Perceptually Guided Simplification of Lit, Textured Meshes

Nathaniel Williams * David Luebke † Jonathan D. Cohen ‡ Michael Kelley † Brenden Schubert †

ABSTRACT

We present a new algorithm for best-effort simplification of polygonal meshes based on principles of visual perception. Building on previous work, we use a simple model of low-level human vision to estimate the perceptibility of local simplification operations in a view-dependent Multi-Triangulation structure. Our algorithm improves on prior perceptual simplification approaches by accounting for textured models and dynamic lighting effects. We also model more accurately the scale of visual changes resulting from simplification, using parametric *texture deviation* to bound the size (represented as spatial frequency) of features destroyed, created, or altered by simplifying the mesh. The resulting algorithm displays many desirable properties: it is view-dependent, sensitive to silhouettes, sensitive to underlying texture content, and sensitive to illumination (for example, preserving detail near highlight and shadow boundaries, while aggressively simplifying washed-out regions). Using a unified perceptual model to evaluate these effects automatically accounts for their relative importance and balances between them, overcoming the need for ad hoc or hand-tuned heuristics.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

Keywords: Level of detail, perceptually motivated rendering, mesh simplification

1 INTRODUCTION

Interactive graphics has come to rely on *level of detail* or *LOD* techniques. These techniques simplify the geometric representation of a scene to reduce its rendering cost, while attempting to preserve visual fidelity. A great deal of excellent research has studied how to simplify polygonal meshes, but the question of how to evaluate visual fidelity to guide that simplification has received less attention. Mesh simplification has been guided primarily by geometric metrics. Usually, however, the important question is not geometric but perceptual: does the simplification *look* like the original?

* University of North Carolina at Chapel Hill
† University of Virginia
‡ Johns Hopkins University
than@cs.unc.edu, *luebke@cs.virginia.edu*, *cohen@cs.jhu.edu*

Of course, researchers in LOD have long recognized the importance of perceptual issues, but have tended to address those issues in an ad hoc fashion. For example, silhouettes are known to play a key role in object recognition and detection, and therefore even the earliest mesh simplification algorithms included heuristics to preserve detail in high curvature regions, which are more likely to project onto the silhouette; see for example the [Rossignac and Borrel 1993] vertex-clustering approach, or the decimation algorithm of [Schroeder et al. 1992]. View-dependent simplification research has also emphasized silhouette preservation; for example, [Luebke and Erikson 1997] enforce a tighter screen-space error threshold for silhouette regions than interior regions. Similarly, the presence and movement of specular highlights across a surface are known to provide important clues to its shape, so [Xia and Varshney 1996] and [Klein and Schilling 1999] describe view-dependent simplification schemes that preserve detail where such highlights are likely to appear. Many researchers have used heuristics and user-specified weights to balance the importance of geometric fidelity during simplification against preservation of appearance-related attributes, such as color, normals, and texture coordinates [Garland and Heckbert 1998, Erikson and Manocha 1999, Hoppe 1999]. At a higher level, [Funkhouser and Sequin 1993] describe a predictive system for choosing LODs to maintain visual quality at constant frame rates; their system accounts for many perceptual factors. Again, the system is fundamentally heuristic, with user-tunable parameters to control the relative importance of various perceptually-motivated factors.

We describe a polygonal simplification algorithm grounded directly in principles of visual perception. Briefly, we estimate the contrast that would be induced by local simplification operations (Figure 1), and the maximum spatial frequency of the resulting distortion, and predict the perceptibility of the operations based on a simple model of low-level vision called the *contrast sensitivity function* or *CSF*. Though the CSF does not provide a complete perceptual model, the resulting system achieves many effects desirable in a simplification algorithm, such as preservation of silhouette boundaries and shading- or illumination-sensitive simplification. More importantly, these effects proceed naturally from the perceptual model. This addresses the question of how, without heuristics or user input, to trade off such factors as silhouette preservation with distortion of a model's underlying color or texture coordinates. Note that we do not claim to achieve such effects for free; for example, we maintain normal cones to determine which regions occupy the visual silhouette. But the importance of that silhouette status—the decision of when to simplify a silhouette region rather than an interior region—derives from the perceptual model.

1.1 Motivation: a pragmatic approach

The primary goal of prior perceptual LOD approaches has been *imperceptible simplification*: create or select a level of detail visually indistinguishable from the original model. We argue that this approach is flawed. First, most interactive systems forced to use LOD to maintain frame rate must compromise ideal visual quality to do so. Put another way, if you can't afford to render the

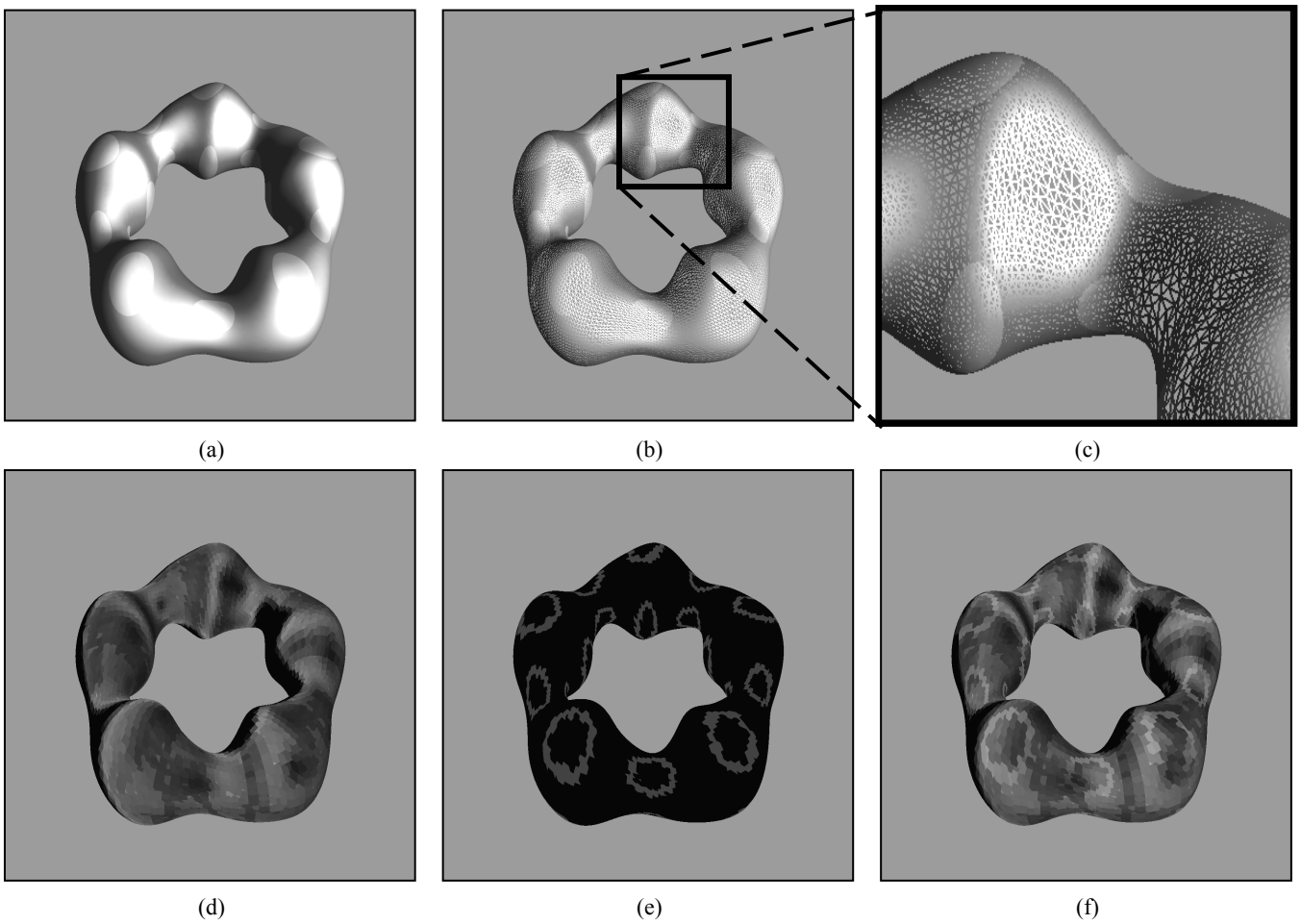


Figure 1: Contrast calculation and simplification effects on a textured and lit torus. The model is shown (a) at full resolution with 57,660 triangles and (b) simplified by 50%. The close-up (c) illustrates preservation of silhouettes and extra simplification in low-contrast areas, such as washed-out specular highlights and deeply shadowed regions. Image (d) shows the contrast due only to dynamic lighting; (e) shows the contrast due solely to the spotted texture; (f) shows the combined contrast used to generate the simplifications shown in (b) and (c).

original model, you probably can't afford to render an indistinguishable approximation. Second, the users of an application in which simplification must not be perceptible are unlikely to trust even a simplification algorithm that claims imperceptibility. A scientist, for example, might prefer to suffer slow frame rates rather than to trust that important features of a visualization are not being simplified away. Finally, and most importantly, at this time it does not appear feasible to evaluate a sophisticated model of visual perception fast enough to be used in interactive rendering.

In particular, the CSF models used by researchers to date (see Section 2.1) provide a reasonable first approximation to low-level perceptibility, but fail to take into account many important perceptual factors. Variations between individual users, adaptation to environmental lighting conditions, temporal sensitivity to flicker or sudden onset (as of a "pop" between LODs), chromatic versus achromatic contrast sensitivity, and facilitation/suppression via visual masking are all effects not modeled by the simple CSF. Some state-of-the-art models for accelerating offline rendering incorporate many of these effects [Ramasubramanian et al. 1999, Myszkowski et al. 2001], but require orders of magnitude longer than the few milliseconds available in interactive rendering. To *guarantee* imperceptible rendering under these conditions currently appears out of reach, and to achieve it in practice requires making conservative decisions that prevent much simplification; for example, [Luebke and Hallen 2001] report that models in their system could be simplified two to three times further without introducing perceptible artifacts. Therefore, we take a pragmatic approach that focuses on perceptually-guided best-effort reduction to a triangle budget.

1.2 Contributions

To briefly summarize the contributions of this paper:

- We extend the perceptual simplification framework of [Luebke and Hallen 2001] to textured models. The result: a more applicable algorithm capable of texture-content-sensitive simplification.
- We also use parametric texture deviation to measure distortion more accurately than the Luebke-Hallen approach. The result is better simplification for a given polygon count.
- We introduce techniques to incorporate the effect of dynamic lighting calculations. We can account for both specular and diffuse effects, under both Gouraud-shaded vertex lighting and per-pixel normal-map lighting. The result is better simplification of lit models.
- We evaluate our results against prior work both visually and with a sophisticated image metric.

Our work builds on research efforts by several groups. Our algorithm relates most closely to the approach of Luebke and Hallen, but applies to a much broader class of models because we account for textures and dynamic lighting. They also emphasize imperceptible simplification, while we focus on the pragmatic approach of perceptually-guided best-effort rendering to a budget. We also incorporate work by [Cohen et al. 1998] on bounding parametric texture deviation, as well as the *Multi-Triangulation* (*MT*) data structure by [DeFloriani et al. 1997, DeFloriani et al. 1998]. We discuss this and other related research in the following sections.

2 RELATED WORK

2.1 Perceptually guided rendering

Perceptually guided rendering is hardly a new field; many researchers have investigated algorithms to accelerate rendering by avoiding computation for which the result will be imperceptible. Examples include [Bolin and Meyer 1998], [Myszkowski et al. 2001], and [Ramasubramanian et al. 1999]. Unlike our work, which targets interactive rendering, most previous perceptually based rendering approaches have examined offline realistic rendering approaches such as ray and path tracing. These frameworks typically require seconds or minutes to create an image, and can therefore employ sophisticated perceptual models such as that described by [Ferwerda et al. 1997]. State-of-the-art perceptual models account for much of the known behavior of the low-level visual system, but are simply too costly for real time rendering. For example, Ramasubramanian et al. report times of several seconds to evaluate a 512x512 image. Such models are clearly out of reach for interactive rendering, which measures frame time in milliseconds.

[Reddy 1997] describes an early attempt to guide LOD selection entirely by a principled perceptual model. Reddy analyzed the frequency content of objects and their LODs in several images rendered from multiple viewpoints. If a high-resolution and a low-resolution LOD differed only at frequencies beyond the modeled *visual acuity*, or greatest perceptible spatial frequency, the system used the low-resolution LOD. Later, Reddy [2001] presented an updated version of this approach for terrains. In similar work, [Scoggins et al. 2000] analyzed the frequency content by transforming a prerendered reference image to frequency space and modulating the resulting spectrum by a perceptually modeled transfer function, then using mean-squared error to choose an appropriate LOD. Both approaches rely on images from just a few viewpoints, which introduces the possibility of sampling error, and both use a small set of discrete LODs, which prevents adaptive simplification (for example to preserve silhouettes).

[Lindstrom and Turk 2000] describe an image-driven approach for guiding the simplification process itself. They render, from multiple viewpoints, each model that would result from many possible simplification operations, and evaluate the cost of each operation by differencing the rendered images from images of the original model. Again, sampling from limited viewpoints and using static LODs have disadvantages for perceptually based simplification, but Lindstrom and Turk's approach has the important benefit that simplification is ultimately guided not by geometric error, nor by some combination of geometric and shading attribute error, but by an estimation of what the effect the simplification will have on the final rendering. This approach is therefore close in spirit to our work, which strives to drive simplification directly by a model of its perceptual effect.

Some simplification algorithms, though not guided by a perceptual model, attempt to preserve the appearance of an object directly by using enough polygons to prevent simplification artifacts larger than half a pixel. [Cohen et al. 1998] track the parametric surface distortion to derive a screen-space bound on the movement of color (represented by a texture map) and lighting (represented by a normal map) across the surface. Similar work by [Schilling and Klein 1998], and later work by the same authors [Klein and Schilling 1999], also deserves mention. In the former, they account for texture distortion using a surface mapping technique similar to that of Cohen et al; in the latter, they account separately for lighting artifacts in vertex-lit models using cones that bound the normals and halfway vectors. Our work improves on these approaches by providing best-effort simplification to a budget, and by using a perceptual model to regulate geometric, texture, and lighting effects in a single framework. This opens up opportunities to simplify more aggressively, for example in the washed-out region of a specular highlight (see Figure 1).

Our approach most closely follows the work of [Luebke and Hallen 2001], who also guide view-dependent simplification with a model of the CSF. The key idea behind the Luebke-Hallen approach is to evaluate local simplification operations according to the *worst-case contrast* and *worst-case spatial frequency* of features they could induce in the image. This provides a principled way to reason about the perceptibility of the resulting simplification. We extend these concepts to a more general and practical framework for simplification of meshes.

2.2 Texture deviation

An appropriate geometric way to measure the error of texture mapped surfaces is to bound the texture deviation [Cohen et al. 1998, Lee et al. 2000, Sander et al. 2001]. The texture deviation is a 3D distance in object space between pairs of corresponding points. The correspondence is established in parameter space. Thus it tells us how far any point on the original surface—for example the point corresponding to a particular texel—may move in 3D when we replace the surface with the simplified version.

One way to use this texture deviation metric in a view-dependent level of detail system is to project it to screen space. We find or approximate the closest point to the eye point of the bounding sphere of some node. Using this distance from the eye to the bounding sphere, we compute the length of the texture deviation vector in screen space. This measures the number of pixels of deviation for the model and bounds the shift of texels in screen-space as a result of simplification.

As we will see in Section 3, the 3D texture deviation may also be used in combination with a node's texture contrast to bound the spatial frequency of its most perceptible feature and compute its imperceptibility distance.

2.3 The Multi-Triangulation

In this section we briefly describe the MT data structure introduced by [DeFloriani et al. 1997]. The MT is a hierarchical model in the form of a directed acyclic graph, represented by a set of nodes connected by a set of arcs. The topmost root node of the graph is called the *source*, and the bottommost node is the *drain*. Figure 2 illustrates a small example.

Each node of the MT represents a small change to the mesh: a refinement operation if we are traversing downward, or a simplification operation if we are traversing upward. We create these nodes from the drain to the root during an offline bottom-up simplification process. Each arc represents one or more mesh triangles. The triangles removed from the model by a simplification operation are stored with the child arcs of the operation's node, and the coarser replacement triangles are associated with its parent arcs. Thus applying the local simplification operation encoded by the end node of an arc A (the node beneath A) will create the triangles encoded in the arc, and applying the simplification encoded by its start node (above A) removes the triangles.

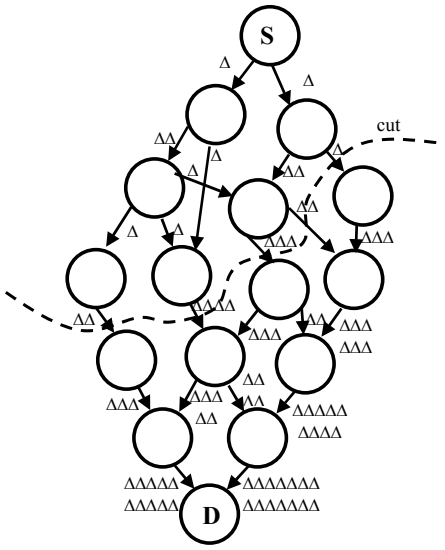


Figure 2: A small Multi-Triangulation containing 24 original triangles. The letters indicate the source (S) and drain (D) nodes. The cut contains 12 triangles.

The arcs of the MT represent the dependencies of one mesh operation on another. So, for example, if we wish to perform the refinement indicated by a node, we must first perform the refinements indicated by all of the node's parents. Performing the node's operation amounts to replacing the primitives of a node's parent arcs with those of its child arcs, or vice versa.

To extract a connected, consistent representation of the surface, we generate a cut of the graph. A cut is a set of arcs that partitions the nodes of the MT, leaving the source node above the cut, and the drain node below it. In addition, if the cut contains arc *A*, then it must not contain any ancestor or descendent of *A*. The triangles of such a cut represent our input surface at some resolution. The cut representing the coarsest level of detail crosses all the child arcs of the source node, whereas the cut representing the finest level of detail crosses all the parent arcs of the drain. We discuss how to generate cuts representing a particular triangle budget in Section 5.1.

For our application, the MT has two major advantages over other well-known simplification hierarchies. In the MT, all triangles in all possible simplifications are explicitly represented, allowing us to precompute accurate object-space error bounds, texture contrasts, and normal cones. In hierarchies, such as the vertex-merging trees of [Hoppe 1997] and [Luebke and Erikson 1997], the exact extent and shape of triangles in the neighborhood of a particular simplification (vertex merge) operation depends on whether nearby vertices have been simplified. A secondary benefit of the MT is rendering efficiency: because the triangles associated with each arc are known in advance, we can easily optimize arc geometry for the graphics hardware using triangle strips and vertex arrays.

3 PERCEPTUAL MODEL

Our underlying perceptual model is the contrast sensitivity function (CSF), which predicts the low-level perceptibility of simple visual stimuli called *contrast gratings* (Figure 3). A contrast grating is sinusoidal luminance pattern; its contrast is a function of its peak luminance values $L_{\min}$ and $L_{\max}$. Contrast grating studies use *Michaelson contrast*, defined as $(L_{max} - L_{min}) / (L_{max} + L_{min})$, and *spatial frequency*, defined as the number of cycles per degree (cpd) of visual arc. The *threshold contrast* at a given spatial frequency is the minimum perceivable contrast for a grating of that frequency, and *contrast sensitivity* is defined as the

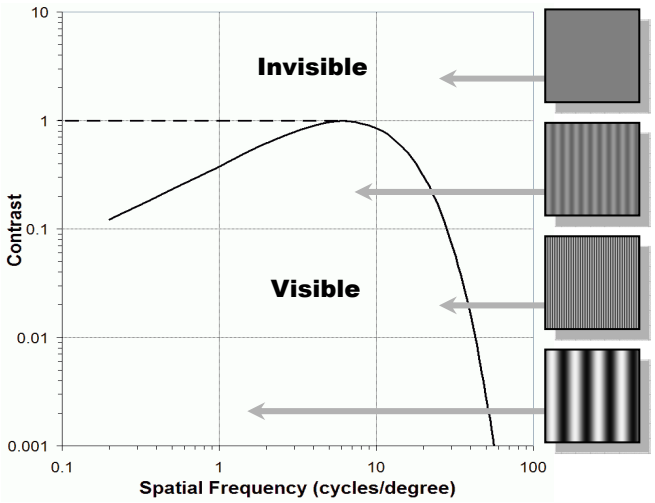


Figure 3. A contrast sensitivity function predicts the threshold perceptibility of a stimulus given its size and contrast. The horizontal dashed line indicates how we clamp the CSF for low spatial frequencies below the maximally sensitive peak. The four contrast gratings illustrate particular combinations of contrast and spatial frequency from the plot. Figure courtesy of Martin Reddy, Pixar Animation Studios.

reciprocal of threshold contrast. The CSF plots contrast sensitivity against spatial frequency, and so describes the range of perceptible contrast gratings. We adapt the approximation by [Rushmeier et al. 1995] of the Daly CSF model [Daly 1993]:

$$C(f) = \left(\frac{0.008}{f^{1.5}} + 1\right)^{-0.2} 1.42\sqrt{f}\, e^{-0.3\sqrt{f}} \sqrt{1 + 0.06 e^{0.3\sqrt{f}}},$$

where C represents contrast sensitivity and f represents spatial frequency in cycles per degree. In practice, we represent this computationally expensive empirical formula with a lookup table.

3.1 Applying the model

We follow Luebke and Hallen's approach of equating local simplification operations to a *worst-case grating*. More precisely, we consider the scale of features of the original surface that the simplification could eliminate. The key observation underlying their approach, which we only summarize below, is that the threshold perceptibility of those features can be conservatively equated to the perceptibility of a grating at the *lowest frequency* and *maximum contrast* possibly induced by that change.

3.2 Spatial frequency

Given that peak contrast sensitivity occurs around 2-4 cycles per degree, and most local simplification operations on a complex model will only affect much higher frequencies, we can assume that contrast at lower spatial frequencies is more perceptible than at higher frequencies.[1] Because the minimum frequency component of an image feature that spans *n* degrees of visual arc is one cycle per 2*n* degrees, the maximum wavelength needed to represent a region of the image is twice the maximum spatial extent of that region. Consequently, we can reduce finding the worst-case frequency induced by a simplification operation to finding the screen-space extent of the affected feature. One of our contributions is an improved method for estimating this extent by using texture deviation.

[1] We ensure that this assumption holds by clamping our worst-case frequency to be no lower than the point of peak sensitivity (Figure 3).

Our approach is motivated by the ability of texture mapping to hide simplification artifacts. This is partially due to a perceptual effect called *visual masking*, in which frequency content in certain channels suppresses the perceptibility of other frequencies in that channel [Ferwerda et al. 1997]. We do not account for visual masking, leaving that as an important and interesting area for future work. But texture mapping is inherently more robust to simplification of the underlying surface than Gouraud shading for another reason: it decouples the surface color from the exact position and number of vertices. Luebke and Hallen permit only prelit Gouraud-shaded models, and bound the spatial extent of a mesh simplification operation with a bounding sphere that contains all triangles involved in the operation. By using a texture-mapped model, we can achieve a better bound on the size of features affected by a local simplification operation. A texture deviation of ε can create or destroy features on the surface no larger than 2ε. The texture deviation induced by a simplification is usually much smaller than the bounding sphere of the simplification neighborhood, leading to a much tighter bound on the screen-space region affected.

3.3 Contrast

Given a worst-case spatial frequency for a simplification operation, determined by the maximum size of any affected features in the image, the next task is to find the maximum contrast of those features. The contrast of a feature is defined by its intrinsic luminance versus the luminance of the surrounding background. We estimate these using the range of luminance covered by the patch of surface affected by simplification. Because our simplification operation is a single edge collapse, this patch is relatively small. This leads to some of the most interesting contributions of our algorithm. By accounting for the intrinsic contrast of the texture map, we achieve *texture-content sensitive simplification*. Incorporating the lighting model into our contrast computation extends our approach to dynamically lit models and enables illumination sensitive simplification. We describe these contrast calculations further in Section 4.

The silhouette status of the surface patch being simplified also affects the maximum resulting contrast. If the patch, or local neighborhood of the simplification, lies on the silhouette, we must account for more than the luminance of nearby points on the surface: a small change may distort the surface and could in principle cover or uncover the brightest or darkest spot in the scene. Because we cannot easily know how much contrast this could cause, we conservatively assign maximal contrast to simplifications we determine are on the silhouette. As a result, silhouette regions of the object are simplified less aggressively – just the behavior one would expect in a perceptually driven simplification algorithm. Note however that even at these higher contrast levels silhouette regions can still be simplified if they represent very fine details (high spatial frequencies).

3.4 Imperceptibility distance

For best-effort perceptual simplification, we would like a model to predict which simplifications will have the least visual effect. Put another way, under the constraints of real-time rendering we will sometimes have to perform perceptible simplifications; we would like to predict which perceptible simplifications will be the least distracting or objectionable. However, the CSF models *threshold performance* of the visual system, predicting the minimal contrast at which a stimulus of a given spatial frequency may become perceptible. Unfortunately, the CSF cannot predict *suprathreshold performance*: given two stimuli, both above threshold contrast, which one is more perceptible?

While a great deal of work has explored threshold behavior of the visual system, much less research has investigated suprathreshold performance. We know of no computational model of suprathreshold perception suitable for interactive rendering; this is a crucial open problem in perceptually driven rendering. As a stopgap measure, Luebke and Hallen suggest inverting the function. Instead of looking up the threshold contrast for a given frequency, they map the contrast associated with a simplification to the spatial frequency at which it becomes visible. Note that for the general CSF this mapping is not necessarily a single-valued function, but because we clamp frequencies below peak sensitivity, the threshold contrast monotonically decreases with frequency. Given the spatial frequency at which a given simplification would become visible, and the screen-space extent of that simplification's effect (which we estimate using the texture deviation), we can compute the *imperceptibility distance*, or distance from the image at which the simplification should be imperceptible. The imperceptibility distance for an LOD is the maximum imperceptibility distance of all the local simplification operations used to generate it. Because it is based on the CSF, we cannot claim that imperceptibility distance necessarily predicts suprathreshold performance, or that simplifying according to imperceptibility distance will necessarily provide the best simplification when viewed from less than that distance. But it at least provides an intuitive physical measure of the fidelity achieved: for a given LOD, the system can report the distance from the screen at which the CSF model predicts the LOD will be indistinguishable from the original model. As we discuss in Section 6, simplifying according to imperceptibility distance seems to do well in practice.

4 PREPROCESSING

We build our MTs by progressive edge collapse simplification with the goal of minimizing object-space texture deviation. We then run a preprocessing stage that augments a basic MT with the structures used by our perceptual run-time simplification. The preprocessing maps nodes in the MT to the triangles in the original model to which they correspond in the texture parameterization, and calculates texture luminance ranges, bounding spheres, and normal cones from those triangles.

To facilitate mapping nodes to their corresponding full-resolution triangles, we build an image pyramid of the original textures. The bottom level of this pyramid represents the full-resolution texture, and we store for every texel a list of the triangles that intersect it. From these lists, we can compute a bounding sphere that contains all triangles that map to that texel, normal cones that bound the normals of the triangles and vertices or normal map, and a luminance range L_{min} – L_{max} for those triangles. We can propagate this information up the pyramid to represent bounding spheres, normal cones, and luminance ranges for progressively larger patches of the original surface.

Once the image pyramid is built, we determine the perceptual structures for a given node by hierarchically rasterizing coarse-resolution the triangles of the node into the pyramid and updating the bounding sphere, normal cones, and luminance ranges according to the regions those triangles cover in the pyramid. If a region of the pyramid is completely covered, we can use the bounds stored with the region directly; if a region partially intersects a triangle, we recursively test the triangle against the next level of the pyramid. The hierarchical evaluation makes the precomputation fairly efficient; preprocessing the armadillo model, with 500,000 triangles and over 100 textures, takes about 15 minutes on a 1.2 GHz PC. We believe this could be further accelerated by clever use of the graphics hardware, but have not felt the need to do so. The image pyramids themselves may be discarded after the data for all the MT nodes have been computed.

A note about calculating luminance: we compute luminance using the standard RGB→Y coefficients for modern CRT monitors in Recommendation 109 [Poynton 1998], gamma

corrected for our display hardware and accounting for the measured ambient light level in our lab. Clearly much more care and calibration would be required to guarantee true imperceptible simplification; however, for our best-effort approach a rough approximation that captures the shape of the curve suffices.

5 RUN-TIME SIMPLIFICATION

Here we describe our framework for run-time perceptual simplification. Our basic algorithm is triangle budget simplification driven by imperceptibility distance. We begin with an overview of our technique for adapting an MT to a budget, followed by a description of how we modify our contrast computation to account for texture content, silhouettes, and dynamic lighting.

5.1 Best-effort MT refinement

Best-effort simplification aims to minimize some error criterion – in our case the LOD's imperceptibility distance – while remaining within the user-specified triangle budget. Recall that each node in the MT can be thought of as a reversible local simplification operation. These local simplifications each incur some error, captured by the node's imperceptibility distance. We can simplify to a budget using a simple greedy top-down algorithm that starts each frame by moving the cut to the source node (simplest model) and iteratively raises the node with the largest imperceptibility distance (thus refining the model in that region). This top-down algorithm is effectively an adaptation of [Luebke and Erikson 1997] budget simplification technique for the MT, and is simple but slow. Traversing from the root usually incurs extra overhead, because every frame many nodes are unnecessarily evaluated, enqueued, shuffled around the heap, dequeued, and raised. We improve the efficiency of this algorithm by using a dual-queue implementation similar to the ROAM terrain simplification algorithm by [Duchaineau et al. 1997]. This approach exploits temporal coherence by beginning each frame with the cut X and queue pair L,R from the previous frame:

```
simplifyBudget(cut X,
               raiseQueue R,
               lowerQueue L,
               int Budget)
   R.updateErrorKeys();
   L.updateErrorKeys();
   while (liftNode != dropNode)
      while (X.numTris > Budget && !L.isEmpty())
         Node dropNode = L.removeTop();
         X.lowerNode(dropNode);
      while (X.numTris <= Budget && !R.isEmpty())
         Node liftNode = R.removeTop();
         X.raiseNode(liftNode);
```

The priority queue R stores nodes directly below the cut (candidates to raise) and L stores nodes directly above the cut (candidates to lower). Each frame the algorithm recomputes the imperceptibility distance of nodes in the queues; it then iteratively lifts the node with the maximum distance and drops the node with the minimum distance until these represent the same node. Again, lifting a node may require lifting parent nodes that are below the cut while dropping a node may require recursively dropping child nodes, and then a node is lifted or dropped, it and its parents or children must be added to the appropriate queue. We also amortize the cost of updating the queues over several frames in a fashion similar to [Duchaineau et al. 1997] and [Hoppe 1997].

5.2 Texture contrast

On textured models, estimating the contrast of a given node is a straightforward process that may be precomputed prior to rendering. Each node represents a mesh simplification operation over the triangles on a given patch of surface. The parameterization of the texture lets us map this patch to the corresponding small patch on the original surface, generating a list of all triangles on the original surface that share the same portion of the texture [Schilling and Klein 1998]. Given the original triangles that map to a node, we can precompute the luminance values of all texels covered by those triangles. Section 4 discussed the details of this preprocess.

Note that it would be incorrect to examine only the texture covered by the simplified triangles in the node, because those triangles may not map to the exact region of the texture mapped to by the original model. This highlights an important point: because we base simplification decisions on the perceptibility of features from the original model, we must take care to always consider the cumulative, rather than incremental, effect of a simplification.

5.3 Silhouettes and visibility

As discussed in Section 3, the silhouette status of a region affects its possible contrast. Accounting for the higher contrast of silhouette regions provides a natural framework for silhouette preservation grounded in perceptual principles. To detect whether nodes are on the silhouette, we use the standard approach described by [Luebke and Erikson 1997] of storing a *silhouette normal cone* with each node that bounds the set of triangle normals; comparing the normal cone, bounding sphere, and view vector lets us quickly decide whether the node might be on the silhouette. The normals that comprise a node's silhouette normal cone come from the triangles in the original model that are associated with the node, and from the triangles of the node itself (because a simplified surface may well contain sharper dihedral angles than the original).

Our normal cone based silhouette test also indicates whether a node is back facing at no additional cost. Back facing nodes and nodes that lie outside the view frustum comprise a special class of invisible nodes that are imperceptible from any distance. We set the imperceptibility distance of invisible nodes to zero to reflect the fact that they cannot locally influence the error associated with simplification. Although these tests incur some overhead to perform, knowing that an invisible node's associated triangles can't be seen saves us some time during rendering.

5.4 Dynamic lighting

We can also account for dynamically lit models in our contrast calculation. In addition to standard Gouraud-shaded vertex lighting, we can apply perceptual simplification to *normal maps* for extremely high quality LODs. Normal maps, once an esoteric feature only available offline or on the most exotic hardware, are now supported on commodity graphics chipsets. Visual quality of simplified models is often drastically increased by the use of normal mapping, so this is a useful mode to support. The choice of normal map versus per-vertex lighting can drastically affect the perceptual quality of the resulting simplification, because per-vertex lighting effects (for example, a specular highlight) are interpolated by Gouraud shading across all triangles in a node. In other words, a color shift caused by applying the local simplification operation encoded by a node can affect the entire region of the image spanned by the node. With normal maps, on the other hand, as with texture maps, the shading is somewhat decoupled from the underlying mesh: the same normals are used for the original and simplified surface, and the extent of a color shift is bounded by the texture deviation. To incorporate lighting effects into our system, therefore, we calculate spatial frequency

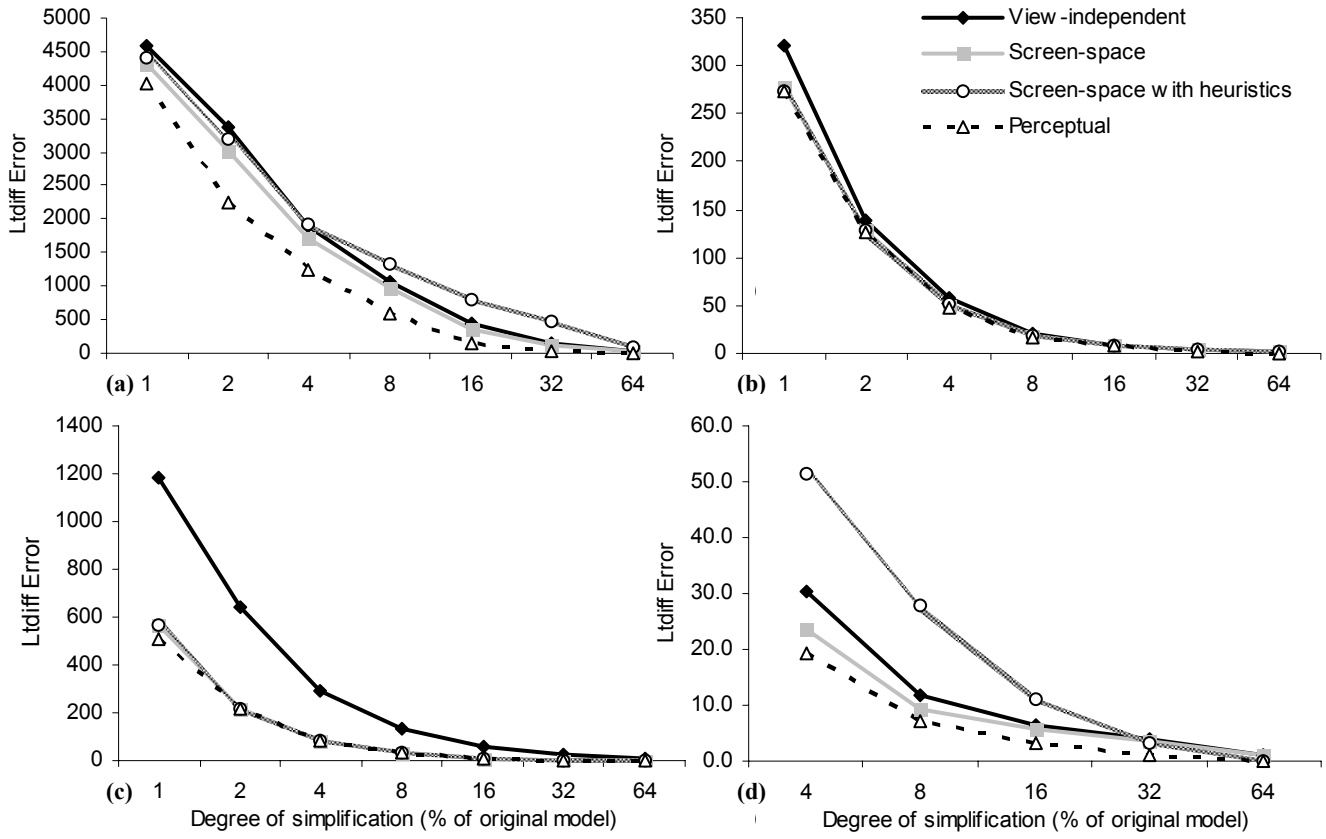


Figure 4: Results of `ltdiff` evaluation on the scenes in Figure 6. (a) 500,000-triangle red armadillo model with per-vertex illumination. (b) The same armadillo with normal maps for per-pixel illumination. (c) 500,000-triangle textured terrain model, no lighting. (d) 238,140-triangle torus with per-vertex lighting and puzzle texture.

using a feature size based on either the projected extent of the texture deviation (for normal map lighting) or the node's bounding sphere (for per-vertex lighting).

Integrating dynamic lighting also requires us to dynamically adjust the contrast associated with nodes. The luminance range associated with a lit node is a function not only of its intrinsic color, but also of the light vector, view vector, and its *shading normal cone*. The shading normal cone, like the silhouette normal cone, simply bounds the normals associated with a node; the only difference is that the silhouette cone is constructed from the original triangles associated with a node, while the shading cone is constructed from the normal map or vertex normals spanned by those triangles.

Our normal mapping algorithm was implemented as a texture combiner program on an nVidia GeForce3, and is simpler than the full OpenGL lighting model. The luminance Y at a vertex is given by:

$$Y = k_a T + k_d T(N \bullet L) + (N \bullet H)^n,$$

where T is the intrinsic surface color read from a texture map, L is the light vector, H is the halfway vector of the Blinn-Phong lighting model, N is from the normal map, k_a and k_d are the ambient and diffuse lighting coefficients, respectively, and n is the specular exponent. The light source and viewer are assumed to be at infinity in this calculation. For per-vertex lighting, we calculate luminance using OpenGL's light model for an infinite directional light source and viewer. We could support more complex lighting models (e.g., point sources), or more than one light, at the cost of some additional computation.

Given the lighting model, we can bound the luminance of the diffuse contribution by calculating the vector encompassed by the shading normal cone that is closest in direction to L and the vector furthest in direction from L. Similarly, we find the range of specular contribution using the halfway vector. Note that this computation is similar to that of [Klein and Schilling 1999].

6 RESULTS AND EVALUATION

Thus far we have described the simplification effects that our algorithm automatically accounts for: silhouette preservation, texture-content sensitive simplification, and illumination sensitive simplification. Here we visually and quantitatively compare the quality of the resulting simplifications to those produced by other algorithms.

We provide a rigorous comparison of our system to a view-dependent implementation of the appearance-preserving simplification (APS) scheme of [Cohen et al. 1998], one of the highest-fidelity simplification algorithms available. APS was the first simplification algorithm to make strong guarantees on the rendered fidelity of LOD; it focuses on bounding the possible screen-space distortion caused by simplification. Like our system, APS measures parametric distortion and factors appearance into color (represented by texture maps) and shading (represented with normal maps). Whereas the original algorithm uses this bound to choose a static LOD, the view-dependent version uses our MT implementation to simplify to a budget while minimizing projected screen-space error of nodes on the MT cut.

Our experiments compare the new perceptual error metric with three others: object-space (view-independent) error, screen-space texture deviation, and screen-space texture deviation with a ten-fold multiplicative factor in silhouette regions. This last variant exemplifies a common heuristic: user-specified weighting

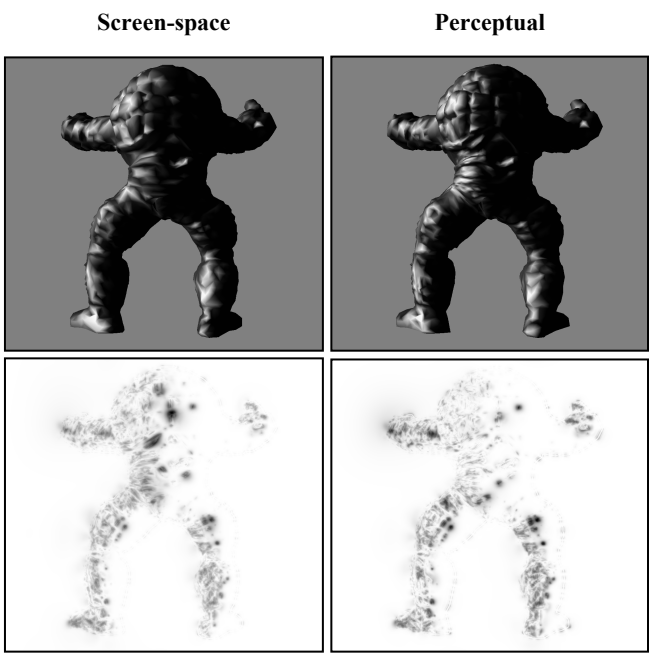


Figure 5. Top: rear view of the 500,000 triangle armadillo model, 98% simplified under per-vertex lighting conditions. Bottom: `ltdiff` visualization of the error associated with simplification from the original model. Darker areas indicate larger error. Notice that perceptually driven simplification exhibits fewer strong errors on the high contrast back than screen-space error.

of silhouette importance (which is difficult to tune). We investigate fully APS-compliant scenarios, using texture and normal maps, as well as those with per-vertex normals and even no normals.

We quantitatively measure simplification quality by pairwise comparison of 1024x1024 images. For a given triangle budget and view of the model we capture images with and without simplification enabled. We calculate the error due to simplification by using Lindstrom's `ltdiff` image metric on these image pairs [Lindstrom 2000]. The `ltdiff` image metric utilizes a computational model of the human visual system tailored to estimate the perceptual differences between 3D models (see Figure 5 and Plate 2). This metric provides a more accurate measure of perceptible error due to simplification than simpler methods such as finding the root-mean-squared error (RMSE) of image differences. For example, the `ltdiff` metric attenuates the phase information associated with silhouette simplification that RMSE over-emphasizes in error measurement. However, because high contrast silhouettes still do affect the `ltdiff` error, we maximize fairness by setting the background color to a mid-level gray in all scenes. In fact, our algorithm often performs better with a black background to conceal dark silhouette edges. We include plots of `ltdiff` error from a single representative view against triangle budget for different scenes, textures, and lighting conditions (Figure 4).

As we expect, using a perceptual model generally provides improved simplification. The benefit is most pronounced on vertex-lit models, primarily because the distortion and tessellation artifacts in specular highlights are highly perceptible (Figure 4a,d). Using normal maps maintains smooth highlights even at low resolutions. Under these conditions the primary differences between our algorithm and APS are the ability to simplify low-contrast regions (washed out highlights or dark shadow), and the ability to preserve high-contrast areas such as silhouettes. Likewise, the perceptual method with texture maps alone does not provide the significant improvement that is found in lighting with per-vertex normals. Except at significant simplification levels, these effects are less important visually.

7 DISCUSSION AND FUTURE WORK

Just as view-dependent algorithms gain benefits and incur costs not present in view-independent systems, our perceptual model provides intelligent simplification not present in other algorithms—aggressive simplification in low-contrast regions, such as uniform texture areas and washed-out specular highlights, along with intelligent refinement at specular highlights and silhouette regions—but comes at a computational cost. Other algorithms have been augmented with manually weighted heuristics to account for most of these opportunities, such as the use of tighter error thresholds for silhouettes. One could argue that evaluating such heuristics probably requires less computation than our perceptual model, and that heuristics could be developed to account for all the simplification effects we support. But this would be missing the point: our chief contribution is a way to *avoid* ad hoc hand-tuned heuristics—or perhaps, in future work, to guide their development—by reasoning directly from principles of visual perception.

7.1 Avenues for future work

While our initial system shows promise, many avenues of future work remain. Perhaps the most important topic for future research is the integration of better perceptual models. We would like to extend our perceptual model to include important effects such as local adaptation (TVI effects), chromatic contrast sensitivity, and temporal effects (flicker sensitivity, sudden onset). In particular, it would seem fruitful to investigate efficient ways to model visual masking. The frequency content of textures and normal maps has a strong effect on the perceptibility of the simplification; we

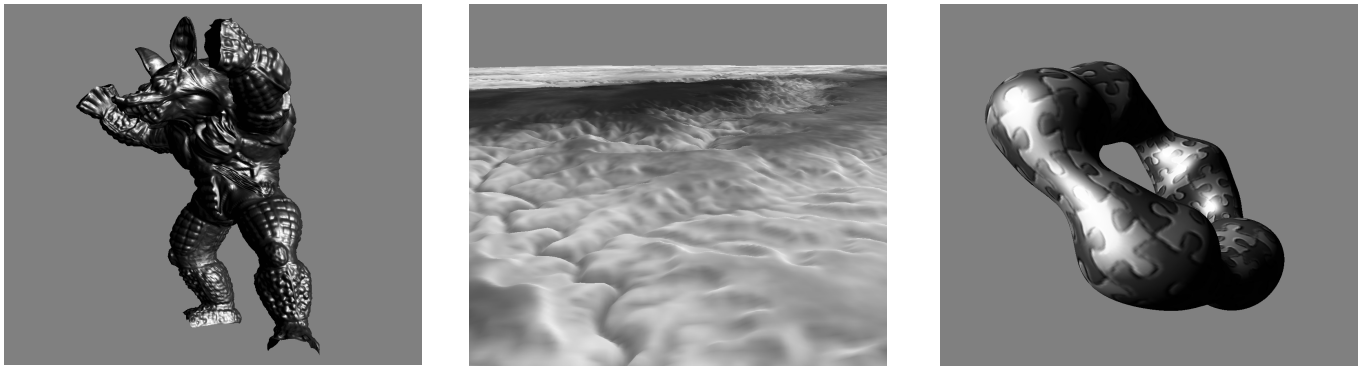


Figure 6. Scenes used to gather the data in Figure 5: 500,000-triangle red armadillo model; 500,000-triangle terrain model, 240,000-triangle torus model.

believe a simple model of visual masking, perhaps based on pre-computed frequency content in the textures, would often enable much more aggressive simplification. Along these lines the work on perceptual texture caching by [Dumont et al. 2001] appears promising for future investigation. More generally, a dire need exists for adequate models of suprathreshold perceptibility that are efficient enough for an interactive framework.

One useful extension would be to account for MIP-map filtering when calculating texture contrast. Many textures have noise or high-frequency components that introduce a great deal of contrast to our algorithm, which simply assigns a node a contrast from the luminance range it covers in the texture. Often these high-frequency components are filtered out in the first or second MIP level, leaving a low-contrast texture that could be simplified much more aggressively. We expect, for example, that our perceptual simplification system would out-perform APS on terrain visualization if equipped with MIP-mapping support (Figure 4c).

We would also like to investigate optimizing the MT construction for perceptual simplification. Currently we simply apply our perceptual metrics to pre-built MTs, which were constructed with the goal of minimizing texture deviation. However, building MTs tailored for given textures should allow the construction process more leeway, for example in areas of low contrast. It also seems helpful to investigate "quick and dirty" parameterizations that could be used to apply our algorithm to non-textured models. A great deal of excellent research has been carried out in the realm of automatic parameterization, but it remains a difficult problem. However, even a simplistic approach should suffice for our method, which simply needs to establish a correspondence between nodes in the MT and the original triangles to which they relate.

ACKNOWLEDGEMENTS

We would like to thank Venkat Krishnamurthy, Marc Levoy, Peter Schröder at Caltech for making the armadillo model available; Hugues Hoppe, the USGS, and Chad McCabe for providing the Grand Canyon model, Greg Turk for the jigsaw puzzle texture, and Peter Lindstrom for the use of his excellent `ltdiff` perceptual error estimation code. This work was supported in part by NSF awards 0092973, 9703080, and 0205586.

REFERENCES

BOLIN, M., MEYER, G. 1998. A Perceptually Based Adaptive Sampling Algorithm. *Proceedings of SIGGRAPH 98*, 299-309.

COHEN, J.D., OLANO, M., AND MANOCHA, D. 1998. Appearance-Preserving Simplification. *Proceedings of SIGGRAPH 98*, 115-122.

DALY, S. 1993. The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity. In A. Watson, ed. *Digital Images and Human Vision*. MIT Press, Cambridge, MA, 179-206.

DEFLORIANI, L., MAGILLO, P., AND PUPPO, E. 1997. Building and Traversing a Surface at Variable Resolution. *Proceedings of IEEE Visualization '97*, 103-110.

DEFLORIANI, L., MAGILLO, P., AND PUPPO, E. 1998. Efficient Implementation of Multi-Triangulations. *Proceedings of IEEE Visualization '98*, 43-50.

DUCHAINEAU, M., WOLINSKY, M., SIGETI, D.E., MILLER, M.C., ALDRICH, C., AND MINEEV-WEINSTEIN, M.B. 1997. ROAMing Terrain: Real-time Optimally Adapting Meshes. *Proceedings of Visualization '97*, 81-88.

DUMONT, R., PELLACINI, F., AND FERWERDA, J.A. 2001. A Perceptually-Based Texture Caching Algorithm for Hardware-Based Rendering. *Proceedings of 2001 Eurographics Workshop on Rendering*, 249-256.

ERIKSON, C. AND MANOCHA, D. 1999. GAPS: General and Automatic Polygonal Simplification. *Proceedings of 1999 ACM Symposium on Interactive 3D Graphics*, 79-88.

FERWERDA, J. A., PATTANAIK, S., SHIRLEY, P., AND GREENBERG, D. P. 1997. A Model of Visual Masking for Computer Graphics. *Proceedings of SIGGRAPH 97*, 143-152.

FUNKHOUSER, T. A. AND SEQUIN, C. H. 1993. Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. *Proceedings of SIGGRAPH 93*, 247-254

GARLAND, M. AND HECKBERT, P. 1998. Simplifying Surfaces with Color and Texture using Quadric Error Metrics. *Proceedings of IEEE Visualization '98*, 263-270.

HOPPE, H. 1999. Optimization of Mesh Locality for Transparent Vertex Caching. *Proceedings of SIGGRAPH 99*, 269-276.

HOPPE, H. 1997. View-Dependent Refinement of Progressive Meshes. *Proceedings of SIGGRAPH 97*, 189-198.

KLEIN, R. AND SCHILLING, A. 1999. Efficient rendering of multiresolution meshes with guaranteed image quality. *The Visual Computer 15*, 9, 443-452.

LEE, A., MORETON, H., AND HOPPE, H. 2000. Displaced Subdivision Surfaces. *Proceedings of SIGGRAPH 2000*, 85-94.

LINDSTROM, P. 2000. *Model Simplification Using Image and Geometry-Based Metrics*. Ph.D. Thesis, Georgia Institute of Technology.

LINDSTROM, P., AND TURK, G. 2000. Image-driven Simplification. *ACM Transactions on Graphics 19*, 3, 204-241.

LUEBKE, D., AND ERIKSON, C. 1997. View-Dependent Simplification of Arbitrary Polygonal Environments. *Proceedings of SIGGRAPH 97*.

LUEBKE, D., AND HALLEN, B. 2001. Perceptually Driven Simplification for Interactive Rendering. *Proceedings of 2001 Eurographics Rendering Workshop*, 223-234.

MYSZKOWSKI, K., TAWARA T., AKAMINE, H., AND SEIDEL, H. 2001. Perception-Guided Global Illumination Solution for Animation Rendering. *Proceedings of SIGGRAPH 2001*, 221-230.

POYNTON, C. 1998. The Rehabilitation of Gamma. *Proceedings of Human Vision and Electronic Imaging III*, 232-249.

RAMASUBRAMANIAN, M., PATTANAIK, S. N., AND GREENBERG, D. P. 1999. A Perceptually Based Physical Error Metric for Realistic Image Synthesis. *Proceedings of SIGGRAPH 99*, 73-82.

REDDY, M.. 1997. *Perceptually Modulated Level of Detail for Virtual Environments*. Ph.D. Thesis, University of Edinburgh.

REDDY, M. 2001. Perceptually Optimized 3D Graphics. *IEEE Computer Graphics and Applications 21*, 5, 68-75.

ROSSIGNAC, J., AND BORREL, P. 1993. Multi-Resolution 3D Approximations for Rendering Complex Scenes. *Modeling in Computer Graphics: Methods and Applications*. Springer-Verlag, 455-465.

RUSHMEIER, H., WARD, G., PIATKO, C., SANDERS, P., AND RUST, B. 1995. Comparing Real and Synthetic Images: Some Ideas About Metrics. *Proceedings of 1995 Eurographics Workshop on Rendering*, 82-91.

SANDER, P.V., SNYDER, J., GORTLER, S.J., AND HOPPE, H.. 2001. Texture Mapping Progressive Meshes. *Proceedings of SIGGRAPH 2001*, 409-416.

SCHILLING, A. AND KLEIN, R. 1998. Rendering of Multiresolution Models with Texture. *Computers and Graphics 22*, 6, 667-674.

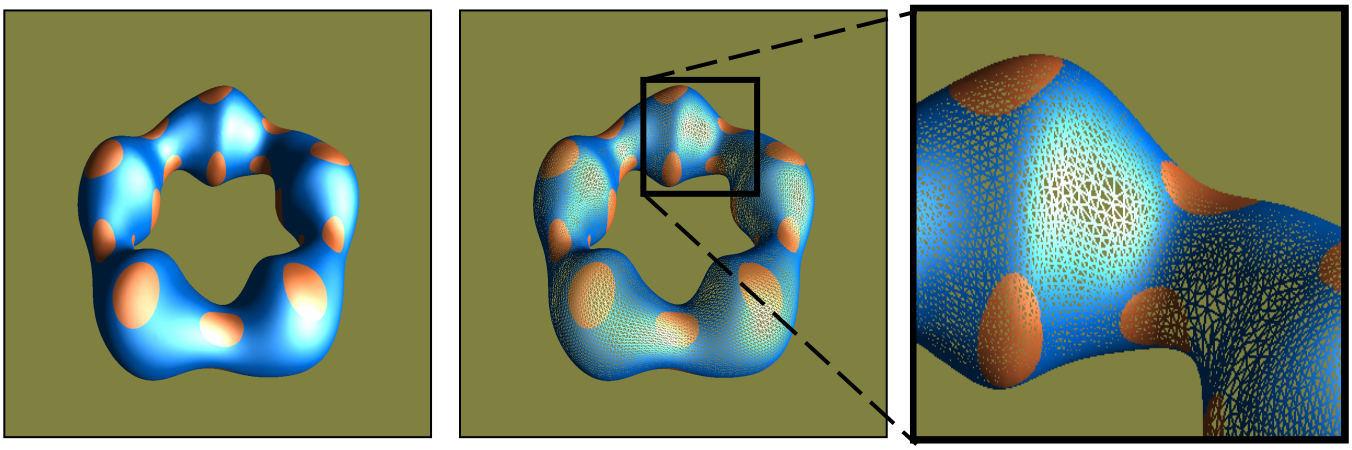
SCHROEDER, W. J., ZARGE, J.A., AND LORENSEN, W. Decimation of Triangle Meshes. *Proceedings of SIGGRAPH 92*, 65-70.

SCOGGINS, R., MACHIRAJU, R., AND MOORHEAD, R.J. Enabling Level of Detail Matching for Exterior Scene Synthesis. *Proceedings of IEEE Visualization 2000*, 171-178.

XIA, J. C. AND VARSHNEY, A.. Dynamic View-Dependent Simplification for Polygonal Models. *Proceedings of IEEE Visualization '96*, 327-334.

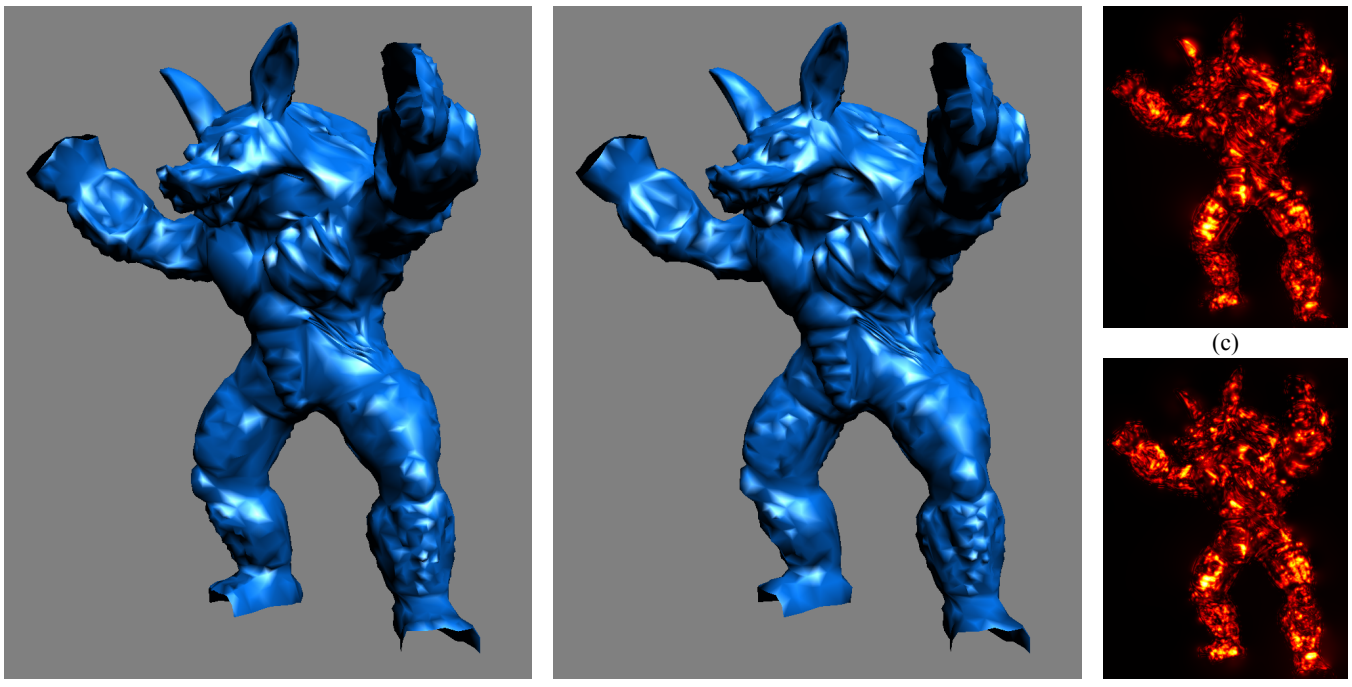
Perceptually Guided Simplification of Lit, Textured Meshes

Nathaniel Williams, David Luebke, Jonathan D. Cohen, Michael Kelley, Brenden Schubert



(a) (b) (c)

Plate 1: Contrast calculation and simplification effects on a textured and lit torus. The model is shown at full resolution with 57,660 triangles (a) and simplified by 50% (b). The close-up (c) illustrates preservation of silhouettes and extra simplification in low-contrast areas, such as washed-out specular highlights and deeply shadowed regions.



(a) (b) (c) (d)

Plate 2: Visual comparison of the 500,000-triangle armadillo model, rendered with per-vertex lighting and simplified by 98% with (a) screen-space deviation and (b) our perceptual metric. (c) and (d) show `ltdiff` visualizations of the resulting error as compared to the full-resolution model for the screen-space and perceptual metrics, respectively. `ltdiff` reports errors of 3,689 for the screen-space metric and 3,123 for the perceptual metric.