VOLUMINATOR 2.0 - SPEEDING UP THE APPROXIMATION OF THE VOLUME OF DEFECTIVE 3D BUILDING MODELS

M. Sindram[a], T. Machl[a], H. Steuer[b], M. Pültz[a], T. H. Kolbe[a]

[a] Chair of Geoinformatics, Technical University of Munich, 80333 Munich, Germany
[b] Clarendon Laboratory, Department of Physics, University of Oxford, Oxford OX1 3PU, United Kingdom

KEY WORDS: Volume Calculation, Semantic 3D City Models, Topological Errors, Octree, GIS, CityGML

ABSTRACT:

Semantic 3D city models are increasingly used as a data source in planning and analyzing processes of cities. They represent a virtual copy of the reality and are a common information base and source of information for examining urban questions. A significant advantage of virtual city models is that important indicators such as the volume of buildings, topological relationships between objects and other geometric as well as thematic information can be derived. Knowledge about the exact building volume is an essential base for estimating the building energy demand. In order to determine the volume of buildings with conventional algorithms and tools, the buildings may not contain any topological and geometrical errors. The reality, however, shows that city models very often contain errors such as missing surfaces, duplicated faces and misclosures. To overcome these errors (Steuer et al., 2015) have presented a robust method for approximating the volume of building models. For this purpose, a bounding box of the building is divided into a regular grid of voxels and it is determined which voxels are inside the building. The regular arrangement of the voxels leads to a high number of topological tests and prevents the application of this method using very high resolutions. In this paper we present an extension of the algorithm using an octree approach limiting the subdivision of space to regions around surfaces of the building models and to regions where, in the case of defective models, the topological tests are inconclusive. We show that the computation time can be significantly reduced, while preserving the robustness against geometrical and topological errors.

1. INTRODUCTION

Computer readable representations of 3D city models enable a plethora of different automatic analysis and simulations. These urban models play an important role in many tasks of different domains and are usually used for multifunctional purposes. These fields of usage include but are not limited to architectural design, urban and regional planning as well as various other domains like sociology and economy that all contribute to the field of smart cities. Due to their multi-faceted capabilities, 3D city models gain importance. They represent a virtual copy of the reality and thus can serve as a common information base and source of information for solving urban questions.

Nevertheless, these models are typically affected by topological and geometrical errors. For the case of urban 3D mapping, (Musialski et al., 2013) state that it is often difficult to acquire coherent and complete data of urban environments. For this reason, many available city models contain incomplete and defective geometries.

Current analysis tools in the field of geographic information systems (e.g. FME) fail at the computation of Key Performance Indicators like the total area or the volume of buildings if the given 3D models are topological incorrect. For instance, in order to compute the volume of a 3D building, the object itself has to be represented by a watertight boundary representation. However, many applications like e.g. the city-wide energy demand estimation as presented in (Kaden and Kolbe, 2013) require accurate data and in particular reliable volume values for further computations and simulations. In view of these facts, there is a clear need to overcome these topological and geometrical errors in an efficient and robust way.

Due to the great importance of the problem and the need of knowledge about the correct volume of buildings, different mitigation strategies have been developed to overcome these topological errors. One is to repair the given geometry by applying topological reasoning (cf. (Zhao et al., 2014), (Wagner et al., 2013), (Bogdahn and Coors, 2010)). Due to the complexity of the objects, this problem still exists; the problem of reconstructing the missing surfaces in a fully automatic way is not completely solved yet. Another approach is to use robust algorithms to deduce important information from the defective geometry. In many cases, these algorithms just cope with numerical errors (cf. (Zhang et al., 1995)). Additionally, many errors typically occurring in geographic models cannot be handled by these algorithms. For instance, (Biljecki et al., 2014) confirm the large impact of geometric errors on the result of an analytical volume computation. Even small geometric errors in the building models lead to large errors in the computed volumes.

Still there is a clear need for an accurate, robust and especially fast method to approximate the volume of topological and geometrical incorrect buildings. Due to the complexity of automatically repairing defective buildings there is a demand for methods finding geometrically and especially topologically incorrect regions of the boundary surfaces.

In the following Section a short review over the basic approach using a regular grid is given. Section 3 presents the extension of the method, especially focusing on the newly introduced spatial subdivision strategy. In Section 4 experiments illustrating the improved performance with regard to run-time and accuracy are shown. Before concluding the paper in Section 6, a theoretically derived estimate about the maximum to expect volume deviation is given in Section 5.

2. VOLUME CALCULATION BASED ON VOXELIZATION

In (Steuer et al., 2015), a robust method for approximating the volume of defective objects (e.g. building features in semantic 3D

city models) that follows adaptation strategies without repairing the objects has been presented. Since this method demonstrated a good accuracy albeit being too slow for analyzing large geographic areas, we extended this method. In this section we give a short summary of the method as proposed by (Steuer et al., 2015) before describing our improvements in the following sections.

In this approach, the bounding box of a building is divided into a 3-dimensional regular grid (see Figure 4). The cells of this grid represent so-called voxels (volume pixels). For each voxel of the grid it is tested wether the center point of the voxel is inside or outside the building. This test is based on idea of the point-in-polygon method of (Sutherland et al., 1974) and is adopted for the 3D case (cf. (Nooruddin and Turk, 2003)).

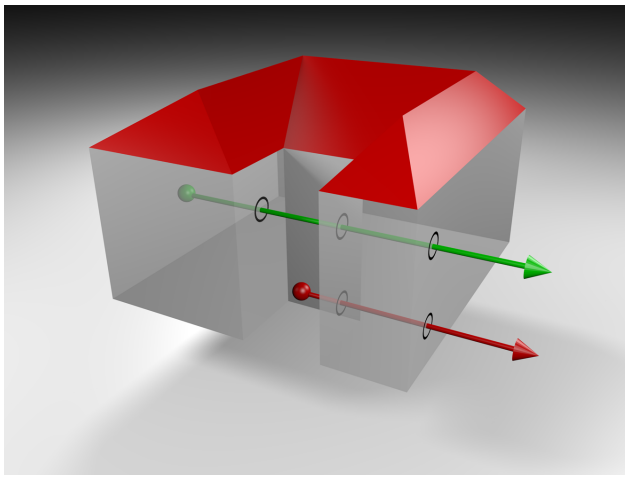

Figure 1: Basic point-in-polyhedron test with the raytracing approach. By counting the number of intersections of a ray starting from a point with all faces the inside/outside decision is made. The green ray intersects an uneven number of intersecting faces and thus the starting point of this ray is considered as lying inside the building. The red ray has an even number of intersections with the faces and thus the starting point is considered being outside the building.

Figure 1 illustrates the principle of the point-in-volume test. Starting from an initial point a ray is casted into an arbitrary direction and the number of intersections with all faces of the object is counted. As can be seen in the case of the green point, the ray has an uneven number of intersections with the faces and thus is interpreted as lying inside the building. In contrast, points at which the number of intersections of the ray with the surfaces is even are interpreted as lying outside the building. This case is illustrated with the red point and ray in Figure 1.

By summing up the volumes of all voxels which have been interpreted as lying inside the building, its volume can be approximated. The accuracy of this method depends strongly on the chosen voxel size in relation to the volume and geometric complexity of the building.

City models, which are provided e.g. by land surveying offices and local municipalities for large parts of Germany and cities like London, Singapore, New York, Berlin, Munich, and Dubai according to the CityGML standard, show amongst others the following topological and geometrical errors (cf. Figure 2):

- duplicated parts of the building's boundary surface
- missing parts of the building's boundary surface (e.g. roof, wall or ground surfaces)

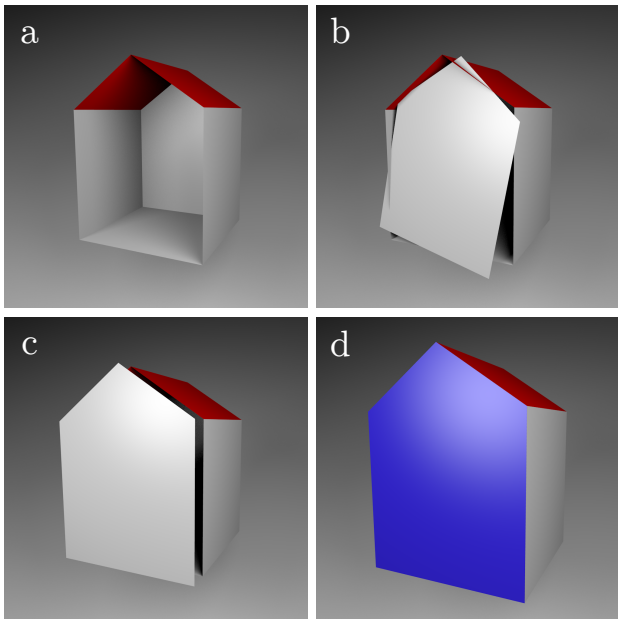

Figure 2: Geometrical and topological errors of volumetric objects in semantic 3D City models: missing (a), tilted (b), translated (c) or incorrectly oriented surfaces (d).

- wrong orientation of the building's boundary surface
- gaps/misclosures between parts of the building's boundary surface

The point-in-polyhedron test, in which only one ray is checked from a starting point leads in these cases to insufficiently accurate results. To make the process more robust with respect to the errors mentioned above, 5 further rays are generated starting from the center point of the voxel in parallel to the coordinate axes and checked for the number of intersections with the building's surfaces. Thus, a probability function can be obtained which can be used to make a decision whether the point lies inside or outside the building.

Figure 3 illustrates the voting approach for a defective buildings boundary surface. In this example the wall facing the observer is missing. If only one ray (red) pointing towards the missing wall surface is tested, the point/voxel would be assumed to lie outside the building model, even though this is not the case. Starting from the point 5 additional rays (green) are being tested for intersection with any parts of the buildings boundary surface. In this example all 5 vote for lying inside since they have an uneven count of intersections with the building model. It may therefore be assumed that at a ratio of 5:1 (probability 5/6) the point actually lies within the building.

It has been shown that in order to minimize the error very high voxel resolutions must be chosen for the approximation of the volume. As shown in Figure 4, large volumes which are likely to lie within the building are checked unnecessarily with very high resolutions neglecting the geometric structure of the model.

3. OPTIMIZED VOLUME CALCULATION

In this paper we introduce an extended approach of the Voluminator (Steuer et al., 2015), which reduces the number of voxels to be tested significantly. For this purpose, an octree approach (Meagher, 1982) with a modified subdivision step is im-

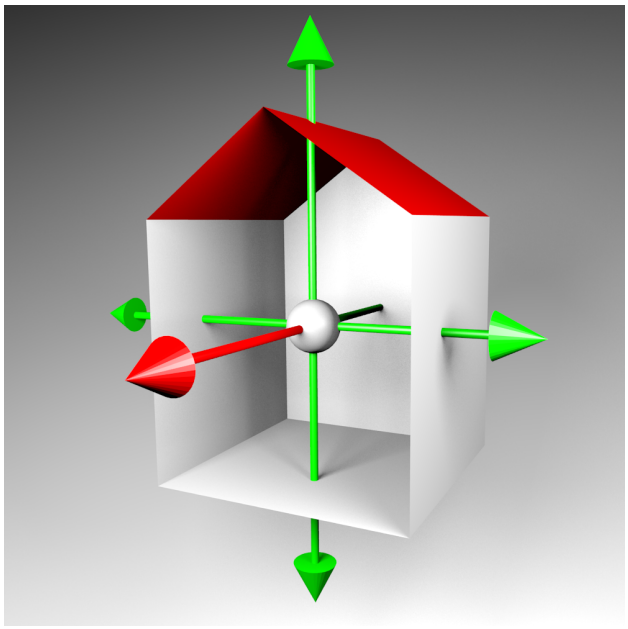

Figure 3: Majority decision of intersections to overcome topological and geometrical errors. There is one missing wall surface in this example. The red ray has no intersection (it votes for the point being outside). The other 5 green rays have an uneven number of intersections (voting for inside). The probability $P(inside) = 5/6$.

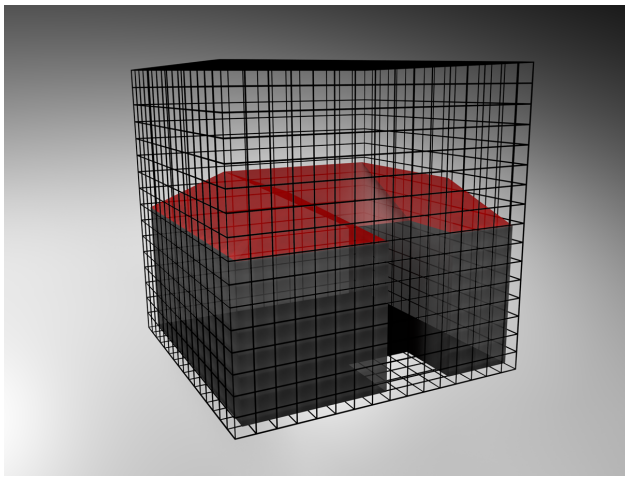

Figure 4: Voxel grid for the original volume approximation approach. The (non-minimal) bounding box of the building is subdivided into a regular grid with a constant cell size.

plemented. In Figure 5 the basic subdivision scheme is illustrated. In the first step, a cube is divided into 8 smaller cubes, which in turn are divided in the next step in the same manner. This can be repeated recursively until a desired and user defined minimum cube size is reached.

In order to apply this method to the problem of the volume calculation for defective building geometries an appropriate subdivision strategy needs to be developed, so that voxels are only subdivided when they come close to the surfaces of the buildings or when the inside/outside test is inconclusive. All voxels that completely lie within the building do not need to be divided furthermore and their volume needs to be summed up to the total volume of the building. Voxels lying completely outside, also do

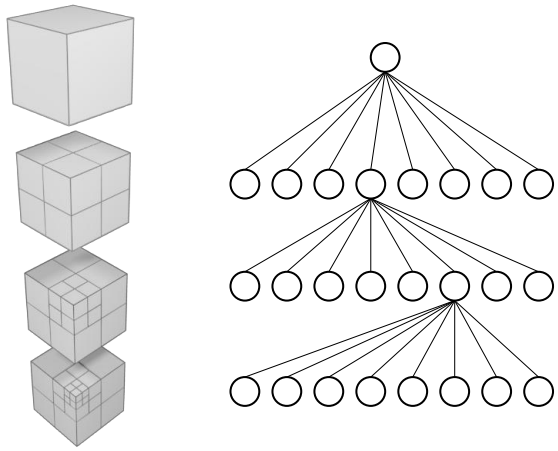

Figure 5: Core concept of the octree approach. The initial cube is recursively divided into further 8 sub-cubes until a predefined termination condition.

not have to be subdivided in the recursive function. This is illustrated in Figure 6. By using the octree approach the number of voxels that must be tested for lying inside the building can significantly be reduced. This leads to a reduction of total computation time as is demonstrated in Section 4.

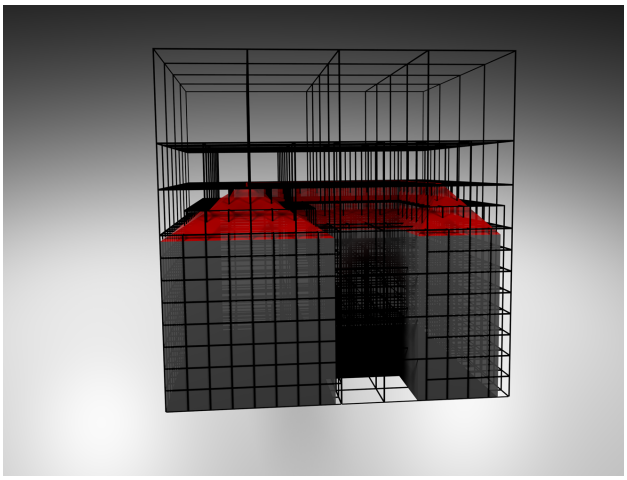

Figure 6: Generated voxel grid for the fast volume calculation method. Starting from an initial voxel the voxel is only subdivided if the voxel contains any part of the building's boundary surface.

3.1 Subdivision Strategy for the Octree Implementation

Unlike in the former approach, in which only the center point of the voxel is tested, each of the 8 vertices of a voxel has to be tested for lying inside or outside the building model. Figure 7 shows that the test for the center point of a voxel (left building in the picture) would vote for inside and the volume of the voxel would be added to the total volume of the building.

In this example (right building in the Figure), considering all vertices of the voxel on the other hand shows that two of the 8 vertices are not lying inside the building. For this reason, the voxel is divided further until either all vertices of the resulting voxels are completely inside or outside the buildings boundary surface or the minimum voxel size has been reached.

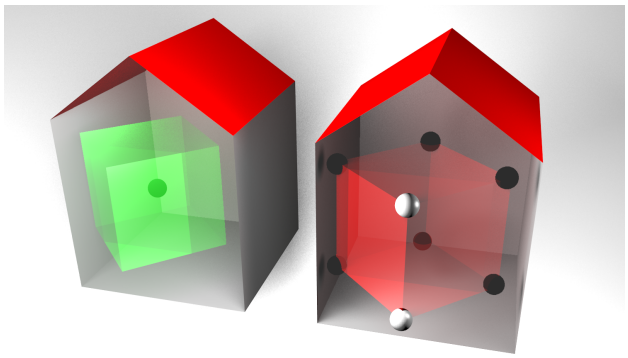

Figure 7: Comparison of different voxel-in-polyhedron approaches. The left building is illustrating the test for the center point of the voxel. This point decides that the voxel is inside the building and the voxel volume is added to the total volume of the building. The right building is illustrating the method for the octree approach. All 8 vertices are tested. If one or more points are outside of the building model the voxel volume is not complete inside.

In some cases it might happen that, although there are all vertices of the voxel outside of the building, the voxel is intersected by surfaces of the building. As illustrated in Figure 8 the tall building is partially enclosed by the blue voxel whose vertices all are lying outside. Without any additional tests this voxel would not be divided any further and its volume would be falsely counted as being outside of the building. In order to avoid this situation, the voxel boundaries are checked for intersections with the edges of the polygons making up the building.

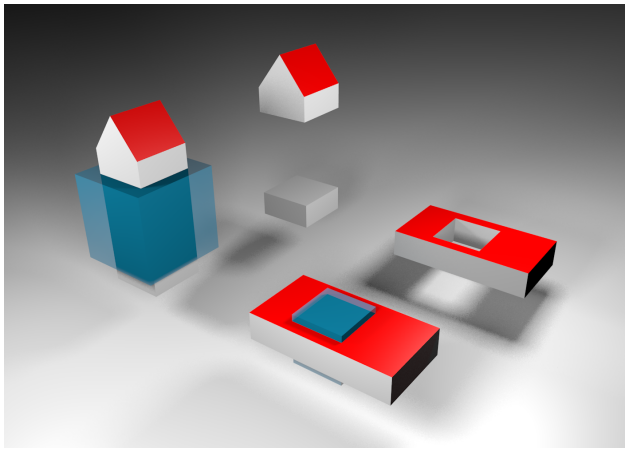

Figure 8: Intersection strategies for dividing the voxels. The tall building on the front left shows the intersection of the building edges with the blue voxel whose vertices are all outside of the building model. The flat building on the front right shows the intersection of the voxel edges with the building. The faulty results obtained when not using additional intersection tests are illustrated by the buildings in the rear part of the figure.

The flat building in Figure 8 is showing a second case of possible intersections. All vertices of the voxel lie outside the building's boundary but the voxel is not intersected by the edges of the building. However the edges of the voxels are intersecting the surfaces of the building. Thus, the previous intersection method would not catch this issue and the voxel would not be divided further (see rear flat building). For this reason, an additional intersection test is implemented which tests the intersection of the edges of the voxel with the surfaces of the building. This information can be derived from the information already obtained in the point-in-polyhedron tests for all vertices of the voxel. It has only to be checked, if the distance of the nearest intersection point is less than the edge length of the voxel. This is done for all the rays of the vertices of the voxel, which make up the voxel. Thus no separate intersection method has to be implemented, which would increase the total computation time significantly.

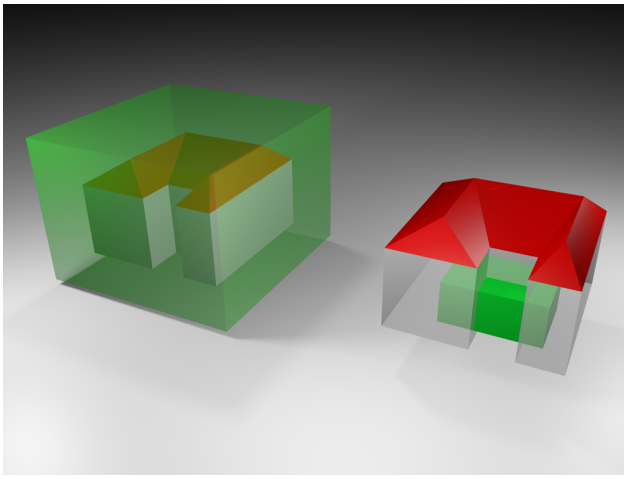

Figure 9: Subdivision strategies for voxels that completely contain the building (left building in the figure) and for voxels with all vertices voting for being inside the building (right building of the figure).

Figure 9 illustrates two different situations for voxels that need to be considered for the overall subdivision strategies. The left building in Figure 9 is showing a voxel that completely contains the building. To make sure that this voxel is divided into subvoxels an additional test is used, which verifies whether the vertices of the building are within a voxel or not. This case is not covered by the strategies depicted in Figure 8. The right building in Figure 9 is showing a voxel with all vertices lying inside the building but cut by a surface of the building. This case is covered by the intersection test between the edges of the voxel and the surfaces of the building.

3.2 Description of the New Algorithm in Pseudo Code

The complete algorithm for approximating the buildings volume was implemented in Java and is described in Algorithm 1 in pseudo code. The city model is imported from a CityGML file using the citygml4j API. Each building is translated to the coordinate origin in order to avoid numerical errors that might occur when using large coordinate values. In order to generate an equilateral voxel the initial voxel is spanned with the maximum edge length of the building's bounding box and and in order to avoid numerical instabilities buffered by an offset of $0.005m$. Each building object of the city model is treated individually.

The global variable $volume$ is assigned the value 0. The $volume$ of the building is calculated recursively by the function CALCVOLUME$(v, B, minVoxSize)$. As parameters of this function the initial voxel v, the polygons B of the building and the minimum voxel size $minVoxSize$ as a termination condition are passed. Within this function, the number of vertices of the voxel which are within the building is assigned to the variable $vPoints$ by calling NUMBERPOINTSINSIDE(v, B). The $boolean$ value $true$ is assigned to the variable $bPoint$ by calling the function AREBUILDINGPOINTSINSIDE(v, B) when a building point

lies within the voxel. If an edge of the building intersects the voxel the value $true$ is assigned to the variable $cutsVoxel$ by calling the function BUILDINGCUTSVOXEL(v, B). The variable $cutsBuilding$ is assigned with the value $true$ when an edge of the voxel intersects at least one building surface. Now it is evaluated whether $vPoints = 8$ AND $cutsVoxel = false$ AND $bPoint = false$. If this query returns $true$, then $volume = volume+$ GETVOXELVOLUME( ). In all other cases, if $vPoints > 0$ OR $cutsVoxel = true$ OR $bPoint = true$ OR $cutsBuilding = false$, the voxel is divided by the function DIVIDEVOXEL(v) into 8 smaller voxels v_i. For each of these voxels v_i the function CALCVOLUME($v_i, B, minVoxSize$) is called recursively if current $voxelSize > minVoxelsize$. Otherwise, the function CENTERPOINTINSIDE( ) checks if the center point of the building is inside or not. If the function returns true, then $volume = volume+$ GETVOXELVOLUME( ). This last step is done to avoid a general underestimation of the approximated building volume.

Algorithm 1 Fast volume calculation

```
volume ← 0.0
function CALCVOLUME(v, B, minVoxelSize)
 vPoints ← NUMBERPOINTSINSIDE(v, B)
 bPoint ← AREBUILDINGPOINTSINSIDE(v, B)
 cutsVoxel ← BUILDINGCUTSVOXEL(v, B)
 cutsBuilding ← VOXELCUTSBUILDING(B, v)
 if vPoints = 8 AND cutsVoxel = false AND bPoint =
false then
  volume ← volume + GETVOXELVOLUME( )
 else
  if vPoints > 0 OR cutsVoxel = true OR bPoint =
true OR cutsBuilding = false then
   voxels ← DIVIDEVOXEL(v)
   if voxelSize > minVoxelSize then
    for all v_i ∈ voxels do
     CALCVOLUME(v_i, B, minVoxelSize)
    end for
   else
    if CENTERPOINTINSIDE() = true then
     volume ← volume + GETVOXELVOLUME( )
    end if
   end if
  end if
 end if
end function
```

4. EXPERIMENTS

In this section, a statistical comparison of the new and the formerly published methods is conducted. The evaluations were performed on an Apple MacBook Pro 15" Retina with a 2.7 GHz Intel Core i7 processor and 16 GB of 1600 MHz DDR3 memory.

4.1 Datasets for Performance Tests

For testing and validating the developed tool and algorithm a defective CityGML dataset was created by using a topological and geometrical correct city model. The original dataset includes 192 buildings of the inner city of Munich. The correct building's volumes are within a range of 24 m^3 to 153900 m^3. Table 1 shows the distribution of the actual volumes of the buildings of the test data set. The average volume of the buildings is $6273m^3$. In Figure 10 the test area is depicted. It shows the area around the Frauenkirche in Munich with the Town Hall and has a typical inner-city building structure in addition to a high variation in the shape and volume of the buildings.

min	q25	q50	mean	q75	max
24	1117	3179	6273	5904	153900

Table 1: Quantiles of the building volumes [m^3] within the test area in Munich.

In a first step the original data of the city model was validated and tested for invalid solid boundaries using FME desktop before the actual volumes were computed. In a next step for each building one randomly selected polygon of the building boundary surface was deleted. The size of the removed surface was at least 5 % of the buildings total outer surface.

The defective buildings were written to a CityGML-file.

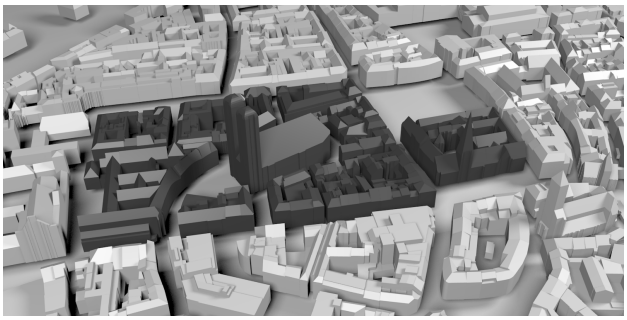

Figure 10: Test area in Munich. The darker area is indicating the test area used for the performance tests. This is the area around the Frauenkirche near to Marienplatz in Munich including the Town Hall.

4.2 Original vs. Optimized Approach

Our approach achieves a similarly high accuracy as the approach previously described in (Steuer et al., 2015) and even surpasses it and has a superior runtime. Figures 13 and 14 in the Appendix show the comparison of the actual to the estimated building volume using the naive and the optimized approach for defective solids in resolutions $2m$ and $0.25m$. The regression analysis shows for both methods a linear relationship between the measured and the approximated volumes with slopes close to 1 and an coefficient of determination (adjusted R-squared) close to 1 indicating a very good correlation and consistency of the results achieved.

The new approach allows to significantly reduce the computation times needed for approximating the defective building solids volumes. This is especially important when high accuracies are required. Figure 11 shows the required overall calculation times for estimating the 192 building volumes (barplot referring to the left y-axis) and the relative estimation error (boxplot referring to the right y-axis) as a function of the voxel resolution for the original (voluminator old; blue) and the new (voluminator 2.0; green) approach. The total computing time results from the sum of the computation times for the individual buildings. The relative error is the difference between the estimated volume v_{est} and the actual volume of the building v_{real} relative to the actual building volume v_{real}.

With an increasing resolution, the relative errors are approaching 0, the computation times increase disproportionally. The original approach tends to an underestimation of the building volumes, the new approach shows a more symmetrical distribution of the relative error around 0 for all resolution levels. In all resolutions the interquartile range of the relative error is smaller than the one of the original method. Extreme values of relative errors occur particularly in buildings with small volumes (v_{real}). For the original

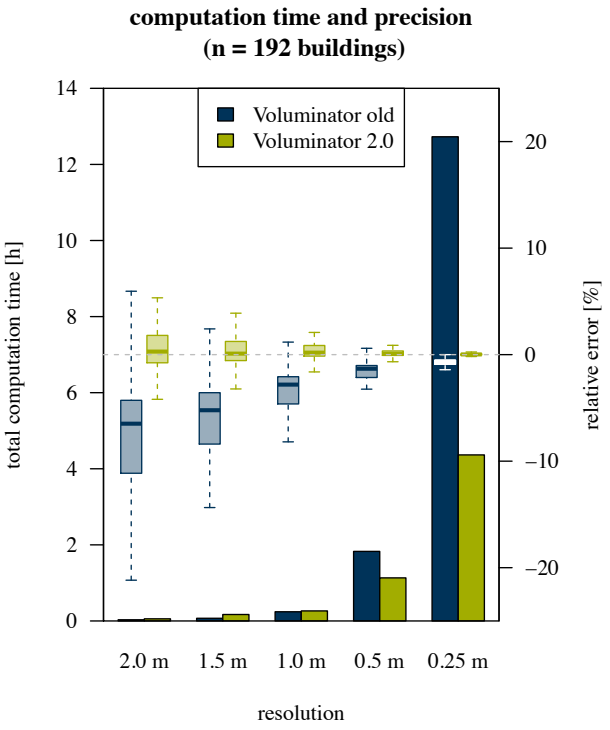

Figure 11: Accuracy (boxplot referring to the right y-axis) and total computation time (barplot referring to the left y-axis) for the original (blue) and the recursive (green) approach of approximating the building volume for defective solid boundaries ($> 5\%$ of the solid boundary surface is missing).

resolution	min	q25	q50	mean	q75	max
0.25 m	-6.0	-0.9	-0.6	-0.9	-0.5	1.1
0.5 m	-14.3	-2.2	-1.3	-2.0	-1.0	1.4
1.0 m	-33.3	-4.6	-2.8	-4.7	-2.1	1.8
1.5 m	-61.1	-8.4	-5.2	-8.1	-3.6	6.2
2.0 m	-59.0	-11.1	-6.5	-10.1	-4.3	13.8

Table 2: Summary of relative errors of the volume calculation for the original method. [%]

approach the relative error for 90 % of the buildings is within a range from -39.8 % (5 %-quantile) to -1.4 % (95 %-quantile) at a resolution of $2.0m$ respectively -3.3 % (5 %-quantile) to -0.1 % (95 %-quantile) at a resolution of $0.25m$. For the new approach 90 % of the buildings show relative errors in a range from -6.6 % (5 %-quantile) to 7.9 % (95 %-quantile) at a resolution of $2.0m$ respectively a range from -0.5 % (5 %-quantile) to 0.7 % (95 %-quantile) at a resolution of $0.25m$. Table 2 and Table 3 give a summary of the relative errors occurring in the old respectively new approach for the test dataset.

For low resolutions ($2.0m$ and $1.5m$), the computation times of the new approach are slightly higher than those of the old method. For resolutions higher than $1m$ the new approach shows significantly shorter computation times for estimating the volume of the 192 buildings within the test dataset. At a resolution of $0.25m$ the total computation time of the new approach is one-third of the computation time of the old method.

resolution	min	q25	q50	mean	q75	max
0.25 m	-2.8	-0.0	0.0	0.0	0.1	2.3
0.5 m	-4.7	-0.1	0.1	0.3	0.3	8.1
1.0 m	-14.3	-0.1	0.2	0.7	0.9	15.3
1.5 m	-15.5	-0.6	0.1	0.5	1.2	18.2
2.0 m	-23.8	-0.8	0.3	0.6	1.8	42.8

Table 3: Summary of relative errors of the volume calculation for the new method. [%]

5. POTENTIAL ACCURACY

By approximating the volume of buildings with voxels deviations from the true volume will occur. The maximum and minimum potential deviation is described below. We found that this error is highly dependent on the ratio of building volume and the selected minimum voxel resolution. In addition, the geometric shape complexity of the building plays a major role. The following methods are describing the potential calculation error for the voxel approach (octree) excluding geometrical and topological errors.

The maximum overestimated or underestimated building volume $Vol_{maxError}$ is calculated by the following sum:

$$Vol_{maxError} <= Vol_{maxSurface} + Vol_{maxEdges} \quad (1)$$

where $Vol_{maxEdges}$ is the volume depending on the length of the edges of the building

$$Vol_{maxEdges} = \frac{\sum_{i=1}^{n} ring}{2} length_{voxel}^2 \quad (2)$$

where $ring$ is the length of a single ring which borders a polygon of the building model and $length_{voxel}$ is the length of one edge of a smallest voxel. $Vol_{maxSurface}$ is the volume depending on the surfaces of the building

$$Vol_{maxSurface} = A_{building} \frac{length_{voxel}}{2} \quad (3)$$

where $A_{building}$ is the total area of the building's surfaces.

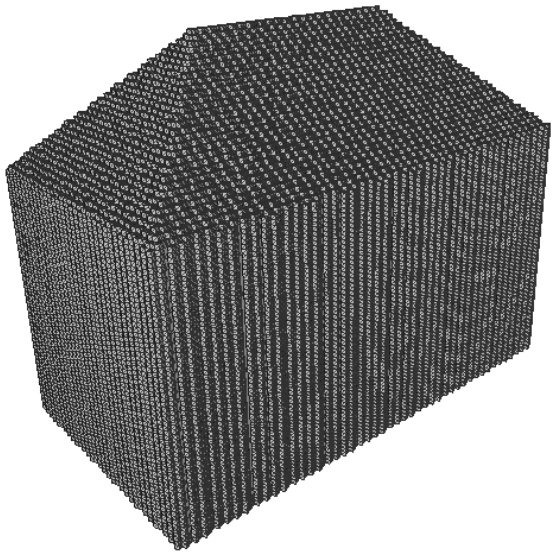

Figure 12: Voxelized test building for the accuracy test. The building was tested with a voxel resolution of $0.05m$. For the visualization this building is depicted in a resolution of $0.5m$

We have calculated the building illustrated in Figure 12 with a voxel resolution of $0.05m$. The building has a real volume of $7159.8m^3$ and a total surface area of $2210.8m^2$. The length of exterior rings of all polygons is $1263.4m$. By using the formulas (1), (3) and (2) these values lead to a maximum value of $56.58m^3$ for calculating the maximum deviation volume $Vol_{maxError}$ and to a relative error of 0.79 % of the maximum that can be underestimated or overestimated. The calculation of the volume with a resolution of $0.05m$ has an approximated volume of $7161.73m^3$, which is a total deviation of $1.93m^3$ and corresponds to a relative deviation of 0.027 %.

6. CONCLUSION AND OUTLOOK

This paper presents an approach for approximating the volume of topologically and geometrically incorrect building models from semantic 3D city models. The approach picks up the idea as shown in (Steuer et al., 2015) and extends it significantly using a spatial subdivision schema based on the octree approach with a novel subdivision strategy. The new approach allows significant improvements regarding computation time needed for high accuracy voxel resolutions in the calculation of the volume.

Due to the fact that semantic 3D city models according to the CityGML standard are mostly available in the Level of Detail 2 (LoD2) we performed our experiments with LoD2 datasets. In the future, a more detailed evaluation of the effects on computation time and accuracy when using different LoDs needs to be done.

First tests conducted on a real-world dataset with synthetically introduced topological errors show a clear reduction of computation time needed for approximating the volume of defective building models with simultaneous improvements regarding accuracy of the approximated volume. At a resolution of 1 m the new approach is as fast as the former method as shown in (Steuer et al., 2015), at a resolution of $0.25m$ the overall computation of the new method needs only a third of the time compared to the old approach. Especially for large areas and high accuracy requirements the approach presented here provides a significant improvement.

Future research should evaluate the presented approach taking into account further kinds of potentially occurring errors like tilted, translated or wrong oriented parts of a buildings boundary. Furthermore intersection tests could be conducted using the computer's Graphics Processing Unit (GPU).

Moreover, research could evaluate the potential of the shown approach for being used to detect and repair defective building boundaries.

REFERENCES

Biljecki, F., Ledoux, H. and Stoter, J., 2014. Error propagation in the computation of volumes in 3d city models with the monte carlo method. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Proceedings of the ISPRS/IGU Joint International Conference on Geospatial Theory, Processing, Modelling and Applications. Toronto, Canada.

Bogdahn, J. and Coors, V., 2010. Towards an automated healing of 3d urban models. In: Proceedings of international conference on 3D geoinformation. International archives of photogrammetry, remote sensing and spatial information science, Vol. 38, Citeseer, p. 4.

Kaden, R. and Kolbe, T. H., 2013. City-wide total energy demand estimation of buildings using semantic 3d city models and statistical data. In: Proc. of the 8th International 3D GeoInfo Conference, Vol. II-2/W1.

Meagher, D., 1982. Geometric modeling using octree encoding. Computer Graphics and Image Processing 19(2), pp. 129 – 147.

Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., Gool, L. and Purgathofer, W., 2013. A survey of urban reconstruction. In: Computer Graphics Forum, Vol. 32number 6, Wiley Online Library, pp. 146–177.

Nooruddin, F. S. and Turk, G., 2003. Simplification and repair of polygonal models using volumetric techniques. Visualization and Computer Graphics, IEEE Transactions on 9(2), pp. 191–205.

Steuer, H., Machl, T., Sindram, M., Liebel, L. and Kolbe, T. H., 2015. Voluminator - approximating the volume of 3d buildings to overcome topological errors. In: F. Bação, M. Y. Santos and M. Painho (eds), AGILE 2015 - Geographic Information Science as an Enabler of Smarter Cities and Communities, Lecture Notes in Geoinformation and Cartography, Springer, Lisbon, Portugal, pp. 343–362.

Sutherland, I. E., Sproull, R. F. and Schumacker, R. A., 1974. A Characterization of Ten Hidden-Surface Algorithms. ACM Comput. Surv. 6(1), pp. 1–55.

Wagner, D., Wewetzer, M., Bogdahn, J., Alam, N., Pries, M. and Coors, V., 2013. Geometric-semantical consistency validation of citygml models. In: Progress and New Trends in 3D Geoinformation Sciences, Springer, pp. 171–192.

Zhang, Z., Deriche, R., Faugeras, O. and Luong, Q.-T., 1995. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Artificial intelligence 78(1), pp. 87–119.

Zhao, J., Stoter, J. and Ledoux, H., 2014. A framework for the automatic geometric repair of citygml models. In: Cartography from Pole to Pole, Springer, pp. 187–202.

ACKNOWLEDGEMENTS

For comparing the runtime performance and accuracy of both approximation approaches we use the real world CityGML dataset from Munich that was thankfully provided by the "Landesamt für Digitalisierung, Breitband und Vermessung (LDBV)" in Bavaria, Germany. LDBV holds the copyright for this dataset.

APPENDIX

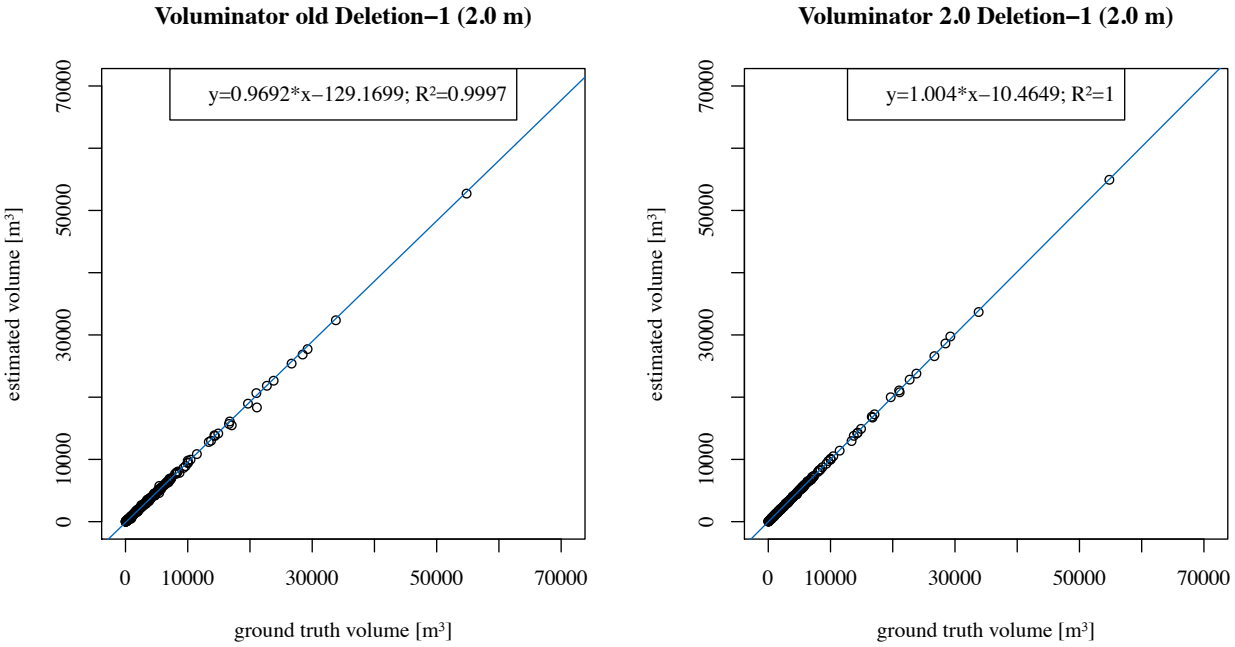

Figure 13: Scatterplot of the actual building volume and the approximated building volume for buildings with defective solid boundaries ($> 5\%$ of the solid boundary is missing; n = 192 buildings) using the old voluminator (left) and the recursive voluminator approach at a resolution of $2m$. For the old voluminator approach the overall computation time was 1.85 minutes for the octree approach the overall computation time was 3.36 minutes.

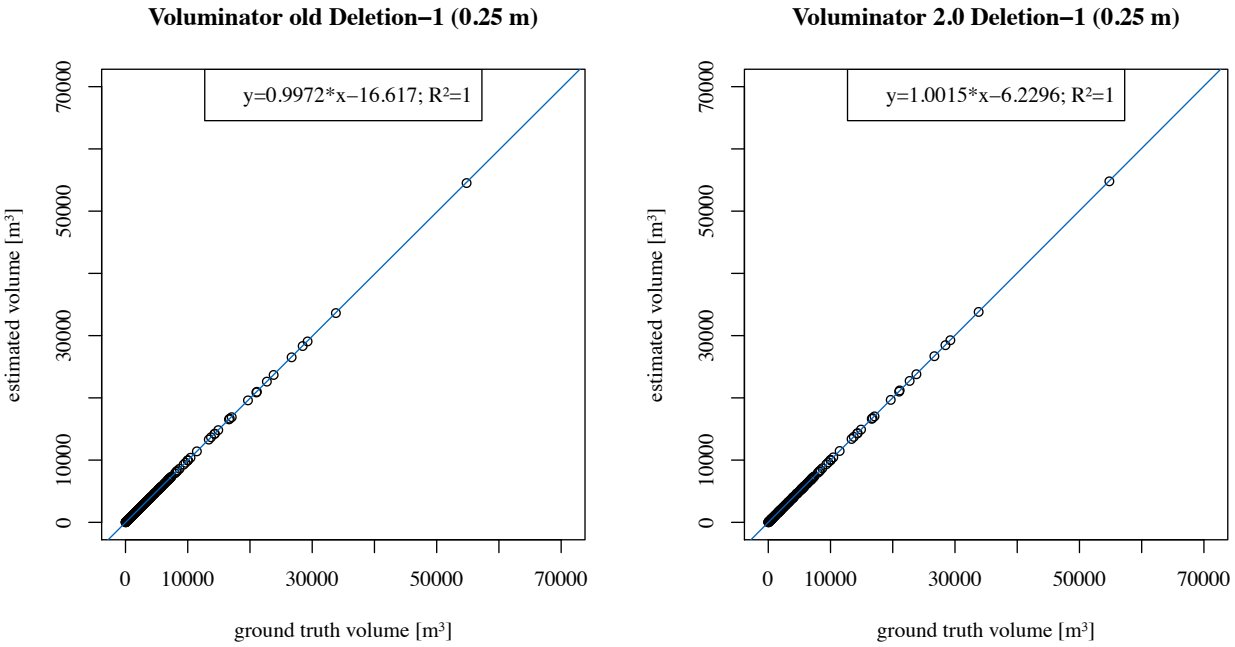

Figure 14: Scatterplot of the actual building volume and the approximated building volume for buildings with defective solid boundaries ($> 5\%$ of the solid boundary is missing; n = 192 buildings) using the old voluminator (left) and the recursive voluminator approach at a resolution of $0.25m$. For the old voluminator approach the overall computation time was 763 minutes for the octree approach the overall computation time was 262 minutes.