

View-Dependent Visibility Estimation for Tree Models

Jessy Lee¹, Jingliang Peng² and C.-C. Jay Kuo¹

¹Ming Hsieh Department of Electrical Engineering and Signal and Image Processing Institute
University of Southern California, Los Angeles, CA 90089-2564

²Department of Computer Science, Sun Yat-sen University, Guangzhou 510275, P. R. China
E-mails: jessylee@usc.edu, jingliap@gmail.com, cckuo@sipi.usc.edu

Abstract—A view-dependent visibility estimation technique for tree models is proposed in this work. While most previous work focused on visibility estimation of large objects in architectural walkthroughs, we consider the visibility estimation problem of tree leaves in complex natural scenes such as forests. Exact calculation of how each leaf object blocks another from a given viewpoint is difficult due to the large amount of computational cost. To address this issue, we propose three simple yet effective methods for efficient visibility estimation of tree leaves: the distance-based, the normal-based, and the layer-plus-normal-based methods. Experimental results show that the distance-based method always produces the best analytical performance, the normal-based method works better in preserving volumetric visual quality at low budget constraints, and the layer-plus-normal-based method preserves volumetric visual quality while yielding excellent analytical performance.

I. INTRODUCTION

Rendering efficiency has always been a fundamental issue in computer graphics. Attempts have been made to achieve such a goal by developing faster hardware as well as more intelligent rendering algorithms. Hidden surface determination is one of the techniques that address this problem for 3D applications. The study is geared towards reducing the number of geometry primitives to be rendered while striving to preserve the original visual quality.

Algorithms were proposed for this purpose, yet scenes tested by them were mostly in an architectural environment or indoor scenes. One observation found in these scenes is that certain primitives can potentially occlude a large number of other primitives, since they tend to be opaque and large in area. This is especially beneficial when modern scene complexity keeps increasing, as reflected in the size of the data set. As a result, assumptions in most algorithms previously proposed are made based on large occluders.

However, such notions may not provide equally satisfying results if the scenes are filled with small entities. Objects such as cloud water droplets, sand, hair, grass and tree leaves are generally small in size. Each object is less likely to be capable of occluding a large set of other 3D models. In contrast with traditional architectural walkthroughs where a large polygon can occlude a large number of others, the tree model of our interest has numerous small spaces between small leaf objects. When a particular area with a high polygon count is being rendered, the regions behind it also need to have some degrees of visibility instead of being set to completely invisible. It is apparent that calculating exactly how leaves block another from the viewpoint is computationally too expensive to be practical, and new visibility estimation techniques are needed.

A view-dependent visibility estimation technique for tree models is proposed to address this challenge in this work. To enable real-time view-dependent visibility estimation, a practical algorithm should be simple yet effective. For a given viewpoint and a given leaf budget, each algorithm determines the priority (or visibility) of each leaf which represents its likelihood of being visible, and selectively renders those leaves with the highest priority until the leaf budget is reached.

The rest of this paper is organized as follows. After a brief review of previous work in Sec. II, we propose three simple yet effective methods for efficient visibility estimation of tree leaves in Sec. III. They are the distance-based, the normal-based, and the layer-plus-normal-based methods. Experimental results are shown in Sec. IV. Finally, concluding remarks are given in Sec. V.

II. REVIEW OF PREVIOUS WORK

To achieve real-time rendering in walkthrough applications, two strategies have been adopted: simplification of geometry and selection of the visible set.

Geometric simplification approaches have been used to reduce the computational cost at run time. In particular, level-of-detail (LOD) algorithms were proposed to render 3D objects in different resolutions based on varying viewing parameters. Gross *et al.* [1] and Lindstrom *et al.* [2] used adaptive LOD techniques to provide view-dependent representations for terrains. Hoppe [3] developed progressive meshes to generate continuous LODs of an input model through a sequence of edge collapses. In general, three major classes of techniques have been adopted in progressive mesh simplification: vertex removal, vertex clustering, and edge collapse. Upon generating different LODs for a mesh, viewing distances to certain areas of the mesh are used to select the appropriate LOD at run time, which achieves a good balance between visual quality and the computational cost. These solutions work well with terrains and large models with reasonable topological complexity.

Instead of generating appropriately simplified 3D models, an estimation of visibility can be performed on geometrical primitives in a region of space, which specifies the priority in rendering of each geometry primitive. Potentially visible sets (PVS) are calculated in many algorithms to determine whether to render certain primitives in a scene. At run time, primitives in PVS are rendered while primitives not in PVS are not set to be visible to save the computational complexity. An opacity value for a region of space can be computed to obtain an approximated visibility.

The above concept was first proposed by Sillion [4] in an effort to speed up computations for radiosity systems. Klosowski and Silva [5] proposed a prioritized-layered projection (PLP) algorithm to perform approximation for occlusion culling techniques. The decision to render a certain polygon is based on the order of projection in an aggressive culling fashion. The algorithm starts with creating several cell partitions in the scenes being rendered. A probabilistic value is calculated indicating how likely the cell will block other cells in the same viewing direction, and one heuristic used to determine a cell’s solidity is by examining how many primitives it includes. Cells are arranged inside a priority queue that places most likely to be visible cells in the first position of the queue. El-Sana *et al.* [6] integrates the PLP approach with LOD calculation to prevent occluded regions to be rendered in high LODs. The types of scenes addressed in these papers are architectural environments and indoor scenes. They provide inspiring rendered results yet none of them try to address scenes with small objects such as leaves, grass, or hair.

III. VIEW-DEPENDENT VISIBILITY ESTIMATION

To facilitate interactive rendering of models with a large number of small primitives and high topological complexity (*e.g.* tree models), we propose in this section three methods for visibility estimation of tree leaves. They are the distance-based, the normal-based, and the layer-plus-normal-based methods as detailed below.

A. Distance-based Method

For a leaf directly facing the viewpoint, the closer it is to the viewpoint, the bigger screen area it will occupy in the rendered image and the less probable that it will be occluded by other leaves. Based on this observation, a straightforward way is to use the Euclidean distance between each leaf and the viewpoint to determine that leaf’s visibility. Leaves with smaller Euclidean distances to the viewpoint are given higher visibility and rendered earlier. As leaves reside farther away from the viewpoint, they are more likely to be occluded by other leaves in between, thus given lower priority and rendered later. The rendering process will stop when the leaf budget (*i.e.* the number of leaves to be rendered) is reached.

B. Normal-Based Method

In addition to the Euclidean distance, the leaf orientation can serve as another metric to approximate the visibility of a leaf. Unlike architectural models, a leaf model seldom occludes all leaves that reside behind the leaf away from the viewpoint. The likelihood of a leaf blocking or being blocked by others often depends on the orientation that it is placed on a tree branch. For a leaf at a fixed position, the more perpendicular its surface is oriented with respect to the viewing direction, the bigger screen image it will produce, the greater its chance of occluding others and the smaller its chance of being completely occluded by others. Thus, in the normal-based method, we estimate the visibility of a leaf according to its orientation relative to the viewing direction.

Such relationship can be quantified using the surface normal of the leaf face with respect to the viewing direction. A visibility value for a leaf is calculated by taking the dot product of the leaf surface normal with the viewing direction. Leaves with high dot product values are assigned higher visibility values, and vice versa.

C. Layer-plus-Normal-Based Method

Through experiments, we have identified advantages and disadvantages of the distance-based and the normal-based methods. The distance-based method produces good results at high leaf budgets, but tends to create holes and empty branches especially in boundary regions at low leaf budgets due to its high bias toward branches closer to the viewpoint. On the contrary, the normal-based method preserves the tree profile better at low leaf budgets, but results in worse image quality at high leaf budgets since it allocates much of the leaf budget to far distance leaves which have very high probability of being occluded by closer leaves. These observations have inspired our exploration of a new method, called the layer-plus-normal-based method, that takes into account both the distance and the normal parameters in visibility estimation.

In this layer-plus-normal-based method, we first partition all tree leaves into several layers based on a certain distance metric. Then, we dynamically distribute the total leaf budget to all layers, with more budget allocated to closer layers. Within each layer, the normal metric is used to pick up the leaves to render within the allocated leaf budget. By always distributing the total leaf budget among layers, we strive to relieve the high distance bias, thus the hole and empty branch problem associated with the distance-based method. By allocating a higher leaf budget to closer layers, we aim to improve the high-leaf-budget image quality as compared with the normal-based method which tends to allocate a higher leaf budget to highly occluded leaves. In this new method, two key issues arisen are how to group leaves into layers, and how to distribute the total leaf budget into layers.

To partition leaves into layers, we calculate the distance of each leaf to the viewpoint, and group leaves within the same range of distances into the same layer. For the distance metric, we do not choose the Euclidean distance. The reason is that it tends to assign low visibility values for leaves around the tree profile, since profile areas tend to have bigger Euclidean distance to the viewpoint. As a result, at a low leaf budget, noticeable gaps and empty branches may be observed around profile regions. Instead, we use the projection distance for layer partitioning. As shown in Fig. 1, a sequence of parallel projection planes $P_{i,i=\{1,2,3\}}$ are placed across the tree model. Each projection plane is perpendicular to the viewing direction. The distance from the viewpoint to each P_i is represented as d_i . A leaf model is set to associate with projection layer G_i if its projected distance on the view direction falls between d_i and d_{i+1} .

The leaf budget allocation is based on the projection distance of each layer. In general, smaller leaf budgets will be assigned to layers with larger projection distances. In our implementation, the leaf budget, l_i , allocated to layer G_i is

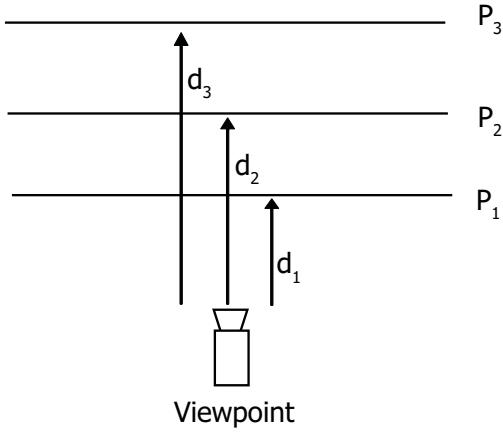


Fig. 1. A tree model is divided into several parallel projection planes, where each plane is perpendicular to the viewing direction.

calculated as

$$l_i = \frac{G - i + 1}{\sum_{k=1}^G k} \times L, \quad i = 1, \dots, G, \quad (1)$$

where G is the total number of layers, L is the total number of leaves in a tree. Note that a layer with a small index i has a smaller projection distance d_i to the viewpoint. If a layer is allocated with less number of leaves than it currently contains, each leaf's normal vector is used to determine whether a leaf is rendered using the method discussed in Sec. III-B. In some cases, layer G_i may be allocated a leaf budget, l_i , more than the number of leaves, n_i , it contains. If this occurs, all leaves in G_i are set to visible and the remaining quota $l_i - n_i$ are added to l_{i+1} .

It is worthwhile to point out that it is usually unaffordable to perform visibility estimation on a leaf-by-leaf basis in real-time walkthrough rendering. Instead, leaves should be clustered into groups based on certain geometrical characteristics, and the visibility estimation should be dynamically performed on groups instead of individual leaves. However, we have implemented our algorithms on a leaf-by-leaf basis and under a static viewpoint setting in this work just in order to demonstrate and compare the effectiveness of the proposed visibility estimation algorithms. When being performed at a coarser granularity, the proposed methods should easily adapt to real-time applications.

IV. EXPERIMENTAL RESULTS

A tree model with 6,143 leaves, as shown in Fig. 2(a), is used in our experiments to compare the performance of three proposed methods. Rendering only a certain percentage of the total number of polygons in a model helps reduce the total rendering time naturally. Under the same budget value, an algorithm may not choose which leaf to render as intelligently as others. The eventual goal is to select the algorithm that produces results closest to the full model yet the rendering cost remains the same as others. In our experiments, the three methods are compared analytically and visually.

A. Analytical Comparison

To compare rendered images analytically, a simple yet commonly used approach is adopted. That is, we calculate incorrect pixels in rendered images against that being rendered with all original leaves in the tree under a certain budget leaf counts. The percentages of incorrect pixels generated by the three methods discussed in the last section are compared in Table I, where the budgets are set to 10%, 30%, 50%, and 80% of the total number of leaves. We see from the table that all three methods are able to produce results fairly similar to the full tree with all leaves visible at a budget value of 80%. As the budget value decreases, significant increases in the incorrect pixel count are observed. The statistics show that the distance-based methods produces results with the least number of incorrect pixels among the three. The normal-based method has the worst performance while the layer-plus-normal-based method has performance lying in-between the other two.

TABLE I
COMPARISON OF THREE VISIBILITY ESTIMATION METHODS

Algorithm	Budget Value			
	80%	50%	30%	10%
Distance-based	3.03%	9.09%	14.03%	18.42%
Normal-based	5.31%	12.87%	16.49%	19.13%
Layer-plus-Normal-based	2.86%	10.06%	15.07%	18.75%

B. Visual Comparison

Even though the distance-based method is able to produce best results in terms of the number of incorrect rendered image pixels, it fails to capture human's perception of natural trees as the budget value decreases. Fig. 2(a) shows the full tree model by setting all leaves to be visible. Fig. 2(b)(c)(d) are results from the tree model rendered at budget value of 50% using the distance-based, the normal-based and the layer-plus-normal-based methods, respectively. Visual perception tells that they resemble more closely to the full tree because more leaves are set to be visible as compared to trees rendered at a lower budget value in Fig. 2(e)(f)(g), where the budget value is set to 30%. The tradeoff is sacrificing the rendering speed for better visual quality.

Fig. 2(e) shows the tree rendered using the distance-based method. Some significant visual artifacts can be observed indicating that some regions of leaves are missing. On the other hand, result from the normal-based method in Fig. 2(f) still manages to keep the overall silhouette of the tree model. Finally, the layer-plus-normal-based method, as shown in Fig. 2(g), is able to preserve the tree's overall silhouette while still keeping incorrect pixel counts low as compared to the normal-based method.

To conclude, the distance-based method yielded rendered results closest to the full tree image in terms of the incorrect number of pixels in our experiments. However, this method has the worst performance under human perception at low leaf budgets. Visual results show that the distance-based method loses the volumetric feel of a natural tree at low leaf budgets because several regions of leaves are noticeably missing.



Fig. 2. (a) Full tree with all leaves rendered (b) Distance-based with 50% of the leaves rendered (c) Normal-based with 50% of the leaves rendered (d) Layer-plus-normal-based with 50% of the leaves rendered (e) Distance-based with 30% of the leaves rendered (f) Normal-based with 30% of the leaves rendered (g) Layer-plus-normal-based with 30% of the leaves rendered

Choosing to render leaves based on each leaf's normal direction is a better solution to preserve the natural volumetric properties in trees. Lastly, the layer-plus-normal-based method is able to keep the volumetric look of a tree while keeping the number of incorrect pixel counts low, which is as competitive as the distance-based method.

V. CONCLUSION AND FUTURE WORK

Since leaf objects are small in size, occlusion-based culling is unrealistic for the purpose of real-time rendering. Instead, visibility estimation techniques should be used. To this aim, three methods were proposed, implemented and experimented to compare the effectiveness of estimating visibility information for tree models in this work. In the future, we plan to adapt and experiment the proposed methods in real-time walk-through environments. To accelerate the visibility estimation, we may cluster leaves into groups, and perform our algorithms on a group-by-group basis. In addition, hierarchical clustering may be utilized to further improve computational efficiency. We may also investigate advanced rendering techniques with the help of visibility estimation techniques. For instance, we

may estimate an transparency value for each group of leaves and use image-based rendering and alpha-blending to achieve realistic and real-time rendering. It is also interesting to explore different methods for leaf clustering, corresponding to concrete visibility estimation and rendering algorithms adopted.

REFERENCES

- [1] M. Gross, R. Gatti, and O. Staadt, "Fast multiresolution surface meshing," in *IEEE Visualization '95*, October 1995, pp. 135–142.
- [2] P. Lindstrom, D. Koller, W. Ribarsky, L. Hughes, N. Faust, and G. Turner, "Real-time, continuous level of detail rendering of height fields," in *SIGGRAPH '96 Conference Proceedings*, August 1996, pp. 109–118.
- [3] H. Hoppe, "View-dependent refinement of progressive meshes," in *SIGGRAPH '97 Conference Proceedings*, August 1997, pp. 189–197.
- [4] F. Sillion, "A unified hierarchical algorithm for global illumination with scattering volumes and object clusters," in *IEEE Transactions on Visualization and Computer Graphics*, September 1995, pp. 240–254.
- [5] J. Klosowski and C. Silva, "The prioritized-layered projection algorithm for visible set estimation," in *IEEE Transactions on Visualization and Computer Graphics*, April 2000, pp. 108–123.
- [6] J. El-Sana, N. Sokolovsky, and C. Silva, "Integrating occlusion culling with view-dependent rendering," in *Proceedings of the conference on Visualization '01*, October 2001, pp. 371–378.