Perceptually Guided Simplification of Lit, Textured Meshes

David Luebke[1], Jonathan Cohen[2], Nathaniel Williams[1], Mike Kelly[1], Brenden Schubert[1]
[1] University of Virginia, [2]Johns Hopkins University

University of Virginia technical report CS-2002-03

ABSTRACT

We present a new algorithm for best-effort simplification of polygonal meshes based on principles of visual perception. Following previous work, we use a simple model of low-level human vision to estimate the perceptibility of local simplification operations in a view-dependent multitriangulation 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 parameterized *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.

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?

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 Rossignac and Borrel's vertex-clustering approach (1993), or Schroeder's decimation algorithm (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 et al (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 1998, Erikson 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. Our system is based throughout 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.

Our work builds on research efforts by several groups. Our algorithm relates most closely to the approach of Luebke and Hallen (2001), but applies to a much broader class of models since 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 parameterized texture deviation, as well as the *multi-triangulation* (*MT*) data structure by DeFloriani et al (1997, 1998). We discuss this and other related research in the following sections.

1.1 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 parameterized texture deviation to measure distortion more accurately than the Luebke-Hallen approach. The result is better simplification for a given polygon count [Figure 5].
- 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 public-domain image comparison toolkit.

1.2 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 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 radiologist, for example, would probably rather suffer slow frame rates than trust that tumors are not being simplified away. Finally, and most important, 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 next section) 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 (e.g., Ramasubramanian 1999, Myszkowski 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 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.

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), Myszkowki 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 (1996). 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. Ramasubramanian et al, for example, 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. 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 parameterized surface distortion to derive a screenspace bound on the movement of color (represented by a texture map) and lighting (represented by a normal map) across the surface. We build on their approach and return to it below. Similar work by Schilling and Klein (1998), and later work by the same authors (Klein and Schilling 1999), also deserves mention. In the first publication, they account for texture distortion using a surface mapping technique similar to that of Cohen et al; in the second, 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 [Figure 4].

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 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*.

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 those inserted by the corresponding refinement operation 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 apply-

ing the simplification encoded by its start node (above A) consumes the 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 refinement 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 4.1.

Advantages of the MT: Since all triangles in all possible simplifications are explicitly represented in the MT, we can precompute accurate object-space error bounds, texture contrasts, normal cones, etc. This is the major advantage of the MT for our application over other well-known simplification hierarchies, such as the vertex-merging trees of Hoppe (1997) and Luebke and Erikson (1997). In those hierarchies 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, vertex arrays, etc.

2.3 Texture deviation

An appropriate geometric way to measure the error of texture mapped surfaces is to bound the texture deviation [Cohen 98, Lee 2000, Sander 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 texel deviation vector in screen space. This measures the number of pixels of deviation model and bounds the shift of texels in screen-space as a result of simplification.

As we will see, 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 perceptibility distance.

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*. 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 contrast that can be perceived in a grating of that frequency, and *contrast sensitivity* is defined as the 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 1992):

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

Where A_D represents contrast sensitivity and f represents spatial frequency in cycles per degree. In practice, we represent this messy and expensive empirically-determined formula with a lookup table for speed.

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

Since 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] Since the minimum frequency component of an image feature that spans n degrees of visual arc is one cycle per $2n$ 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.

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 [Ferwada 96]. 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 much tighter bound on the screenspace region affected [Figure 5].

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

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

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. Since 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. Since 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. Since 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 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 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.

4.1 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's (1997) budget simplification technique for the MT, and is simple but slow. Traversing from the root usually incurs extra overhead, since 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 from the last frame. One priority queue stores nodes below the cut (candidates to lift) and another stores nodes above the cut (candidates to drop). 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 and Hoppe (1997).

4.2 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 1998). Given the original triangles that map to a node, we can precompute the luminance values of all texels covered by those triangles. Section 5.1 discusses the details of this preprocess.

Note that it would be incorrect to examine only the texture covered by the simplified triangles in the node, since those triangles may not span the entire texture spanned by the original model. This highlights an important point: since 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.

4.3 Silhouettes

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 (2000) 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 (since a simplified surface may well contain sharper dihedral angles than the original).

4.4 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 texture deviation to 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, since 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 using a feature size based on 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 range at a vertex is given by:

$$Luminance = K_{ambient} * TexVal + K_{diffuse} * TexVal * (N \bullet L) + (N \bullet H)^n$$

where *TexVal* 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, and N is from the normal map. 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).

5 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 an arbitrary 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 on 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 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 over 100 textures and over 400,000 triangles, takes about 3 minutes. We believe this could be further accelerated by clever use of the graphics hardware, but have not felt the need to do so.

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 seems sufficient.

6 RESULTS AND EVALUATION

The preceding sections and figures demonstrate the visual results of our approach, and highlight the simplification effects that it 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.

As a fair comparison, we decided to contrast our system with a view-dependent implementation of the *appearance-preserving simplification* or *APS* scheme of Cohen et al (1998). We should emphasize that this is a rigorous comparison against one of the higher fidelity simplification algorithms available. APS was the first simplification algorithm to attempt 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 parameterized 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 multitriangulation implementation to simplify to a budget while minimizing projected screen-space error of nodes on the MT cut. We also compare our system to a view-independent implementation of APS that simplifies the multitriangulation according to object-space texture deviation, as well as a heuristic approach that uses APS but multiplies the screenspace texture deviation tenfold for silhouette regions. [Figures 1-3].

We also make a limited comparison to the Luebke-Hallen approach—limited because a fair comparison is difficult. The Luebke-Hallen algorithm does not support dynamic lighting, so we use a prelit model acquired from a Cyberware laser scanner. Also, the Luebke-Hallen approach does not support textured models, but it would be grossly unfair to compare a Gouraud-shaded model with significant reduction in polygon count to a similarly-reduced texture-mapped model. We therefore extend the Luebke-Hallen approach to render with and compute contrast from the texture map. Beyond lighting and texturing, the major differences between our algorithms are the use of the multitriangulation versus VDSlib vertex merging, and the use of node bounding sphere versus texture deviation to estimate feature size and spatial frequency. We decided to implement the Luebke-Hallen algorithm in our MT framework using node radius rather than texture deviation; as Figure 5 shows, the tighter bound provided by texture deviation improves the quality of the simplification.

In addition to visual inspection, we also report two quantitative measures of image fidelity: RMS and JND. RMS is simple root-mean-squared error based on the pairwise difference of pixels. RMS is much-criticized as an image fidelity metric, but does give an intuitive feeling for how much the pixels are changing. The JND metric represents the *just-noticeable-difference* count returned by *DCTune*, a public-domain software package by Watson (1994) used for optimizing the discrete cosine transform (DCT) basis functions to design custom JPEG quantization matrices. One feature of DCTune, designed to evaluate quanitization errors in image compression, takes as input two images and returns a measure of their similarly in JNDs. DCTune uses a perceptual color space and accounts for luminance masking (local adaptation) as well as contrast masking (facilitation and suppression of one pattern by another). We include plots of RMS error and JNDs against triangle budget for different models, textures, and lighting conditions [Figures 1,2,3,5].

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. 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. Except at certain 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 Luebke and Erikson's 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.

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 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. Note that by not accounting for MIP-mapping we are at least treating the model conservatively, since MIP-mapping should only reduce contrast and hence perceptibility.

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, but 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.

REFERENCES

Bolin, Mark. and G. Meyer. "A Perceptually Based Adaptive Sampling Algorithm", *Computer Graphics*, Vol. 32 (SIGGRAPH 98).

Cohen, J, M. Olano, and D. Manocha. "Appearance-Preserving Simplification," *Computer Graphics*, Vol. 32 (SIGGRAPH 98).

Daly, Scott, "The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity", *Proceedings of SPIE*, Vol. 1616, pp. 2-15. 1992.

DeFloriani, Leila, Paola Magillo, and Enrico Puppo. Building and Traversing a Surface at Variable Resolution. Proceedings of IEEE Visualization '97. pp. 103-110.

DeFloriani, Leila, Paola Magillo, and Enrico Puppo. Efficient Implementation of Multi-Triangulations. Proceedings of IEEE Visualization '98. pp. 43-50.

Duchaineau, M., M. Wolinsky, et al. (1997). "ROAMing Terrain: Real-time Optimally Adapting Meshes". *Proceedings of IEEE Visualization 97*

Dumont, R., Pellacini, F., & Ferwerda, J. A. (2001). A perceptually-based texture caching algorithm for hardware-based rendering. *Proceedings Eurographics Workshop on Rendering 2001*, Springer (2001).

Erikson, C. and D. Manocha (1999). "GAPS: General and Automatic Polygonal Simplification". *1999 ACM Symposium on Interactive 3D Graphics*.

Ferdwada, James, S. Pattanaik, P. Shirley, and D. Greenberg. "A Model of Visual Masking for Realistic Image Synthesis", *Computer Graphics*, Vol. 30 (SIGGRAPH 96).

Funkhouser, Tom, and C. Sequin. "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments", *Computer Graphics*, Vol. 27 (SIGGRAPH 93).

Garland, Michael and Paul Heckbert. Simplifying Surfaces with Color and Texture using Quadric Error Metrics. Proceedings of IEEE Visualization '98. pp. 263-270.

Hoppe, Hughes. "View-Dependent Refinement of Progressive Meshes", *Computer Graphics*, Vol. 31 (SIGGRAPH 97).

Hoppe, Hugues H. New Quadric Metric for Simplifying Meshes with Appearance Attributes. Proceedings of IEEE Visualization '99. pp. 59-66.

Klein, Reinhard and Andreas Schilling. Efficient rendering of multiresolution meshes with guaranteed image quality . The Visual Computer . vol. 15 (9). 1999 . pp. 443-452 .

Lee, Aaron, Henry Moreton, and Hugues Hoppe. Displaced Subdivision Surfaces. Proceedings of SIGGRAPH 2000.

Lindstrom, Peter and Greg Turk. Image-driven Simplification. ACM Transactions on Graphics. vol. 19(3). 2000. pp. 204-241.

Luebke, David and Benjamin Hallen. Perceptually Driven Simplification for Interactive Rendering. Proceedings of Eurographics Rendering Workshop. 2001.

Luebke, David, and C. Erikson. "View-Dependent Simplification of Arbitrary Polygonal Environments", *Computer Graphics*, Vol. 31 (SIGGRAPH 97).

Myszkowski, Karol, Rakehiro Tawara, hirolyuke Akamine, Hans-Peter Seidel. "Perception-Guided Global Illumination Solution for Animation Rendering. *Proceedings of SIGGRAPH 2001*. 221-230.

Poynton, C. "The rehabilitation of gamma", In *Proceedings of Human Vision and Electronic Imaging III*, vol 3299, pp 232-249. SPIE, San Jose, CA (1998).

Ramasubramanian, Mahesh, S. Pattanaik, and D. Greenberg. "A Perceptually Based Physical Error Metric for Realistic Image Synthesis", *Computer Graphics*, Vol. 33 (SIGGRAPH 99).

Rossignac/Borrel 93

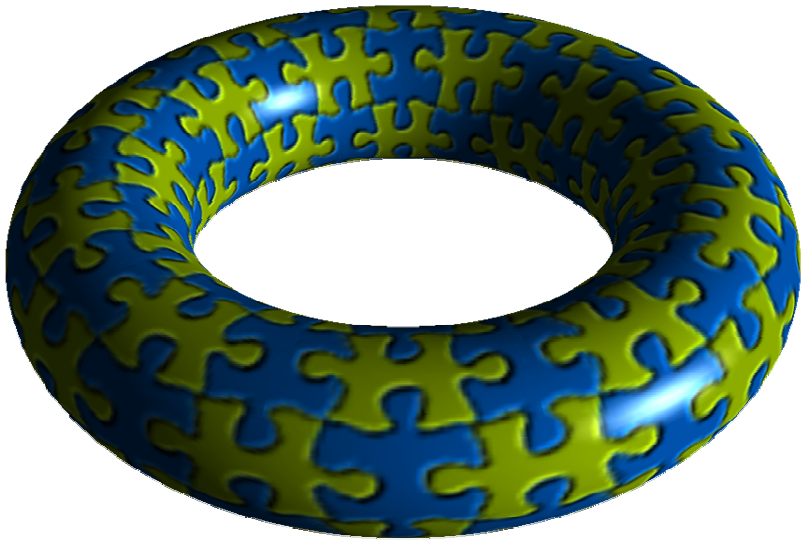
Reddy, Martin. "Perceptually-Modulated Level of Detail for Virtual Environments", Ph.D. thesis, University of Edinburgh, 1997.

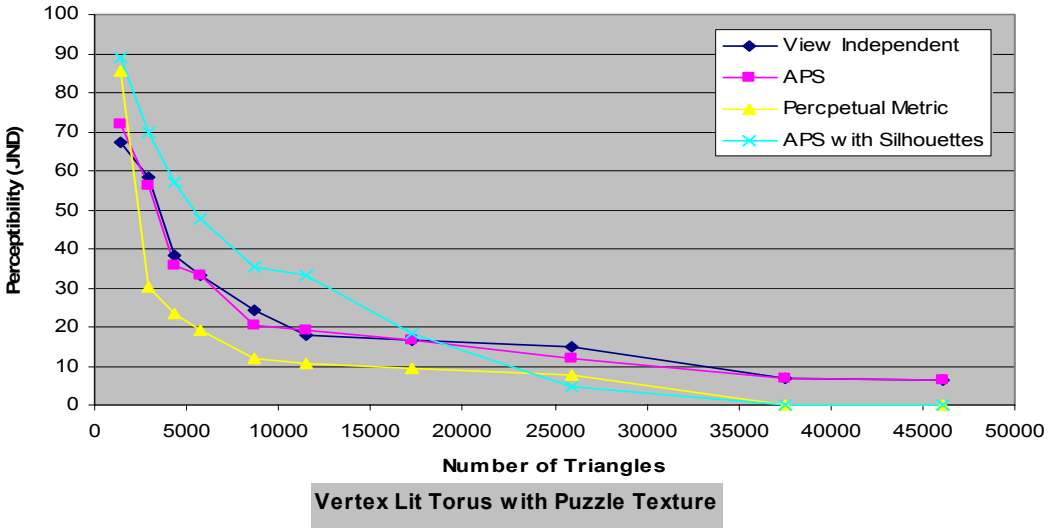
Schilling, A., R. Klein. "Rendering of Multiresolution Models with Texture" *Computers & Graphics* vol. 22, no. 6, pp. 667-674, Dec. 1998.

Schroeder, W. J., J. A. Zarge, et al. (1992). "Decimation of Triangle Meshes", *Computer Graphics (*SIGGRAPH 92).

Scoggins, R., Machiragju, R., and Moorhead, R. J. "Enabling Level of Detail Matching for Exterior Scene Synthesis". In *Proceedings of IEEE Visualization 2000*.

Xia, Julie and Amitabh Varshney. "Dynamic View-Dependent Simplification for Polygonal Models", *Visualization 96*.





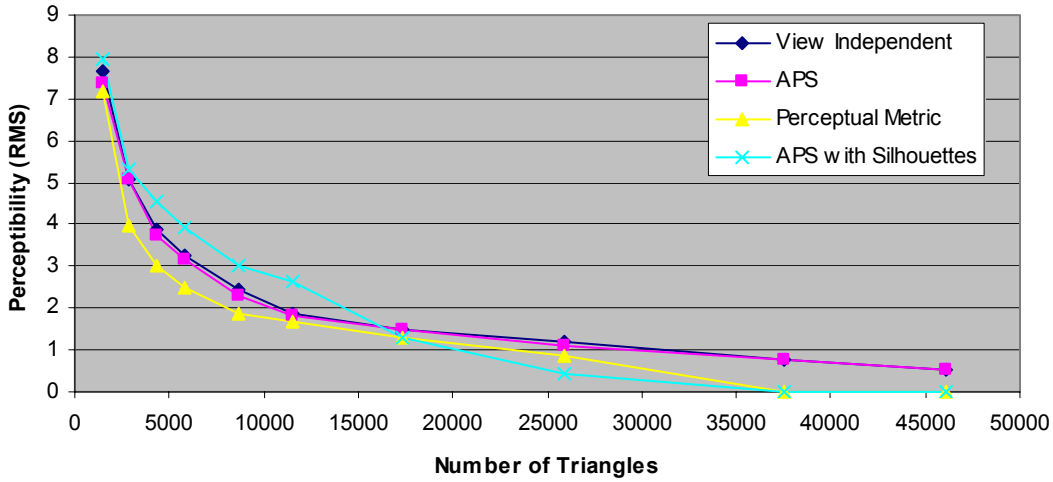


Figure 1: Comparison of different algorithms on the vertex-lit torus model (top), reported in Just Noticeable Differences using the DCTune software package (middle) and RMS pixel difference in luminance (bottom). The ability of the perceptual metric to predict high perceptible artifacts in regions of specular highlights, and allocate more triangles to those regions, gives it an advantage over other algorithms here.

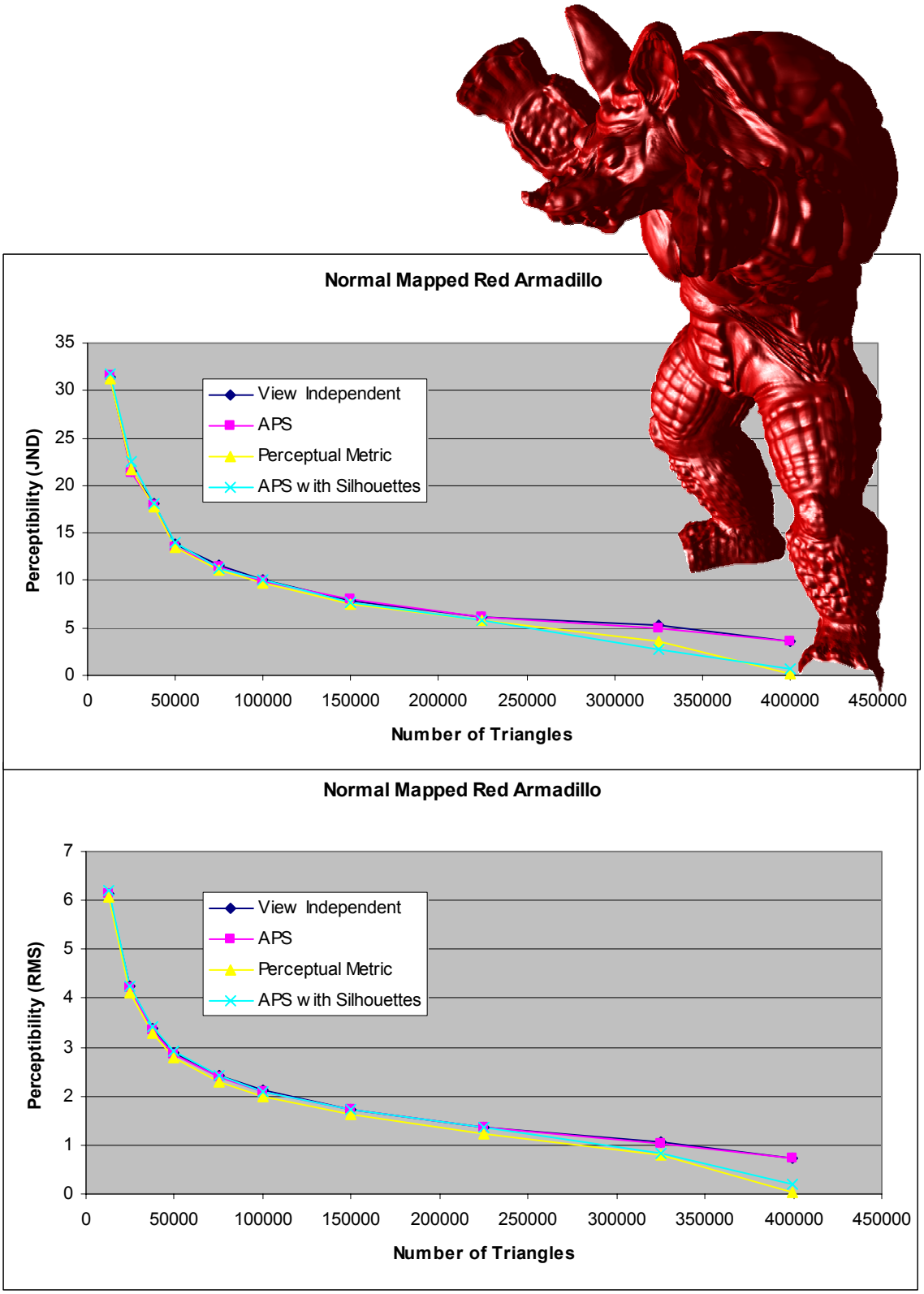


Figure 2: Comparison of different algorithms on the normal-mapped armadillo model (inset), reported in Just Noticeable Differences using the DCTune software package (top) and RMS pixel difference in luminance (bottom). As expected, with normal mapping enabled the benefits of perceptual simplification are comparatively slight.

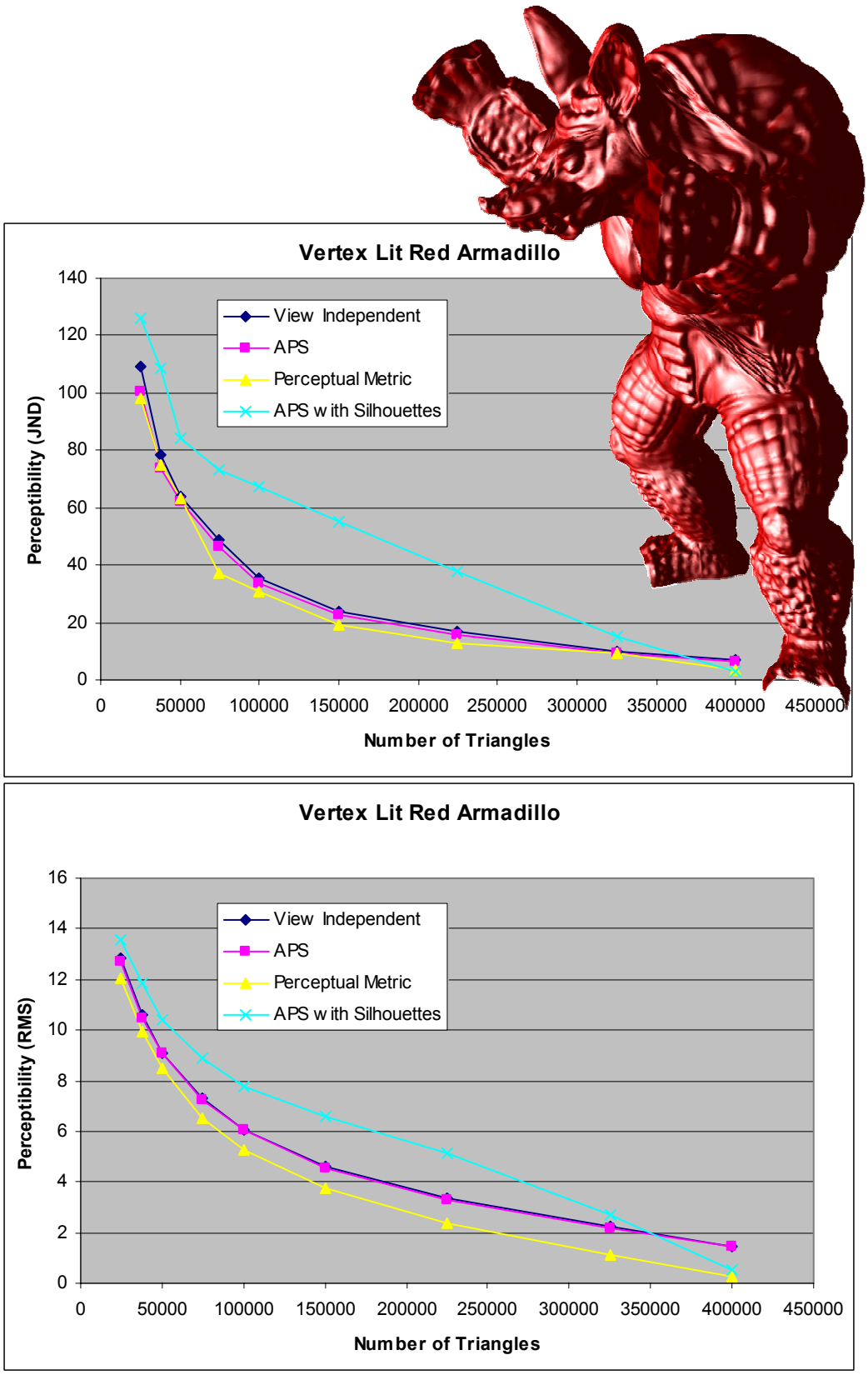
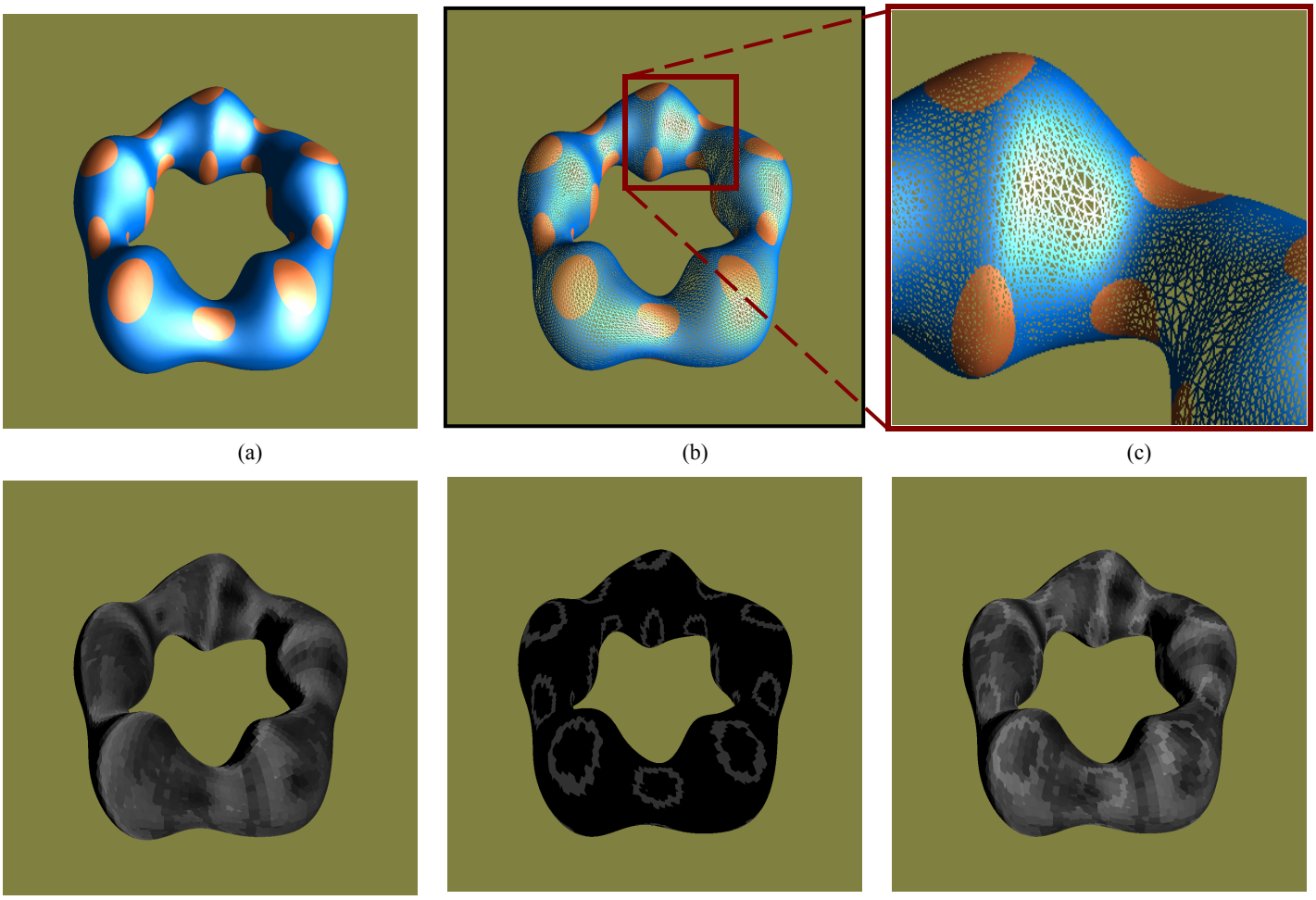


Figure 3: Comparison of different algorithms on the vertex-lit armadillo model (inset), reported in Just Noticeable Differences using the DCTune software package (top) and RMS pixel difference in luminance (bottom). The ability of the perceptual metric to predict high perceptible artifacts in regions of specular highlights, and allocate more triangles to those regions, gives it an advantage over other algorithms here.



(a) (b) (c)

(d) (e) (f)

Figure 4: Contrast calculation and simplification effects. The original model shown at full resolution (a) with 57660 triangles and simplified (b) 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 texture; (f) shows the combined contrast used to generate the simplifications shown in (b) and (c).

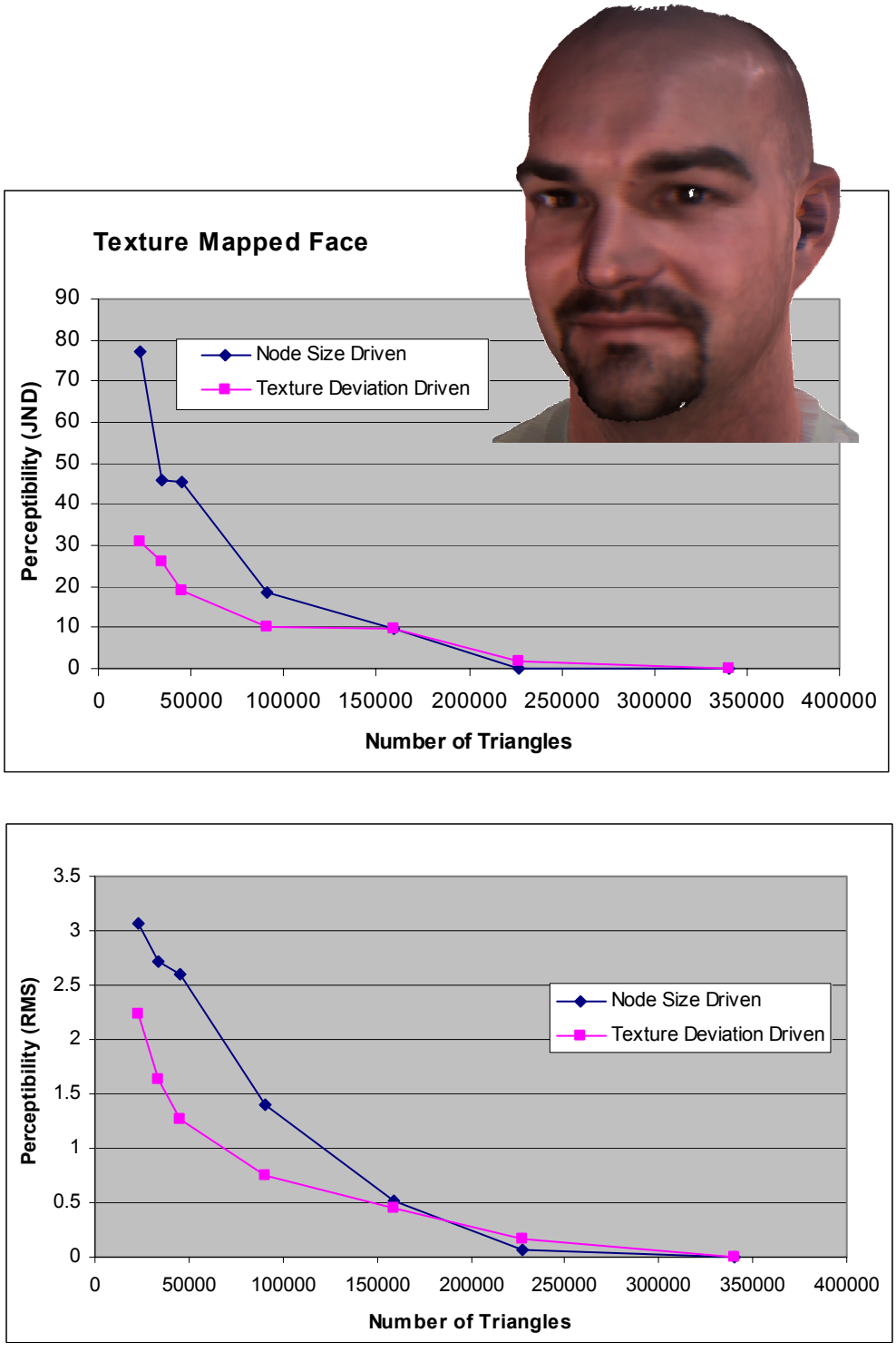


Figure 5: Comparison of Luebke-Hallen approach (driven by node radius) and our perceptual simplification approach (driven by texture deviation) on a pre-lit textured model (a laser scan of a human face, inset). Note that the holes (in the left pupil and beard) are artifacts of the data and not our simplification algorithm).