

City Reconstruction and Visualization from Public Data Sources

J. R. Menzel¹, S. Middelberg¹, P. Trettner¹, B. Jonas¹ and L. Kobbelt¹

¹RWTH Aachen University, Germany



Figure 1: Rendering of automatically generated procedural content (left), image-based reconstructed buildings (middle) and our inverse procedurally generated buildings (right).

Abstract

We present a city reconstruction and visualization framework that integrates geometric models reconstructed with a range of different techniques. The framework generates the vast majority of buildings procedurally, which yields plausible visualizations for structurally simple buildings, e.g. residential buildings. For structurally complex landmarks, e.g. churches, a procedural approach does not achieve satisfactory visual fidelity. Thus, we also employ image-based techniques to reconstruct the latter in a more realistic, recognizable way. As the manual acquisition of data required for the procedural and image-based reconstructions is practically infeasible for whole cities, we rely on publicly available data as well as crowd sourcing projects. This enables our framework to render views from cities without any dedicated data acquisition as long as there are sufficient public data sources available. To obtain a more lively impression of a city, we also visualize dynamic features like weather conditions and traffic based on publicly available real-time data.

1. Introduction

City planning, (pedestrian) navigation and virtual tourism are just a few of the many use-cases of realistic three-dimensional city models. However, a manual creation of rich and detailed models required for those applications is infeasible, which is why the automated reconstruction of cities is an active field of research. Those reconstructions can be based on a variety of different input data sources, including city maps, street level photos, aerial images or laser scans. These differ essentially in both difficulty of acquisition and reconstruction quality. Thus, a major challenge in automated city reconstruction is to keep the data acquisition effort low and, at the same time, to ensure that the visualization reflects the city in a faithful and visually appealing manner.

We present a city reconstruction and visualization framework which handles different types of buildings based on different reconstruction techniques. All of the data we use for reconstruction and visualization is publicly available or can be acquired by crowd-sourcing. Figure 1 shows three views of a city visualization, each

displaying different kinds of reconstructions. The vast majority of buildings is procedurally generated from 2D maps acquired from the collaborative *OpenStreetMap* (OSM) project [osm16]. While such a fully-automatic procedural approach works well with simple residential buildings, structurally more challenging buildings demand for more elaborate techniques, as depicted in Figure 2.

Thus, to visualize such buildings, we use image-based techniques. The *City Reconstructions* framework from Untzelmann *et al.* [USMK13] fits well for our purpose as it provides the collaborative image-based reconstruction of point clouds from input images and also uses OSM data to align the building point clouds to a common coordinate frame. We present a semi-interactive tool to post-process these point clouds and to extract watertight 3D models. Furthermore, to exploit the regular structures and symmetries common to most urban buildings, we propose an image-based inverse semi-procedural modeling technique.

To further enhance the realistic impression of our city visualization we add pedestrian and traffic simulations based on the same

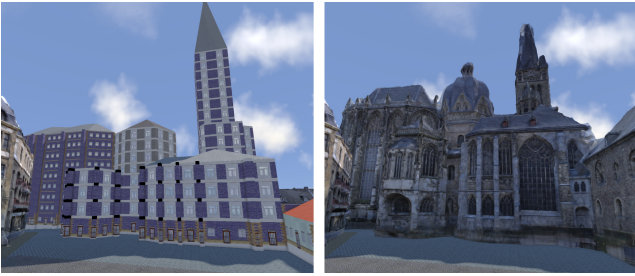


Figure 2: The Aachen cathedral reconstructed procedurally from OSM data (left) and reconstructed from images (right).



Figure 3: Realistic traffic lanes are generated from OSM data and used for the visualization as well as the traffic simulation.

2D map data as well as weather rendering based on real-time data as seen in Figure 3.

2. Related Work

Various results from procedural modeling can and are applied to cities and their visualization. Müller *et al.* worked on procedurally modeling cities in general [PM01] and buildings in particular [MWH*06]. They also published an approach for image-based inverse procedural modeling of facades [MZWVG07]. Later, their work led to the commercial product Esri CityEngine [Mue07]. Procedural models for architecture are often based around various grammars and their extension, with one particularly expressive one presented by Schwarz *et al.* [SM15]. We incorporate the approach by Krecklau *et al.* [KBK13] which evaluates facade grammars on the GPU and additionally generate novel geometry on the fly, resulting in a rich visualization.

The major challenges in non-procedural approaches for city visualization are the vast amount of data that have to be managed and the investigation of techniques that are able to visualize it efficiently. Liono *et al.* [LSS15] analyzed the management and retrieval of data sources for urban visualization and concentrated on the difficulties posed by heterogeneous data. Gaillard *et al.* [GVB*15] optimized the visualization process itself to allow it to run in a web browser. Common techniques to make the rendering scalable are *culling* and *level of detail management*. Yang *et al.* [YPL07] use general GIS data to evaluate visibility for de-

sign decisions in urban planning. Gal *et al.* [GD12] calculate visibility explicitly for procedurally generated cities. Wonka [Won01] uses visibility information to perform culling of geometry to reduce the rendering load. Yilmaz *et al.* [YG07] also perform occlusion culling for urban scenarios but with a conservative slicing algorithm.

In practice, high quality geometric and texture data for buildings and landscapes is typically created by artists. A viable alternative to this costly and time-consuming method are image-based techniques, like *Structure-from-Motion* (SfM) [BP12, ASS*09, FFGG*10, FCSS10, SPF10].

Our visualization uses the SfM framework from Untzelmann *et al.* [USMK13] which aims to reconstruct buildings using many images taken from users in a collaborative online system with consumer grade hardware. The BigSfM project [big] has a similar approach but tries to automatize it for internet-scale heterogeneous image collections. Wu *et al.* extract schematic surface representations from SfM reconstructions [WACS12]. Xiao *et al.* [XFZ*09] apply semantic segmentation and structure analysis on facade images to obtain convincing 3D street models. Their approach for the detection of repetitive facade patterns shares some concepts with our inverse procedural approach we discuss in Section 4. Vahl and Lukas [VL13] do not use street-level imagery but extract facade models from oblique aerial images.

While image-based techniques result in realistic reconstructions, the acquisition of images for all buildings of a city is practically infeasible. Previous approaches try to overcome this issue by exploiting internet photo collections [ASS*09, FFGG*10, FCSS10, SPF10]. However, such freely available online databases usually cover only touristic landmarks or inner cities sufficiently but not residential areas or suburbs. Procedural techniques, on the other hand, scale well but fail to represent complex buildings in a pleasing way. Thus, our approach to city visualization combines the strengths of both techniques and relies on publicly available, crowd-sourced data.

3. Procedural City Reconstruction from Public Data

Our city reconstruction and visualization framework *Virtual Aachen* generates a city model from various sources. We parse the OSM database to leverage different data sources in our framework. The street layout and ground is represented by a 3D mesh directly derived from the street and lane information in OSM. While nodes in OSM can store the elevation as well, this data is only rarely present so we ignore the elevation from that source and use the height values from the height-maps provided by a NASA mission [Ram]. The elevation data has an accuracy of one meter and samples the majority of the populated earth surface at up to one sampling point per 30 meters with a minimum resolution of one sampling point per 90 meters.

OSM is rich in data normally not associated with street maps. Points-of-interests like public mailboxes, phone booths, park benches and traffic lights are individually mapped. We add those and other pieces of city furniture into our model by adding an instance from a mesh database (see Figure 4). Individual trees get instanced in a similar fashion as city furniture but we also add trees



Figure 4: Points of interests based on OSM data are instanced at their actual locations.

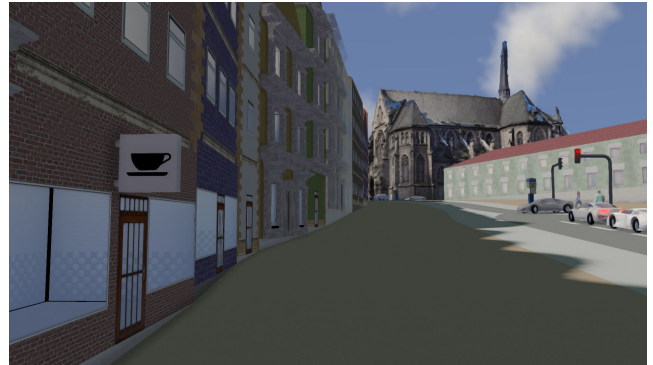


Figure 6: Detail objects on procedural facades. When the type of a store is known, a matching sign is placed on the facade.



Figure 5: The city of Aachen with a procedural forest in the foreground.



Figure 7: Different weather conditions: Sunny, cloudy and raining.

in a randomized pattern in areas which are mapped as forests as depicted in Figure 5.

As we have already knowledge of the street layouts, we add this data to a traffic and pedestrian simulation to let the city appear more lively. OSM includes information about one-way streets, speed limits and street types which can be used to enhance the plausibility of the simulation. Real-time traffic data is also acquired and constrains the traffic simulation where it is available.

Whenever we lack an image-based representation of a building, we generate it from OSM data. Not only the outline of a building can get stored in OSM, we found that for larger cities the data also includes the height of buildings (as number of floors), the roof shape, roof and facade color, as well as the usage of a building. Most stores for example are mapped correctly by store type. All of this information is used for our procedural building generation. We generate procedural facades based on the grammars presented by Krecklau et al. [KBK13] while extending the method by utilizing OSM metadata, for example the actual location of the entrance and the type of a store by adding store signs as depicted in Figure 6. As the image-based reconstructed buildings discussed in the following section are mapped to the corresponding OSM building outline within the *City Reconstructions* framework [USMK13], we

can easily replace procedural buildings by more realistic representations.

The renderer further queries the current weather at the center of the city map from a web service [for] to mimic the actual weather condition and lighting as depicted in Figure 7. The current sun position gets calculated based on the time of day as well as the geographical location of the city (see Figure 8).

Ultimately, all used data sources are either based on or mapped to OpenStreetMap to avoid mismatches of the positions. Both, the procedural data sources as well as the image-based sources rely on crowd-sourcing to ensure good scalability and low cost. However, our framework is not limited to this scenario.

4. Image-Based Building Reconstruction

The approach of Untzelmann et al. [USMK13] is well-suited for our multi-source city visualization, as it offers the reconstruction of cities on a building level, with the resulting point clouds readily aligned with the OSM coordinate frame. Furthermore, the framework comes along with apps for iOS and Android that permit citizens to contribute to the reconstruction of their own city by uploading images of local buildings. This goes well with the also crowd-sourced OSM data.

We utilize *Poisson reconstruction* to extract a closed triangle

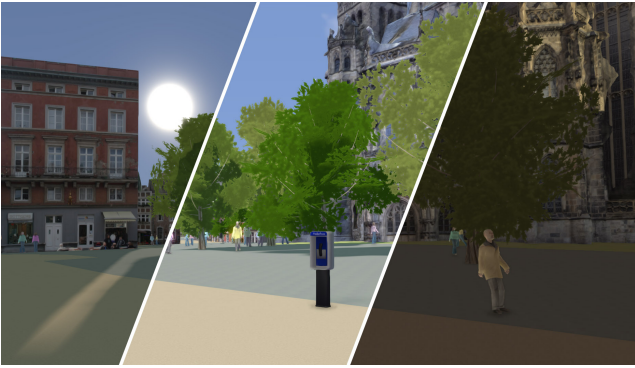


Figure 8: The same location at different times of day.



Figure 9: Comparison of a reconstructed mesh before (left) and after (right) post-processing. With our tool, it took less than 10 minutes to obtain the mesh on the right.

mesh from the unstructured building point clouds [KBH06]. However, as depicted in Figure 9, lack of data in a point cloud manifest themselves in artefacts of the resulting mesh. A common issue in image-based reconstruction are not only partially observable building parts, like roofs or inaccessible parts of the facades. Also, image noise or false-positive feature matches can lead to wrongly reconstructed geometry. To overcome these issues, we developed a tool for semi-automatic post-processing of point clouds (Section 4.1).

Most urban facades exhibit a highly regular structure, e.g. windows that are arranged in a regular grid. Exploiting such regularities can significantly improve the visualization of such buildings and, at the same time, reduce rendering time and memory load. In Section 4.2 we briefly present our algorithm to detect such regularities.

4.1. Semi-Interactive Point Cloud Post-Processing

Figure 10 shows a screenshot of the tool we developed for the semi-automatic post-processing of point clouds. It offers an intuitive interface to select points that project within a circular window centered at the cursor, whose size can be altered with the scroll wheel. Selected points can be removed or mirrored around an axis that can be specified via drag and drop. Especially the latter is useful for

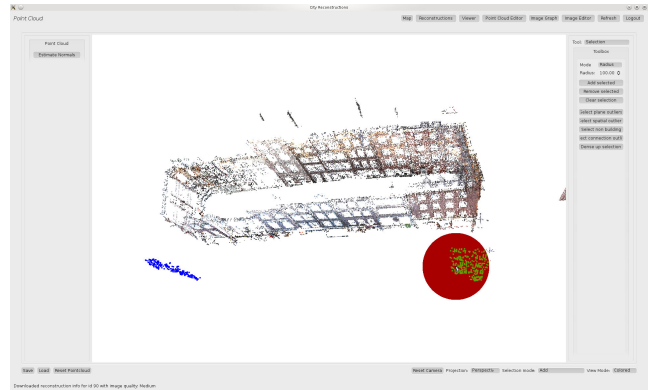


Figure 10: Screenshot of our tool for the semi-automatic post-processing of point clouds.

buildings for which not all facades are publicly accessible. Well-reconstructed facades can simply be mirrored to create a plausible, although not necessarily faithful visualization for such buildings, as displayed in Figure 9. Additionally, our tool provides step-wise smoothing and decimation of the Poisson-reconstructed meshes to remove noise and to reduce their complexity.

4.2. Inverse Procedural Facade Modeling

For the detection of structural regularities we use the input images along with the camera poses and the point cloud we obtain from the *City Reconstructions* framework. We further rectify the images by detecting horizontal and vertical vanishing points and computing a rotation that puts them to infinity [BP12]. As a first step, we have to identify the rectangular areas spanned by the individual facades, i.e. we have to separate the facades of neighbouring buildings from each other. We detect strong vertical edges in the rectified images to split the facades horizontally and use the point cloud to estimate the vertical extent and an average depth \bar{d} of a facade.

Having separated the facades, our goal is to detect *symmetry groups*, i.e. sets of rectangular areas, such that all members of a symmetry group correspond to the same structural facade element. To achieve this, we first split a facade into rectangular areas aligned with the facades horizontal and vertical axes, such that each area contains either only a structural instance, e.g. a window, door or balcony, or only wall (Figure 11). The structural areas are then clustered based on their pixel intensities to build the symmetry groups.

The facade splitting is done first vertically, to split the facade into alternating *structural floors* and *wall floors*, and then performed along the horizontal direction for each structural floor. We briefly explain the vertical splitting only, as the subsequent horizontal splitting is analogous.

To split structural floors from wall floors, we assume that the wall covers most of the facades area and that the facade wall has constant depth. Thus, we employ a voting scheme to detect the most dominant facade depth. We discretize the facade into $1\text{cm} \times 1\text{cm} \times 1\text{cm}$ voxels, such that the first two dimension correspond to the facade's principal axes and the third dimension corresponds to facade depth.

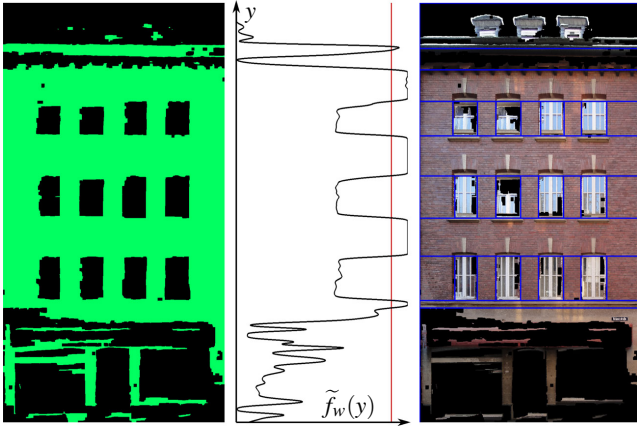


Figure 11: Structural splitting of a facade. Left: Green pixels belong to the set \mathcal{D}_w . Middle: The function $\tilde{f}_w(y)$. The red line corresponds to the splitting threshold τ_{split} . Right: Vertically and horizontally split facade.

Here, we consider only depths within 25cm distance to \bar{d} . We now let each voxel $(x, y, d)^T$ vote for the depth d iff

$$e_{ph}(x, y, d) \leq \tau_{ph} \quad \wedge \quad e_{ph}(x, y, d) \leq \alpha e_{ph}(x, y, d \pm 1) . \quad (1)$$

Here, e_{ph} indicates the photometric error of a voxel $(x, y, d)^T$ that measures the variance of the pixel intensities at the voxel's projections onto the input images. The first condition ensures that this error is below a threshold τ_{ph} and the second condition requires that e_{ph} is by a factor α smaller compared to both voxels neighbors along the depth dimension non-minimum suppression. The latter prevents that homogeneously textured facade regions tamper the result with false votes.

We now accumulate the votes for each depth d along the xy -plane and take the depth with the maximum number of votes as wall depth d_w . Additionally, we assign each pixel $(x, y)^T$ to depth d_w and add it to the set of all wall pixels \mathcal{D}_w , if the voxel (x, y, d_w) satisfies Condition 1. Due to the second term in Condition 1 this yields only a very sparse subset of actual wall pixels. Thus, we iteratively expand \mathcal{D}_w by adding neighbors of pixels in \mathcal{D}_w that satisfy only the first term in Condition 1. This is done as long as no more pixels can be added followed by morphological filtering to close remaining small holes in the wall. This results in a fairly dense set \mathcal{D}_w as shown in Figure 11. We then define a function $f_w(y) = \sum_x \mathbb{1}_{\mathcal{D}_w}(x, y)$ that counts the number of all wall pixels for each pixel row and a smoothed version $\tilde{f}_w(y) = g_\sigma(y) * f_w(y)$, where g_σ is a Gaussian kernel with standard deviation σ , which we set to 5 throughout our experiments. Vertical splits are placed between those rows at which $\tilde{f}_w(y)$ falls below or exceeds a threshold τ_{split} . The parameters τ_{ph} , α and τ_{split} are user-controllable and set to $\tau_{ph} = 0.0025$, $\alpha = 1.25$ and $\tau_{split} = 0.8$ for our experiments.

Having computed the symmetry groups, we can match each group to a manually prepared library of structural instances to replace it with a high-quality library instance in the visualization. This is also beneficial from a privacy-preserving viewpoint, as replacing, e.g., windows with anonymous library instances prevents



Figure 12: Left: A Poisson-reconstructed building. Right: The same building reconstructed with our procedural approach. The windows have been automatically replaced by a library instance.

that one can look inside the buildings. Figure 12 shows a building that is visualized both, as a Poisson-reconstructed and as a procedurally reconstructed building. As can be seen, the procedurally reconstructed building looks much cleaner, with the facades separated by sharp edges.

5. Rendering

Our real-time rendering framework has to be able to visualize a wide variety of different data types and needs to be flexible enough to be extended to additional rendering techniques if needed (e.g. new kinds of compressed or procedural meshes and facades). A well known technique in modern real-time rendering is Deferred Shading [DWS*88]. In a first render pass, all the data needed for the illumination is collected in a so called G-Buffer. A second, screen-space pass can then use this data to perform the actual lighting calculations. The main benefit is an elimination of overdraw of already shaded pixels as these computations can often be the main bottleneck in modern real-time rendering applications. This is especially true for our procedural facades.

One important disadvantage is the requirement for a G-Buffer data structure which has to be compatible with all rendering techniques to be used. As our procedural facades require a complex grammar evaluation, the required data for the shading differs substantially from the image-based reconstructed buildings and the inverse semi-procedural generated buildings. A common G-Buffer layout proves to be too complex in this case which results in bandwidth limitations during the first rendering pass and a too complex uber-shader during the second pass. In addition to this, such an architecture would not be flexible enough for later additions of other novel geometry representations.

To eliminate the problem of overdraw we instead opted for a Z-Prepass renderer which iterates over the geometry twice. During the first pass only the Z-Buffer gets filled while the second pass will perform the shading without any unneeded overdraw due to hardware accelerated early Z testing. The main benefit is the complete decoupling of different rendering techniques for different building representations. Different geometry representations which require different rendering techniques are organized in modules with a common API. At least they have to provide a method to render the objects into a Z buffer (used for the first pass as well as for

shadow map generation) and a color pass which renders into the final framebuffer. A third method to provide rendering of transparent objects is optional.

It has been shown that the lighting can also get decoupled from the geometry representations by extending this Z-Prepass renderer into a Forward+ renderer [HMY12]. This has advantages in scenarios with many light sources, e.g.. when adding realistic night rendering of the city with thousands of light sources.

While the procedural facades have an implicit level of detail as the grammar will only get evaluated for visible pixels due to the Z-prepass, the reconstructed buildings need an explicit level of detail. For this, building variants with a reduced geometric complexity are used. In addition to this, view frustum culling for the ground and buildings as well as the instanced objects like trees is performed.


6. Future Work

Data acquisition, inverse procedural modeling and efficient rendering are the main challenges in extending the number of buildings per city which we can represent using our inverse semi-procedural representation. The data acquisition component can scale well with the use of crowd sourcing as we have already implemented. Our current implementation uses facades made up from building blocks which were detected during the inverse modeling. This reduced the amount of memory needed to represent large city areas. However, in future research we want to extend this to model the facades from 3D elements and procedural material description when possible and use the actual photo based texture only as a fallback for more uncommon facades. This will reduce the memory required even further which also allows for more efficient rendering.

7. Conclusion

We presented a framework for city reconstruction and visualization which is based on freely available data and crowd sourced images. It combines different building representations based on the availability of input data and the building type. Our rendering architecture can freely mix and visualize those different datasets as one city representation.

As one of these building representations we introduced a novel inverse semi-procedural representation of buildings based on crowd sourced images together with a reconstruction pipeline.

Acknowledgements: We gratefully acknowledge support by PRE-Serv, Europäischer Fonds für regionale Entwicklung - Investition in unsere Zukunft . We also thank all those that contributed to the Virtual Aachen Project, including Niko Abeler, Janis Born, Christian Mattes and Ole Untzelmann.

References

- [ASS*09] AGARWAL S., SNAVELY N., SIMON I., SEITZ S., SZELISKI R.: Building Rome in a Day. In *ICCV* (2009). 2
- [big] Bigsfm. <http://www.cs.cornell.edu/projects/bigsfm/>. Accessed: 2016-07-19. 2
- [BP12] BAZIN J. C., POLLEFEYS M.: 3-line ransac for orthogonal vanishing point detection. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012). 2, 4
- [DWS*88] DEERING M., WINNER S., SCHEDIWIY B., DUFFY C., HUNT N.: The triangle processor and normal vector shader: A vlsi system for high performance graphics. *SIGGRAPH Comput. Graph.* 22, 4 (June 1988), 21–30. 5
- [FCSS10] FURUKAWA Y., CURLESS B., SEITZ S. M., SZELISKI R.: Towards internet-scale multi-view stereo. In *CVPR* (2010). 2
- [FFGG*10] FRAHM J.-M., FITE-GEORGE P., GALLUP D., JOHNSON T., RAGURAM R., WU C., JEN Y.-H., DUNN E., CLIPP B., LAZEBNIK S., POLLEFEYS M.: Building Rome on a Cloudless Day. In *ECCV* (2010). 2
- [for] The dark sky company forecast api. <https://developer.forecast.io/>. Accessed: 2016-07-14. 3
- [GD12] GAL O., DOYTSHER Y.: Fast visibility analysis in 3d procedural modeling environments. In *Proceedings of the 3rd International Conference on Computing for Geospatial Research and Applications* (2012). 2
- [GVB*15] GAILLARD J., VIENNE A., BAUME R., PEDRINIS F., PEY-TAVIE A., GESQUIÈRE G.: Urban data visualisation in a web browser. In *Proceedings of the 20th International Conference on 3D Web Technology* (2015). 2
- [HMY12] HARADA T., MCKEE J., YANG J. C.: Forward+: Bringing deferred lighting to the next level. In *Proc. EUROGRAPHICS 2012* (2012). 6
- [KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson Surface Reconstruction. In *Symposium on Geometry Processing* (2006). 4
- [KBK13] KRECKLAU L., BORN J., KOBELT L.: View-Dependent Realtime Rendering of Procedural Facades with High Geometric Detail. *Computer Graphics Forum* (2013). 2, 3
- [LSS15] LIONO J., SALIM F. D., SUBASTIAN I. F.: Visualization oriented spatiotemporal urban data management and retrieval. In *Proceedings of the ACM First International Workshop on Understanding the City with Urban Informatics* (2015). 2
- [Mue07] MUELLER P.: Applied procedural modeling. In *ACM SIGGRAPH 2007 Courses* (2007). 2
- [MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., VAN GOOL L.: Procedural modeling of buildings. In *ACM SIGGRAPH 2006 Papers* (2006). 2
- [MZWVG07] MÜLLER P., ZENG G., WONKA P., VAN GOOL L.: Image-based procedural modeling of facades. In *ACM SIGGRAPH 2007 Papers* (2007). 2
- [osm16] Openstreetmap. <http://www.openstreetmap.org>, 2016. [Online; accessed 14-September-2016]. 1
- [PM01] PARISH Y. I. H., MÜLLER P.: Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001). 2
- [Ram] RAMIREZ E.: Shuttle radar topography mission. <http://www2.jpl.nasa.gov/srtm/mission.htm/>. Accessed: 2016-07-12. 2
- [SM15] SCHWARZ M., MÜLLER P.: Advanced procedural modeling of architecture. *ACM Trans. Graph.* (2015). 2
- [SPF10] STRECHA C., PYLVANAINEN T., FUA P.: Dynamic and Scalable Large Scale Image Reconstruction. In *CVPR* (2010). 2
- [USMK13] UNTZELMANN O., SATTLER T., MIDDELBERG S., KOBELT L.: A Scalable Collaborative Online System for City Reconstruction. In *The IEEE International Conference on Computer Vision (ICCV) Workshops* (2013). 1, 2, 3
- [VL13] VAHL M., LUKAS U. V.: Toward Automated Façades Generation from Oblique Aerial Images. In *Eurographics Workshop on Urban Data Modelling and Visualisation* (2013). 2
- [WACS12] WU C., AGARWAL S., CURLESS B., SEITZ S. M.: Schematic Surface Reconstruction. In *CVPR* (2012). 2

- [Won01] WONKA P.: *Occlusion Culling for Real-Time Rendering of Urban Environments*. PhD thesis, 2001. [2](#)
- [XFZ*09] XIAO J., FANG T., ZHAO P., LHUILLIER M., QUAN L.: Image-based street-side city modeling. In *ACM SIGGRAPH Asia 2009 Papers* (2009). [2](#)
- [YG07] YILMAZ T., GÜDÜKBAY U.: Conservative occlusion culling for urban visualization using a slice-wise data structure. *Graph. Models* (2007). [2](#)
- [YPL07] YANG P. P.-J., PUTRA S. Y., LI W.: Viewsphere: A gis-based 3d visibility analysis for urban design evaluation. *Environment and Planning B: Planning and Design* (2007). [2](#)