FICUP
An International Conference on
Urban Physics
B. Beckers, T. Pico, S. Jimenez (Eds.)
Quito – Galápagos, Ecuador, 26 – 30 September 2016

Skyline-control Based LoD Generation for Solar Analysis in 3D Cities

Gonzalo Besuievsky[1], **Benoit Beckers**[2], and **Gustavo Patow**[1]

[1]Geometry and Graphics Group, University of Girona, Spain

[2]Urban System Engineering Department, Compiegne University of Technology, France

Keywords: Level of Detail, 3D City model, Solar simulation

Abstract. *Solar simulation for 3D city models may be a complex task if detailed geometry is taken into account. For this reason, the models are often approximated by simpler geometry to reduce their size and complexity. However, geometry details, as for example the ones provided in a roof, can significantly change the simulation results if not taken into account. The classic solution to deal with a detailed city model on a district scale is to use a Level-of-Detail (LoD) approach for geometry reduction.*

In this paper, we present a new LoD method for 3D city models for performing accurate solar simulation with detailed geometry models. Given a Point of Interest to analyze, the method works by automatically detecting and preserving all roofs that really have significant impact on the simulation and simplifying the rest of the geometry. We perform a test for a detailed district model showing that for this case our method can reduce the geometry size to 5% *of the original model, preserving almost the same accuracy.*

1 Introduction

The estimation of the solar potential and the access to sunlight in urban areas is a very important issue in building energy performance. The computation accuracy for solar simulation is directly influenced by the urban geometry, both at the neighborhood and at the building scale. Because of this, a well-defined geometrical model of the urban environment is mandatory for accurate analysis and assesment.

One of the main difficulties concerning geometry detailed models is the amount of data to deal with, affecting the memory storage and the time processing. The classical solution is to introduce Levels of Detail (LoD) into the model in order to simplify it. Most of the simplification methods for simulation work by rough approximating the buildings to simple shapes like bounding boxes and basic roof geometry. As a consequence, building elements that influence the simulation, like a roof water tank or an antenna, may propagate errors in a lighting assessment or in a Sky View Factor (SVF) study.

In this paper, we propose a new LoD method for urban models. Our main goal is to provide models for performing solar simulation at full geometry resolution. Given a 3D detailed city model and a region of interest, the method can obtain a simplification model feasible for computing solar radiation for that region. By computing the skyline impact over the urban geometry, the method preserves all important roof details. A fast ray-casting engine is used against a rough approximation of the model to decide which geometry level should be instantiated.

The main contribution of our work is that it provides a fast, accurate and automatic model simplification for dealing with solar simulation in detailed urban models. We tested our reduced model results to perform an annual daylight simulation. Our results show that, for the district model used, we can obtain accurate values with the model reduced to 5% of the original size, improving considerably the execution time.

2 Previous Work: LoD in Urban models

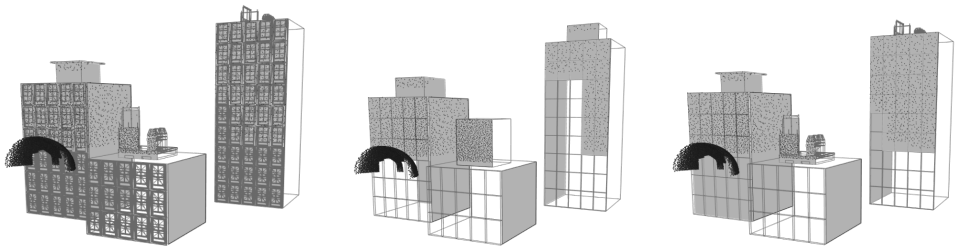
Level-of-detail techniques have been largely developed in computer graphics with the aim of reducing geometry since the first work presented by [Clark, 1976], followed by the seminal work by [Luebke and Erikson, 1997]. The interested reader can refer to the book at [Luebke et al., 2002] for a more exhaustive and complete survey of general LoD techniques.

As far as urban models are concerned, a few LoD proposals come from the use of a semantically well-defined dataset structure, as for example the CityGML schema as defined at [Kolbe, 2009]. CityGML differentiates between five consecutive LoD-levels, where objects become more detailed with increasing LoD

regarding both geometry and thematic functionality differentiation. They range from a coarsest level in LOD0, which is essentially a two and a half dimensional digital terrain, to full interior structures like rooms, stairs, and furniture in the top level LOD4. In that approach, the different levels of detail are pre-made and not adapted for specific simulation purposes.

For procedural modeling, [Parish and Müller, 2001] presented an initial proposal intended for city generation based on the L-system recursive nature. Automatic LoD-generation is obtained by starting from the building envelope as axiom, and the output of each rule iteration represents a refining step in the building generation. Although it is simple and automatic, this approach does not provide control on geometric building details. Recently, new approaches have been proposed to integrate LoDs mechanism in the procedural processing. In [Besuievsky and Patow, 2013a], the authors developed a rewriting method of the rulesets for the buildings for further replacing the geometric operators, which produces the right level of detail for each asset according to some user-defined criteria. In [Besuievsky and Patow, 2013b], they propose a highest level of detail by enabling selection, from entire buildings up to whole blocks, for geometric reduction. These works focus more on solving rendering problems, whereas in our approach we target more on the model preparation for simulation analysis.

Concerning solar energy simulation, defining the optimal LoD at the neighborhood scale is not a simple problem and most of the approaches are taken from an empirical perspective. In [Rodriguez et al., 2012], a study of the sensitivity of the geometry used is carried out taking into account the solar flux computation, where different levels of detail elements (windows and roofs) are evaluated for a neighborhood-scale model. In [Besuievsky et al., 2014], a configurable LoD system based on procedural models is presented for daylight simulation. The system allows the configuration of different criteria for approximating the full geometry for different computations. [Biljecki et al., 2014] provided a formal and consistent framework to define discrete and continuous levels of detail (LODs), by determining six metrics that constitute it, and discussed their quantification and relations. Following this initial work, in [Biljecki et al., 2015], they studied the propagation of positional error in 3D GIS, and applied this computation to the estimation of the solar irradiation of building roofs. Next, [Biljecki et al., 2016] studied the variety of LOD1 and LOD2 geometric references that are commonly employed in LoD models, and performed numerical experiments to investigate their relative difference when used as input for different spatial analyses of a 3D building model. Their results show that two different models generated from different geometric references, but with the same LoD, may yield substantially different results when used in a spatial analysis.



(a) Full model: 85.19% (b) Bbox: 83.32% (c) LoD propossal: 85.19%

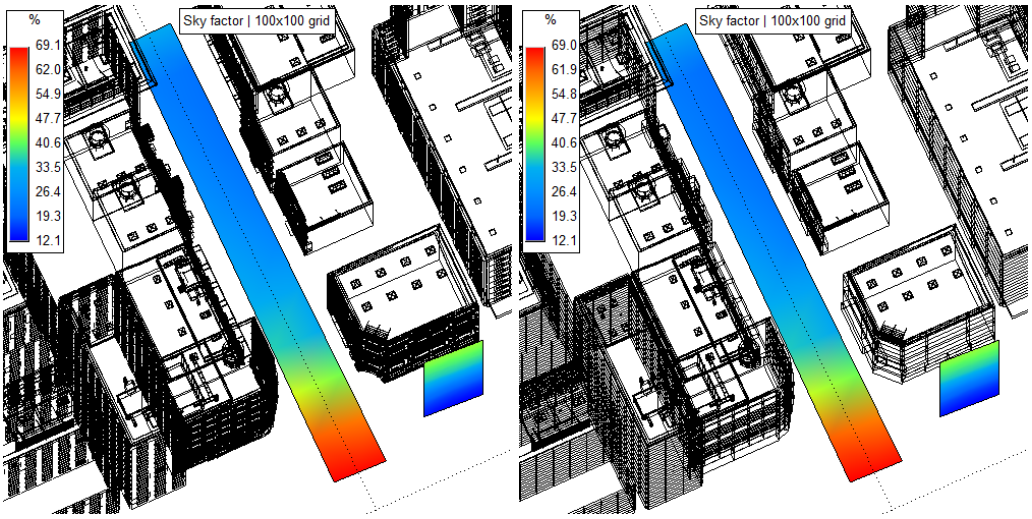
Figure 1: Sky View Factor test for three different model representations.

3 Skyline Approximation Foundation

In this section, we analyze the influence that the city skyline details have on the simulation. For this analysis, we use the Sky View Factor (SVF) as a metric for studying the influence of the roof details on a given building. The SVF, currently used in daylight assessment, is defined as the percentage of sky visible from a surface, taking into account the angle of inclination to the sky vault. It is a pure geometrical parameter that has a physical meaning. Fig. 1 shows three simple buildings modeled at different detail resolutions: full detail resolution, bounding box approximation represented by the envelope of each building, and a mix of the previous ones, putting full resolution only to the roof. We compute the SVF for a virtual point in front of the buildings, getting the same value for both the full model and for the last approximation. We noted that most of the relevant details for solar analysis belong to the skyline given by the roof, and that could be obtained from the silhouette. We complete our test by computing the average SVF with Heliodon [Beckers and Masset, 2008] on two virtual surfaces on a 3D city model: an horizontal plane at the street level and a vertical plane in front of the facade of a building (see Fig. 2). By comparing the results of the bounding box approximation with full roof resolution city against the full model, we observe a relative error lower than 10^{-2} in both cases. Considering that the number of polygons used in the approximation representation is 10 times smaller than the original model, we tuned our LoD proposal following these observations.

4 LoD Generation Method

Given a 3D city model, the method works in two steps. In a first pre-processing step, all geometry assets (like windows, balconies or roofs) are replaced by their respective bounding boxes. We call the resulting model the *rough* model. In the second step, a LoD operator decides, from a given Point of Interest (POI), the simplification level according to a defined criteria by intersecting a bundle of rays



(a) Full model: Hplane=34.9%, Vplane=28.2%

(b) LoD propossal: Hplane=34.6%, Vplane=28.1%

Figure 2: Average Sky View Factor for horizontal and vertical planes in two different representations of a 3D city model.

from the POI with the rough model. The rest of this section describes the method in detail.

4.1 Criteria

For our geometry selection/replacement to work, we designed a simple criteria based on the observation that the relevant details for solar computation are the ones that belong to the building silhouettes (see Section 3). For the rest of the geometry, in most cases it will be enough to approximate it by its visible bounding box planes. We also noted that silhouettes are more relevant at the city skyline. Our LoD model define then three particular geometry levels:

- **Level 0:** The geometry is omitted from the model
- **Level 1:** The geometry is instantiated as its bounding box
- **Level 2:** The geometry is represented in full details.

4.2 LoD algorithm

The algorithm that sets the corresponding simplification levels using the previous criteria is presented at Algorithm 1. The algorithm takes as input a fully detailed city model and a POI, to obtain the simplified representation. After generating the bounding boxes for the rough model (method $generateBBox$, see Section 4.3) it

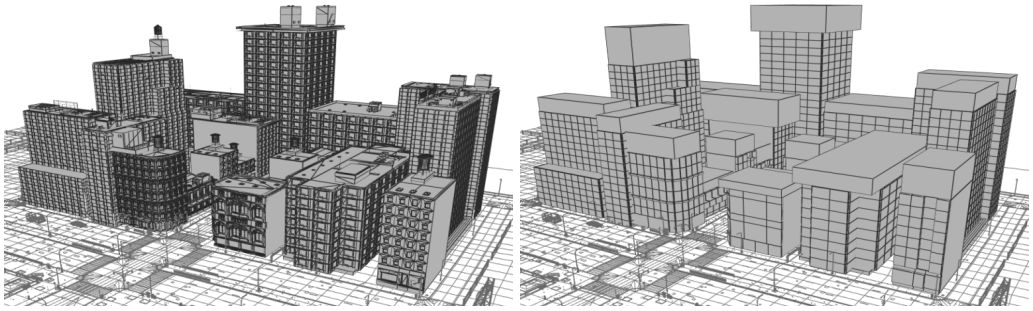
casts a bundle of rays from the POI (Section 4.4). The algorithm automatically replaces all affected roofs with geometric details (method $isSilohuette$, see Section 4.5), and discards all geometry not participating in the SVF computation (level 0 in the above criteria).

Algorithm 1 LoD Algorithm

Require: $3DCM$: 3D City Model
Require: POI: Point of Interest
  $P \leftarrow generateBBox(3DCM)$
  $rayCastBundle(POI, P)$
  for each p_i in P **do**
    $n \leftarrow p_i.hits()$
    if $n = 0$ **then**
      $p_i.setLevel(0)$
    else
      if $p_i.isRoof()$ **and** $p_i.isSilohuette()$ **then**
        $p_i.setLevel(2)$
      else
        $p_i.setLevel(1)$
      end if
    end if
  end for

4.3 Bounding box Building Generation

The bounding box city model is generated in a pre-processing step. It is assumed here that the building models are well structured into assets that represent the basic constructive elements, like windows, doors, balconies or roofs. In general, these elements concentrate most of the geometric complexity. These assumptions about the building structure are reasonable, considering that the 3D building models used in practice are usually generated by Architectural CAD systems that already provide these structures. However, if the models are given as a raw polygon soup, as for example when obtained from acquisition techniques (e.g., with LIDAR), a detection and classification step should be applied to structure the model. Our current implementation is based on procedural urban models [Müller et al., 2006], which implicitly provide this organization: the buildings are generated from the iterative appliction of a set of rules, each one resulting in a product subject to further processing by successive rules. In this model, the constructive geometric elements are attached to the model in a final rulesset application to the final products by use of the *Insert* command.



(a) Full model (b) Bounding box approximation model

Figure 3: The urban model.

Here we describe the algorithm for procedural models, and a similar procedure should be followed for other kind of 3D models after a structuring stage, as mentioned above. Given a detailed procedural urban model, the bounding box generation algorithm iterates over all products of each building detecting the *Insert* command placements. For each asset, a bounding box of the corresponding insertion is then computed. Then, new building instances are generated, replacing all geometry insertions by their corresponding bounding boxes (see Fig. 3(b)).

4.4 Bundle distribution

We generate the ray bundle distribution by randomly sampling the sky vault using the equal-area cell distribution method described in [Beckers and Beckers, 2014]. By using such specific distribution, we can quickly approximate the SVF directly by just counting the number of impacts with the city model from a given POI.

4.5 Silhouette detection

The problem for detecting the silhouette of an object has been solved by several geometric techniques. One of the widest uses is for visualization in non-photorealistic rendering [Markosian et al., 1997]. In our case, we can simplify the problem with the observation that we are only interested in impact detection in the upper hemisphere of directions with the city skyline and that all candidates to analyze are rectangles produced by the bounding box building approximation described above.

With these considerations, we use the bundle of rays to build a routine for silhouette detection. After casting rays from a given POI, we can split the rays of the obtained distribution into two sets: the ones that impact the geometry and the ones that go straight to the sky. We used this last set to decide if a polygon roof belongs to the silhouette or not: our algorithm works by projecting such candidates to the

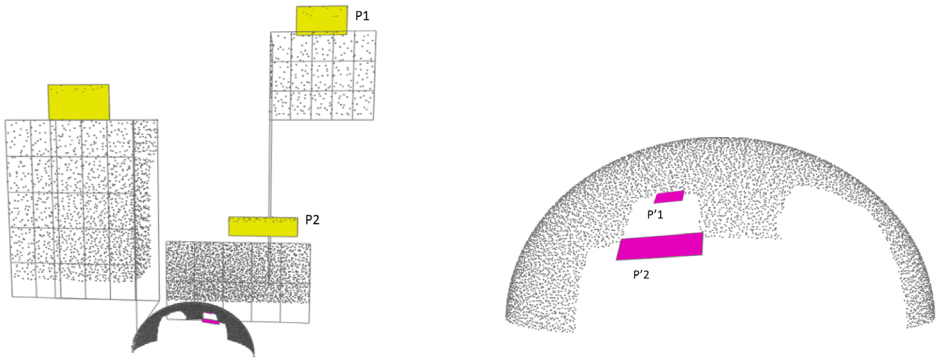


Figure 4: Silohuette algorithm. Top candidate polygons are projected onto a sphere center at the POI (left). By comparing their projection to the sky points, P1 is classified as silhouette and P2 not (right).

POI distribution bundle of rays. When analyzing the projection of the top line of the polygon, if there is any sky point of the distribution close enough (i.e., whose distance is below a given user-defined threshold) to the line, then we can conclude that the polygon is part of the skyline, otherwise it is not. Fig. 4 shows a graphic example using the same buildings models of Fig. 1. The yellow top polygons of the left are candidates for being silhouette from a given POI. Projecting P1 and P2 to the bundle distribution of the sky points, our algorithm discards P2 from the skyline and includes P1.

4.6 Model Aggregation

The final step of the method is to merge all instances of geometry obtained in Levels 1 and Level 2 into a single model. The resulting geometric model is then exported for being used by any daylight simulation application.

5 Results

In this section, we analyze the performance of our LoD approximation. First of all, it should guarantee that the solar simulation will not be affected by the simplification method. We use the SVF computation to validate our approximation. To show its potential use, we compute the solar impact at the windows of a facade for a given model. We also analyze the importance of using detailed geometry by comparing the simulation result to a rough approximation model.

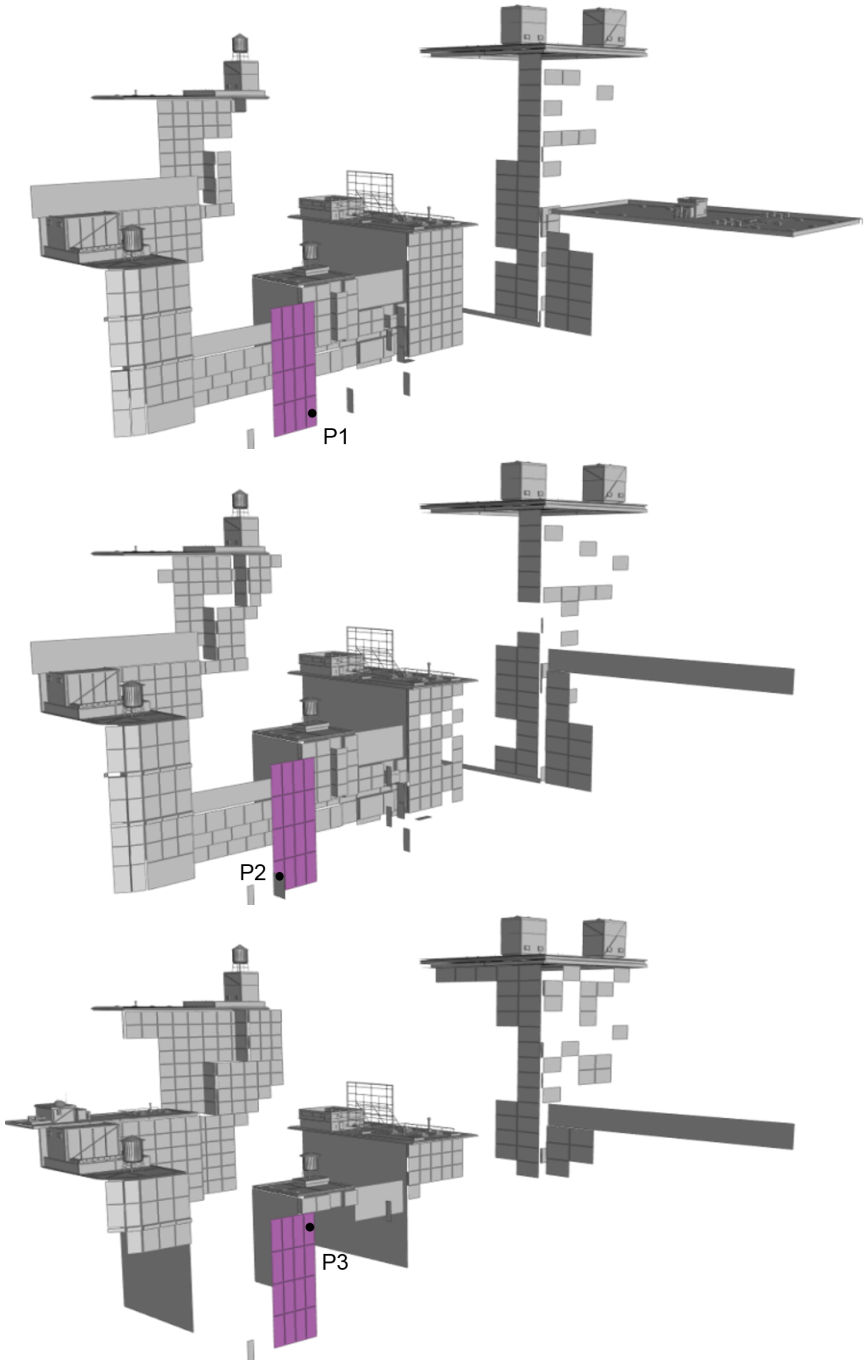


Figure 5: Simplified model for 3 window selected in the facades. The POI is set at each center of the corner's window of the facade.

5.1 LoD from POI

To analyze the accuracy of our method, we computed the SVF using a detailed model composed by 297K polygons (see Fig. 3(a)). We tested our LoD model for a virtual facade. We generated the LoD model for the corner windows of the facade, and the POIs were set at the center of each window. For the resulting models (see Fig. 5) we computed the SVF for both the full and the approximated models. The relative error is computed as $E_r = \|SFV_{Full} - SVF_{LoD}\|/SFV_{Full}$ (see Table 1). On average, the model is reduced to 16.8K polygons, which represents around only 5% of the full input model. The processing time to generate the reduced models is around 10 seconds. Our implementation is written with Python routines using SideFX's Houdini [Side-Effects-Software, 2015] as a develpment platform.

	P1	P2	P3	P4
SVF_Full	12.59	15.02	24.57	25.59
SVF_App	13.31	14.70	26.60	24.22
Error	0.027	0.021	0.082	0.053
# Polygons	20039	16881	19201	11388

Table 1: Sky View Factor results comparison for the reduced models.

5.2 Solar Impact Computation

To test the usability of the models, we performed a solar impact computation for the whole year using Heliodon [Beckers and Masset, 2008]. The model is localized in Barcelona, Spain. We compute the total daylight hours each window receives using a 15-minute time step. Table 2 shows the resulting hours using the full model and each LoD model for the corresponding window at the corners facade (see Fig. 6). We observe that the relative error in average is around 2%, which is significantly small. The savings in the simulation processing time are also significant: for the full model, the simulation takes around 5 minutes for each window, while the same simulation takes only 15 seconds using the approximated models. Finally, we did the same simulation with the bounding box building models (Fig. 3(b)) to analyze the relevance of using geometric details. In this case, the error, on average for the four windows, is around 13%, too large for accurate simulation purposes. This demonstrates, again, the validity of our assumption, that details should be preserved, but only for the skyline silhouettes.

	P1	P2	P3	P4
Daylight_Full (h)	889.1	923.6	1272.6	1357.8
Daylight_LoD (h)	875.8	932.9	1230.5	1299.7
Daylight_Bbox (h)	784.9	774.9	1084.3	1142.6
Error(Full-LoD)	0.015	0.009	0.033	0.042
Error(Full-Bbox)	0.117	0.191	0.033	0.188

Table 2: Daylight hours comparison for the whole year.

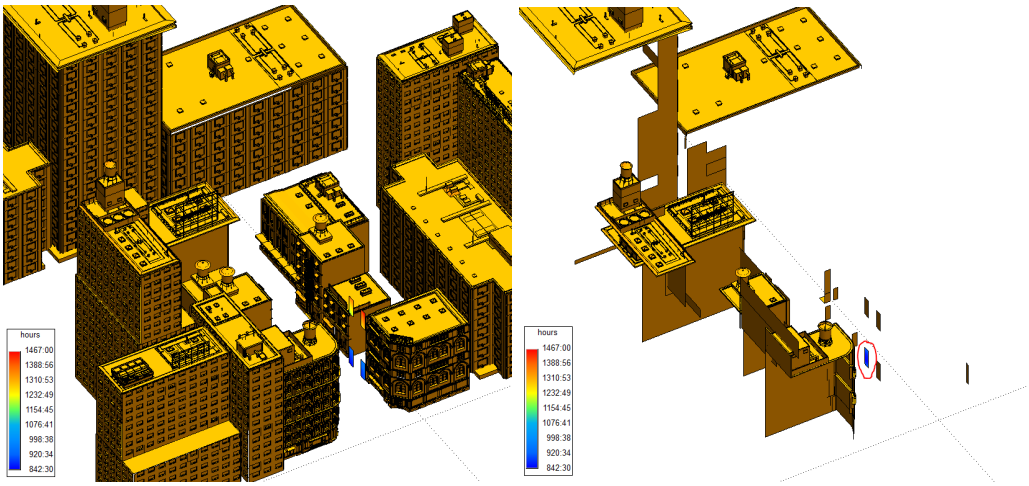


Figure 6: Year-daylight simulation: Full detailed urban model (left) and LoD model for P1 (right).

6 Discussion and Perspectives

Although several LoD techniques were developed to simplify city models for different purposes, only a few of them focus on actual solar simulations for driving the simplification of the model. Comparing our approach to the POI method presented in [Besuievsky et al., 2014], we can observe a significant improvement in the accuracy of the simplification. Whereas in the mentioned technique simplification is implemented by a set of configurable distances to the POI, here we drastically simplify the model by keeping only the geometry that has the largest impact on the final calculation. If visualization of the simulation results is desired, then they can easily be mapped to the original model.

Another important result of this work is the analysis of the relevance the geometric details may have on solar simulation. As it is shown in [Biljecki et al., 2015], among the different applications of 3D city models, the solar impact, like computing solar panels irradiance, is frequently needed. However, building models are usually simplified to LoD1 or LoD2 in CityGML format using only basic

roof geometry, which may lead to significant differences from the real models, and thus seriously distorting the subsequent calculations. We show here that certain detailed geometry may be important in the impact study, and we provide a method to handle it. Our technique for detecting the skyline can be extended for other relevant architectural elements, as for example balconies and other protruding elements, necessary for other kinds of simulations (e.g., wind, pollution, pedestrian simulations).

Our simplification is point-of-view based, as we use the POI for reference. For further development we plan to extend the technique for simplifying the model from an area of interest, like a whole facade or a street. This would allow more general calculations than with a single POI without the burden of repeating the same simplification over and over again, but from slightly different origins.

Acknowledgements

This work was partially supported by grant TIN2014-52211-C2-2-R project from Ministerio de Economía y Competitividad, Spain.

References

[Beckers and Beckers, 2014] Beckers, B. and Beckers, P. (2014). Sky vault partition for computing daylight availability and shortwave energy budget on an urban scale. *Lightning Research and Technology*, 46(6):716–728.

[Beckers and Masset, 2008] Beckers, B. and Masset, L. (2008). Heliodon2 software, references & manuals. http://www.heliodon.net/ (in French & Spanish).

[Besuievsky et al., 2014] Besuievsky, G., Barroso, S., Beckers, B., and Patow, G. (2014). A Configurable LoD for Procedural Urban Models intended for Daylight Simulation. In Besuievsky, G. and Tourre, V., editors, *Eurographics Workshop on Urban Data Modelling and Visualisation*. The Eurographics Association.

[Besuievsky and Patow, 2013a] Besuievsky, G. and Patow, G. (2013a). Customizable lod for procedural architecture. *Computer Graphics Forum*, 32(8).

[Besuievsky and Patow, 2013b] Besuievsky, G. and Patow, G. A. (2013b). City-level level-of-detail. In Association, E., editor, *XXIII Congreso Español de Inforática Grafica*, pages 29–36.

[Biljecki et al., 2015] Biljecki, F., Heuvelink, G. B. M., Ledoux, H., and Stoter, J. (2015). Propagation of positional error in 3D GIS: estimation of the solar

irradiation of building roofs. *International Journal of Geographical Information Science*, 29(12):2269–2294.

[Biljecki et al., 2016] Biljecki, F., Ledoux, H., Stoter, J., and Vosselman, G. (2016). The variants of an LOD of a 3D building model and their influence on spatial analyses. *ISPRS Journal of Photogrammetry and Remote Sensing*, 116:42–54.

[Biljecki et al., 2014] Biljecki, F., Ledoux, H., Stoter, J., and Zhao, J. (2014). Formalisation of the level of detail in 3D city modelling. *Computers, Environment and Urban Systems*, 48:1–15.

[Clark, 1976] Clark, J. H. (1976). Hierarchical geometric models for visible surface algorithms. *Commun. ACM*, 19:547–554.

[Kolbe, 2009] Kolbe, T. H. (2009). Representing and exchanging 3d city models with citygml. In *Lecture Notes in Geoinformation and Cartography*, page 20. Springer Verlag.

[Luebke and Erikson, 1997] Luebke, D. and Erikson, C. (1997). View-dependent simplification of arbitrary polygonal environments. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 199–208.

[Luebke et al., 2002] Luebke, D., Watson, B., Cohen, J. D., Reddy, M., and Varshney, A. (2002). *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA.

[Markosian et al., 1997] Markosian, L., Kowalski, M. A., Goldstein, D., Trychin, S. J., Hughes, J. F., and Bourdev, L. D. (1997). Real-time nonphotorealistic rendering. SIGGRAPH '97, pages 415–420, New York, NY, USA.

[Müller et al., 2006] Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L. (2006). Procedural modeling of buildings. *ACM Trans. Graph.*, 25:614–623.

[Parish and Müller, 2001] Parish, Y. I. H. and Müller, P. (2001). Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 301–308. Press.

[Rodriguez et al., 2012] Rodriguez, D., Besuievsky, G., Patow, G., and Beckers, B. (2012). Procedural models to better compute solar flux at the neighbourhood scale. In *Proceedings of Flow modeling for urban development*.

[Side-Effects-Software, 2015] Side-Effects-Software (2015). Houdini.