COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2014; **25**:245–255
Published online 16 May 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1582

SPECIAL ISSUE PAPER

A hybrid level-of-detail representation for large-scale urban scenes rendering

Shengchuan Zhou[1]*, Innfarn Yoo[2], Bedrich Benes[2] and Ge Chen[1]

[1] College of Information Science and Engineering, Ocean University of China, 238 Songling Road, 266071 Qingdao, China
[2] College of Technology, Purdue University, 401 N.Grant Street, West Lafayette, 47906 IN, USA

ABSTRACT

A novel hybrid level-of-detail (LOD) algorithm is introduced. We combine point-based, line-based, and splat-based rendering to synthesize large-scale urban city images. We first extract lines and points from the input and provide their simplification encoded in a data structure that allows for a quick and automatic LOD selection. A screen-space projected area is used as the LOD selector. The algorithm selects lines for long-distance views providing high contrast and fidelity of the building silhouettes. For medium-distance views, points are added, and splats are used for close-up views. Our implementation shows a 10× speedup as compared with the ground truth models and is about four times faster than geometric LOD. The quality of the results is indistinguishable from the original as confirmed by a user study and two algorithmic metrics. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS
massive urban models; hybrid rendering; level of detail

***Correspondence**
Shengchuan Zhou, College of Information Science and Engineering, Ocean University of China, 238 Songling Road, 266071 Qingdao, China.
E-mail: shc.zhou@gmail.com

1. INTRODUCTION

Rendering large urban data requires processing of huge amounts of data, and although many researchers presented a wide variety of approaches, it still poses an open problem.

Our key observation is that human visual acuity is sensitive to edges [1]. Urban scenes, especially of modern cities, are composed from buildings that have a clearly defined boundary with the environment. Because of this, the usual level-of-detail (LOD) techniques, such as point-based algorithms and triangular mesh simplifications, will often generate noise for boundaries. We hypothesize that urban scenes can be simplified by points and lines that will support an efficient rendering and will be also visually plausible.

We introduce a new hybrid rendering approach that aims to provide urban model simplification at medium and long distances. Our method produces similar visual quality results compared with textured meshes. The main contributions are as follows: (1) a hybrid representation for urban models by points and subdivided lines that are created through recursive Wang tiles and color clustering; (2) a novel hybrid strategy that dynamically selects and renders points and lines according to the raster area of objects, which avoids expensive texture operations and compacts the data in the memory; and (3) a comprehensive quality assessment including perceptual study and algorithm-based metrics, which show that our method is indistinguishable from mesh rendering.

Figure 1 shows our work applied to an urban scene containing 21.31k building objects, modeled at a resolution of 0.2k–5k triangles and 302k texture images with a resolution of 32 × 32 to 1024 × 1024 pixels (5.21 GB of geometric data and 42.32 GB of textures). The simplified scene was rendered at interactive frame rates on a Xeon 2.0-GHz PC (Intel Corporation, Santa Clara, CA, USA) with Nvidia GTX 480 graphics processing unit (GPU; Nvidia Corporation, Santa Clara, CA, USA), and the building edges are preserved as can be seen in the inset images. In Figure 1a, 58.12 million points and 1.26 million lines are rendered at 51 milliseconds per frame, and in Figure 1b, 77.85 million points are rendered at 83 milliseconds per frame. By introducing our hybrid representation, the sharpness of the resulting image and the building structure is well preserved, and the number of rendered primitives is reduced. Perceptual evaluations show that

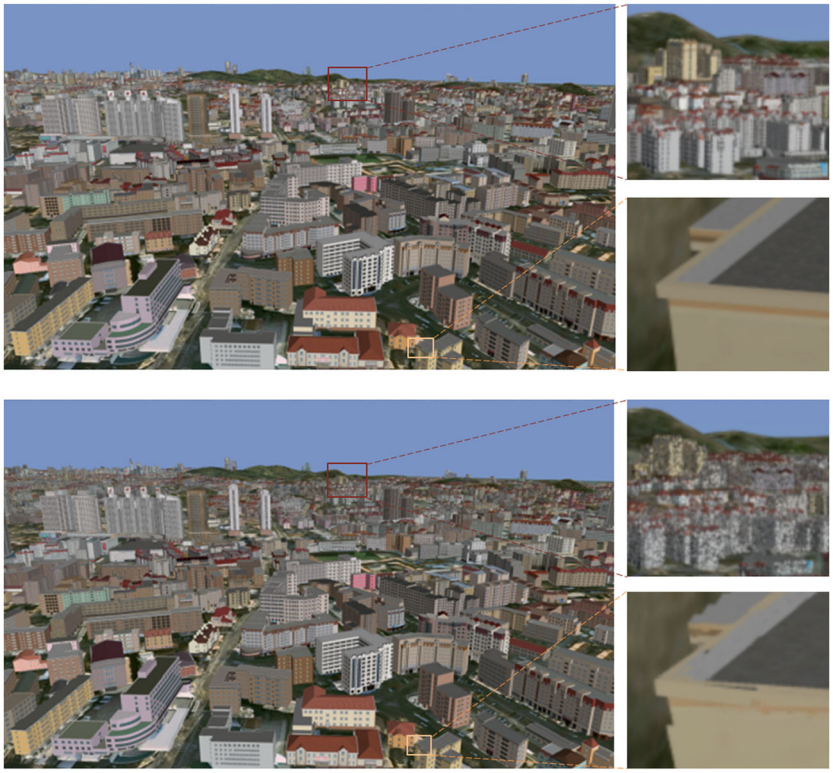


Figure 1. Hybrid level-of-detail rendering of urban-scale architecture models.

our approach produces similar visual quality compared with textured triangle meshes and our method provides an average 10× speedup compared with a rendering of the full-textured geometry.

2. RELATED WORK

Large-scale city geometry rendering is usually addressed by LOD techniques. The first algorithms can be traced as early as the work of Clark, who introduced hierarchical geometric models in [2] and Jones [3] who addressed the hidden-lines elimination problem. Later, spatial subdivision [4,5] and occlusion culling [6,7] were used to calculate visibility for successive frames with the objective to reduce the amount of rendered geometry. Other methods included LOD [8,9] to reduce the scene complexity and out-of-core rendering [10,11]. Research works such as [12,13] combined the aforementioned methods into frameworks for rendering massive models. In our work, we combine point-based and line-based representations into a unified framework that addresses both the rendering quality and the amount of in-core data. We also provide various evaluations of our approach.

Image-based urban model rendering has been addressed by Sillion *et al*. [14] who introduced impostors, and Maciel *et al*. [15] used textured clusters. Decoret *et al*. [16] used multiple layers to address parallax artifacts and later used impostor clouds for extreme simplification in [17]. Wimmer *et al*. [18] introduced a data structure for encoding the appearance of urban models. The BlockMap representation [19] encodes the geometry and the sampled appearances of buildings and stores them as textured prisms. Omni-directional relief impostors were also used to render urban scenes by selecting precomputed relief maps [20]. Most of the image-based urban rendering approaches work on building aggregations, which allow for a more compact simplification, provide better performance for distance views, but are prone to parallax error. Our approach works for individual buildings, focuses on improving visual quality, and reduces memory usage by using high-contrast simplified geometry as LOD primitives.

An alternative to point-based rendering is to use *hybrid representations.* An inspiration for our work is the approach of Deussen *et al*. [21] who proposed a rendering framework that combined points, lines, and polygons for complex plant ecosystem visualization. Merhof *et al*. [22] implemented a hybrid visualization system that combines triangles with point sprites. Approaches that mix multi-resolutional polygons and point rendering were introduced in [23] and [24]. However, these systems cannot be directly applied to massive urban models.

3. SYSTEM OVERVIEW

Figure 2 shows an overview of our approach. The input to our system is the building geometry and textures. Lines and

Comp. Anim. Virtual Worlds 2014; **25**:245–255 © 2014 John Wiley & Sons, Ltd.
DOI: 10.1002/cav

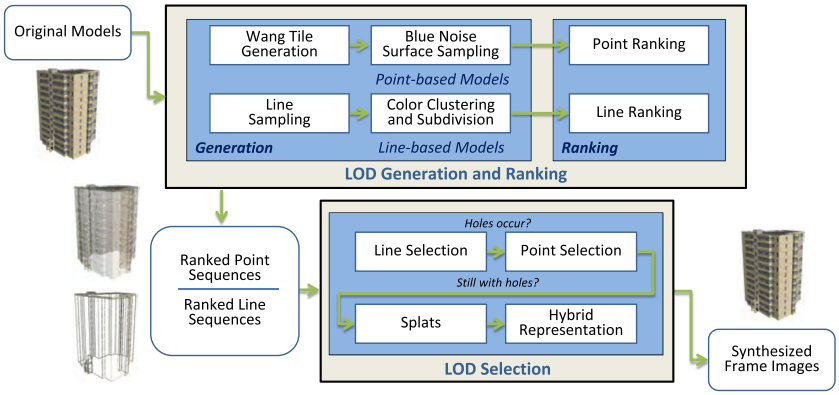


Figure 2. System overview. Input polygonal textured mesh is preprocessed, and a set of lines and points that cover the model in a progressive way is generated. The simplified representation is stored in a linear structure for easy access. The level of detail is selected from the projected area, and lines and points are added gradually. Splats are used for close-ups.

points are first generated by being extracted from the input. The lines are generated by analyzing the model geometry and image-based color clustering. The points are extracted via progressive recursive blue noise Wang tile sampling. The lines and points are then ranked and stored in linear structures for each input model. The results of this step are progressive sequences of points and lines per model where simplified representations of an urban model can be generated by selecting subsets from the sequences.

We build a novel LOD rendering model that uses the dual lines and points representation. First, the surface area of each building is precomputed. We then determine the number of lines and points that need to be rendered through a projection transformation of the surface area at runtime. We then select the required number of primitives from the progressive sequences. For the distance views, only lines are rendered. Points and subdivided lines are added gradually with the decreasing viewing distance. For extreme close-ups, this simplification cannot produce hole-free models, so we use a splat-based rendering to guarantee correct coverage of the models.

4. LEVEL-OF-DETAIL GENERATION AND RANKING

The input model is converted to LOD representation that represents the geometry as a set of points and perceptually salient lines. Here, we describe how these data structures are generated.

4.1. Points and Lines Generation

Point distribution should follow several important characteristics in order to produce visually plausible results.

(1) Point samples should be evenly distributed over their spatial domain to assure good coverage of the input that is guaranteed by Poisson distribution.
(2) Regularly distributed points can produce regular patterns that are visually disturbing. The blue noise sampling removes the regularity by transferring low-frequency artifact (aliasing) to high-frequency noise.
(3) The blue noise should be progressive, so that it allows for changing the point density without violating conditions 1 and 2.
(4) Although applied as preprocessing, the sampling should be fast enough to be applied to many buildings.

A method that has these four characteristics is the recursive Wang tiles generation of [25]. We used several pregenerated sets of Wang tiles that contain progressive blue noise points. These sets provide different sampling rates per polygon area.

Lines generation is controlled by the angle between two adjacent mesh faces. If the angle is less than a certain threshold (175° in our implementation), a single adjacent edge will be extracted and represented as outlines of a main structure. We use the prevailing color from the input as a single color of the entire line to avoid texture. In most cases, this is a shared color between two adjacent edges. If two edges have different colors, the brighter color is used, because it causes a stronger visual response [26]. We do not use an average because the averaged color would blur the result or could lead to incorrect color, not represented in the input. Using the brighter color, we can guarantee that the color is consistent to at least one facade. An edge with a strong color variation (Figure 3) is further subdivided according to its color gradient.

The color information is converted to Lab color space in which the measured color differences are proportional to the human perception and can be calculated as the Euclidean distance [27]. We convolve the color of the pixels on each line with a 1D Gaussian kernel to attenuate high-frequency noise. Then a 1D differentiation kernel $[-1, 0, 1]$ is used to cluster the color elements by calculating perceptual color differences of neighboring pixels. The vertices that subdivide the lines as well as the different

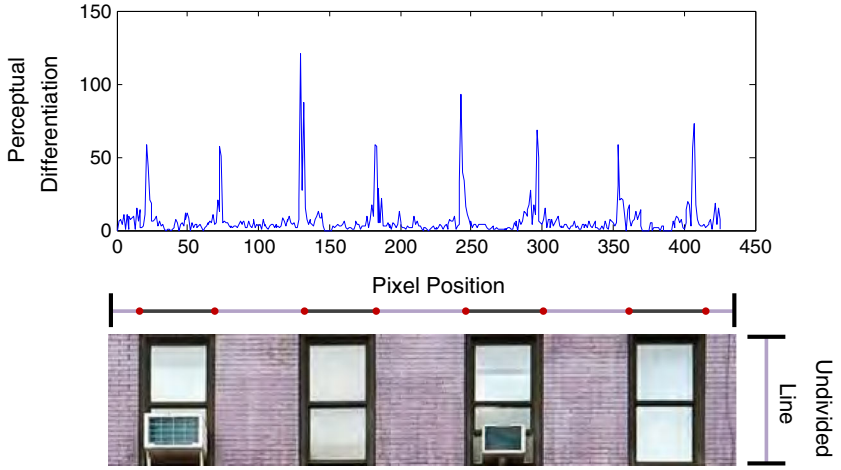


Figure 3. Lines are subdivided according to their local perceptual color differences.

color clusters correspond to local maxima of the convolved space (Figure 3). The length of each line is then used for line-based ranking and raster area calculations.

4.2. Point and Line Rankings

The previous steps simplify the model by representing it as lines and points. In the next step, they are ordered to allow for rendering that supports adaptive LOD selection.

The *point ranking* creates a point sequence for an input model. Any prefix of the sequence is a simplified point-based representation, and it incrementally adds details. This progressive representation allows us to generate simplification according to a given projected area that is determined in the LOD selection stages. Although we could just randomly select the points, the progressive representation maintains the blue noise characteristic for varying point densities.

It is achieved during the point sampling, where each point sample in the pregenerated sets of Wang tiles is assigned with a rank number. The resulting ranking defines a progressive sequence that is used during the LOD selection. We generate the desired progressive sequence for the input model by sorting the entire point set according to the rank number and storing the various points in increasing order.

Line ranking creates a line-based LOD model so that the complete lines can be replaced by subdivided lines according to the LOD selection. We store the subdivided lines and the completed lines in two sequences. To avoid overlaps and z-fighting, we replace a completed line when any one of its subdivided line sections is visible.

In the first step, we assign each complete line with a unique *id* as a rank number. We then organize the subdivided lines into groups according to the rank and construct a sequence of line groups. After that, we search for the longest subdivided line in each group and use the length as a sort key to sort the line groups as well as the corresponding complete lines in increasing order. A subdivided line group shares the same rank id with the corresponding complete line; they have the same index in their arrays. By rendering a prefix from the complete lines array, we avoid their subdivisions by rendering the corresponding suffix of the subdivided lines array.

Let us denote the number of completed lines by N. The line-based model representation will contain sublevels i, $i \in [0, N]$, composed by the first $N - i$ lines from the ranked line sequence and the last i subdivided line groups from the ranked group sequence. The ranked structures can be directly compiled into a Vertex Buffer Object and Vertex Array Object in OpenGL to increase the rendering performance. Moreover, the LOD selection and rendering can be carried out by a single OpenGL function call.

5. LEVEL-OF-DETAIL SELECTION

The ranked sequences provide a flexible way to render the model at various LODs. The actual number of points and lines is chosen automatically from the ranked sequences.

Our objective is to select representations and to render models that minimize the number of rendering primitives while respecting a user-given screen-space error tolerance ϵ for maintaining visual quality. We use $\epsilon = 0$ in our examples, which corresponds to the optimization of the pixel coverage for error minimization.

Let the input model M consist of n triangles $M = \{T_1, \ldots, T_n\}$ with surface areas A_t in world space and raster areas A'_t under a given perspective projection. The value of A'_t can be approximated by

$$A'_t \approx A'_u \sum_{i=1}^{n} A_i \tag{1}$$

where A'_u is the raster area of a screen-aligned unit square in model position.

DOI: 10.1002/cav

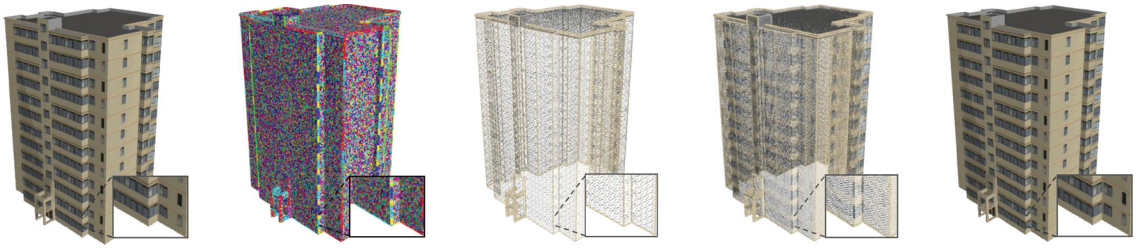


Figure 4. An original model (a) is sampled into different points and lines (b) and displayed using a selection of different point and line rankings (c–d). In a close-up view, the points are replaced by geometric splats (e).

Let x denote the resolution of the screen in pixels. Given the total field of view θ and the distance in the viewing direction from the eyepoint to the model d, the unit raster area A'_u and unit raster length L'_u can be calculated as

$$A'_u = L'_u \cdot L'_u \approx \left(\frac{x}{2d \tan \theta/2} \right)^2 \tag{2}$$

Let us assume the original model is approximated by a point set $S_p = \{P_1, P_2, \ldots, P_m\}$ and a line set $S_l = \{L_1, L_2, \ldots, L_n\}$ (Figure 4). The point set has a raster area A'_p, and the line set has a raster area A'_l. To obtain a faithful rendering output, especially to avoid gaps, we need

$$A'_t - A'_l - A'_p \leq \epsilon \tag{3}$$

5.1. Line Selection

We use the visible raster area as the LOD selector for lines. A subdivided line group is rendered if and only if the longest subdivided line of the group is visible (large than 1 pixel); otherwise, the corresponding completed line is rendered. We first select a suffix from the ranked sequence of the subdivide line groups using the projected length as criterion, then we render the corresponding prefix from the ranked complete lines sequence for avoiding overlapping.

Let us assume that the input model is approximated by a line set $S_l = \{L_1, L_2, \ldots, L_n\}$. And let us denote the line length by l_i. The length of the line in the image plane can be approximated by $l'_i = l_i L'_u$, where L'_u corresponds to the projection transformation. The raster area A'_l covered by the line set is then

$$A'_l = \sum_{i=1}^{n} l'_i = \sum_{i=1}^{n} l_i L'_u \tag{4}$$

We use a similar condition to (3) to decide what is rendered. $A'_t - A'_l \leq \epsilon$. If the condition is true, we render the model only by lines. Otherwise, we compute the raster area that needs to be covered by points to fill the gaps.

5.2. Point Selection

Points fill the gaps between line structures and provide facade details. We first render the lines, and the number of points required to cover the facade is then calculated.

In our approach, each point is represented as a point sprite (i.e., planar circular element aligned with the image plane) with size W'_p. The number of points depends on the difference between the approximated raster area of triangles A'_t and the approximated raster area of lines A'_l, which are calculated by Equations (1) and (4). The number of points n_p is then

$$n_p = j_p \left(A'_t - A'_l \right) / W'^2_p \tag{5}$$

The coefficient j_p slightly increases the point density to avoid gaps and holes caused by the blue noise sampling and anti-aliasing (we use $j_p \in [1.0, 1.2]$ in our experiments). The coefficient j_p is used only for points because the line length does not change with subdivision.

5.3. Splat Selection

Our approach aims at medium-distance and long-distance views. If the object is too close to the camera, the actual number of pregenerated points can be smaller than the number of required points n_p. In this situation, we use tessellated splats to cover the gaps between the rendered areas.

The tessellation engine shares exactly the same data model (e.g., VBOs) with point-based representation and processes on GPU, so data and rendering context switching are not needed. This method is scalable and efficient for out-of-core streaming and saves approximately half of the memory by avoiding textures. We approximate the covered area by splats oriented in the direction of the normal vector. Furthermore, the tessellated splats have assigned the color of the corresponding point samples, and they are rendered with α-texture for blending with neighboring and overlapping primitives.

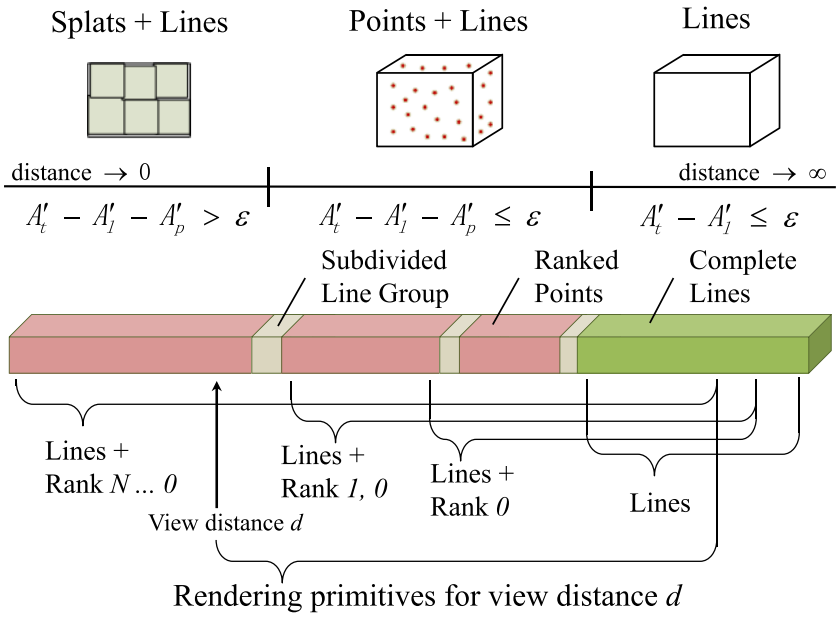


Figure 5. Hybrid representation by using lines, points, and point-based splats.

5.4. Hybrid Representation

After the representations are selected and the number of each primitive is determined, we generate simplified models from ranked sequences by synthesizing their hybrid representation.

Figure 5 shows the process of selecting the primitives from the data structure. We first render a suffix from the ranked sequence of subdivided line groups, using raster area as criterion. Then the completed lines that have not been replaced are rendered, using the corresponding prefix from the ranked line sequence.

Let us denote the sampling quantity of point-based rank level i by n_i. If two neighboring rank levels, r and $r+1$, exist in the ranked-point sequence and they have

$$\sum_{i=0}^{r} n_i \leq n_p \leq \sum_{i=0}^{r+1} n_i \tag{6}$$

we then render the first n_p points from the sequence. Otherwise, the entire point set is processed as tessellated splats.

Points and their normals are sampled from every mesh face. During the rendering, we cull points that are not facing the camera. After that, we use the hardware fast approximate anti-aliasing for anti-aliasing and removing image space high-frequency signals in coarser levels.

6. RESULTS AND EVALUATION

Level-of-detail techniques degrade the visual quality of the output. We evaluate our method by measuring visual quality of the generated images, and we also provide results of the rendering performance. For the performance evaluation, we use a large 3D urban model data set. For comparisons of visual quality and performance, we created five test scenes showing different urban areas with the aim of covering various scenarios. Each scene contains several hundreds of architectural models. The ground truth images are generated by rendering the original model. The input city model was obtained from Digital Qingdao Urban Data Set and provided by the Qingdao Urban Planning Bureau. It contains 5.2-GB geometric models and 42.3 GB of textures.

Table I. Size of tested scenes (GB).

	Scene				
	1	2	3	4	5
Mesh	0.20	0.30	0.30	0.20	0.20
Texture	2.10	1.70	2.40	2.30	1.50
Total	2.30	2.00	2.70	2.50	1.70
Line	0.05	0.06	0.05	0.05	0.04
Point	1.01	0.96	1.10	0.91	0.72

Because the complete scene was not possible to render by the use of triangle meshes, we created five test scenes from the same urban data set showing different urban areas that were renderable (Table I).

6.1. Performance Analysis

Results have been measured on a desktop PC with Intel Xeon E5-2620 at 2.0 GHz with 8-GB of memory and an Nvidia GTX480 GPU with 1.5 GB of memory. We compared our approach with polygonal representation ren-

DOI: 10.1002/cav

Table II. Rendering time per frame (milliseconds).

	Scene no.				
	1	2	3	4	5
Ground truth	423	368	530	504	354
GLoD	149	126	205	241	107
Hybrid	34	32	44	42	31

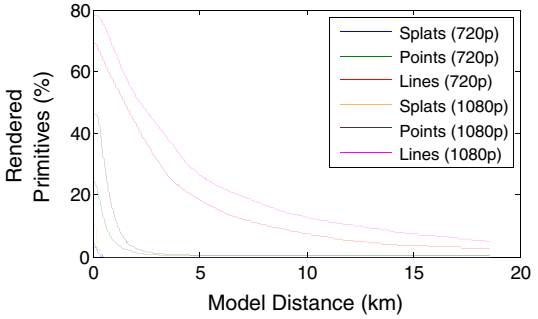


Figure 6. Number of rendering primitives at different view distances.

dering that is commonly used in urban scene visualizations and provides ground truth for our rendering. The large scene did not fit into the GPU memory, so we could not report the speedup. The rendering cost is summarized in Table II, where each value represents the rendering time per frame in milliseconds of a given scene. We rendered uncompressed videos at HD720p resolution, and the duration of each video sequence was 30 seconds. The results show that our approach is about 10× faster compared with ground truth scene rendering and four times faster than the geometric LOD rendering by using [28].

Figure 6 and the accompanying video (supporting information) show the number of rendering primitives at different view distances. The number of rendered points has a substantially faster convergence rate than the number of lines. This means most of the distant models were rendered only as lines. Also, objects that project to a small screen area are rendered by using a smaller number of primitives. Only some close-ups required a higher detail and were rendered by using splats.

We also tested our system by rendering the entire urban data set (Figure 1). The simplified visible model set fits into the GPU memory and was rendered at interactive frame rates by using our approach.

6.2. Perceptual User Study Evaluation

We evaluated the quality of the rendered images by using a psychophysical experiment that assessed if our approach produces changes that are noticeable to the user. We have developed an online evaluation system for the study. A brief demographic survey asking the gender, computer graphics experience, and presence of visual deficiencies, such as glasses, introduced the testing.

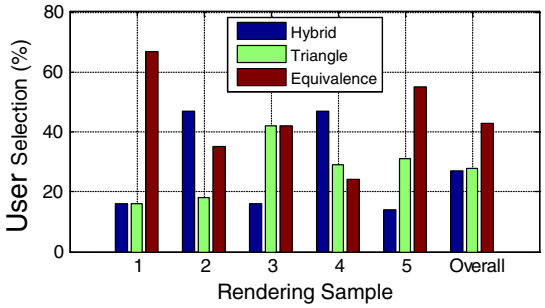


Figure 7. Perceptual user study compared the level of detail with the ground truth.

Table III. Statistics of user study results.

	μ	σ^2	Minimum	Maximum
Hybrid	1.382	1.019	0.00	4.00
Triangle	1.432	1.071	0.00	4.00

We then showed the five video pairs, which compared the rendering results of the five given urban scenes. The test videos were created and played at HD720p resolution, and we carefully removed all the background such as terrain and sky to eliminate bias. Each video pair was divided into left and right, and one scene randomly showed the ground truth, whereas the other showed the simplified scene renderings. Both videos were synchronized, and the users could not interrupt them. The participants were asked, "Which video looks better," and they had three choices as an answer: left, right, and hard to distinguish. There was no time limit to complete the test, so the user could replay the videos for comprehensive evaluation.

We used 81 human subjects, and the aggregate survey results are shown in Figure 7.

To prove if there is a significant difference between the ground truth and the simplified scene, we built a statistical model based on the results of our survey and performed a two-sample paired Wilcoxon signed-rank test with a significance level of 0.05. We created paired samples by counting how many times participants selected the hybrid representation as better and how often they preferred the ground truth. Because each participant reviewed five video pairs, we obtained the individual ratings on a scale $[0, 1, \ldots, 5]$, where 5 indicates the highest visual quality of the specified rendering method.

The result shows that 33 participants ranked the ground truth as better than the hybrid representation, 30 rated the hybrid representation as better, and 18 ranked the two methods as hard to distinguish. A statistical analysis is reported in Table III. The similar mean and rank values indicate that the two rendering methods were perceived to be similar. The low standard deviation indicates that the choices of each user tends to be very close to the mean value.

The null hypothesis is that the visual quality of hybrid representation is equivalent to that of textured triangle

mesh. According to the statistical analysis, the Z score is -0.195, and the p-value is 0.846 by the Wilcoxon signed-rank test. Because the p-value for the two-tailed hypothesis is much greater than the significance level ($\alpha = 0.05$), we can conclude that the Wilcoxon test supports the null hypothesis and shows that *there was no significant difference in visual quality between the two rendering methods in our test.* The simplified representation shows results that are similar to the ground truth.

6.3. Dynamic Range Independent Metric

A further comparison was introduced by using algorithm-based metric Dynamic Range Independent Video Quality Assessment and Dynamic Range Independent Metric Online (DRIVQM) [29] that measure visual differences between two (sequences of) images. The result of this test is a figure with significantly different pixels colored in red, identical pixels in gray, and pixels that are close to a noticeable difference in green. Figure 8 shows in the top row the results of the DRIVQM test of three frames rendered by our hybrid rendering approach compared with the ground truth. Some small areas are significantly different in terms of DRIVQM (displayed in red), but most of the pixels are either identical or on the edge of a hardly noticeable difference. The diversity of pixels increases if we only use point-based representation (middle row) or if we remove the tessellation procedures (bottom row).

We conclude that the DRIVQM test indicates the importance of mixing different methods and using hybrid representation. If point, line, and splat LODs are used simultaneously, they provide a better visual quality. Moreover, there is no significant difference between the ground truth images and the simplified ones if the hybrid representation is used, which is a finding congruent with the user study from the previous section.

6.4. Precision and Recall Metric

The last method we used to evaluate our algorithm was the precision and recall algorithm of [30] that has been applied in pattern recognition and information retrieval.

In this automatic evaluation, true positives are pixels that are rendered exactly the same in the original and in the simplified versions; false positives are pixels that are rendered only for the simplified model; and false negatives are pixels that are rendered in the original model, but not in the tested scene. Precision (p-value) is defined as the ratio of true positives and the sum of true positives and false positives. Recall (R-value) is the ratio of the true positives and all

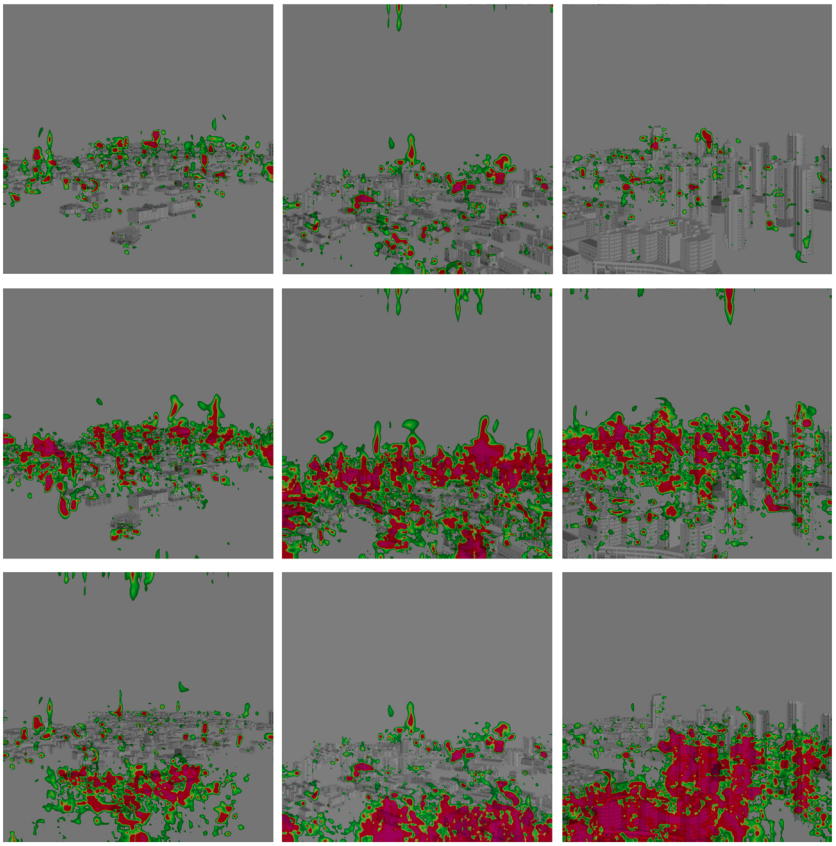


Figure 8. Top: DRIVQM test of the complete hybrid representation (lines, points, and splats); middle: point-based representation; and bottom: the hybrid representation without tessellation.

DOI: 10.1002/cav

Table IV. Precision and recall test results (PR values).

	Video				
	1	2	3	4	5
Points, lines, and splats	0.905, 0.903	0.881, 0.881	0.916, 0.910	0.911, 0.908	0.910, 0.914
Hybrid without lines	0.644, 0.648	0.554, 0.555	0.678, 0.679	0.713, 0.718	0.684, 0.682
Hybrid without splats	0.762, 0.763	0.784, 0.773	0.829, 0.811	0.910, 0.908	0.557, 0.558

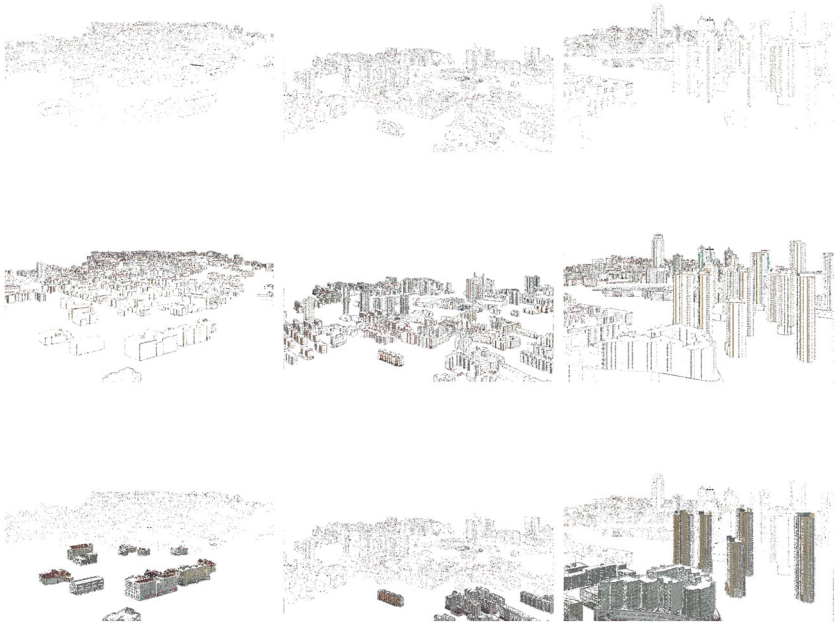


Figure 9. Top: wrongly set pixels in hybrid representation (lines, points, and splats), middle: in point-based representation; and bottom: in hybrid representation without tessellation.

relevant items (sum of true positives and false negatives). That is, the ratio of correctly set pixels and the number of correctly set pixels plus the number of pixels that should have been rendered but are not covered by the simplified model. A p-value $= 1$ and R-value $= 1$ mean that the two images are exactly the same; however, a p-value $= 0$ and R-value $= 0$ mean that the two images are totally different.

We applied the precision and recall algorithm on the five video pairs and compared the rendering results of textured triangle mesh and the hybrid representation. The PR values of each video pair are shown in Table IV, and several examples are shown in the top row of Figure 9.

The PR values from our testing results are very close to the value of 1.0, meaning that the results of the two rendering methods are similar. There are only a few wrongly set pixels, and they probably resulted from sampling stages.

Similar to the DRIVQM, the PR scores worsen significantly if we use point-based representation or remove tessellation procedures only, and the number of wrongly set pixels increases significantly as shown in the second row and the third row of Figure 9. These results support our conclusion from the DRIVQM test that if the three LOD representations are used simultaneously, the results improve in quality.

The wrongly set pixels in the point-based representation mostly occur in the architectural borders and facades near the viewpoint. This indicates that the line rendering is helpful for protecting the building outlines as well as the sharpness that confirms the intuition behind the key observation of this approach.

7. CONCLUSION AND FUTURE WORKS

We have presented a novel LOD method to represent and render massive urban models. The main idea is to render the urban models by using a hybrid representation of lines, points, and splats. Our method achieves interactive frame rates, and we have presented a comprehensive evaluation by using three different evaluation metrics. The evaluations indicate that our method produces visually similar results compared with textured mesh rendering, and we have achieved 10× speedup on our hardware and with our implementation.

There are some limitations to our approach. Our preprocessing step extracts edges directly from the polygonal mesh that causes dependence on the quality of the input. Although the resulting edges are independent on the input, a mesh-independent approach, such as sampling of the input geometry, could be used. Our approach is not suitable for extreme close-ups. A possible way for improving close-ups rendering would be by replacing low-frequency geometries with triangular meshes and seamless integration with splats. Also, impostor-based LOD methods such as [31,32] could be applied, and different perceptual evaluation techniques could be used [33]. A direct comparison between our method and impostor-based LOD is not possible. However, Hilbert and Brunette [32] show that the performance gain for urban scenery was 25%, while our method provides an approximate speedup of 10 times over the polygonal model. In addition, impostor-based LOD methods are not suitable for a closed-up view, and our approach can be more efficient in representing objects that are located far away from the camera, as our approach does not need frequent texture switching and can be completely integrated with GPU.

Also, our approach extracts only the silhouette edges from the input. However, there may be significant edges inside the facades that could also be extracted and used, thus decreasing the amount of point samples.

ACKNOWLEDGEMENTS

The implementation of precision and recall algorithm is provided by the Work Group Computer Graphics and Media Design, University of Konstanz. This work was supported by Projects of International Cooperation and Exchanges NSFC (613111035).

REFERENCES

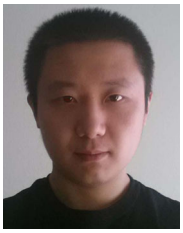
1. Burr DC, Morrone MC, Spinelli D. Evidence for edge and bar detectors in human vision. *Vision Research* 1989; **29**(4): 419–431.
2. Clark JH. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM* 1976; **19**(10): 547–554.
3. Jones CB. A new approach to the hidden line problem. *The Computer Journal* 1971; **14**(3): 232–237.
4. Luebke D, Georges C. Portals and mirrors: simple, fast evaluation of potentially visible sets. In *Proceedings of I3D*. ACM Press: New York, NY, USA, 1995; 105ff.
5. Teller SJ, Séquin CH. Visibility preprocessing for interactive walkthroughs. *SIGGRAPH Computer Graphics* 1991–07; **25**(4): 61–70.
6. Airey JM, Rohlf J, Brooks FP. Towards image realism with interactive update rates in complex virtual building environments. *SIGGRAPH* 1990; **24**(2): 41–50.
7. Downs L, Möller T, Séquin CH. Occlusion horizons for driving through urban scenery. In *Proceedings of I3D*. ACM Press: New York, NY, USA, 2001; 121–124.
8. Hoppe H. Progressive meshes. In *Proceedings of SIGGRAPH*. ACM Press: New York, NY, USA, 1996; 99–108.
9. Sander P, Snyder J, Gortler S, Hoppe H. Texture mapping progressive meshes. In *Proceedings of SIGGRAPH*. ACM Press: New York, NY, USA, 2001; 409–416.
10. Varadhan G, Manocha D. Out-of-core rendering of massive geometric environments. In *Visualization, 2002. VIS 2002*. IEEE Computer Society Press: Washington DC, USA, 2002; 69–76.
11. Wald I, Dietrich A, Slusallek P. An interactive out-of-core rendering framework for visualizing massively complex models. In *ACM SIGGRAPH 2005 Courses*. ACM Press: New York, NY, USA, 2005, Article No: 17, http://dl.acm.org/citation.cfm?id=1198756sss.
12. Gobbetti E, Marton F. Far voxels: a multiresolution framework for interactive rendering of huge complex 3D models on commodity graphics platforms. *ACM Transactions on Graphics* 2005; **24**(3): 878–885.
13. Yoon S, Salomon B, Gayle R, Manocha D. Quick-VDR: out-of-core view-dependent rendering of gigantic models. *IEEE TVCG* 2005; **11**(4): 369–382.
14. Sillion F, Drettakis G, Bodelet B. Efficient impostor manipulation for real-time visualization of urban scenery. In *Computer Graphics Forum*, Vol. 16. Eurographics Association Press: Aire-la-Ville, Switzerland, Switzerland, 1997; 207–218.
15. Maciel PW, Shirley P. Visual navigation of large environments using textured clusters. In *Proceedings of I3D*. ACM Press: New York, NY, USA, 1995; 95–ff.
16. Decoret X, Sillion F, Schaufler G, Dorsey J. Multi-layered impostors for accelerated rendering. *Computer Graphics Forum* 1999; **18**(3): 61–73.
17. Decoret X, Durand F, Sillion F, Dorsey J. Billboard clouds for extreme model simplification. *ACM Transactions on Graphics* 2003; **22**(3): 689–696.
18. Wimmer M, Wonka P, Sillion F. Point-based impostors for real-time visualization. In *Proceedings of the 12th Eurographics Conference on Rendering*. EGWR'01. Eurographics Association Press: Aire-la-Ville, Switzerland, Switzerland, 2001; 163–176.
19. Cignoni P, Benedetto MD, Ganovelli F, Gobbetti F, Marton F, Scopigno R. Ray-casted blockmaps for large urban models visualization. *Computer Graphics Forum* 2007; **26**(3): 405–413.
20. Andujar C, Diaz J, Brunet P. Relief impostor selection for large scale urban rendering. In *IEEE*

Virtual Reality Workshop on Virtual Citiscapes: Key Research Issues in Modeling Large-Scale Immersive Urban Environments, Reno, Nevada, USA, 2008, http://upcommons.upc.edu/e-prints/handle/2117/15881.
21. Deussen O, Colditz C, Stamminger M, Drettakis G. Interactive visualization of complex plant ecosystems. In *Proceedings of Visualization*. IEEE Computer Society Press: Washington DC, USA, 2002; 219–226.
22. Merhof D, Sonntag M, Enders F, Nimsky C, Hastreiter P, Greiner G. Hybrid visualization for white matter tracts using triangle strips and point sprites. *IEEE TVCG* 2006; **12**(5): 1181–1188.
23. Cohen J, Aliaga D, Zhang W. Hybrid simplification: combining multi-resolution polygon and point rendering. In *Proceedings of Visualization*, San Diego, California, USA, 2001; 37–539.
24. Guthe M, Borodin P, Balázs A, Klein R. Real-time appearance preserving out-of-core rendering with shadows. In *Proceedings of EGSR*, Konstanz, Germany, 2004; 69.
25. Kopf J, Cohen-Or D, Deussen O, Lischinski D. Recursive Wang tiles for real-time blue noise. In *ACM SIGGRAPH*. ACM Press: New York, NY, USA, 2006; 509–518.
26. Jameson D, Hurvich LM. Theory of brightness and color contrast in human vision. *Vision Research* 1964; **4**: 135.
27. Connolly C, Fleiss T. A study of efficiency and accuracy in the transformation from RGB to CIELAB color space. *IEEE Transactions on Image Processing* 1997; **6**(7): 1046–1048.
28. Cohen J, Luebke D, Duca N, Schubert B. Glod: a geometric LOD system at the OpenGL API level. In *Proceedings of Visualization*, Seattle, Washington, USA, 2003; 85.
29. Aydin TO, Čadík M, Myszkowski K, Seidel H. Video quality assessment for computer graphics applications. In *ACM TOG*, Seoul, South Korea, 2010, Article No: 161, http://dl.acm.org/citation.cfm?id=1866187.
30. Wald I, Dietrich A, Slusallek P, Neubert B, Pirk S, Deussen O, Dachsbacher C. Eurographics Symposium on Rendering (Poster Session), Saarbrücken, Germany, 2010.
31. Day AM, Willmott J. Compound textures for dynamic impostor rendering. *Computers & Graphics* 2005; **29**(1): 109 –124.
32. Hilbert K, Brunnett G. A hybrid LOD based rendering approach for dynamic scenes. In *Proceedings of Computer graphics international, 2004*, Grete, Greece, June 2004; 274–277.
33. Hamill J, McDonnell R, Dobbyn S, O'Sullivan C. Perceptual evaluation of impostor representations for virtual humans and buildings. *Computer Graphics Forum* 2005; **24**(3): 623–633.

SUPPORTING INFORMATION

Supporting information may be found in the online version of this article.

AUTHORS' BIOGRAPHIES



Shengchuan Zhou is a Ph.D. student in the College of Information Science and Engineering at Ocean University of China. He was a visiting researcher at Purdue University in 2012. He received his M.S. degree from Ocean University of China and B.S. degree from Yantai University. His current research interests include urban simulation, image-based modeling and rendering, real-time 3D graphics, and GPU programming.

Innfarn Yoo is a Ph.D. student at Purdue University in the Department of Computer Graphics Technology. He received his Masters degree from Purdue and a bachelor's degree from Konkuk University in South Korea, majoring in mathematics. He also has 5 years of work experience in the field of gaming industry. He is recently concentrating on animation, photo-realistic rendering, and parallel programming.

Bedrich Benes is an Associate Professor and Purdue Faculty Scholar in the Computer Graphics Technology department at Purdue University. He obtained his Ph.D. and M.S. degrees from the Czech Technical University. His research is primarily in the areas of procedural modeling, real-time rendering, and 3D computer graphics in general. To date, he has published more than 70 peer-reviewed publications.

Ge Chen is a Professor of Satellite Oceanography and Marine Information Technology and Dean of the College of Information Science and Engineering at Ocean University of China. He obtained his Ph.D. and B.S. degrees from the Ocean University of China. His current research interests include geographic information system, virtual reality, and virtual geographical environment.