INSTANCE-AWARE SIMPLIFICATION OF 3D POLYGONAL MESHES

Tahir Azim[⋆†] *Ewen Cheslack-Postava*[⋆] *Philip Levis*[⋆]

[⋆] Stanford University, Stanford, CA, USA
[†] NUST-SEECS, Islamabad, Pakistan
{tazim, ewencp, pal}@cs.stanford.edu

ABSTRACT

Virtual worlds and games increasingly deliver 3D meshes over the Internet using instanced file formats, such as COLLADA. However, existing simplification algorithms do not account for instancing in their inputs, operating instead on triangle soups or indexed triangle meshes. This makes them unsuitable for highly instanced meshes, since expanding an instanced mesh to an indexed triangle mesh and then simplifying it can result in a larger output file size. The high cost of delivering these larger files over the Internet results in long network latencies and low performance.

This paper presents instance-aware simplification (IAS), an algorithm designed to efficiently simplify instanced 3D meshes. To ensure smaller output file sizes, IAS incorporates the existing compression of mesh instancing into its cost metrics. Unlike existing algorithms, IAS strictly reduces file size as it simplifies, so that IAS-simplified meshes of a given file size have higher visual quality than meshes simplified using existing algorithms. On highly instanced models, IAS results in simplified versions that are orders of magnitude smaller than existing algorithms for a given triangle count.

Index Terms— 3D Meshes, Simplification, Instancing.

1. INTRODUCTION

Increasing numbers of virtual worlds are now hosted as Internet-based services, and deliver their 3D models over the network. These models often need to be delivered to the client in a simplified form that has fewer triangles than the original model but similar visual quality. From vertex clustering to edge contraction, many approaches exist to generate low-resolution, simplified versions of 3D models. These existing approaches generally assume that a model is represented as an indexed triangle mesh, consisting of a set of vertices and a set of triangles indexing into those vertices. However, many real-world model formats, including the COLLADA standard and 3ds, describe a 3D model as an instanced mesh (or equivalently, a scene graph). An instanced mesh is composed of a set of submeshes which are instanced using different transformation matrices (Figure 1). Instances are organized hierarchically such that the transformation matrix of an instance is the product of all transformation matrices from the root down to the instance.

A key benefit of instancing is that it enables greater compression of models with symmetric or identical sub-parts. Examples include leaves on a tree, slats on a chair, or a building's architectural features. Substantial recent work has focused on automatically detecting instances within large meshes, so they can be represented in a more compressed format [1].

Unfortunately, this compression is at odds with existing simplification algorithms. Existing algorithms are unsuitable for simplifying instanced meshes because reducing the triangles in a mesh can *increase* its file size. This is especially problematic if the simplified version of the mesh has to be delivered over the network, since the download time for a graphical client would increase substantially. This problem occurs because existing algorithms require expanding the instanced model into an indexed triangle mesh before they can be simplified. On expansion, the models can become much larger because submesh data must be duplicated for each new instance. The same submesh data may have to be duplicated hundreds or even thousands of times in an input mesh. Even after removing a few triangles, the expanded mesh's file size will remain larger than the original mesh. As more graphical content is downloaded dynamically over the Internet, there will be a need for algorithms that can simplify instanced models while reducing their file size.

This paper presents instance-aware simplification (IAS), an algorithm that simplifies instanced meshes without first expanding them. Instead, IAS simplifies an instanced mesh by computing the cost of and collapsing submesh edges directly. This ensures that the file size of a simplified instanced mesh decreases as its level of detail is reduced. Our results show that on meshes having more than 1.75 instances per submesh, IAS consistently yields smaller output file sizes than quadric simplification, while introducing comparable geometric error. For highly instanced meshes where one submesh is instanced hundreds of times, the file size of the simplified mesh is smaller by orders of magnitude, while the geometric error remains approximately the same. In addition, IAS runs up to

This work is funded by the National Science Foundation (NeTS-ANET Grants #0831163 and #0831374) and conducted in conjunction with the Intel Science and Technology Center - Visual Computing.

2.6 times faster on such meshes. We have incorporated IAS into the Sirikata virtual world platform. We refer the reader to [2] for details on how IAS fits within Sirikata.

The next section overviews related work in more detail. Section 3 introduces IAS, a new simplification algorithm designed to efficiently simplify instanced models. Section 4 demonstrates the benefits of using IAS on a large set of 3D models. Section 5 then discusses some implications and trade-offs of using IAS for simplification.

2. RELATED WORK

A vast body of work exists in the area of simplifying polygonal 3D models. Most of this prior work addresses how to simplify models stored on local disk, which is connected to the GPU through a high-bandwidth channel. For this reason, existing work focuses on reducing the triangle count of 3D models, not their file sizes. This paper targets systems where 3D models are loaded over the Internet, a much lower-bandwidth channel. This makes model file sizes a crucial concern. In that way, this paper presents an additional tool in the toolbox of a game/virtual world developer: it aims to complement, not replace, existing techniques.

Quadric simplification [3, 4], perhaps the most popular mesh simplification algorithm, uses an incremental approach that aims to minimize an error metric at every step. For every edge, it finds the optimal point at which the edge collapse would introduce the least error. Follow-up techniques extend this approach to consider other attributes, such as texture coordinates and normals [5, 6]. View-dependent simplification [7] and perceptually modulated level-of-detail (LOD) [8] exploit the camera's current viewpoint for high quality rendering, but can only operate at runtime on a client.

Billboard clouds [9] are an alternative technique for extreme simplification that projects a 3D model onto a set of planes with textures and transparency maps. This allows a complex model to be represented with something as simple as a textured cube. However, this only yields a good approximation when the object being viewed is distant and has parallax limitations. Further, the additional textures can increase memory cost and computing the billboards is time-consuming. Similar approaches are used by image-based techniques, such as hierarchical image caching [10] and impostors [11]. One positive of these approaches is that they can work for all kinds of meshes, including instanced meshes, without any modification.

A number of metrics exist to measure visual error on simplified meshes. One of the more well-established metrics is MetroMn [12], which has been shown in studies to correlate strongly with human perceptions of quality [13]. However, all of these algorithms assume that a model is an indexed triangle mesh and do not account for geometry reuse or instancing.

Finally, a great deal of recent work aims to find symmetrical substructures within 3D models. [1] provides a good overview of these approaches. Finding these symmetrical substructures can compress a complex shape into a smaller set of instanced submeshes. These algorithms are complementary to our work, as they create more highly instanced models.

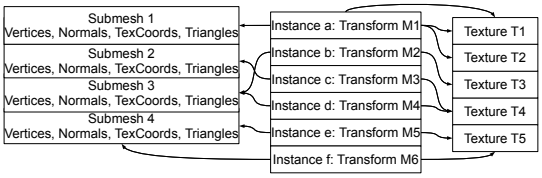


Fig. 1. The structure of an instanced mesh file.

3. IAS: INSTANCE-AWARE SIMPLIFICATION

When an instanced mesh is expanded into an indexed triangle mesh, it results in duplicated geometry and a much larger model. Simplifying such an expanded mesh is a one-way transformation: once edges in one copy of a submesh are collapsed, that copy cannot be easily factored back into a single submesh. As a result, a simplified instanced mesh can reduce graphical complexity while *increasing* file size. Our instance-aware simplification algorithm (IAS) addresses this problem.

Before we explain IAS, we provide some background on quadric simplification. We use quadric simplification as a base algorithm due to its excellent speed/performance trade-off and support for multiple moderately complex objects [14], both of which are important for games and virtual worlds.

3.1. Background: Quadric Simplification

Quadric mesh simplification executes in two phases. During the initialization phase, the algorithm assigns an error quadric, Q, to each vertex, v. Q is computed on the basis of the planes (triangles) neighboring v, and is given by $Q = \sum_{p \in planes(v)} Q_p$, where Q_p is the quadric for plane p and is computed as:

$$\mathbf{Q_p} = area(p). \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \quad (1)$$

Here a, b, c and d are the normalized coefficients of the equation $ax + by + cz + d = 0$, which defines the plane p, while $area(p)$ is the area of the triangle corresponding to plane p. With this formulation, given a vertex w, $w^T Qw$ is a measure of the distance of vertex w from the set of planes in planes(v).

For every edge (v_1, v_2), assuming that quadrics Q_1 and Q_2 are associated with v_1 and v_2, it then computes an optimal contraction target $\bar{v}$ for which the cost is given by

$$cost(\bar{v}) = \bar{v}^T (Q_1 + Q_2)\bar{v}. \quad (2)$$

Assuming $K = (Q_1 + Q_2)$, $\bar{v}$ is computed as:

$$\bar{\mathbf{v}} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{12} & k_{22} & k_{23} & k_{24} \\ k_{13} & k_{23} & k_{33} & k_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (3)$$

In the edge collapse phase, edges are collapsed iteratively in increasing order of these error values, with the cost of vertices neighboring v_1 and v_2 updated after each collapse.

3.2. Instance-aware Simplification Algorithm

Instance aware simplification converts the hierarchical representation of an instanced mesh M into a flat representation of a list of instances. The transform associated with each instance is the product of all transforms from the root of the hierarchy to the instance itself. Each instance indexes into a list of a submeshes, which contain geometry information in indexed triangle mesh format. Formally, $M = \{I_1, I_2, I_3, ..., I_n\}$, where n is the number of instances in the mesh. Each instance $I_j =< T_j, S_k >$, where T_j is the transformation matrix associated with I_j and S_k is the submesh I_j references. The submesh S_k is from a list of submeshes $\{S_1, S_2, ..., S_m\}$, such that $1 \leq k \leq m$ and $m \leq n$. Each submesh S_k consists of a set of vertices and a set of triangles referencing those vertices.

IAS uses the observation that the error quadric Q associated with a vertex v is derived from the set of planes neighboring v. In an instanced mesh, therefore, Q for a submesh vertex v can be computed by accounting for all the neighboring planes that exist in all instances of the submesh. Suppose T is the transformation matrix for a given instance of a submesh, and v is a submesh vertex which maps to x in that instance. Since $x = Tv$, we can write the distance of x from its set of neighboring planes as

$$x^T Q x = (Tv)^T Q (Tv) = v^T T^T Q T v \quad (4)$$

where Q is computed from the neighboring planes in that instance.

Then, $(T^T Q T)$ is the error quadric giving the distance of the submesh vertices from the neighboring planes in the instance. Summing it over all instances, the error for a submesh vertex is

$$E_s = \sum_{i \in I} T_i^T Q_i T_i \quad (5)$$

where I is the set of of instances of the submesh, T_i is the transform associated with instance i, and Q_i is the quadric computed for the instantiated vertex.

Using this new quadric, we can find optimal contraction targets for submesh edges and simplify an instanced mesh down to a target triangle count as follows:

1. For each instance i applying T_i to submesh S_i:
   (a) For each triangle t in submesh S_i:
      i. Transform t by applying T_i to each of its vertices.
      ii. Compute Q_p, the error quadric for the transformed triangle, using Equation 1.
      iii. Compute the error quadric for the untransformed triangle t as $T_i^T Q_p T_i$ and add it to the error quadrics for each of t's untransformed vertices.
2. In each submesh S_i, compute the optimal contraction target $\bar{v}$ and its cost for each submesh edge (v_1, v_2) using Equations 3 and 2, where Q_1 and Q_2 are the submesh error quadrics for v_1 and v_2 respectively.
3. Collapse submesh edges in increasing order of their cost. Compute how many triangles become degenerate after each collapse and decrement the number of triangles in the model by that times the number of instances of the submesh. At each step, since only a submesh edge is collapsed, the cost has to be updated only for neighboring vertices in that submesh.

As the algorithm proceeds, all edges of a submesh may collapse due to which the submesh may not remain visible. This is particularly a problem when there are many instances of a very simple submesh, such as the leaves of a tree: a single edge collapse in the submesh may cause all leaves to disappear. To handle this case, IAS employs a variation of stochastic simplification [15]. First, it does not collapse an edge if that causes a submesh to disappear. Second, to deal with many instances of a simplified submesh, it eliminates some instances of the submesh, while scaling up the remaining instances to preserve surface area. To do this, it computes the number of instances of the submesh needed to achieve the target triangle count. It keeps and scales up that number of instances while removing the rest. In doing so, it prioritizes keeping those instances which, if scaled up, remain within the original bounding box of the model.

Finally, IAS handles boundary edges in a special manner due to their importance. A boundary edge is an edge that exists in only one triangle. IAS considers an edge to be a boundary edge as long as it is a boundary edge within its submesh. For each boundary edge, IAS generates a perpendicular constraint plane running through the edge. It then computes the quadric for this constraint plane, weights it by the length of the edge and adds it to the quadrics for the endpoints of the edge. This results in much better results than simply marking such edges as incollapsible since it still allows small boundary edges to be collapsed in place of other longer edges.

4. RESULTS

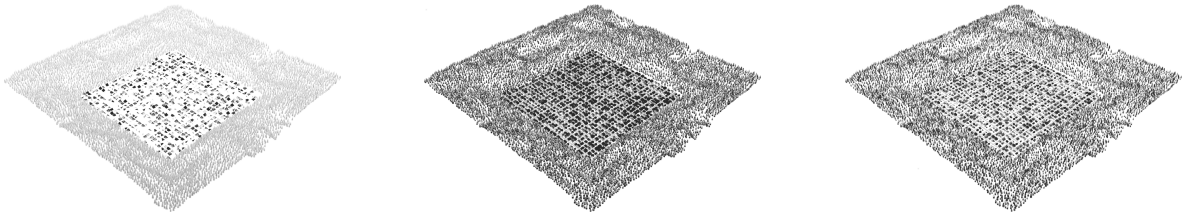
Table 1 shows results for instance-aware simplification (IAS) on four sample models with different degrees of instancing,

Model	Submeshes	Instances	Instances per submesh (IPS)	Triangles	Time (Quadric)	Time (IAS)	Time Reduction	Size (Quadric)	Size (IAS)	Size Reduction
Bunny	1	1	1	20,000	390 ms	1890 ms	0.2:1	235 KB	235 KB	1:1
Patio Chair	6	68	11.33	2,240	40 ms	43 ms	0.9:1	32 KB	31 KB	1:1
Maple Tree	18	9324	518	1,818,074	13 s	5 s	2.6:1	17 MB	110 KB	154:1
Village	79	13523	171.18	1,254,696	30 s	17 s	1.8:1	31 MB	6 MB	5:1

Table 1. Performance comparison of IAS and Quadric simplification for the three models shown in Figures 2– 4 as well as the Stanford bunny. Each model was simplified to 20% of its original triangle count. Instance-aware simplification outperforms quadric simplification in speed for highly instanced meshes while simultaneously better reducing file sizes.

(a) Quadric (20% triangles, 17 MB) (b) Original, 546 KB (c) IAS (20% triangles, 110 KB)

Fig. 2. Maple tree (518 instances per submesh) from Table 1 simplified to 20% of the original triangle count. Each leaf is an instance of the same submesh. Using a simplification algorithm that is not instance aware causes the file to increase in size by a factor of 30. Note how IAS preserves the appearance of individual leaves while reducing file size by 80%.



(a) Quadric (6MB) (b) Original (16 MB) (c) IAS (6 MB)

Fig. 3. Village (171 instances per submesh) from Table 1 simplified to 6MB in size. Quadric simplification removes all of the roads, reduces all of the homes to their roofs, and turns trees into simple crossed triangles. Although each road segment is small, their repeated, regular presence makes them a visually important feature of the scene.

(a) Quadric, 32 KB (b) Original, 92 KB (c) IAS, 31 KB

Fig. 4. Patio chair (11.3 instances per submesh) from Table 1 simplified to approximately one third of the file size. IAS preserves the slats while quadric simplification removes some of them.

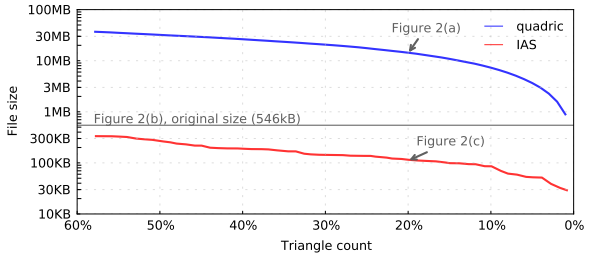


Fig. 5. For the tree model in Figure 2, IAS is able to maintain the same triangle count as quadric simplification at one hundredth the file size. Even after 99% simplification, the quadric simplified model is larger than the original file.

measured by their instance count per submesh (IPS). Figures 2–4 show the visual results. We do not show visual results for the Stanford bunny because IAS devolves to standard quadric simplification, so it is visually identical.

The dense maple tree and village (Figures 2 and 3) are highly instanced meshes with over a million triangles each and on average over 100 instances per submesh. Simplification with IAS results in a much smaller output mesh that has greater visual quality. In the case of the maple tree, IAS reduces file size by a factor of 5 and maintains the visual quality of the leaves; quadric simplification distorts the leaves while increasing file size by a factor of 30. In the case of the village, IAS achieves significantly better visual quality for the same target triangle count.

Figure 5 shows the size of the tree model under IAS and quadric simplification. Simply applying quadric simplification expands the mesh to 36MB. Reducing it to 1% of its original triangles is 891KB, larger than the original file size. In contrast, IAS strictly decreases file size as LOD reduces.

Using IAS, simplification of these heavily instanced models proceeds faster than quadric simplification, because IAS's edge collapse operates only on submesh edges. There are fewer submesh edges than edges in the overall mesh and collapsing a single submesh edge effectively collapses multiple edges in the overall mesh.

Submeshes in the patio chair are not highly instanced (IPS=11.33). IAS takes slightly longer than quadric simplification since it must perform additional matrix multiplications to compute quadrics for the transformed submeshes. IAS recognizes the visual importance of the slats in the chair back (an instanced submesh) and so maintains them, while quadric simplification produces holes.

Finally, the Stanford bunny mesh has exactly one instance per submesh. It takes longer to simplify it using IAS, but the visual error remains the same. For such meshes with few instances per submesh, existing algorithms are more efficient.

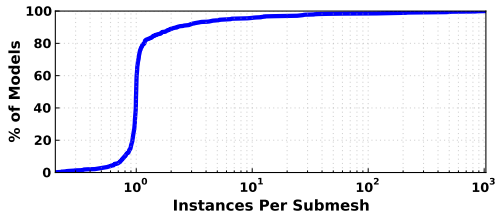


Fig. 6. CDF showing the distribution of instances per submesh in our dataset.

4.1. Running IAS on a model dataset

In order to test if IAS is generalizable to all user-generated 3D models, we tested IAS on a dataset of 748 models uploaded by a group of students while building their own virtual worlds. These models vary widely in complexity from 1 to 1.8 million triangles, and from 1 to almost 1000 IPS (Figure 6). We use this dataset because it contains a representative sample of models used to construct virtual worlds.

Each model in the dataset was simplified to 20% of its original triangle count using both quadric simplification and IAS. The solid blue line in Figure 7 compares the file size of the resulting simplified models using the two algorithms. Unexpectedly, in almost 20% of models, quadric simplification actually results in smaller file sizes than IAS. Most models where quadric simplification outperforms IAS have a low degree of instancing, measured by its IPS. On the other hand, if the IPS is higher than 1.75, IAS performs worse in only a handful of cases. Therefore, IAS switches to quadric simplification whenever the IPS of the input mesh is less than 1.75.

The dashed red line in Figure 7 shows that, with this optimization, quadric simplification outperforms IAS in only 1% of models. In these models, the submeshes are so small that it is more efficient to store them in an expanded format instead of storing their instance transformations. Among the remaining models, about 16% see more than 20% reduction in file size using IAS. Some complex models even see more than 99% reduction in file size using IAS compared to quadric simplification. On the average, IAS results in 57% smaller simplified file size than quadric simplification.

Finally, the dashed black line in Figure 7 shows that on almost 98% of models, IAS also achieves the same or lower geometric error compared to quadric simplification. In the remaining 2% of models, while the percentage error introduced by IAS relative to quadric simplification is sometimes large, the absolute error value remains small.

5. DISCUSSION

If every instance of a submesh has the same scaling, then existing simplification approaches can be trivially applied by multiplying the edge collapse cost from one instance by the number of instances. This approach creates artifacts when

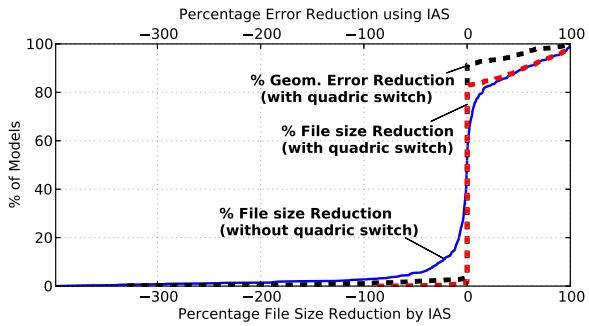


Fig. 7. CDFs showing percentage reduction in file size and geometric error achieved by IAS compared to quadric simplification. Metro's Hausdorff distance is used to measure geometric error. The dashed red and black lines show the results if IAS switches to quadric simplification when the instances-to-submeshes ratio is less than 1.75.

there are multiple submeshes and their instances have very different scalings, as a large instance of a submesh has greater visual impact than a small one. In our user uploaded dataset of 748 models, 15% of models have this property.

IAS does not compute quadrics across submeshes. It knows less about the overall mesh, which could result in lower quality outputs. However, our evaluations in Section 4 show that, in practice, IAS typically results in higher quality results. This is mainly because IAS does not allow instances of a submesh to diverge. For example, unlike existing algorithms, IAS keeps the leaves of the tree (Figure 2) and the slats of the patio chair (Figure 4) unchanged after simplification. Our current IAS implementation does not optimize other vertex attributes such as texture coordinates. Extending the quadric to include these attributes is part of future work.

6. CONCLUSION

This paper presents a new algorithm to simplify meshes for delivery over a wide area network such as the Internet. The algorithm, instance-aware simplification, simplifies models while respecting their existing compression and internal coherence through instancing. This generates simplified instanced meshes that have higher visual quality than those generated by existing algorithms having the same output file size.

7. REFERENCES

[1] Niloy J Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan, "Symmetry in 3d geometry: Extraction and applications," in *Computer Graphics Forum*. Wiley Online Library, 2013, vol. 32, pp. 1–23.

[2] Tahir Azim, Ewen Cheslack-Postava, and Philip Levis, "Displaying large user-generated virtual worlds from the cloud," Tech. Rep. 2013-01, Stanford University, Computer Science, 2013.

[3] Rémi Ronfard and Jarek Rossignac, "Full-range approximation of triangulated polyhedra.," in *Computer Graphics Forum*, 1996, vol. 15, pp. 67–76.

[4] Michael Garland and Paul Heckbert, "Surface simplification using quadric error metrics," in *Proc. ACM SIGGRAPH*, 1997.

[5] Michael Garland and Paul S. Heckbert, "Simplifying surfaces with color and texture using quadric error metrics," *Visualization Conference, IEEE*, 1998.

[6] Hugues Hoppe, "New quadric metric for simplifiying meshes with appearance attributes," in *Visualization Conference, IEEE*, 1999, VIS '99.

[7] David Luebke and Carl Erikson, "View-dependent simplification of arbitrary polygonal environments," in *Proc. ACM SIGGRAPH*, 1997.

[8] Nathaniel Williams, David Luebke, Jonathan D. Cohen, Michael Kelley, and Brenden Schubert, "Perceptually guided simplification of lit, textured meshes," in *Proc. Symp. Interactive 3D Graphics*. 2003, ACM.

[9] Xavier Décoret, Frédo Durand, François X. Sillion, and Julie Dorsey, "Billboard clouds for extreme model simplification," in *Proc. ACM SIGGRAPH*. 2003, ACM.

[10] Jonathan Shade, Dani Lischinski, David H. Salesin, Tony Derose, and John Snyder, "Hierarchical image caching for accelerated walkthroughs of complex environments," in *Proc. ACM SIGGRAPH*, Aug. 1996.

[11] Simon Dobbyn, John Hamill, Keith O'Conor, and Carol O'Sullivan, "Geopostors: a real-time geometry / impostor crowd rendering system," in *Proc. Symp. Interactive 3D Graphics*. 2005, ACM.

[12] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, no. 2, pp. 167–174, 1998.

[13] Benjamin Watson, Alinda Friedman, and Aaron McGaffey, "Measuring and predicting visual fidelity," in *Proc. ACM SIGGRAPH*, 2001.

[14] David P. Luebke, "A developer's survey of polygonal simplification algorithms," *IEEE Comput. Graph. Appl.*, vol. 21, no. 3, pp. 24–35, May 2001.

[15] Robert L Cook, John Halstead, Maxwell Planck, and David Ryu, "Stochastic simplification of aggregate detail," in *ACM Transactions on Graphics (TOG)*. ACM, 2007, vol. 26, p. 79.