all algorithms for automated texture mapping is the visibility checking. Nowadays most algorithms for texture extraction use visibility check based on z-buffer or polygon intersection in the image plane. Thus, the visibility of particular face is calculated for every frame independently. However, in many 3D city models some polygons are modeled which can be never visible for the camera because of e.g. neighboring buildings. In this paper we discuss how the topological relations between buildings and semantics can be used for visibility checking and texture mapping. First, the influence of the automatic 3D reconstruction on the geometry and topology of the 3D city models is discussed. Then, the strategy for enrichment of the model with information about invisible faces or their parts in neighboring buildings is presented. Further, the possibilities for storage of the geometry of touching buildings are reviewed considering the *CityGML* standard. Finally, two procedures for storage of the invisible faces: in the geometry and in the texture are presented.

1. INTRODUCTION

3D city models are applied in more and more fields. The 3D structures of the city models are usually created from data which were generated with methods of photogrammetry and remote sensing. Models are often stored in different data formats using different properties. Because of this 3D city models are often not interoperable. To enable the interoperability of the 3D geo-data from many sources a common information model for the representation of 3D urban objects – *CityGML* has been recently developed (Gröger et al., 2008). *CityGML* provides many components in several Levels of Detail (LoDs). Thus the 3D city models stored in *CityGML* can be a detailed representation of urban areas.

1.1 Related work

In many cases, textures are an important component of the 3D city models. A realistic appearance of the buildings is important for e.g. virtual sightseeing, urban planning, disaster management or building inspection. Further, the textures can be used to detect objects on façades and roofs (Mayer & Reznik, 2005; Mueller e al., 2007; Ripperda & Brenner, 2007; Hoegner & Stilla, 2009). These detected objects can be embedded in the 3D city models as vector data. Thus, it is not sufficient to display the textured 3D city model correctly, but the texture should also represent the real geometry and appearance of the face. Particularly, the position of the objects seen in the texture, e.g. windows, doors, should be accurate, as well as the parts of face covered by other faces should be marked in the texture.

Faces can be occluded by objects modeled in the 3D city model (e.g. other buildings), by unmodeled and not exactly modeled objects (e.g. traffic signs, trees, street-lamps, pedestrians, cars). Usually, for detection of modeled occlusions, the depth-buffer method is applied. It is a basic method adopted from computer graphics removing hidden surfaces. The depth-buffer is a matrix storing for every pixel the distance from projection centre to the model surface. This method is often presented with some variations.

Karras et al. (2007) proposed a method where every triangulated 3D mesh is projected onto projection plane and for every triangle occupied pixels get identity number (ID) of the triangle. For pixels with more than one ID, the closest one is chosen. Frueh et al. (2004) used a modified depth-buffer, storing additionally the product of a triangle's normal vector with the camera viewing direction at each pixel. Using information about this vector product, non-occluded edges can be detected. Abdelhafiz & Niemeier (2009) integrated digital images and laser scanning point clouds. They used a "Multi Layer 3D Image algorithm" which classifies the visibility on two stages: point stage and surface stage. The visible layer and back layers are applied. Occluded vertexes are sent to a back layer, while visible vertexes appear on the visible layer. An image is used for texture mapping of a mesh, if all three vertexes of it are visible in this image.

In some works polygon-based methods have been proposed. In contrast to depth-buffer, these methods are more accurate and resolution independent. In Kuzmin et al. (2004), all polygons are projected onto image plane. For every polygon overlaying polygons are found and intersected. For every resulting polygon the visibility is determined.

In both methods for the occlusion detection, the computational effort is high. Therefore, the number of polygons projected onto image plane needs to be as small as possible. For this purpose clipping with the image frame and culling of the back-faces can be applied. However, this method does not remove faces which are never seen, e.g. touching walls of neighboring buildings. This problem is especially remarkable in dense build-up areas, where the buildings form blocks and rings with high ratio of touching walls (Fig.1A). Besides, in some cases the invisible parts of faces get textures which do not represent the real appearance of this covered plane (see Fig. 1B).

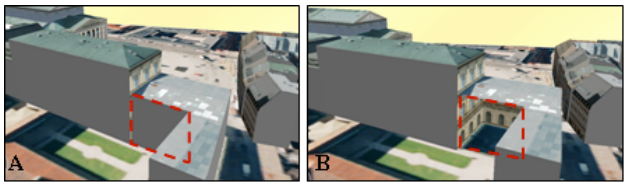

Figure 1. An example of 3D city models near Max-Joseph-Platz in Munich, Germany. A –rear view of touching buildings; B – after removing of the back wall a texture for an invisible wall is seen (model from http://www.citygml.org/1539/)

Furthermore, differentiating between parts of faces invisible in a particular frame and parts of faces generally invisible is an important task for generation of occlusion-free textures. It is possible to complement the missing part of the texture from another frame using multiple images. Identification of invisible regions helps to avoid the search for missing parts of the texture, which are never visible.

1.2 Overview

Aim of this article is to address the role of topology and semantics for texture mapping. We concentrate mainly on the faces or their parts which are invisible because of the model geometry. We assume that the input 3D model does not have complete information about topology and includes some inaccuracies in geometry. This case is rather normal in practice. Reasons for this situation, which are associated with the automatic 3D reconstruction of the 3D building models, are discussed in Chapter 2. In Chapter 3, we present a strategy for finding the parts invisible in the model. Afterwards, we follow two ideas to store this information. The first idea is to store the invisibility in the geometry and the second idea is to store invisible regions in textures. For better understanding of both ideas, first we discuss in Chapter 4 the possibilities for storing of the geometry of touching buildings. In Chapter 5 we present the first method, i.e. storage of the invisible faces or their parts in the geometry. We focus on the topological and semantical model of *CityGML*. In this method, we obtain a new model with changed geometry including information about invisible faces. Alternatively, invisible regions can also be stored in textures, which we present in Chapter 6. In this Chapter we introduce a concept of "informative texture" which is the output of the process in this case. We conclude the paper in last chapter with a short summary.

2. AUTOMATICALLY CREATED 3D CITY MODELS

Creation of the 3D city models manually is a very time consuming task. Thus, many algorithms for automatic building extraction have been developed. Usually for 3D model generation, point clouds from aerial laser scanning (e.g Wang & Schenk, 2000), aerial imagery (e.g. Jibrini et al., 2000) or their combination (e.g. Rottensteiner & Briese, 2003) are used. Mainly two groups of methods for reconstruction are discussed in literature for building reconstruction: data-driven methods (e.g. Vosselman, 2002; Wang & Schenk, 2000) and model-driven methods (e.g. Suveg & Vosselman, 2004). Typically, in data-driven methods clustering of surface normal directions and 3D Hough transform are used for the detection of planes. On the contrary, in the model based methods shape primitives are fitted to the point clouds.

In the model-driven methods often the constructive solid geometry (CSG) modeling is applied (Brenner, 2004). This representation uses combination of volume primitives and Boolean operations and is implemented widely in computer aided design (CAD). On the one hand, modeling with CSG ensures topological correctness (e.g. common points for all polygons of one primitive) but on the other hand, it can cause some ambiguity in the geometry of the model. In some implementations, a complex building can be modeled as a combination of solids, where one is inside of another, which does not reflect the reality.

Alternative modeling approach is the boundary representation (B-rep), which is created directly from measured points, lines or planes. In B-rep often the topology is not correct and only the external building hull is included.

However, in both methods often only the geometry is modeled and the relations between features are not considered. Therefore, the topological relations between objects and semantics need to be built in post processing. This step of creation of correct 3D city models must not be skipped, because the semantically and topologically correct data can only allow further applications and analysis.

Unfortunately, algorithms for automated creation of topology and semantics are still rather a research topic and in practice the 3D city models are produced as geometrical models with basic semantics. Therefore, often the information from the 3D model cannot be used for detection of the invisible faces. Furthermore, the invisible regions are not even stored as separate polygons. Thus, a technique for detection of these invisible polygons in building models is needed. In Chapter 3, we introduce a strategy for detection of neighboring faces and of their common part.

3. DETECTION OF INVISIBLE FACES

3D city models in *CityGML* standard should store geometry, semantics and topology of the buildings and other city objects. For automatic texturing not only the geometry but also information about semantics and topology should be taken into account. Unfortunately, often the existing 3D city models do not include topological relations. As mentioned in the Chapter 2, also the geometry requires some corrections. Various algorithms are needed to ensure the topological, semantical and geometrical correctness. One of the main objectives of this paper is to discuss the problem of touching faces, so in this Chapter we propose a procedure for detection of this relation. The proposed method is based on vector data stored in the 3D model. Thus we chose vector-based geometric analysis instead of image-based visibility detection.

The detection of touching faces is implemented pair wise. For every pair of faces is investigated if the polygons: (i) lie in the same plane, (ii) have reversed orientation and (iii) intersect. In case of three positive answers, the intersection is carried out and the common part is recognized as invisible. In reality, due to digitization, rounding errors or inaccurate modeling (e.g. no snapping), the faces of touching buildings does not touch exactly in the digital model (see Fig. 2). To solve this problem two strategies can be considered.

1. 3D model can be corrected even before detecting touching faces. For this purpose the points which represent the same point (corner) in the real world need to be found. Then during an adjustment the geometry of the model should be changed considering the following conditions:

- the same corners in the reality should be represented by one point in the model;
- the planarity of the faces needs to be kept;
- the right angles should be kept.

2. A tolerance can be introduced while searching the touching faces. That means, faces lying in almost the same plane (angle between planes $\alpha < \alpha_{max}$ or the distance between planes $d < d_{max}$) are also taken into account. For determining the intersection, a mean plane is created and the faces are projected on this plane orthogonally. After intersection, the back projection of new polygons can be carried out.

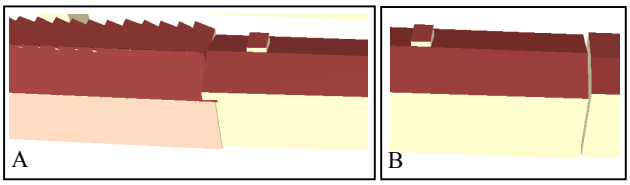

Figure 2. Examples of not exactly touching faces in the model: A. overlapping buildings; B. gap between the buildings

These detected invisible parts of the model have to be stored at least for the texturing process. This can be reached by changing the model geometry and using a new geometry for texturing (see Chapter 5). Alternatively, the invisibility can be marked in the textures which preserve changes in the model (see Chapter 6).

4. CONSIDERING TOPOLOGICAL AND SEMANTICAL RELATIONS

Touching buildings can be observed in dense city areas as well as in residential districts as houses with garages. In Fig. 3A we present such an example, where a garage (red) was built touching the residential house (blue). These buildings fulfill a topological 3D relation "meet" (Zlatanova et al., 2004). It means that two faces, one from the garage and one from the house, partially occlude each other. Hence, they have a common, invisible part. In a 3D building model these faces can be stored in few different ways. Basically for every of both faces we can consider three cases:

- the face is stored as one polygon including the common part (in Fig. 3 first column for the residential house, first row for the garage)
- the face is stored as one polygon and the common part as a whole (second column for the residential house, second row for the garage)
- the face is stored as two polygons, the common part is a separate polygon (third column for the residential house, third row for the garage)

Combining these 3x3 possibilities we get 9 different ways to store the model of two touching buildings (Fig. 3B-J). The examples presented in the Fig. 3 have various topological senses as follows. In presented order can be noticed that the cases on the main diagonal (B,F,J) handle both faces in the same way. Case B, where both faces are one polygon including the common part, does not represent any topology and its influence to the geometry of the model. In this solution removing a building does not require any changes in the geometry in the rest of the model. Case F suggests that the buildings are one entity and there is a connection between them. In this case, after removing one of the buildings a hole in another building is developed, therefore the original geometry has to be completed. Nevertheless, this solution is optimal for texture mapping, because covered parts of faces are not included in the model. Thus for every face in the 3D model a texture is expected. In case J the topological relations reflect in the geometry and this solution does not require changes of geometry after removing a building.

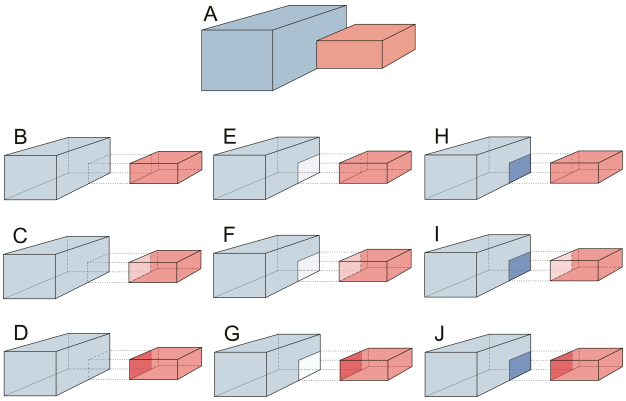

Figure 3. Possibilities for storing of the touching buildings

While cases C, D, E and H do not make any sense from topological point of view, cases G and I can be considered as an important solution. Advantage in these solutions is that the common geometry is stored only once. On the one hand, it reduces the size of the model because of lower number of polygons, but on the other hand, it is not clear to which building the common polygon should belong. To ensure closing of both buildings a link to the common geometry can be applied (Fig. 4.). This solution is presented in *CityGML* specification.

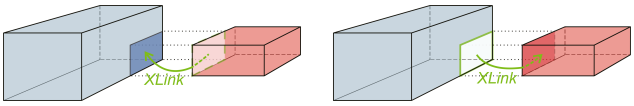

Figure 4. Storage of the touching buildings with link to the common geometry

In *CityGML* the topology is implemented using the *XML* concept of *XLinks*. Each geometry object that is shared by different geometric aggregates or a different thematic feature is assigned a unique identifier, which may be referenced by a *GML* geometry property using a *href* attribute (Gröger et al., 2008). This solution helps to avoid storing of the same geometry multiple times. An example of the *CityGML*-code using *XLinks* is presented in the Fig. 5.

Considering the topological relations of the 3D model, the invisible faces could be identified directly in the code. But it should be mentioned that sharing the geometry does not determine the invisibility for texturing. The change of orientation (*gml:OrientableSurface orientation='-'>…*) could be understood as assured invisibility, because two of the same geometries with reversed orientation have to cover one another. However we suggest using semantics to make it clear which polygon is invisible, because theoretically, it is possible that a shared geometry refers to a feature inside of the building.

```
<!-- .... -->
<bldg:Building gml:id="ID_Building_001">
<!-- .... -->
  <bldg:WallSurface gml:id="ID_Wall_001_0003">
<!-- .... -->
  <gml:surfaceMember>
   <gml:Polygon gml:id="ID_Plygon_001_W0003_001">
     <gml:exterior>
      <gml:LinearRing gml:id="ID_Ring_001_W0003_001">
       <gml:pos srsDimension='3'>12.0 0.0 0.0</gml:pos>
<!-- .... -->
      </gml:LinearRing>
     </gml:exterior>
   </gml:Polygon>
  </gml:surfaceMember>
  <gml:surfaceMember>
   <gml:Polygon gml:id="ID_Plygon_001_W0003_002">
    <gml:exterior>
     <gml:LinearRing gml:id="ID_Ring_001_W0003_002">
      <gml:pos srsDimension='3'>12.0 4.0 0.0</gml:pos>
<!-- .... -->
     </gml:LinearRing>
<!-- .... -->
<bldg:Building gml:id="ID_Building_001_1">
 <bldg:boundedBy>
<!-- .... -->
  <bldg:WallSurface gml:id="ID_Wall_002_0003">
<!-- .... -->
  <gml:surfaceMember>
   <gml:OrientableSurface orientation="-">
    <gml:baseSurface xlink:href="#ID_Plygon_001_W0003_002"/>
   </gml:OrientableSurface>
  </gml:surfaceMember>
<!-- .... -->
```

Figure 5. Part of the *CityGML*-code using XLinks to implement the topology

At the semantic level, real-world entities are represented by features, such as buildings, walls, windows, or rooms. The description also includes attributes, relations and aggregation hierarchies (part-whole-relations) between features (Gröger et al., 2008). The visibility can be explicitly defined by creation of an attribute. In *CityGML* it could be stored using generic attribute (see Fig. 6). This attribute has to be stored only once. The *XLink* referring to the geometry (to the polygon) automatically refers also to the attribute. Thus, geometries with both orientations should be recognized as invisible.

```
<!-- .... -->
  <gml:surfaceMember>
   <gml:Polygon gml:id="ID_Plygon_001_W0003_002">
    <gml:metaDataProperty>
     <gen:stringAttribute name="visibility">
      <gen:value>invisible</gen:value>
     </gen:stringAttribute>
    </gml:metaDataProperty>
<!-- .... -->
```

Figure 6. Part of the *CityGML*-code with implemented generic attribute defining the visibility of the polygon

5. STORAGE OF OCCLUSIONS IN THE GEOMETRY

One solution for preserving the information about the invisible faces and their parts is to change the geometry. According to the *CityGML* topological model, every geometric element should be stored only once. For every two touching faces the intersection is carried out. The intersection of two polygons results in new polygons and reorganization of the geometry (Fig. 7). Alternatively, after intersection the invisible polygons can be removed from the model and this new model (like in Fig. 3F) can be used for texturing. However, as discussed in Chapter 4, this way of modeling can cause ambiguity in interpretation of the model, especially synth etic connection between buildings.

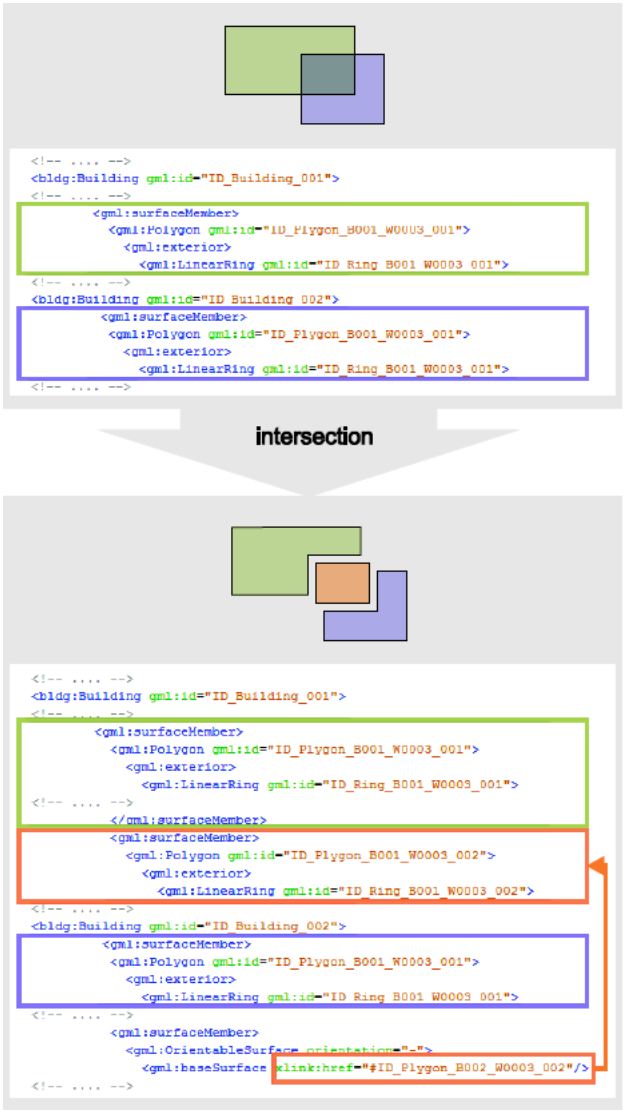

Figure 7. Changes in the *CityGML*-code after intersection of neighboring faces

6. STORAGE OF OCCLUSIONS IN TEXTURES

Texture is a rectangular image assigned to a polygon which depicts its appearance. Usually, a texture is used for visual improvement of the 3D city models. For speeding-up the display of textured models many separate textures can be packed in a patchwork (Kaul & Bohn, 2008) or the same texture can be assigned to more than one face of similar buildings. However, in some cases textures are used for further processing and feature extraction. Thus, it should be kept in mind that texture is a potential source of information. So every texture should be geometrically accurate. Further, occluded areas should also be marked, even if they are synthetically complemented for visualization (e.g. interpolated from the surrounding pixels). This is a signal during e.g. feature

extraction that no feature can be extracted from this area. For 3D spatial information system this regions should be labeled as "no information".

Following the idea of an "Informative Texture", at the beginning of the texturing for every face, a matrix representing its texture in certain resolution is created. Every element (pixel) of this matrix can get certain value. Thanks to this idea the invisible parts of the textures can be stored without changing the model geometry. In practice, these parts can be marked with e.g. value "-1" while the pixels of visible parts get intensity values between 0 and 255, computed from the input image. Further, it is possible to create a second matrix, kind of a map for the texture, where additional information about the texture is stored, e.g. number of the original frame from the image sequence or the quality.
The process of texture creation is presented in the Fig. 8.

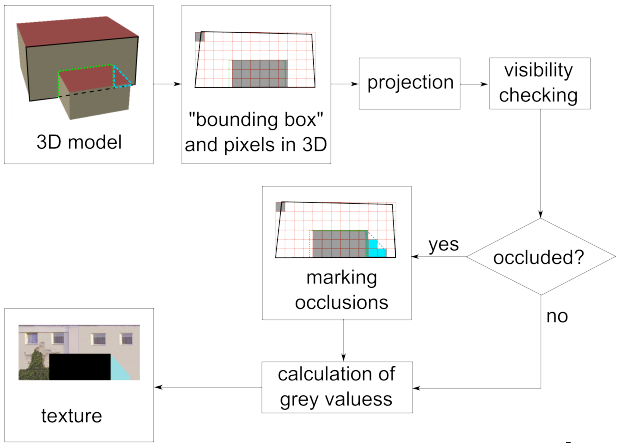

Figure 8. A workflow for texture extraction in one image

In some cases, the face is not rectangular. Then a bounding box should be created for this face. Based on the size of the face and potentially the best resolution of the input image, the pixel size needs to be determined. Position of every pixel in 3D should be calculated. Pixels which do not cover the face (bounding box can be bigger than the face) should be marked. Similarly the invisible parts should also be marked. According to orientation parameters of the camera, every pixel, except pixels marked in the previous step, has to be projected onto the image. Then the visibility checking is carried out. If an occlusion occurs, than occluded pixels should also be labeled. However, this label has other meaning than the previous one. This part of the texture can be found in other images and replaced. Finally, an intensity value is computed for every visible pixel.

7. CONCLUSIONS

In this paper, we discussed the possibilities of storing touching faces in the 3D city models and presented a method for detection of the common parts. Later we addressed the role of topology and semantics for the texturing process

We concentrated mainly on storage of information of invisible faces or their parts and introduction to the concept of "Informative Texture". *A priori* information of invisible faces improves the efficiency of the automatic texturing process. This knowledge about the invisible regions makes the evaluation of the completeness of the automatic texture mapping possible.

Moreover, classification of city models into semantic objects can be used to improve the texturing process. If the city model includes many objects types, only some of them can be filtered for texturing process (e.g. buildings). Thus, classification into further classes (semantic objects) like *WallSurface, RoofSurface* and *GroundSurface* etc. can also be helpful as well. Usually, the *GroundSurface* and all interiors (e.g. *interiorWallSurface*) as well as *ClosureSurface* cannot be textured at all. Thus these classes can be extracted in the beginning only to exclude them from texturing process. In most of the cases, the *RoofSurface* cannot be textured with terrestrial images. Therefore, when texturing with terrestrial data, *RoofSurface* can also be not taken into consideration. The *BuildingInstallation* class is used for building elements like balconies, chimneys, dormers or outer stairs, strongly affecting the outer appearance of a building (Gröger et al., 2008). Usually these objects also do not need to be covered with a specific texture. Instead of that, a specific appearance can be set according to the type of the object.

Finally, to summaries, this concept of an "Informative Texture" which can be used to store not only the appearance, but also some information about the face, e.g. information about the visibility of face, will have advantages in further applications. Efficiency in feature extraction from the "Informative Texture" will also improve because of the *a priori* information about the regions where no feature can be extracted.

8. REFERENCES

Abdelhafiz A, Niemeier W (2009), In: Godhoff F, Staiger R. Terrestrisches Laserscanning (TLS 2009). Yes, we Scan! Beiträge zum 91. DVW-Seminar am 19. und 20. November 2009 in Fulda. Schriftreihe eds DVW. Band 60/2009

Brenner C (2004) Modelling 3D objects using weak primitives. In: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXV, Istanbul, 2004.

Frueh C, Sammon R, Zakhor A (2004) Automated Texture Mapping of 3D City Models With Oblique Aerial Imagery, Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'04)

Gröger G, Kolbe T H, Czerwinski A, Nagel C (2008) OpenGIS® City Geography Markup Language (*CityGML*) Encoding Standard. OGC 08-007r1, Version: 1.0.0

Hoegner L, Stilla U (2009) Thermal leakage detection on building facades using infrared textures generated by mobile mapping. Joint Urban Remote Sensing Event (JURSE 2009). IEEE

Jibrini H, Paparoditis N, Deseilligny M P, Maitre H (2000) Automatic building re-construction from very high resolution aerial stereopairs using cadastral ground plans. 19th ISPRS Congress, Amsterdam, 16-23 July. IAPRS Vol. XXXIII

Karras G, Grammatikopoulos L, Kalisperakis I, Petsa E (2007) Generation of Or-thoimages and Perspective Views with Automatic Visibility Checking and Texture Blending. Photogrammetric Engineering & Remote Sensing, Vol. 73, No. 4, April 2007: 403-411

Kaul K, Bohn C A (2008) A genetic texture packing algorithm on a Graphical Processing Unit. 3ia2008 - The 11th International Conference on Computer Graphics and Artificial Intelligence

Kuzmin Y P, Korytnik S A, Long O (2004) Polygon-based true orthophoto gen-eration, International Archives of Photogrammetry, Remote Sensing and Spa-tial Information Sciences, 35 (Part 3): 529-531

Mayer H, Reznik S (2005) Building façade interpretation from image sequences. In: Stilla U, Rottensteiner F, Hinz S (Eds) CMRT05. IAPRS, Vol. XXXVI, Part 3/W24. Vienna, Austria, August 29-30, 2005

Mueller P, Zeng G, Wonka P, Van Gool L (2007) Image-based procedural model-ing of facades. ACM Trans. on Graphics (SIGGRAPH 2007) 26, 3, 85:1–85:9

Ripperda N, Brenner C (2007) Data driven rule proposal for grammar based fa-cade reconstruction. International Archives of Photogrammetry, Remote Sens-ing and Spatial Information Sciences, vol. XXXVI- 3/W49A, pp. 1-6, 2007.

Suveg I, Vosselman G (2004) Reconstruction of 3D building models from aerial images and maps. ISPRS Journal of Photogrammetry & Remote Sensing 58 (2004) 202– 224

Vosselman G (2002) Fusion of laser scanning data, maps, and aerial photographs for building reconstruction. In: Geoscience and Remote Sensing Symposium, IEEE International.

Wang Z, Schenk T (2000) Building extraction and reconstruction from lidar data. International Archives of Photogrammetry and Remote Sensing. Vol. XXXIII, Part B3. Amsterdam 2000

Zlatanova S, Rahman A A, Shi W (2004) Topological models and frameworks for 3D spatial objects. Computers & Geosciences 30 (2004) 419-428

Rottensteiner F, Briese Ch (2003) Automatic generation of building models from lidar data and the integration of aerial images. ISPRS, Vol. XXXIV, Dresden, 2003

ACKNOWLEDGEMENTS

The authors would like to thank FGAN-FOM, Ettlingen, for providing images of the flight campaign.