Memory Efficient Billboard Clouds for BTF Textured Objects

Jan Meseth, Reinhard Klein

Institute for Computer Science II
Computer Graphics Group
University of Bonn
Römerstr. 164, 53117 Bonn, Germany
Email: {meseth,rk}@cs.uni-bonn.de

Abstract

Efficiently rendering highly structured models distant from the viewer constitutes a difficult task since the geometric complexity has to be reduced extremely while simultaneously preserving the overall visual quality. Recently, billboard clouds have been introduced as an new solution to this problem. They achieve acceptable performance by coarsely approximating the geometry of a model while storing surface details as textures. Yet, important surface detail due to self-shadowing, reflectance properties and changing silhouettes is lost.

In this paper we introduce Bidirectional Texture Function (BTF) textured billboard clouds which drastically increase the visual quality by preserving view- and light-dependent effects like reflection properties, changing silhouette and changing shadows while preserving the fast rendering performance. In order to utilize the gains of BTFs, we propose two new methods for generation of memory efficient billboard clouds optimized for connected and unconnected models that preserve the normals of the model much better than previous approaches.

1 Introduction

Objects with very complex structure and appearance are highly common in everyday life and therefore contained in many scenes modelled and rendered in computer graphics. Experiencing virtual life e.g. using terrain rendering requires visualization of highly complex landscapes with mountains, ridges and many more geographical features. In addition, natural scenes usually contain vast amounts of vegetation like grass, bushes and trees. Modelling and rendering such complex scenes at full detail currently exceeds the capabilities of computer graphics systems by far and most likely the increasing capabilities of future hardware will not satisfy the ever-growing needs of increased realism.

For many years already, researchers have focused their work on reducing the complexity of rendered scenes. One of the standard approaches to reduce the rendered data is to adjust the complexity of the displayed model to the perceivable quality, which is called level of detail (LOD). Sophisticated and highly efficient methods have been proposed for displaying models of varying complexity at varying distance. Although these methods follow very different approaches, all of them try to optimally balance the amount of rendered primitives and additional memory e.g. used for textures against the achievable rendering quality.

Recently Décoret et al. [9] proposed billboard clouds (BCs) as an efficient rendering representation. The basic idea of the approach is to coarsely approximate the geometry of a rendered object by a set of quads and encoding fine-grained detail like surface roughness, surface color or the silhouette in normal and color textures. Since current graphics boards are highly optimized to handle textures, such representations can be rendered very efficiently. Compared to other approaches, BCs achieve limited appearance preservation while at the same time requiring only very few geometric primitives to be rendered. A huge advantage of this method is that it can handle arbitrary polygon soups and even point clouds unlike most standard simplification algorithms.

Like all image-based approaches, BCs tend to use much texture memory. Employing the BC generation method proposed by Décoret et al. this amount may well be a couple MBs for models projected to at most 100×100 pixels. Nevertheless,

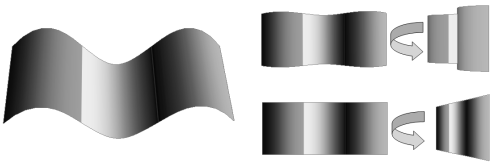

Figure 1: View-Independence Problem. The figure shows a curved surface (left and top), compared to an approximating textured quad (bottom). While the colors and the silhouette of the quad are correct for the frontal view (middle), they become incorrect for other view directions (right).

alternative representations like point-clouds require similar amounts of memory to achieve comparable visualization results[1].

However, existing BC schemes have deficiencies as well. The first will be called *view-independence* problem in the remainder of the paper: view- and light dependent reflectance properties cannot be represented by a single texture and are therefore lost in the BC representation. In addition, other effects like view-dependent silhouettes and occlusions or light-dependent self-shadowing due to small surface detail are not preserved (compare figure 1).

A second problem is the *normal sampling* problem: the normals of approximating quads may be completely different than the normals of the approximated faces which can lead to missing pixels in the resulting image (see figure 3).

In this paper, we propose novel BC construction algorithms optimized for connected meshes and polygon soups that improve the abovementioned problems. The normal sampling problem is solved by providing explicit control over normals represented by a plane, the view-independence problem is significantly improved by utilizing view- and light-dependent textures (i.e. Bidirectional Texture Functions (BTFs)). Since BTFs require even more storage than textures, we additional control the amount of required texture memory. In addition, our methods generate hierarchical BCs which can be employed for LOD rendering and easily extend to BC generation from point clouds.

The following text is structured as follows: in section 2 we describe related research results and distinguish our work from existing publications. Section 3 provides details on our solution to the view-independence problem. Section 4 describes and analyzes an existing BC construction technique before introducing our new methods. In section 5 we present results of our methods before we conclude.

[1] Our tests showed that rendering the plant model in figure 4 using QSplat [25] requires about 120k points for a displayed image of about 100×100 pixels resulting in about 500 kB of memory for storage of the points and their normals.

2 Related Work

2.1 Appearance Preserving Extreme Simplification

Appearance preserving extreme simplification is concerned with generating LOD representations containing a few hundreds of polygons that well represent the appearance of the original object for very distant viewing. Approaches in this area are typically image-based since they produce LOD representations sensitive to the projected screen size and not to the geometric complexity of the input geometry.

Among the earliest image-based approaches are static impostors, proposed by Maciel and Shirley [20], which replace large parts of the geometry by a single textured polygon. The approaches of Schaufler et al. [29] and Shade et al. [32] dynamically update the texture to match the current viewpoint. Later approaches aimed at improvings parallax effects using layered impostors [30, 8], layered depth images [33, 24] or more complex, textured geometry [37, 15] which makes single impostors valid or acceptable for a larger set of viewpoints at the cost of increased texture, geometry and rendering complexity.

The recent approaches of Décoret et al. [9] and Andujar et al. [2] introduce and utilize the already described concept of BCs. Unlike most previous methods this solution is view-independent making it very efficient for real-time rendering. Unfortunately, the presented methods generate BCs which require much texture memory and fail to solve the abovementioned normal-sampling and view-dependence problems. Another view-independent approach is followed by Decaudin and Neyret [7]: they sample objects into 3D textures which are efficiently rendered using volume visualization techniques. Although highly efficient rendering is possible, 3D textures require even larger amounts of texture memory.

Standard geometric simplification algorithms (for an overview see [19]) are highly efficient for objects containing large or moderate numbers of triangles but usually fail to reproduce the appearance of extremely simplified versions due to complex silhouettes and surface details. Among the notable exceptions are appearance preserving simplification [5], which controls the deviation of texture placement in the original and simplified model, illumination-dependent refinement [18], which dynamically increases the polygon count in highlight regions, resampling of surface details into textures [4] and the silhouette clipping approach of Sander et al. [26], which additionally corrects the silhouette of the textured polygonal model. Unfortunately, none of them produces good results for meshes of arbitrary topology and complexity.

Geometric simplification targeted at point-based rendering [12, 25, 23] represents another group of methods for extreme simplification. Like the triangle-based simplification approaches the number of primitives of simplified objects is related to the complexity of the input model, making them less performant than image-based methods. Yet, the optimal adjustment to the screen resolution is much easier, making them applicable more widely than specialized algorithms for extreme simplification. An interesting combination of point-based and image-based rendering was employed by Wimmer et al. [39] which utilzes point-clouds with view-dependent colors for realistic visualizations of distant architectural models but still suffers from a relatively large number of points required for sufficient visual quality.

2.2 Geometry Clustering

Memory efficient billboard construction implies finding an optimal set of textured quads such that the appearance of the object is best preserved at minimal costs. In other words: efficient billboard-clouds represent an optimized set of possibly overlapping clusters of geometry where each cluster is well approximated by a textured quad.

Finding optimal clusters of geometry has been the topic of various publications. Focusing on simplification of connected triangular meshes Kalvin and Taylor [17] presented a bottom-up approach for merging adjacent, planar faces. The merge-criterion is based on thresholding the maximum normal deviation and maximum geometric error with respect to a best-approximating plane, and additional threshold constraints ensuring compact shape and avoidance of foldovers. A very similar approach was applied to the field of mesh generation by Sheffer et al. [34]. A following publication of Inoue et al. [14] introduced an ordering scheme for the merge operation based on a weighted sum of terms measuring flatness, boundary smoothness and merge area resulting in a deterministic algorithm. Unfortunately, choosing appropriate weights is highly unintuitive and requires various tries. The improved approach of Sheffer [35] allows clustering into smooth, not necessarily planar regions but suffers from the limitation to connected models and the unintuitive selection of weights. Although targeted at a different application area, the publications of Garland et al. [11] and Sander et al. [27] employ very similar ideas to decompose a connected mesh into separate charts suitable e.g. for parametrization. Like related approaches, their merge-criteria are computed as a weighted sum of components.

Clustering approaches based on top-down strategies are frequently used in computer graphics employing spatial data structures like kd-trees, octrees or grids. One such approach is clustering of point-clouds with normals and colors for efficient rendering [16] but many others exist as well.

3 BTF Textured Billboard Clouds

Billboard clouds (BCs) are textured quads that approximate the geometry and surface detail of a model at a coarse LOD. Besides convincing renderings of the simplified model, this representation provides the possibility to efficiently cast approximate shadows since the silhouette of the object is well preserved.

Nevertheless, since the silhouette is view-dependent, such shadows will be incorrect for most light-directions using simple textures. In addition, changing surface appearance due to reflectance properties of the surface material, occlusion and interreflection cannot be preserved.

Such information can efficiently be stored as a view- and light-direction dependent texture which is called a Bidirectional Texture Function (BTF) and which was first introduced by Dana et al. [6]. Figure 2 shows the increased quality of BTF textured BCs compared to standard textured ones.

Unlike the original intention of the BTF, which

Figure 2: Comparison of standard BC with texture and normal map (left) with BTF textured one (middle) and original model (right). The resolution of the texture and BTF are optimized for distant viewing (small images).

stores reflectance for a flat sample and therefore requires sampling of view- and light-directions on the hemisphere only, the BTFs used for representations of BCs represent non-flat regions and therefore sampling of the complete sphere (for view-directions this can be neglected since rendered primitives will be invisible for view-directions outside the hemisphere). Hence, for this purpose we generalize the BTF to a 6D function $BTF(x, y, \theta_v, \phi_v, \theta_l, \phi_l)$ of the surface location (x, y), view direction (θ_v, ϕ_v) and light-direction (θ_v, ϕ_v) (represented both by spherical coordinates) where $\theta_v \in [0, \frac{\pi}{2}]$, $\theta_l \in [0, \pi]$, and $\phi_v, \phi_l \in [0, 2\pi[$. Construction of such BTFs can efficiently be done using either rasterization hardware as in [9] for changing view- and light-directions in combination with a shadowing algorithms or using a simple raytracer. Resulting BTFs feature accurate sampling, which is especially important for e.g. forest scenes since trees contain nearly arbitrarily oriented leaves (see figure 8).

Since BTFs require much more memory than simple textures, one needs to choose a reasonable compression algorithm for rendering (we chose the algorithm from Müller et al. [21]) and an efficient scheme for BC construction that minimizes the number of texels required for coding of surface detail. Such schemes reduce the amount of memory produced during BTF construction (which ranges up to several GBs) and reduce the compression error since additional texels tend to introduce additional variance into the BTF data.

4 Billboard Cloud Construction

Constructing optimal BCs is a very difficult since computationally expensive task. One needs to concurrently minimize:

1. the amount of memory for the textures,
2. the number of primitives that approximate the geometry, and
3. the loss of visual quality of the rendered model (which includes minimizing the view-independence and normal-sampling problems).

Since generation of optimal BCs is an NP hard problem [9], an optimal solution is unpractical. Therefore, in the following subsections we will first discuss an existing greedy algorithm that computes a solution to the problem. Afterwards, we will present our new improved approaches and compare them to the first one.

4.1 Hough Space

Décoret et al. [9] suggest to build BCs using the 3D equivalent of the Hough transform [13]: a plane is represented by the spherical coordinates (θ, ϕ) of its normal and its distance d to the origin. All faces of the original mesh are inserted into a regular grid which represents s spatial subdivision of 3D Hough space. For each cell C of the grid *valid* and *missed* faces are determined. A face F is considered valid with respect to C if there exists a plane $P \in C$ such that the Euclidean distance between P and the vertices of F is smaller than the prescribed approximation error ϵ_a. F is considered missed if there exists a plane $P \in C$ such that the distance between P and the vertices of F against the direction of the normal of P is larger than ϵ_a but smaller than $\epsilon_a + \epsilon_m$ where ϵ_m can as well be chosen by the user.

Extraction of planes for the BC is done in a greedy fashion based on the accumulated, projected areas of valid and missed faces stored in the grid cells. After determination of each new plane, the contributions of faces within ϵ_a distance of the new plane are removed from the grid cells. The process terminates when all faces are covered.

As a next step, for each extracted plane all points of the original mesh within ϵ_a distance are projected onto it and the result is stored in a texture. To optimize texture use, textures are split into parts if unconnected, compact regions are detected.

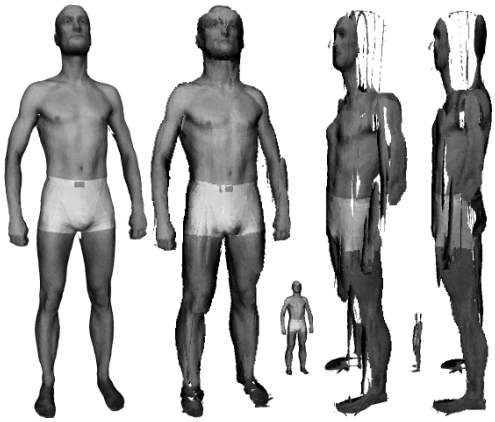

Figure 3: Normal sampling problem of the Hough space approach. Due to the lengthy shape of the body (left: original model with 25k triangles), many parallel planes were chosen to represent the geometry. While the frontal view of the BC (second from left) looks correct, pixels are missing when seen from the side even at the desired projection size (small images).

The advantage of this approach is the applicability to arbitrary triangle soups and the small number of resulting textured quads (see table 1).

Unfortunately the approach does not explicitly handle the deviation of face normals from the normal of the textured quad which represents them (for a bad example see figure 3). It is only due to the redundant sampling of surface points that only few cases exist where the problem really becomes apparent. As a result, BCs generated using this method require two-sided rendering.

Even worse, the method provides no control on the amount of required texture space since Hough space is insensitive to Euclidean distances and therefore it can easily occur that very distant geometries are represented by the same plane. The approach for texture optimization can resolve a very limited number of such cases only, leading to excessive texture memory requirements in many cases. Since texture space is already the limiting factor for application of BCs, other generation methods are required for reasonable simplification errors. In the following, our two new techniques are represented which provide solutions to these problems.

ϵ_{g_c}	Hough		HFC		Simpl.	
4	34	29k	200	34k	1.2k	2.5k
2	116	334k	566	139k	3.1k	17k
1	448	2.9M	1.5k	500k	6.0k	67k
.5	1.2k	22M	1.9k	1.4M	9.8k	313k
4	34	30k	60	6.9k	66	4k
2	62	175k	93	26k	168	17k
1	164	1.6M	193	112k	195	70k
.5	434	14M	478	432k	404	280k
4	4	0.9k	99	0.8k	70	0.6k
2	10	6.1k	102	3.1k	95	3.3k
1	30	67k	121	12k	194	24k
.5	88	600k	200	43k	419	150k

Table 1: Comparison of the BC construction algorithms. Per method, model and chosen geometric error ϵ_{g_c} (specified as percentage of the longest side of the bounding box) the number of geometric primitives (left column) and the number of texture pixels (right column) is given. Each block of four rows is associated to a specific model (top: plant, middle: Max Planck head, bottom: man).

4.2 Mesh Simplification

Our first method for memory-efficient construction of BCs is based on the standard way of reducing the complexity of models by successively removing vertices, edges or faces of a given model: mesh simplification algorithms [19]. In contrast to the previous Hough space approach which extracts globally optimal planes, mesh simplification algorithms are usually based on local optimization.

In principle, standard mesh simplification algorithms allowing for topology simplification can be employed for determination of approximating geometry for BC generation. Yet, since an accurate evaluation of the geometric error between the original mesh and the simplified mesh is required, methods guaranteeing tight error bounds like the one of Borodin et al. [3] are preferred.

As table 1 shows, applying these techniques to connected, smooth and preferrably manifold meshes yields models of rather low polygon count and minimal texture space requirements. In addition, the deviation of normals from the normal of the approximating triangle can simply be controlled using normal cones [36].

For unconnected meshes like trees, mesh simplification algorithms performed much worse in our tests (although texture requirements remain very low): the number of triangles increases significantly compared to the Hough space approach and the results largely reduce in rendering quality due to the

complex silhouettes of unconnected meshes which cannot be represented adequately. Enlarging the BC triangles to compensate for this problem unfortunately increases the texture requirements significantly (in our experiments the required texture space was doubled approximately).

4.3 Hierarchical Face Clustering

Our second new approach to BC generation is based on ideas from hierarchical face clustering [11]. The key idea of this method is to iteratively merge pairs of adjacent surface patches based on some energy function. Since face clustering is limited to connected meshes, we generalized the definition of adjacent patches.

4.3.1 Spatial Proximity

One of the central points of hierarchical face clustering is the concentration on adjacent patches which is necessary for charting but hinders general BC construction. For our needs, working on adjacent patches makes sense only insofar as they result in connected areas and therefore efficient texture use. Yet, since arbitrary meshes consist of many unconnected parts, the approach misses many perfectly reasonable opportunities to merge geometry. We therefore generalize the notion of adjacency to spatial closeness which removes the limitation to connected input meshes. Spatial closeness can efficiently be computed even for large meshes using a spatial data structure (SDS) like a grid or an octree.

An important parameter for computation of spatially close parts is the definition of closeness. The most intuitive approach will relate closeness to Euclidean distance. Following this definition, one has to compute all pairs of primitives that are no further apart than a predefined threshold. Unfortunately, to balance the amount of possible pairs during the merging process, this threshold needs to be increased over time. This introduces additional complexity into the algorithm.

Therefore, we define a primitive p_1 to be close to another one p_2, if at most $n-1$ other primitives have a smaller Euclidean distance to p_2 than p_1. This methods leads to an easy yet efficient control of the number of possible merges and requires the user to define a single threshold only. During our experiments, we determined that an n of around 50 leads to very good results which increase at most slightly for higher values of n. Nevertheless, this parameter varies from model to model since it depends on the model's geometric structure.

Another problem closely related to the run-time efficiency is the choice of an adequate number of spatial subdivisions introduced by the SDS. Our implementation generates an initial subdivision based on the axis-aligned bounding volume of the model and the number of its primitives. This subdivision is adjusted at runtime (the spatial resolution in each dimension is halved as soon as the number of primitives reduces by a factor of eight) resulting in a good balancing of the number of subdivisions to the number of remaining primitives.

4.3.2 Cost Function

Generalized hierarchical face clustering can be formulated as a minimization problem on the proximity graph. We define the proximity graph to consist of nodes representing primitives of the object and weighted edges connecting spatially close primitives, where weights represent costs of merging two nodes. The graph is simplified by merging connected nodes until a predefined number of nodes remains or until any further merge operation requires costs above a predefined threshold. The sum of costs for simplification is to be minimized.

In contrast to standard face clustering approaches, which try to achieve well parameterizable, compact charts, we want to minimize the amount of texture space. For our needs, the merge costs will therefore be combined of three different parts that measure geometric approximation error (i.e. maximum distance to a fitting plane), normal deviation and texture waste.

Given a pair of adjacent patches (P_1, P_2), the geometric error ϵ_g is simply half the length of the smallest side of the smallest oriented bounding box (OBB) containing P_1 and P_2. The OBB and the respective best approximating plane p_a can efficiently be computed using principal component analysis. The normal deviation error ϵ_n is defined as:

$$\epsilon_n = max\left\{acos\left\langle n\left(f\right)|n\left(p_a\right)\right\rangle | f \in P_1 \cup P_2\right\} \quad (1)$$

with $n(x)$ denoting the normal of face x. Texture waste ϵ_t is a more accurate version of the shape error from Garland et al. [11] optimized to our needs and is computed as:

$$\epsilon_t = \frac{\sum_{f \in P_1 \cup P_2} area(f)}{area(p_a)} \quad (2)$$

Since weighting these errors to compute the merge costs is a very difficult task (since appropriate weights are hard to determine) we utilize only one to define an ordering among the valid pairs while the others serve as hard rejection criteria defining validity. Since experience from LOD research shows that hierarchical LODs (HLODs) [10] lead to much better performance than continuous LODs, we need only be concerned about discrete LODs for hierarchical BCs. Therefore, setting a threshold for ϵ_g for the most accurate HLOD level and doubling it for each coarser level is a reasonable choice that sorts possible merges into clusters assigned to the different HLOD levels. Since the major reason for including normal deviation in the evaluation is the minimization of the normal sampling problem, ϵ_d can be thresholded as well. A setting of $\epsilon_{d_m} = 60^\circ$ e.g. guarantees that no such problems can occur for spheres and cylinders (compare as well figure 5).

Based on these definitions, the cost of a possible merge can be computed as

$$MergeCost = \begin{cases} \infty & \epsilon_n > \epsilon_{n_m} \\ \infty & \epsilon_t > \epsilon_{t_m} \\ \epsilon_t + k\epsilon_{t_m} & else \end{cases} \quad (3)$$

where $k \in N$ is either zero if $\epsilon_g \leq \epsilon_{g_m}$ or otherwise determined by $2^{k-1}\epsilon_{g_m} \leq \epsilon_g \leq 2^k\epsilon_{g_m}$.

Pairs resulting in too high normal deviation or texture waste are rejected. Pairs with valid normal deviation and texture waste are grouped into two categories: if the geometric error is below the threshold ϵ_{g_m} for the finest HLOD level, their merge cost is equal to the texture waste. If the geometric error is above the threshold for the finest level, the appropriate LOD level k for which the geometric error is valid is determined and the cost is computed as $\epsilon_t + k\epsilon_{t_m}$ which assures that no such merge is executed before all possible merges from the previous LOD levels are performed.

This new method has the big advantage that the required amount of texture space can be reduced significantly compared to the Hough space approach while preserving the high visual quality of resulting BCs (see figure 6). In addition, generated BCs contain relatively few textured triangles and the inherent control over normal deviation eliminates most visible cases of the normal sampling problem.

5 Results

During our experiments, we generated BCs with each of the above schemes. While standard hierachical face clustering and mesh simplification of connected meshes turned out to be the fastest methods due to the restricted search-space (few seconds), the runtimes of the more general methods for unconnected models turned out to be very similar (few minutes). As stated above already, the results generated by the methods vary greatly.

Computation of BTFs on the BCs was done using either rasterization or a simple raytracer. While rasterization provides run-time advatanges (about 2 hours for the model in figures 2 and 7 - bottom row), raytracing allows better quality due to pixel-correct shadows and interreflections (see top row of figure 7 and figure 8). Fortunately, both approaches can easily be parallelized.

The raw BTF requires about 30k memory per texel in the BC (about 1.8 GB for the model in figures 2 and 7). Using the Hough space approach, about 18 GB would be required.

The final amount of data per BTF textured BC was significantly reduced to about 17 MBs for the presented models by applying the compression method of Müller et al. [21], which enables efficient rendering using standard programmable graphics hardware (see [31]).

6 Conclusions and Future Work

In this paper we have analyzed various problems of BCs, some of which are related to the respective construction method and some are inherent to the approach. We proposed BTFs as solutions to the view independence problem and presented new, memory-aware BC construction algorithms that solve the normal sampling problem.

As future work, the suitability of view-dependent displacement maps [38] for representation of surface detail of BCs should be investigated. In addition, a comparison between the rendering quality and speed of BCs and 3D texture based approaches [7] might provide interesting insights. Finally, combinations with the occlusion culling ap-

proach of Sayer et al. [28] should be tested since their approach might nicely combine with our view-dependent BTF silhouettes.

Acknowledgements

This work was partially funded by the European Union under project RealReflect (IST-2001-34744). The man model was gratefully provided by the Virtual Try-On project.

References

[1] A. Adamson and M. Alexa, "Approximating and Intersecting Surfaces from Points", Proc. of Symposium on Geometry Processing 2003, pp. 230–239, 2003.

[2] C. Andújar, P. Brunet, A. Chica, J. Rossignac, I. Navazo, and A. Vinacua, "Computing Maximal Tiles and Application to Impostor-Based Simplification", to appear in Proc. of Eurographics 2004.

[3] P. Borodin, S. Gumhold, M. Guthe, and R. Klein, "High-Quality Simplification with Generalized Pair Contractions", Proc. of Graphicon'2003, pp. 147–154, 2003.

[4] P. Cignoni, C. Montani, R. Scopigno, and C. Rocchini, "A General Method for Preserving Attribute Values on Simplified Meshes", IEEE Visualization 1998, pp. 59–66, 1998.

[5] J. Cohen, M. Olano, and D. Manocha, "Appearance Preserving Simplification", Proc. of SIGGRAPH 1998, pp. 115–122, 1998.

[6] K. Dana, B. van Ginneken, S. Nayra, and J. Koenderink, "Reflectance and Texture of Real World Surfaces", IEEE Conference on Computer Vision and Pattern Recognition, pp. 151–157, 1997.

[7] P. Decaudin and F. Neyret, "Rendering Forest Scenes in Real-Time", Proc. of EG Symposium on Rendering, pp. 93–102, 2004.

[8] X. Décoret, F. Sillion, G. Schaufler, and J. Dorsey, "Multi-Layered Impostors for Accelerated Rendering", Computer Graphics Forum, 18(3), pp. 61–73, 1999.

[9] X. Décoret, F. Durand, F. Sillion, and J. Dorsey, "Billboard Clouds for Extreme Model Simplification", Proc. of SIGGRAPH 2003, pp. 689–696, 2003.

[10] C. Erikson, D. Manocha, and W. Baxter, "HLODs for Faster Display of Large Static and Dynamic Environments", Proc. of Symposium on Interactive 3D Graphics, pp. 111-120, 2001.

[11] M. Garland, A. Willmott, and P. Heckbert, "Hierarchical Face Clustering on Polygonal Surfaces", ACM Symposium on Interactive 3D Graphics, pp. 49–58, 2001.

[12] J. Grossmann and W. Dally, "Point Sample Rendering", Proc. of 9th EG Workshop on Rendering, pp. 181–192, 1998.

[13] P. Hough, "Method and Means for Recognizing Complex Patterns", US patent 3,069,654, 1962.

[14] K. Inoue, T. Itoh, A. Yamada, T. Furuhata, and K. Shimada, "Clustering a Large Number of Faces for 2-Dimensional Mesh Generation", Proc. of 8th Internat. Meshing Roundtable, pp. 281–292, 1999.

[15] S. Jeschke and M. Wimmer, "Textured Depth Meshes for Real-Time Rendering of Arbitrary Scenes", Proc. of the 13th EG Workshop on Rendering, pp. 181–190, 2002.

[16] A. Kalaiah and A. Varshney, "Statistical Point Geometry", Proc. of Symposium on Geometry Processing 2003, pp. 107–115, 2003.

[17] A. Kalvin and R. Taylor, "Superfaces: Polygonal Mesh Simplification with Bounded Error", IEEE Computer Graphics and Applications, 16(3), pp. 65–77, 1997.

[18] R. Klein and A. Schilling, "Efficient Rendering of Multiresolution Meshes with Guaranteed Image Quality", The Visual Computer, 15(9), pp. 443–452, 1999.

[19] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner, "Level of Detail for 3D Graphics", Morgan Kaufmann, ISBN 1558608389, 2002.

[20] P. Maciel and P. Shirley, "Visual Navigation of Large Environments using Textured Clusters", ACM Symposium on Interactive 3D Grapics, pp. 95–102, 1995.

[21] G. Müller, J. Meseth, and R. Klein, "Compression and real-time Rendering of Measured BTFs using local PCA", Vision, Modeling and Visualisation 2003, pp. 271–280, 2003.

[22] NVidia Corporation, "Mipmapping Normal Maps", 2004.

[23] H.-P. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: Surface Elements as Rendering Primitives", Proc. of SIGGRAPH 2000, pp. 335–342, 2000.

[24] V. Popescu, A. Lastra, D. Aliaga, and M. De Oliveira Neto, "Efficient Image Warping for Architectural Walkthroughs using Layered Depth Images", IEEE Visualization, pp. 211–215, 1998.

[25] S. Rusinkiewicz and M. Levoy, "QSplat: A Multiresolution Point Rendering System for Large Meshes", Proc. of SIGGRAPH 2000, pp. 343–352, 2000.

[26] P. Sander, X. Gu, S. Gortler, H. Hoppe, and J. Snyder, "Silhouette Clipping", Proc. of SIGGRAPH 2000, pp. 327–334, 2000.

[27] P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe, "Multi-Chart Geometry Images", Proc. of Symposium on Geometry Processing 2003, pp. 146–155, 2003.

[28] E. Sayer, A. Lerner, D. Cohen-Or, Y. Chrysanthou, and O. Deussen, "Aggressive Visibility for Rendering Extremely Complex Foliage Scenes", Proc. of IK 2004.

[29] G. Schaufler and W. Stürzlinger, "A Three-Dimensional Image Cache for Virtual Reality", Computer Graphics Forum, 15(3), pp. 227–236, 1996.

[30] G. Schaufler, "Per-Object Image Warping with Layered Impostors", Proc. of the 9th EG Workshop on Rendering, pp. 145–156, 1998.

[31] M. Schneider, "Real-Time BTF Rendering", Proc. of the 8th CESCG, pp. 79–86, 2004.

[32] J. Shade, D. Leschinski, D. Salesin, T. DeRose, and J. Snyder, "Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments", Proc. of SIGGRAPH 1996, pp. 75–82, 1996.

[33] J. Shade, S. Gortler, L.-W. He, and R. Szeliski, "Layered Depth Images", SIGGRAPH 1998, pp. 231–242, 1998.

[34] A. Sheffer, T. Blacker, and M. Bercovier, "Clustering: Automated Detail Suppression using Virtual Topology", Trends in Unstructured Mesh Generation, ASME Press, pp. 57–64, 1997.

[35] A. Sheffer, "Model Simplification for Meshing using Face Clustering", Computer-Aided Design, 33, pp. 925–934, 2001.

[36] L. Shirmun and S. Abi-Ezzi, "The Cone of Normals Technique for Fast Processing of Curved Patches", Computer Graphics Forum, 12(3), pp. 261–272, 1993.

[37] F. Sillion, G. Drettakis, and B. Bodelet, "Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery", Computer Graphics Forum, 16(3), pp. 207–218, 1997.

[38] X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum, "Generalized Displacement Mapping", Proc. of EG Symposium on Rendering 2004, pp. 227–233, 2004.

[39] M. Wimmer, P. Wonka, and F. Sillion, "Point-Based Impostors for Real-Time Visualization", Proc. of EG Workshop on Rendering 2001, pp. 163–176, 2001.

Figure 4: Plant model. Left and middle: original model (12k triangles). Right: BCs for 0.5%, 1% and 2% approximation error (compare table 1).

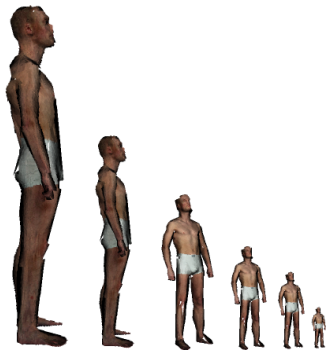

Figure 5: BC of man model from HFC (1% approximation error, no BTF): the normal sampling problem is solved by setting a maximum normal deviation of 60°. While the silhouette is incorrect in closeups (left), artifacts vanish when the desired projection size is approached (right).

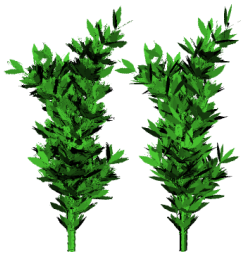

Figure 6: Comparison of visual quality of BCs generated from Hough Space (left) and HFC (right) at 1% approximation error. Although the visual quality is very similar, the right model requires 6 times less texture memory (see table 1).

Figure 7: Views of BTF textured Max Planck head BC (123 textured quads) for varying view and light-directions. The appearance of the surface material varies drastically. Top: lacquered wood with shadows encoded in the BTF (see ear and nose), bottom: plasterstone BTF without shadows.

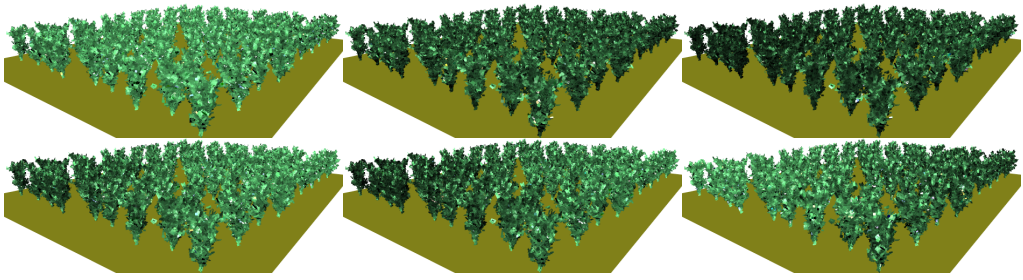

Figure 8: Relighting of distant wood scene. Occlusion and shadowing are correctly represented in the BTF and therefore the images of the distant trees due to the correct resampling used for BTF construction. Such results are not achievable with normal maps due to the large variance of normals per pixel even using improved techniques like mipmapping of normal maps [22].