# Comparing Simplification and Image-Based Techniques for 3D Client-Server Rendering Systems

W. Pasman and F.W. Jansen, *Member*, *IEEE*

**Abstract**—A mathematical model is presented for comparing geometric and image-based simplification methods. Geometric simplification reduces the number of polygons in the virtual object and image-based simplification replaces the object with an image. Our model integrates and extrapolates existing accuracy estimates, enabling the comparison of different simplification methods in order to choose the most efficient method in a given situation. The model compares data transfer and rendering load of the methods. Byte size and expected lifetime of simplifications are calculated as a function of the desired visual quality and the position and movement of the viewer. An example result is that, in typical viewing and rendering conditions and for objects with a radius in the order of one meter, imposter techniques can be used at viewing distances above 15 meters. Below that, simplified polygon objects are required and, below one meter distance, the full-resolution virtual object has to be rendered. An electronic version of the model is available on the web.

**Index Terms**—Real-time rendering, dynamic geometry simplification, imposters, resource load, thin client, mathematical model.

## 1 Introduction

Various geometric and image-based simplification techniques, such as simple imposters [65], [42], meshed imposters [69], [16], and simplified polygon models [66] have been developed to adapt the complexity of a scene to the available bandwidth and capacity of the rendering engine and network. These simplification and imposter techniques often preserve, for a given viewpoint, crucial aspects of the objects, such as contour and front image, while sacrificing geometric accuracy for other viewpoints. This introduces the notion of lifetime for the simplified representation: As the viewpoint changes, the visual distortion will grow and will run the simplification obsolete. The representation then will have to be refreshed or replaced with another form of geometric simplification.

We assume that the rendering and simplification are separated and can be characterized as a server-client architecture, where the server holds the complete scene model database and supplies the rendering client with a scene description of reduced size with the appropriate level of detail. This simplified scene can be rendered in real time while making as few sacrifices to the image quality as possible. A similar client-server setup can be found in web-based systems where the data transfer capacity and latency of the communication link, as well as the temporal validity of parts of the model, impose additional constraints on the model complexity [8], [32]. Our focus is on thin clients, such as mobile phones and mobile augmented reality systems [72], where it is essential to manage the CPU, memory, and communication load on the client. Nevertheless, our model also applies to fat-client setups. We will restrict the object representations to single-resolution meshes with multi-resolution textures in order to avoid excessive complexity in the modeling.

Intuitively, simple imposters are cheapest to transmit and render and remain correct relatively long if the object is far away. But, for nearby objects, the viewpoint changes relatively rapidly and quickly outdates the imposter. In that case, a simplified polygon object is preferable. For objects at moderate distance, the meshed imposter seems to be the most appropriate. See Fig. 1. Shade et al. [68] suggests a similar use of various simplification methods.

Our purpose is 1) to compare the different simplification methods and 2) to select the appropriate representations dynamically given limited resources. For this, we present a model for these simplification methods that relates the load on the communication link and the load on the rendering engine to the visual accuracy.

We start with a discussion of previous work. In Section 3, we give an overview of our approach and list the assumptions we made. In Section 4, we derive distortion formulas for the various simplification methods. These formulas are inverted in Section 5 to estimate the required number of polygons and the size of the textures given a desired accuracy. These sizes can then be converted into a byte size. The communication load is then estimated by dividing the estimated byte size by the planned lifetime of the virtual objects. In Section 6, results are plotted for the various simplification methods to compare the load on the communication link and on the rendering engine. A partial validation of the model is given in Section 7. Conclusions are summarized in Section 8.

• *The authors are with Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands.*
*E-mail: W.Pasman@twi.tudelft.nl, F.W.Jansen@cs.tudelft.nl.*

*Manuscript received 11 Dec. 2000; revised 11 Feb. 2002; accepted 18 May 2002.*
*For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 113279.*

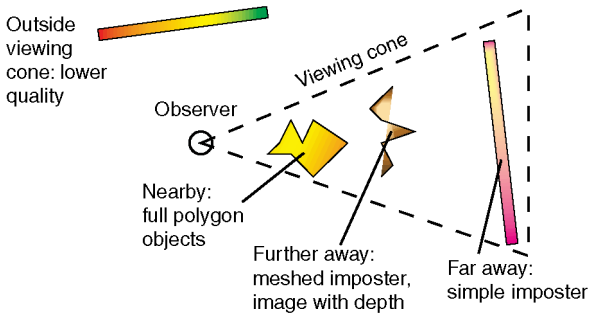


Fig. 1. Various simplification methods are most effective at different distances.

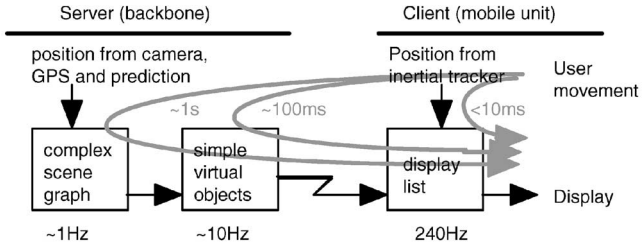


Fig. 2. Latency layered rendering system.

## 2 Previous Work

### 2.1 Latency in Thin-Client Rendering Architectures

This section reviews latency issues with respect to thin-client architectures. Latency is the time lag between the moment the user changes his viewpoint and the moment the images for his new viewpoint are actually presented to him. Latency is a problem with many virtual reality (VR) and augmented reality (AR) applications as it will cause incorrect placement and floating of virtual objects, and may even cause simulator sickness. For VR, latencies of 60 to 150 ms are very noticeable [1] and, for car driving, 30 ms seems the maximum [51]. Prediction has been used to alleviate these problems [5]. However, such predictions give acceptable results only when the predicted time is very small and problems become worse as headsets become lighter and the user can move around less encumbered. With AR, the latency requirements are even tighter as displacements between real and virtual objects can be observed directly. The definitive answer to resolving latency problems seems to be to adjust the rendering pipeline architecture to allow low latency rendering [49], [54], [60].

Regan and Pose [60] inserted five frame buffers instead of the usual single frame buffer between the rendering engine and display. Each of the five buffers has its own rendering rate, ranging from 3.75 to 60 Hz, and the virtual objects are assigned to the appropriate buffer according to their refresh requirements. The buffers contain an image which completely surrounds the viewer, enabling quick generation of images for new viewing directions. The currently visible part from each of the buffers is merged in real time during display scan-out, giving latencies in the order of a few microseconds only. Although a very clean solution, this approach heavily increases the amount of display memory required and the number of pixels to be drawn into the buffers and the low latency is reached only when the viewpoint rotates, but not when it translates as well. Furthermore, this approach still requires the client to have the full, unsimplified virtual objects available for rendering, posing high loads on the network connection.

An extreme form of thin-client rendering architecture is a mobile AR system, where the user is wearing a see-through display that receives its information over a wireless link [72], [56]. Here, the constraints are extremely severe: AR requires a latency of less than 10 ms between a change of viewpoint and the refresh of the display [54], [6], [51]. Moreover, the mobile link has a limited bandwidth (2-10 Mbit/s), which may fail now and then, and introduces by itself a latency of approx. 100 ms. Finally, the client should be lightweight and low-power, which limits the possibilities for local storage and processing.

In a distributed rendering system, a latency requirement of less than 10 ms can only be met with a latency-layered structure. A first coarse approximation is rendered instanteously in the client which is then successively refined with additional information from the backbone. In [35], an approximate image is rendered using a coarse polygon model and its appearance is then improved by sending additional texture information from the backbone. A similar approach is presented in [43], where a first approximation is derived by warping the old image for the new viewpoint. The image is then corrected with an incremental image update sent by the backbone. Although these methods mask the visual effects of the network latency, the extra rendering effort in the client increases the overall latency within the client itself.

Fig. 2 shows the structure that we chose for our mobile AR system [72]. This structure aims to provide the client with the appropriate simplifications in time, without sacrificing image quality. The server holds the model database and compiles the scene graph every second into a viewpoint dependent representation. With a new viewpoint estimate based on vision tracking, obtained a few times per second, the simplified scene graph is further adapted and compiled into a display list. This display list and the associated imposter textures are then transferred to the mobile unit and repeatedly rerendered for new viewpoint estimates based on inertial tracking.

To keep the latency below 10 ms, the client generates image parts just ahead of the display's raster beam. The display is subdivided into four horizontal slices and the image for each slice is rendered with a new viewpoint (every 4 ms). In this way, we can use conventional polygon rendering hardware to render approximately 350 texture-mapped polygons with a maximum latency of 10 ms [54], [55].

### 2.2 Choices Concerning Scene Simplification

Various methods have been proposed to simplify objects in a virtual scene (Fig. 3). The simple imposter [65], [3], [42], [19], [67], also called billboard in VRML [73], replaces the object or a group of objects with a single image. The meshed imposter [69], [16], [15], [61], sometimes called textured depth mesh [2], also consists of a single image, but now the image is mapped onto a depth mesh that roughly approximates one side of the object. The simplified polygon

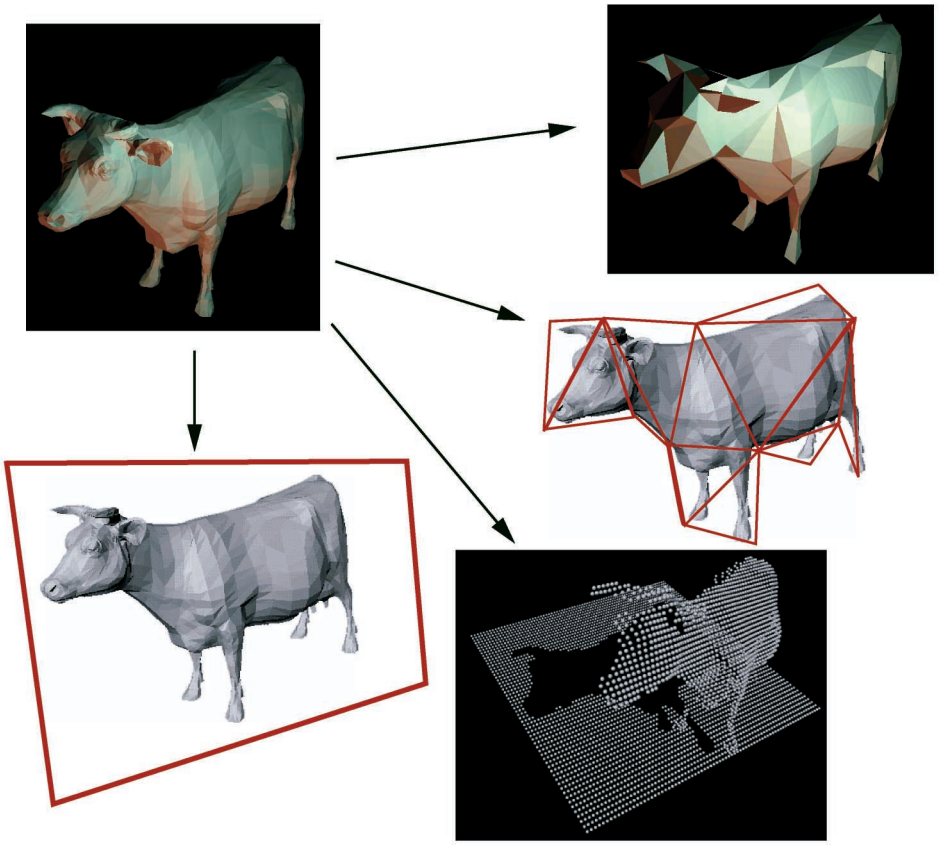

Fig. 3. A number of scene simplification methods. From top left, in clockwise order: the original object, simplified polygon object, meshed imposter, IwD, and simple imposter.

object is a full 3D approximation of the original 3D object, but with fewer polygons. The image with depth (**IwD**) [48], [43], [44] is a single image where each image pixel has a separate depth value. Finally, the layered depth image (**LDI**) [68], [44] is an extension on the IwD, where each pixel stores multiple color/depth pairs.

Often, the image is placed on a billboard rotating only around the vertical axis in order to keep it facing as well as possible toward the observer [73]. We use a more general simple imposter that can rotate about two axes, which is perfectly aimed at the observer at each refresh, but does not rotate between refreshes [65].

Other simplification mechanisms exist, but they are less relevant for thin-client rendering systems. We do not consider light fields and volume models because of their heavy resource usage. We further exclude viewport-remapping [60], image morphing [77], [34], and image inter and extrapolation [13], [45] because we assume 3D rendering capabilities to be available in the client and rerendering a meshed imposter from a new viewpoint is of the same complexity as warping one viewport image (see our discussion in Section 6.2), while warping is less accurate.

## 2.3 Types of Error Metrics

To estimate the rendering and communication load given a certain target quality, we need an error metric that rates the quality of simplified objects. The metric will be used to determine which simplification method is best in which situation, to control the simplification (pre)process, and to use the optimal level of detail given the size and importance of the object and the varying distance to the viewer. Most metrics available are designed to optimize the resulting images for human perception. Two types can be distinguished: image-based and geometry-based error metrics.

### 2.3.1 *Image-Based Metrics*

Image-based analyses have mainly been developed in the last 10 years for rendering systems aiming at maximally realistic images. The importance of various image parts is estimated from their contrast, color, and other properties [31], [7] and this importance is used to allocate resources. Resources can be accurately targeted by estimating the reduced contrast sensitivity of the human system for objects in motion [79], high background illumination levels, high spatial frequencies, and high contrast levels [59].

Unfortunately, these advanced image-based analyses developed for ray tracing can not yet be used for real-time rendering. First, these techniques are still too CPU intensive to apply in real time, usually it takes seconds to minutes for analysis of the scene alone (e.g., [38]) and a lot of further processing is required to get a single image or a general simplification of the object. Furthermore, in real-time interaction, the user will probably put more attention to objects he is working with and is less guided by the spectral properties of the objects in his surroundings. Finally, real-time rendering requires careful deployment of the available resources in order to have an acceptable image ready within the constraints set by the system and the user and strict

resource scheduling is usually not an issue with ray-tracing techniques.

There have been a few attempts to use image-based metrics for real-time rendering. Horvitz and Lengyel [31] proposed using a model of perceptual degradation caused by rendering simple imposters that are affinely warped for the final rendering (sprites) with fewer resources and a model predicting on which sprite the attention of the user is. However, in the prototype implementation [71], every aspect of perceptual-based rendering is absent. Lindstrom and Turk [38] estimated the impact of simplification by comparing the simplified polygon object and the original object rendered from multiple sides. Luebke and Hallen [39] presented a model based on geometric estimations and did a more thorough analysis estimating changes in the image with respect to contrast and spatial frequencies. However, this approach ignores textures. The most promising step toward using image-based analysis for real-time rendering seems to analyze textures in a preprocessing step and to estimate their effect in the final rendering [18].

#### 2.3.2 Geometry-Based Metrics

The most basic geometric distortion measure is the Euclidean distance between the simplified and the original surface. Usually, this metric is expressed relative to the object's size; we refer to this as **relative distortion**. Several variants exist, using the mean or maximum distance and Hausdorff, Fretchet, or Minkowski distance [74]. But, we are not aware of results indicating which one is best for predicting task performance.

Estimating the worst-case screen space errors (**visual distortion**) from the geometry of the objects [28], [36], [40], [78] is more optimized for human perception and more appropriate for measuring quality in a complete scene. Cohen et al. [14] also estimate texture error on the basis of geometry.

Considering a moving viewpoint and a limited time and bandwidth to adapt the geometric representation and to refresh the image caches and imposters, it is crucial to have a formula for the life span of the different objects in the scene. Shade et al. [67] calculate a spherical region around the current viewing position which guarantees a minimal angular discrepancy between the real and imposter position of an object. With an estimated trajectory for the moving viewing position, a lower bound on the number of frames is calculated for which the cache will remain valid. In [10], a similar life span is estimated, but progressive meshes are used to allow a more flexible caching of the geometry. In both approaches, however, no tradeoff between imposters, IwD, LDI, and simplified polygon object is made.

With a number of polygon simplification methods, the amount of simplification attainable is limited for objects containing many separate parts and holes. Even when objects are clustered and holes filled, the practical gain with polygon simplification is moderate. For instance, Garland and Heckbert [24] show, for a human foot model containing many holes, that the accuracy reachable with a certain polygon quotum can only be doubled, in optimal cases tripled, when using clustering as compared to normal polygon simplification.

Imposter methods seem more efficient with respect to grouping issues. Usually, a separation is made between the nearby geometry, represented as full or simplified polygon models, and far geometry, which is represented by simple or meshed imposters. However, current approaches have focused on only one type of imposters and do not provide an explicit error measure and cost function to switch between the different representations. For simple imposters, Schaufler and Stürzlinger [65] estimate the worst-case visual distortion from the object's bounding box, viewing parameters and observer's change in viewing position. Other derivations for simple imposters [67], [19], [42] are similar. None of these estimations determine the optimal position to place the imposter, but assume the imposter is placed at the center of the object. For meshed imposters, Decoret et al. [16] derive distortions due to overlap of imposters in order to find an optimal assignment of objects to imposters. They adopt Schaufler and Stürzlinger's formulas, but it seems that this will result in far too pessimistic error estimations.

### 2.4 Error Metrics Compared

There are a number of image distortions that are hard to capture with geometry-based metrics, although they can be visually disturbing.

An important point is the effect of polygon simplification on attributes attached to vertices and faces: These have to be interpolated appropriately when the vertices and faces are shifted around or removed [25], [30]. Error measurements by Hoppe on color attributes suggest that the attribute distortions behave in a similar way with respect to the number of polygons as the geometry distortions. Garland and Heckbert [25] indicate that, theoretically, the distortions for texture attributes are expected to be lower than color attributes because minimizing texture distortion requires only an additional u and v parameter to be minimized in error, while, for color attributes r, g, b, and, optionally, transparency, parameters have to be minimized. The approach of Cohen et al. [14] guarantees a maximum visual error derived from geometric arguments and seems the best answer to this problem.

For imposters, geometry-based metrics fail to capture the effect of visibility gaps [43], [16] and rubber-sheet distortion [57], [45]. There are a number of techniques to alleviate such gap problems: interpolating pixels to fill the gap [45], [44], [43], warping and overlaying multiple meshed imposters or images with depth [16], [47], [57], calculating the missing pixels in the server [43] or using layered depth images instead of images with depth. But, again, their impact on task performance is unknown. For our model, we will ignore these problems.

Watson et al. [75], [76] did comparisons of both metrics with respect to task performance. They measured the time it takes humans to name a simplified object, human preferences, and similarity ratings with the original object. They found that, for drastic simplifications [75], both the metric by Bolin and Meyer [7], MSE, and maximum 3D distance are a reasonable predictor for the naming time. A later study with moderate simplifications [76] showed that Metro's mean, max, and mean square error measurements [11] are good at predicting the similarity rating and

preference. However, all these metrics are quite bad at predicting naming times, the best being a correlation of 30 percent reached both with Metro's mean, MSE, and Bolin and Meyer's metric.

Concluding, it is unlikely that image-based error metrics will be the basis of error metrics for real-time rendering in the coming years. Both metrics have only limited value for the actual task performance. We feel that other information available at the geometry and application level, such as task knowledge, user goals, and preferences, could be used to improve task-related predictions. Thus, we will focus on geometric distortions for our model. In the following subsections, geometric error metrics are discussed in more detail.

### 2.5 Error versus Rendering Resources

Many polygon simplification methods try to minimize geometric distortion [66], [62]. Accurate accumulation of the distortion from each simplification step is potentially expensive, but, for instance, Garland and Heckbert [24] made an efficient approximation using quadrics. To model the typical errors with these simplification methods, Funkhouser and Séquin [23] proposed formulas to model the relative distortion and rendering costs as a function of the number of faces, vertices, and pixels for rendering. These are discussed in more detail in Section 4.1.

Progressive meshes [27], [20] are an efficient way to store a series of simplifications of an object with decreasing relative distortion. Combining geometric error measures with progressive meshes allows us to efficiently select the proper geometry for some viewing distance. In order to optimize for viewing direction and field of view, local refinement of meshes is required. For terrain models, various methods were proposed [37], [29], [40], and gains of $100\times$ on the amount of polygons and $10\times$ on the amount of texture were reported [37]. For more arbitrary 3D models, extending the progressive mesh to a progressive mesh tree is essential in order to enable efficient local refinement and this also enables handling of other geometric factors such as surface normals and silhouette, corners and surface discontinuities [28], [78], [40].

The common method to trigger the refresh of imposters is to set a threshold on their geometric distortion [65], [67]. More advanced, for each precomputed LOD, the costs and benefits of rendering can be compared [23], [42], [46]. Typically, this cost/benefit ratio is used to maximize the possible quality within a certain rendering budget, for instance, to keep a certain frame rate. Funkhouser and Séquin [23] showed that, in general, this optimization problem is NP-complete and they propose a greedy algorithm to reach a point at least 50 percent from the optimum. Maciel and Shirley [42] extend this model for scene graphs, but run in problems estimating the optimum. Mason and Blake [46] came with a practical approximation to solve the resource scheduling problem for scene graphs. Aliaga and Lastra [3] used a similar benefit/cost ratio. Although their tradeoff is dynamic, the different simplification levels are fixed during preprocessing and the meshed imposters are preconstructed based on an octree cell subdivision.

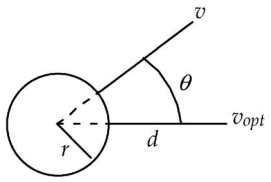


Fig. 4. The circle indicates the object with radius $r$ Optimal viewpoint $v_{opt}$ is at distance $d$ from the object's surface. Current viewpoint $v$ is at angle $\theta$ from $v_{opt}$.

So far, no work is known to us that directly links the visual accuracy to the rendering and communication load and, at the same time, allows us to select the most appropriate simplification method and level. Most proposed techniques require a large set of precomputed object representations. This seems suboptimal for imposters as the number of possible positions will be large compared to the number of actual viewpoints during walkthroughs and, hence, there will be a mismatch between the actual viewpoint and the viewpoints used for rendering the imposters. Instead, our model assumes dynamically updated imposters, which assures that the imposter exactly fits the requirements, which in turn should result in a longer lifetime of the new imposter and a lower communication load.

## 3 Overview and Assumptions

For our model, we will derive a formula that estimates the load on the communication link for the different simplification methods. The formula will be modeled as a function of the radius $r$ of the object, the distance $d$ from the viewer to the object, the maximum acceptable visual distortion $D$, and the available number of polygons $N$.

As discussed, the model will be based on the visual distortion as estimated from the maximum geometric distortion. We chose to use the standard and freely available Metro tool [11] to measure the one-sided Hausdorf distance between the reference and simplified surface. Both the mean and maximum distance will be discussed. Relative distortion will be expressed as a fraction of the object's bounding sphere radius and will be referred to as $D_{rel}$, all other distortions $D$ (with or without subscript) refer to the visual distortion in radians.

The formulas will be constructed in a number of steps:

1. Estimate the distortion of an object. We estimate the visual distortion $D$ of a rendering of an object of radius $r$, which was simplified to $N$ polygons, where the front side of the virtual object is at distance $d$ and the object is viewed at an angle $\theta$ from the optimal viewpoint $v_{opt}$ (Fig. 4). The optimal viewpoint is especially relevant for imposters and images with depth: It is the center of projection from which the imposter is generated. It is called optimal viewpoint because the simplified object will be rendered undistorted when the user stands at that viewpoint.
2. Invert the distortion formulas. Inverse formulas are constructed, to estimate the required number of

polygons $N$ given a maximum visual distortion and the required lifetime of the simplified objects.
3. Estimate texture size. Estimate the size of the textures from the object size and distance, the maximum acceptable visual distortion, and the required lifetime.
4. Calculate communication link load. Convert the number of texture pixels and polygons into a number of bytes and divide the total number of bytes by the lifetime to find the communication load.

The entire model and numerous figures are available for Mathematica and in HTML and can be found on the Internet: http://graphics.tudelft.nl/~wouter/publications/model.

### 3.1 Assumptions

Numerous assumptions are needed to enable comparison of the effects of all parameters on the final image quality. Many assumptions may look somewhat arbitrary or over-simplified. We chose to keep it simple in order to keep overall complexity of the model as low as possible. More future research is required in order to determine which simplifications need refinement and in which cases.

Several estimations, interpolations, and extrapolations will be done, considering partial results from the literature. These will be discussed where introduced. Here, we discuss the more general assumptions.

Distortion caused by latency due to head movements and viewpoint changes will be separated from static geometric scene distortion. Strictly, latency also causes geometric distortion, but realistic latencies introduce such tremendous distortions, even at moderate head movement speeds, that all other distortions become negligible. Even latencies as low as 10 ms [54] may cause visual distortions in the order of degrees with moderate head movements.

Often, the distortions of frontal and side view can be estimated accurately, but it has to be estimated for in-between positions. As it seems reasonable to assume that the distortion will not grow rapidly when close to the front view, we use a sinusoidal function to model this.

For several calculations, the size of the virtual object matters. To simplify matters, we assume a basically spherical object, looked at from the outside. This does not mean that we cannot handle large objects, but it means that large objects have to be split into several smaller objects. We will discuss neither how such splits can be done nor the impact of such splits on performance. Instead, we assume that the accuracy of the rendering of the entire scene is reached if all parts that the scene consists of (or is split into) are rendered with sufficient quality.

For meshed imposters, we will ignore rubber sheet distortions and, for images with depth, we ignore visibility gaps. This will result in slightly optimistic distortion values for such simplifications.

To account for missing back faces of imposters, simple area weighing will be used. Ignoring missing parts seems just as unfair as using 100 percent distortion in case parts are lacking.

Virtual objects are assumed to be triangle-based indexed face sets. This choice was made to allow use of the many existing tools and estimations from literature and because it fits with our low-power approach for wearable augmented reality [72].

For rendering, perfect bitmap caching is assumed, although it is not clear how feasible this is in practice. It might be necessary to insert some constant overhead factor, but this seemed not essential for our model.

In order to model the walking behavior of the observer, it is assumed that he will nicely walk around virtual objects as he would walk around normal objects.

## 4 Distortion Due to Simplification

This section estimates the distortion that the various simplification methods introduce. Formulas are derived for polygon simplification, simple and meshed imposters, images with depth, and layered depth images.

### 4.1 Polygon Simplification

When curves are plotted for the relative distortion against the number of polygons on a log-log scale (Fig. 5), it shows that most objects have a relatively large linear part. Garland's QSlim simplification software [58] was used to simplify the objects and Metro [11] was used to calculate the distortions. These curves usually have a slant of $-45°$, which indicates that the curve behaves roughly as $D_{rel} = k/N$, where $k$ is some constant. A heuristic model in [23] also proposes this formula, but only for flat shaded objects, while suggesting $D_{rel} = k/N^2$ for gouraud shaded objects. However, their formulas are more focused on appearance than on geometric distortion.

This result can be given theoretical support, as follows. Fig. 6 shows a cross-section of a part of the surface and two approximations. The rough approximation has a maximum distortion $e$. The four smaller surfaces of the finer approximation of the same area have a maximum distortion $e'$. The figure shows only a cross-section of the surface, but, on a 2D surface, we now would have four subfaces replacing the large surface.

From the figure, we can see that $s = r\cos 2\beta$, $e = r - s = r - r\cos 2\beta = r(1 - \cos 2\beta)$, and

$$e' = r(1 - \cos\beta).$$

We get, for the distortion ratio,

$$\frac{e'}{e} = \frac{r(1 - \cos\beta)}{r(1 - \cos 2\beta)},$$

which converges quickly to $1/4$ if $\beta \to 0$. Thus, if we quadruple the number of polygons $N$, the distortion $D_{rel}$ gets four times as small. This amounts to the equation $D_{rel} = k/N$. To estimate $k$, Fig. 5 also shows the lines $D_{rel} = k/N$ for a few $k$. When using the mean relative distortion, $k = 1$ seems a good estimate for surface models such as the cow and the bunny, while $k$ increases if the object has many fine details, is built from several smaller objects, or is filled with objects instead of being only a surface, as is shown dramatically by the tree's curve. For the maximum relative distortion, k = 10 is a reasonable estimate for simple models. The maximum distortion curves are more irregular than the mean distortion curves. The maximum distortion curve

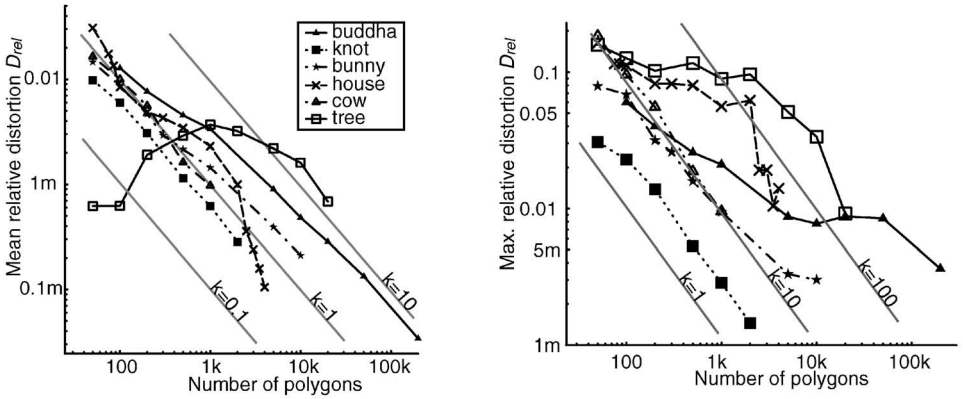


Fig. 5. Mean and maximum relative distortion. Cow from [58]. Tree and House (Maybeck Studio) from [4]. Buddha and Bunny from [70]. Knot is a pipe bent into a three-dimensional lissajour figure.

should be used for the guaranteed-quality mechanism used in the model presented in this paper, but we expect that the mean distortion curves will be better indicators of the global quality of the object and therefore can be used in our model when a minimum quality is required for the average appearance only.

To convert a relative distortion to a visual distortion, we multiply with the angle the full object subtends in the observer's field of view, $2\arctan(r/d)$, which gives, for the visual distortion of simplified polygon objects:

$$D_{polygon\_simpl} = 2k\arctan(r/d)/N.$$

### 4.2 Viewpoint Dependent Polygon Simplification

The model presented here focuses on contour preservation as the human visual system is highly sensitive to distortions at the contours [33], while details in the front face and highlights can also be suggested by adding a texture to the object. Gu et al. [26] even proposed clipping simplified renderings with a smooth contour, to improve the subjective quality impression. Luebke and Erikson [40] mention 1 percent relative distortion when only the back faces of a sphere are simplified, for a total of 3,388 triangles, while 1 percent silhouette error and 20 percent internal error is reached with 1,950 triangles. From this, we estimate that the front view relative distortion is half the distortion of a viewpoint independent simplification with the same amount of polygons, while the side view will have 10 times higher distortion. We assume that the distortion will grow sinusoidally between optimal ($\theta = 0$) and worst ($\theta = 90$) viewpoint: We scale up the function $0.5 - 0.5\cos 2\theta$ to start at 0.5 and to end at 10 and multiply this with $D_{polygon\_simpl}$. And, again, a conversion is required from relative to visual distortion, giving, for the visual distortion of viewpoint dependent simplified polygons,

$$D_{vpt\_dep\_simpl} = \frac{2k\arctan(r/d)(21 - 19\cos 2\theta)}{N},$$

for $-90 \le \theta \le 90$.

### 4.3 Simple imposters

For simple imposters, the distortion depends on the placement of the imposter. The optimal placement and the resulting distortion will be calculated.

To find the maximum visual distortion when rendering simple imposters, we take the point $p_f$ on the front face and a point $p_b$ on the back face of the original object as far as possible from the imposter surface (Fig. 7), as parallax shifts depend on the distance from the projection plane. The imposter is at distance $d_{imp}$ from the viewer, the surface of the original object at distance $d$. For convenience, we set $e_f = d_{imp} - d$ and $e_b = d + 2r - d_{imp}$, where $r$ is the object's radius.

For simplicity, we assume that the center of projection, which is the optimal viewpoint $v_{opt}$, is on a line through $p_f$ and $p_b$ and, therefore, both points are projected to $p'$ on the

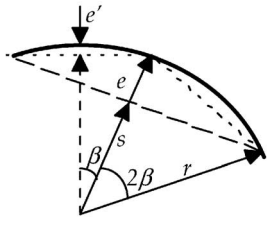


Fig. 6. Cross-section of an object (solid line), an approximatation with one surface (dashed), and an approximation with four surfaces (two stippled lines).

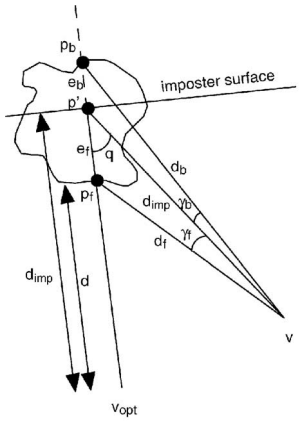


Fig. 7. Visual distortion $\gamma$ caused by movement of viewpoint away from the optimal viewpoint.

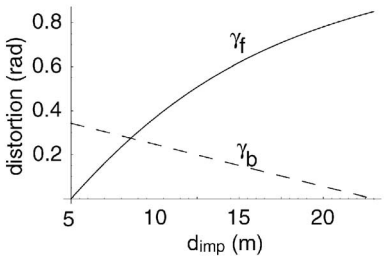


Fig. 8. Visual distortion introduced by simple imposters of a point in front and back of the object.

imposter. Theoretically, in some cases, off-axis projection is an option for creating the imposter image, especially if the original object is essentially flat, but the discussion is out of the scope of this paper. The observer keeps a constant distance $d_{imp}$ from the imposter. $\theta$ is the angle between $v_{opt}$ and the actual viewpoint $v$, relative to the point $p'$. Now, we can derive the visual distortion $\gamma$:

$$\begin{aligned}\gamma_f &= \arcsin(e_f \sin(\theta)/d_f)\\ \gamma_b &= \arcsin(e_b \sin(\theta)/d_b)\\ \gamma &= \max(\gamma_f, \gamma_b).\end{aligned}$$

Fig. 8 shows $\gamma_f$ and $\gamma_b$ as a function of the distance to the imposter plane, for a setting with $d = 5$ m, $\theta = 25°$, and $r = 9$ m. With the imposter at a five meter distance, the object's front side would coincide with the imposter plane and there will be no distortion in the front of the object. At a distance of 23 m, the object's back side will coincide with the imposter plane. The relative distortion is the maximum of the two distortions, which is minimal where $\gamma_f = \gamma_b$. As can be seen, the optimal imposter distance is not in the center of the original object (which would be at 14 m), but closer to the viewpoint. This is also intuitively correct as the back side of the object is less visible and smaller than the front face.

Solving the equation $\gamma_f = \gamma_b$ gives

$$\gamma_{\min}(d, r, \theta) = \textit{ if } \theta = 0 \textit{ then } 0 \textit{ else } \left|\arcsin\left(\frac{\cos\frac{\theta}{2}(r - d + S - 2r\cos\theta)}{\sqrt{2}\sqrt{d^2 + dr + r(r+S) - (d(r+S) + 2r(2r+S))\cos\theta + 2r(d+2r)\cos^2\theta}}\right)\right|,$$

where

$$S = \sqrt{(d+r)^2 - 4r^2\cos\theta + 4r^2\cos^2\theta}.$$

The back of the object is not displayed on the imposter and, because the imposter is flat, even looking from the side results in 100 percent visual distortion or $\gamma = 2\arctan(r/d)$: the angle in the field of view that the original object covers. But, the formula we derived only calculates the maximum distortion of objects depicted on the imposter and does not estimate the distortion of not rendered parts. To account for this, we add the front and back face distortions, weighted with the area they cover on the screen, using a weighing formula

$$D = D_{backface}(1 - \cos 2\theta)/2 + D_{frontface}(1 + \cos 2\theta)/2.$$

We then get for the total visual distortion of simple imposters, assuming optimal placement of the imposter:

$$D_{simple\_imposter} = \frac{1 - \cos 2\theta}{2} 2\arctan\left(\frac{r}{d}\right) + \frac{1 + \cos 2\theta}{2}\gamma_{\min}(d, r, \theta).$$

Imposters with a simple texture map cannot handle changing lighting conditions such as moving lamps. This problem can be alleviated by using bump maps, but then it is probably better to use an IwD to get higher accuracy with the same transmission load.

### 4.4 Meshed Imposters

A meshed imposter can be seen as a polygon object with only the front faces available. Therefore, the distortion in the front faces is comparable with a simplified polygon model with double the number of polygons, but with an additional $\sin\theta$ term because the texture is optimized for the optimal viewpoint: $D_{front} = 2\arctan(r/d)k/(2N)\sin\theta$. Again, we have no backfaces. Area weighing as with the simple imposter gives

$$D_{meshed\_imposter} = ((1 - \cos 2\theta) + (1 + \cos 2\theta)k/(2N)\sin\theta)2\arctan\left(\frac{r}{d}\right).$$

### 4.5 Images with Depth

If the texture is sampled appropriately, layered depth images don't have any distortion in the front faces. As with meshed imposters, there are rubber sheet and visibility gap problems, but we ignore these. Again, the main source of distortion is the missing back face. Thus, we get, for the visual distortion of images with depth:

$$D_{IwD} = (1 - \cos 2\theta)\arctan(r/d).$$

The distortion probably is worse as side faces are represented quite roughly, but it is hard to model this accurately.

### 4.6 Layered Depth Images

As discussed in the previous work section, LDIs can solve visibility gap problems in side views. But, an LDI does not seem suited for rendering backfaces, given a practical average number of pairs per pixel of 1.24 [68] and the need for six LDIs to allow viewing from arbitrary viewpoints [50]. The side faces will still be modeled quite roughly due to limits on the number of color+depth pairs per pixel. So, we estimate the distortion equal to the distortion of images with depth.

## 5 Conversion of the Distortion Formulas

### 5.1 Inversion of the Formulas

Now that we have estimated the distortion as a function of —among others—the number of available polygons, we can invert these formulas to find the number of required polygons given the maximum acceptable visual distortion $D$ (Table 1). The maximum acceptable visual distortion can be determined by the application, user, or scene designer, and it makes sense also to stay below the maximal resolution of the user's display.

TABLE 1
Number of Polygons Required to Reach Visual Distortion Requirement $D$

| Simplification method | Number of polygons required $N$ |
|---|---|
| polygon simplification | $\lceil 2k\arctan(r/d)/D \rceil$ |
| viewpoint dependent simplification | $\lceil (21-19\cos 2\theta)k\arctan(r/d)/(2D) \rceil$ |
| meshed imposter | $\left\lceil \frac{(\sin\theta+\sin 3\theta)k\arctan(r/d)}{2(D+2\arctan(r/d)(-1+\cos 2\theta))} \right\rceil$ |
| simple imposter, LDI, IwD | $N=1$ |

*The minimum $N$ is set to 4 except in the last row.*

### 5.2 Integration of Lifetime

The formulas did not yet account for the minimal lifetime of the simplified object $T$. Assuming an observer walking with speed $v$, the observer can approach the object down to $d - vT$ meter if he started at distance $d$. If the observer can come very close to the object, the distortion will go to infinity, no matter how many polygons we spend on the object. Therefore, we set a minimum distance *MinDist* and, if the observer could come closer, we use the minimum distance as the basis for further distortion requirements.

The angular distance the observer can walk around the object $\theta$ needs more attention. If the observer has time $T$ to walk around, he can walk a distance $W = vT$. If the observer starts at *MinDist*, we find $\theta = W/(r + MinDist)$. If the observer is closer than *MinDist* from the object's surface, we also use this formula, to avoid excessive refreshes. If the observer cannot reach the surface of the object within time $T$, we get $\theta = \arcsin(W/(d+r))$. If the observer can reach *MinDist* within time $T$, we need a combination of these formulas (Fig. 9). We calculate the distance to the side of the object

$$\begin{aligned} d_{side} &= \sqrt{(r+d)^2 - (r+MinDist)^2} \\ &= \sqrt{(d-MinDist)(d+MinDist+2r)} \end{aligned}$$

and we then get for the maximum $\theta$ reachable within time $T$:

$$\theta = \begin{cases} d \le MinDist & \Rightarrow & W/(r+MinDist) \\ W > d_{side} & \Rightarrow & \arcsin(d_{side}/(d+r)) \\ & & +(W-d_{side})/(r+MinDist) \\ W \le d_{side} & \Rightarrow & \arcsin(W/(d+r)). \end{cases}$$

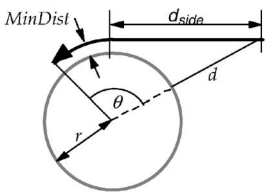


Fig. 9. Maximum angular distance the observer can walk around the object $\theta$. Thick black: the walking path. Thick gray: boundary of the object.

### 5.3 Conversion to Byte Size

To estimate the data size of a conventional polygon model, we assume that each polygon is a triangle with three vertices. A minimum of four vertices per model will be used. If we assume an average indexed faceset, each vertex is reused six times. Each vertex consists of three position coordinates, three normal coordinates, and two texture coordinates. Putting it all together, we get

$$bits = polygons\left(bits\_per\_face + \frac{3 * bits\_per\_vertex}{vertex\_reuse\_rate}\right),$$

where $vertex\_reuse\_rate = 6$. Using current mesh compression techniques, such as difference coding, quantization, and entropy coding [52], [63], [17], we need four to six bits for each of the vertex and texture coordinates. Using a limited set of 4,096 normal directions [17], we need only 12 bits for the normals. This gives a worst-case total of $5 * 6 + 12 = 42$ bits per vertex. For coding the connectivity, Rossignac [63] presented an algorithm compressing to two bits per triangle. Together with Pajarola [52], he presented a version supporting progressive meshes using 3.7 bits per triangle. We conservatively set $bits\_per\_face = 4$ as we will consider incremental transport of the meshes. Plugging in those values gives the formula $bits = 25\ polygons$.

Polygon objects can also be transmitted incrementally. To estimate the byte size of such an incremental format, the difference is used between the byte sizes of the equivalent nonincremental polygon model at the current distance and the worst-case reachable distance.

### 5.4 Required Texture Size

For the estimation of the number of polygons, we assumed a constant polygon size over the entire object, which was determined using the smallest reachable distance $d$. For the textures, we want a more accurate estimation as the textures are so much larger than the polygon data. As long as the observer stays far from the object, we can simply divide the angular size of the object by the maximum visual distortion to get the number of pixels to be spent in one dimension on the object and, squaring this value, we get the number of pixels required for the front face of the object:

$$FarTextureSize(d) = \left(\frac{2\arctan(r/d)}{D}\right)^2.$$

Again, if the observer could reach the surface of the object, we would need an infinitely high resolution and we use *MinDist* instead.

If the observer comes close to the minimum distance and walks around the object, we want to refresh only parts of the object and not the entire object. For instance, if he is walking at a speed of 1 m/s in front of a 20 m wide building and the lifetime of the simplified version of the virtual building was planned to be three seconds, we may want a texture with only three meters of high resolution instead of the full 20 meters. As an approximation to this behavior, we use the FarTextureSize equation only if the observer cannot reach the minimum distance. If the lifetime $T$ is enough for the observer to reach the minimum distance *MinDist*, we prepare for the worst case where the observer would go as quickly as possible to the object and then scan its surface from this closest distance. Because we can't predict in which direction he will scan the surface, we have to prepare a texture area following from the distance the observer can walk along the surface of the object: $d_{near} = W - (d - MinDist)$. Actually, we have to prepare more because the observer can see more of the object than only the pixel in front of him. To avoid difficult mathematics, we make some approximations.

When the observer can walk a distance $d_{near}$ along the object's surface, he can scan $d_{near}/(r + MinDist)$ radians of the surface, corresponding to an area

$$\begin{aligned} scanarea(d_{near}) &= \int_0^{d_{near}/(r+MinDist)} 2\pi r^2 \sin\theta d\theta \\ &= 2\pi r^2(1 - \cos(d_{near}/(r + MinDist))). \end{aligned}$$

Of course, the area is at most the full sphere, $4\pi r^2$. To find the number of pixels corresponding to this area, we have to divide this area by the area of a single pixel at the smallest viewing distance, which is $(MinDist \tan D)^2$, giving

$$scanpixels(d_{near}) = scanarea(d_{near})/(MinDist\ \tan D)^2.$$

But, the observer can see more than the pixels straight under him. To account for this, we use a slightly higher $d_{near}$ as if the observer had more time to scan the surface: $vispixels(d_{near}) = scanpixels(d_{near} + d_0, r)$. The extra distance $d_0$ is chosen such that the equation connects smoothly to the FarTextureSize equation at $d_{near} = 0$. Solving $vispixels(0) = FarTextureSize(MinDist)$ gives

$$d_0(r) = (r + MinDist)$$
$$\arccos\left(1 - \frac{2}{\pi}\left(\frac{Mindist\ \arctan(r/MinDist)\tan D)}{r\ D}\right)^2\right).$$

Combining all this and plugging in the lifetime $T$ and observer speed $v$, we get

$$TextureSize(r, v, T) =$$
$$\begin{cases} vT > d - MinDist: & vispixels(vT - (d - < MinDist)) \\ else: & FarTextureSize(d - vT). \end{cases}$$

This is both including front and back face and is in pixels. Fig. 10 illustrates the texture size as a function of the life time and the distance to the object. The number of pixels still has to be multiplied with the number of bytes per pixel, which varies from three (RGB), four (RGBA), to five (RGBA+Depth). The IwD uses RGBAD. For the LDI, we use 1.24 RGBAD+1 as 1.24 is a practical depth of an LDI [68] and we need one extra byte per pixel to indicate the number of layers per pixel.

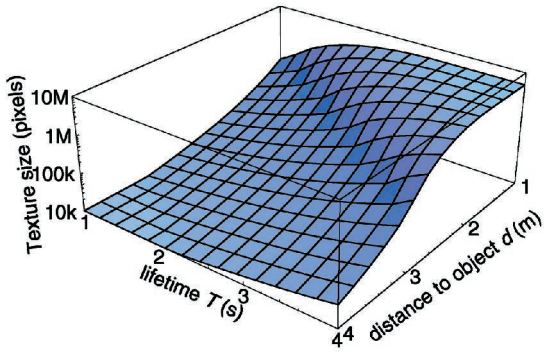


Fig. 10. Uncompressed texture size as a function of the lifetime $T$ and the distance to the object $d$. Distortion $D = 0.004$ rad, object size $r = 0.5$ m, $v = 0.7$ m/s, and $MinDist = 0.2$ m.

Especially for imposter methods that need frequent refresh, there will be a strong temporal coherence between the textures. Therefore, MPEG-like video compression methods [41], [9] can be exploited effectively to compress the textures. Current MPEG compression can reach compression factors up to 75:1. For the other methods—IwD, LDI, and polygon simplification methods (both incremental and nonincremental)—refresh rates will be typically quite low and jumpy. We expect a JPEG variant will be more effective here, with typical compression ratios of 20:1 [12], [61].

### 5.5 Communication Link Load

To find the average communication link load, we divide the model size as found in the previous sections by the lifetime $T$, incorporating the probability that a model refresh is really required. The user can move in six directions (three degrees of freedom and two directions on each axis), but only a few of these directions actually require updating of the polygon model. We have to refresh a polygon model (either normal or incrementally coded) only if the observer gets closer to the model, a probability of 1/6. All other models, including the viewpoint dependent polygon simplification, have to be refreshed also when the observer moves around the object, a probability of 5/6.

## 6 Results

In this section, results on the communication link load and CPU and memory usage are plotted and discussed.

### 6.1 Results on the Communication Link Load

Fig. 11 shows the results for an object size $r = 1$ m and a maximal acceptable visual distortion $D = 1$ mrad, which is approximately the size of a pixel on a $640 \times 480$ display with a field of view of $40° \times 35°$. *MinDist* is 0.2 m, the observer speed $v = 1$ m/s and the object complexity $k = 5$ in all examples in this section. Incremental transport of the full polygon objects gives on average the lowest transport costs.

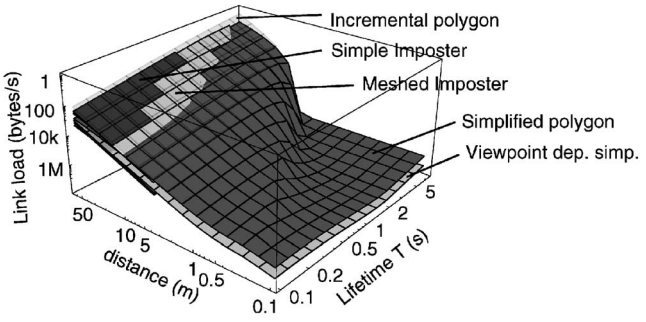


Fig. 11. Average load on the communication link. Maximum distortion $D = 0.001$ and object size $r = 1$ m. Incrementally transported simplified polygon objects are shown transparent to expose underlying layers.

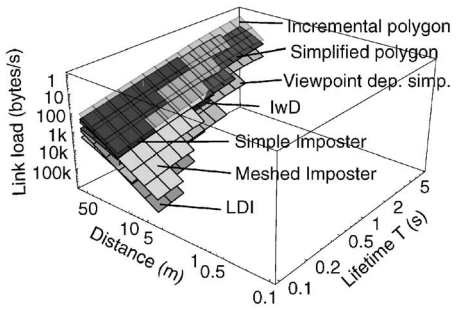


Fig. 12. As Fig. 11, but now the number of polygons $N$ is limited to 500.

Thus, our model predicts that the added costs for the larger mesh complexity are compensated by the lower refresh costs.

Without such incremental transport, simplified polygons perform worse, in which case imposters can help save transport costs. The imposter models break down at some distance because the distortion due to the missing back face gets too large and, for simple imposters, also because of their flatness. The ridge in the polygon distortion model is caused by the possibility of the observer reaching the object, making it necessary to transfer the texture of the complete 3D model. If we aim at a lifetime of one second, this particular setting shows that a polygon model is necessary for distances below 15 meters to reach the required visual distortion. Simple imposters can be only at large distances, 50 meter and further.

For larger models, the imposters are less effective. If we increase the object radius to 10 m, the meshed imposter, IwD, and LDI can be used at distances higher than 30 m; simple imposters give too high distortions up to at least 100 m. With very short lifetimes, a meshed imposter can be used at small distances because the observer's speed around a large object is relatively smaller than around a big object, but the gain compared to a simplified polygon model is small.

If this can be considered a typical situation, the communication load is not a real argument to use imposters at large distances. However, the number of polygons in imposters is usually much smaller than polygon objects with comparable distortion and, in case of tight polygon budgets, it is attractive to switch to simpler models as soon as possible.

Fig. 12 shows what happens if we use at most 500 polygons for the simplified model. The incremental polygon and simplified polygon methods now break down at distances smaller than 30 m because the distortion requirement cannot be reached with the given amount of polygons. The viewpoint dependent simplification improves this range to down to 10 meters. In such a situation, the imposter and LDI methods offer an advantage.

Clearly, polygon-limited cases introduce situations where certain distortion targets cannot be reached. If a rendering is still required, quality has to be reduced. By accepting higher visual distortions ($D = 0.1$), but still requiring high texture quality (with $D = 0.001$ as before), the simplified polygon models could still be used at smaller distances (Fig. 13), while the textures at least suggest low distortions. Of course, when lower geometry distortion is acceptable, the imposters can also be used at smaller distances.

## 6.2 CPU and Memory Load

We did a similar analysis of CPU and memory load, using a complexity analysis of currently available algorithms. But, we can summarize the results without such an analysis:

1. For the imposter and polygon models, the memory required in the client is dominated by the texture size and, thus, follows directly from the estimated texture sizes.
2. The textures should be at a resolution closely matching the final projected size on the display if we want to have a close to 1 pixel visual distortion. Then, the load on the texture memory (number of bytes accessed per second) is determined by the area covered by the virtual objects. Assuming perfect caching, each pixel is read only once from texture memory and, therefore, the memory load for the various methods is different only because of the varying number of bytes for RGB, RGBA, RGBAD, and layered RGBAD pixels. For images with depth, a hybrid, 2-stage rendering method has been developed [68], [64], but this requires storage of intermediate images and therefore gives a higher memory load.
3. To compare the CPU load involved in the rendering, it suffices to realize that the main bulk of calculations

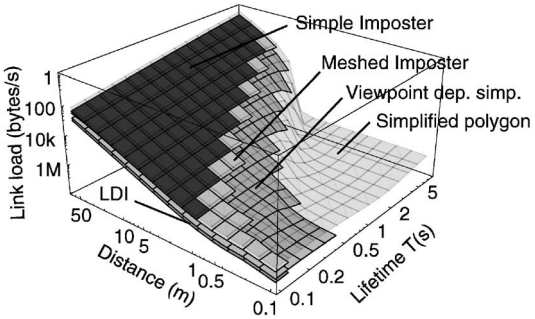


Fig. 13. As Fig. 12, with lower geometric quality $D = 0.1$, but with the same texture quality. The incremental polygon curve was removed and the simplified polygon curve was made transparent to shown the other curves.

involves the per-pixel perspective calculation. This is required both for polygon rendering, imposters, and for depth-per-pixel texture maps such as the image with depth. Therefore, the rendering loads are comparable. Some tricks can be used to approximate perspective mapping with cheaper functions, but these tricks can be used for all methods we discussed.

## 7 Validation

Now we have described a model, how good is it? First, we would like to repeat that our model is the first, possibly oversimplified, attempt to model the current simplification techniques. For instance, we already saw that the formula $D = k/N$ does not appropriately model nonsurface models like a tree or a house containing furniture. Also, the ratio of silhouette to interior polygons that we used was estimated very roughly, using only results from the literature on a sphere. It remains to be seen whether our approximations are sufficient and more elaborate models have to be developed to come to a more accurate comparison.

For our validation, we used a prototype system [53] that uses the model to schedule the available resources in the client to the various virtual objects in the scene. In this system, many problems have been only partially resolved, such as latency variations, caching problems, texture memory limitations, absence of geometry compression, and lack of texture support for simplified polygon objects. Furthermore, there are other issues, such as the resource scheduling, also taking part of the resources. In all, it probably is dangerous to compare performance results of the prototype system with our model. It is likely that the theoretical model will give a more balanced estimate than measurements on a prototype system as it can be checked that the same criteria are rigorously applied to all available methods in a comparable way, which is hard to see in thousands of lines of code.

However, a part of the question about validation is about how good the resulting images look on the screen and whether similar settings with different methods do give similar-looking images. To answer this, some basic comparisons were done and, below, some snapshots and SNR measurements are presented. Of course, SNR is just as dubious to assess image quality as is geometric distortion, but there is no definitive measurement to assess image quality.

To generate a few images for comparison, simple imposters, meshed imposters, and simplified polygon objects were rendered with an equal distortion target. The cow and house model were used (see Fig. 5) because the cow is very regular and almost perfectly follows the k = 10 line, while the house behaves very irregularly. The cow is relatively flat and an imposter giving a side view will give lower distortions than an imposter giving a front view, when viewed from the same distance from the optimal viewpoint. Both cases are shown in the figures. Fig. 14a shows the result for $D = 0.02$. In the simplified polygon version, many details get lost, such as the horns of the cow and window details. Many apparently fine details are still in the house; this is related to the fact that QSlim does not optimize for geometric distortion. The simple imposters look very good, much better, in fact, than the simplified polygon version, but this is mainly caused by lacking texture on the simplified polygon version. The geometric distortion relevant here is a vertical foreshortening, hardly visible in these static images. A disturbing artifact can be seen in the roof of the house in the meshed imposter. This is caused by the grid of the underlying meshed imposter, which becomes somewhat jumpy at the edge of the house. There is a light stripe attached to the horn of the cow. This is because of our relatively naive way to generate the vertices for meshed imposters, by just sampling a few pixels instead of a full check of the depths of all pixels. Therefore, in unfortunate cases, a mesh vertex may be set at the far clip distance, while there are much closer pixels in one of the faces containing this vertex. In such a case, part of the mesh is stretched out toward the far clipping plane. In Fig. 14b, $D = 0.001$. With such a low distortion target, distortions are essentially invisible.

To make a further comparison of the image quality using geometric distortion, we compared the signal to noise ratio (SNR) with the geometric distortion (Fig. 15). Here, we use the mean geometric distortion, again determined with Metro [11], instead of the maximum distortion, as the SNR gives a kind of average over the entire image. Note that a log-scale is used to plot the geometric distortion, again matching the log-nature of the SNR. Judging from this figure, the mean geometric distortion has a clear relation to the SNR.

## 8 Conclusions

The intuitive model was affirmed by a theoretical analysis of the load on the communication link. A first experimental validation of our model also looks promising. Although we focused on thin-client systems, our model also applies to fat-client setups. But, for fat-client setups, additional analysis may be necessary to estimate the load on the server as well.

If there is no polygon budget in the client and the communication load is the main factor for considering simplification, our model suggests that a combination of incremental transmission and compression will offer the lowest communication load. In the presence of a tight polygon budget, a possible mechanism suggested by our analysis is to refresh the models at a rate of one hertz, to use polygon models at distances closer than 15 meters, meshed imposters between 15 and 50 meters, and simple imposters at larger distances. Exact switching distances depend on the size of the virtual object and will, in practice, also be related to the polygon budget.

In practice, the throughput of the communication link can vary drastically, especially if a mobile link is used as in our mobile augmented reality system [72]. The model assumes that the separate objects are all represented by an appropriate simplification and we estimated that the renderings will look acceptable even in case of a temporal breakdown of the communication link. However, it is not clear how to incorporate varying link performance in the model and more research is required to estimate how important this is and how to model it.

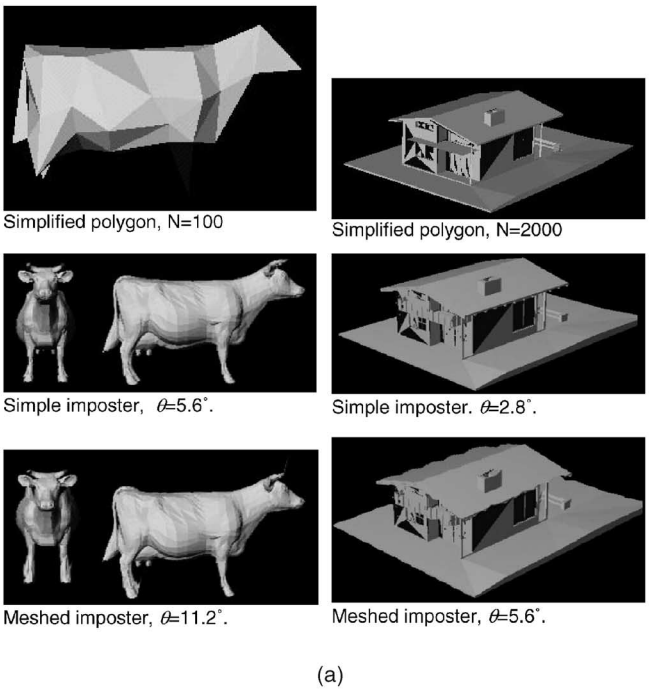


Fig. 14. (a) Comparison of various simplification methods targeting at $D = 0.02$ (10 pixels at the resolution of $640 \times 480$, vertical fov $22.6°$). $d = 5,000$, $r = 570$ (cow) and $984$ (house). (b) Now, $D = 0.001$ (1 pixel at rendered resolution).

The model as presented here involves a geometric distortion measure. It remains to be seen whether a geometric distortion measure is a useful measure in practice, for instance, to estimate task performance. However, comparison of the geometric distortion measure with an SNR measure showed no reason to worry about this distinction. More accurate measures are probably highly dependent on the specific task at hand and cannot be solved

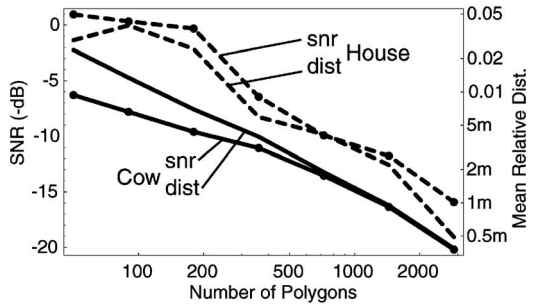


Fig. 15. Comparison of mean relative distortion and SNR. Vertical placement of SNR and dist curves is arbitrary, chosen such that right sides of the cow curves match.

at the general level of our model. Instead, we suggest propagating these issues to the application layer [53].

Our more recent work focused on dynamic simplification of a virtual scene and resource allocation, using the model derived in this paper. Results on this have already been published [53].

The large textures, required as the observer comes close to the object, will pose storage, rendering, and transport problems. Splitting large textures into several smaller ones is not trivial as this may cause cracks on the polygon boundaries because of the texture discontinuity [22]. Systems exist to deal with large textures, for instance, SGI's clip maps [21], but it is not clear whether it is realistic to use those on a thin client and what happens with texture coordinates if the mesh needs to be refreshed.

## ACKNOWLEDGMENTS

This work was funded by the UbiCom interdisciplinary research initiative (DIOC) at Delft University of Technology. The authors would like to thank the anonymous reviewers for their detailed comments on this paper.

## REFERENCES

[1] D.G. Aliaga, "Virtual Objects in the Real World," *Comm. ACM,* vol. 40, no. 3, pp. 49-54, Mar. 1997.
[2] D. Aliaga et al., "MMR: An Integrated Massive Model Rendering System Using Geometric and Image-Based Acceleration," *Proc. Symp. Interactive 3D Graphics (I3D),* pp. 199-206, Apr. 1999, http://www.cs.unc.edu/~aliaga/publications.html, May 2002.
[3] D.G. Aliaga and A.A. Lastra, "Automatic Image Placement to Provide a Guaranteed Frame Rate," *Proc. SIGGRAPH,* pp. 307-316, 1999, http://www.cs.unc.edu/~aliaga/research.html, May 2002.
[4] *DesignWorkshop,* Artifice Inc., Eugene, Ore., 1999,http://www.artifice.com, May 2002.
[5] R. Azuma and G. Bishop, "A Frequency-Domain Analysis of Head-Motion Prediction," *Proc. SIGGRAPH,* pp. 401-408, Aug. 1995, http://cs.unc.edu/~azuma/azuma_AR.html, May 2002.
[6] R.T. Azuma, "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments,* vol. 6, no. 4, pp. 355-385, 1997, http://www.cs.unc.edu/~azuma/ARpresence.pdf, May 2002.
[7] M.R. Bolin and G.W. Meyer, "A Perceptually Based Adaptive Sampling Algorithm," *Proc. SIGGRAPH,* pp. 299-310, July 1998.
[8] A.T. Campbell, "Technical Review: QoS-Aware Middleware for Mobile Multimedia Communications," *Multimedia Tools and Applications,* vol. 7, nos. 1-2, pp. 67-82, 1998.
[9] *ISO/IEC JTC1/SC29/WG11 N, Short MPEG-2 description,* http://garuda.imag.fr/MPEG4/syssite/syspub/index.html, May 2002.

[10] J. Chim et al., "Multi-Resolution Model Transmission in Distributed Virtual Environments," *Proc. ACM Symp. Virtual Reality Software and Technology,* pp. 25-34, 1998, http://www.cs.cityu.edu.hk/~rynson/pub-cgvr.html, May 2002.

[11] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring Error on Simplified Surfaces" *Computer Graphics Forum,* vol. 17 no. 2, pp. 167-174, June 1998. http://vcg.iei.pi.cnr.it/~rocchini, May 2002.

[12] D. Cline and P.K. Egbert, "Interactive Display of Very Large Textures," *Proc. IEEE Visualization,* pp. 343-350, 1998.

[13] D. Cohen-Or, "Model-Based View-Extrapolation for Interactive VR Web-Systems," *Proc. IEEE Computer Graphics Int'l Conf.,* pp. 104-112, 248, June 1997.

[14] J. Cohen, M. Olano, and D. Manocha, "Appearance-Preserving Simplification," *Proc. SIGGRAPH,* pp. 115-122, 1998.

[15] L. Darsa, B. Costa, and A. Varshney, "Walkthroughs of Complex Environments Using Image-Based Simplification," *Computers and Graphics,* vol. 22, no. 1, pp. 55-69, 1998, http://www.cs.umd.edu/~varshney, May 2002.

[16] X. Decoret, G. Schaufler, F. Sillion, and J. Dorsey, "Multi-Layered Impostors for Accelerated Rendering," *Proc. Eurographics, Computer Graphics Forum,* vol. 18, no. 3, pp. 61-73, 1999, http://graphics.lcs.mit.edu/~gs/research/egmmi, May 2002.

[17] M. Deering, "Geometry Compression," *Proc. SIGGRAPH,* pp. 13-20, 1995.

[18] R. Dumont, F. Pellacini, and J.A. Ferwerda, "A Perceptually-Based Texture Caching Algorithm for Hardware-Based Rendering," *Proc. Eurographics Workshop Rendering,* http://www.graphics.cornell.edu/pubs/2001 and http://www.graphics.cornell.edu/~jaf/publications/publications.html, 2001.

[19] P. Ebbesmeyer, "Textured Virtual Walls: Achieving Interactive Frame Rates during Walkthroughs of Complex Indoor Environments," *Proc. Virtual Reality Ann. Int'l Symp. (VRAIS '98),* pp. 220-227, Mar. 1998.

[20] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," Technical Report #95-01-02, ftp://ftp.cs.washington.edu/tr/1995/01/UW-CSE-95-01-02.d/UW-CSE-95-01-02.PS.Z, 1995.

[21] G. Eckel, "Cliptextures," *IRIS Performer Programmer's Guide,* chapter 10, SGI Technical Publications, Silicon Graphics Inc., Mountain View, Calif., http://techpubs.sgi.com/library, 1995.

[22] G. Francini, "Surface Texture Estimation of 3D Model Objects," Panorama project AC092, http://uranus.ee.auth.gr/~alex/panorama/deliverables.html, 1998.

[23] T.A. Funkhouser and C.H. Séquin, "Adaptive Display Algorithm for Interactive Frame Rates during Visualization of Complex Virtual Environments," *Computer Graphics (SIGGRAPH '93 Proc.),* pp. 247-254, Aug. 1993.

[24] M. Garland and P.S. Heckbert, "Surface Simplification Using Quadric Error Metrics," *Proc. 24th Ann. Conf. Computer Graphics & Interactive Techniques (SIGGRAPH '97),* pp. 209-216, 1997.

[25] M. Garland and P.S. Heckbert, "Simplifying Surfaces with Color and Texture Using Quadric Error Metrics," *Proc. IEEE Visualization '98,* pp. 263-269, 542, http://www.cs.cmu.edu/~garland/quadrics/quadrics.html, 1998.

[26] X. Gu, S. Gortler, H. Hoppe, L. McMillan, B. Brown, and A. Stone, "Silhouette Mapping," Technical Report TR-1-99, Dept. of Computer Science, Harvard Univ., Mar. 1999, http://research.microsoft.com/~hoppe.

[27] H. Hoppe, "Progressive Meshes," *SIGGRAPH '96 Proc.,* pp. 99-108, 1996, http://research.microsoft.com/~hoppe.

[28] H. Hoppe, "View-Dependent Refinement of Progressive Meshes," *Proc. SIGGRAPH '97,* pp. 189-198, 1997.

[29] H. Hoppe, "Smooth View-Dependent Level-of-Detail Control and Its Application to Terrain Rendering," *Proc. IEEE Visualization,* pp. 35-42, 1998.

[30] H. Hoppe, "New Quadric Metric for Simplifying Meshes with Appearance Attributes," *Proc. IEEE Visualization '99,* pp. 59-66, Oct. 1999, http://research.microsoft.com/~hoppe.

[31] E. Horvitz and J. Lengyel, "Perception, Attention, and Resources: A Decision-Theoretic Approach to Graphics Rendering," *Proc. 13th Conf. Uncertainty in Artificial Intelligence (UAI '97),* pp. 238-249, Aug. 1997. ftp.research.microsoft.com/pub/ejh/dtgraph.ps.

[32] J.H. Kim and A.A. Chien, "Rotating Combined Queueing (RCQ): Bandwidth and Latency Guarantees in Low-Cost, High-Performance Networks," *Proc. 23rd Ann. Int'l Symp. Computer Architecture,* pp. 226-236, 1996.

[33] J.J. Koenderink, "What Does the Occluding Contour Tell Us about Solid Shape," *Perception,* vol. 13, pp. 321-330, 1984.

[34] J. Lengyel and J. Snyder, "Rendering with Coherent Layers," *Proc. SIGGRAPH '97,* pp. 233-242, 1997.

[35] M. Levoy, "Polygon-Assisted JPEG and MPEG Compression of Synthetic Images," *Proc. SIGGRAPH,* pp. 21-28, 1995.

[36] P. Lindstrom, D. Koller, L.F. Hodges, W. Ribarsky, N. Faust, and G. Turner, "Level of Detail Management for Real-Time Rendering of Phototextured Terrain," Technical Report GIT-GVU-95-06, Georgia Inst. of Technology, Jan. 1995, http://www.cc.gatech.edu/gvu/people/peter.lindstrom.

[37] P. Lindstrom, D. Koller, W. Ribarsky, L.F. Hodges, N. Faust, and G.A. Turner, "Real-Time, Continuous Level of Detail Rendering of Height Fields," *Proc. SIGGRAPH '96,* pp. 109-118, Aug. 1996.

[38] P. Lindstrom and G. Turk, "Image-Driven simplification," *ACM Trans. Graphics,* vol. 19, no. 3, pp. 204-241, 2000, http://www.gvu.gatech.edu/people/peter.lindstrom/papers.

[39] D. Luebke and B. Hallen, "Perceptually Driven Simplification for Interactive Rendering," *Rendering Techniques,* S. Gortler and K. Myszkowski, eds., London: Springer-Verlag, 2001, http://www.cs.virginia.edu/~luebke/publications.html.

[40] D. Luebke and C. Erikson, "View-Dependent Simplification of Arbitrary Polygonal Environments," *Proc. 24th Ann. Conf. Computer Graphics & Interactive Techniques (SIGGRAPH '97),* pp. 199-208, 1997.

[41] Moving Pictures Expert Group, "ISO/IEC JTC1/SC29 WG11," 2000, http://mpeg.telecomitalialab.com.

[42] P.W.C. Maciel and P. Shirley, "Visual Navigation of Large Environments Using Textured Clusters," *Proc. 1995 Symp. Interactive 3D Graphics,* pp. 95-102, 1995, http://www2.cs.utah.edu/~shirley/papers.

[43] Y. Mann and D. Cohen-Or, "Selective Pixel Transmission for Navigating in Remote Virtual Environments," *Proc. Eurographics '97,* vol. 16, no. 3, pp. C201-C206, 1997, http://www.math.tau.ac.il/~daniel.

[44] W.R. Mark, "Post-Rendering 3D Image Warping: Visibility, Reconstruction, and Performance for Depth-Image Warping," doctoral dissertation, UNC Computer Science Technical Report TR99-022, Univ. of North Carolina, Apr. 1999, http://www.cs.unc.edu/~billmark/research.html.

[45] W.R. Mark, L. McMillan, and G. Bishop, "Post-Rendering 3D Warping," *Proc. 1997 Symp. Interactive 3D Graphics,* pp. 7-16, Apr. 1997, http://www.cs.unc.edu/~billmark.

[46] A.E.W. Mason and E.H. Blake, "Automatic Hierarchical Level of Detail Optimization in Computer Animation," *Computer Graphics Forum (Eurographics '97 Proc.),* vol. 16, no. 3, pp. 191-200, 1997.

[47] L. McMillan and G. Bishop, "Plenoptic Modeling: An Image-Based Rendering System," *Computer Graphics Ann. Conf. Series (SIGGRAPH '95),* pp. 39-46, 1995.

[48] L. McMillan, "An Image-Based Approach to Three-Dimensional Computer Graphics," PhD thesis, Univ. of North Carolina at Chapel Hill, UNC Technical Report TR97-013, 1997, http://graphics.lcs.mit.edu/~mcmillan/Publications/diss.pdf.

[49] M. Olano, J. Cohen, M. Mine, and G. Bishop, "Combatting Rendering Latency," *Proc. 1995 Symp. Interactive 3D Graphics,* pp. 19-24 and 204, Apr. 1995, www.cs.unc.edu/~olano/papers/latency.

[50] M.M. Oliveira and G. Bishop, "Image-Based Objects," *Proc. Symp. Interactive 3D Graphics (SI3D '99),* pp. 191-198, 1999.

[51] P. Padmos and M.V. Milders, "Quality Criteria for Simulator Images: A Lterature Review," *Human Factors,* vol. 34, no. 6, pp. 727-748, 1992.

[52] R. Pajarola and J. Rossignac, "SQUEEZE: Fast and Progressive Decompression of Triangle Meshes," *Proc. Computer Graphics Int'l (CGI 2000),* pp. 173 -182, June 2000.

[53] W. Pasman and F.W. Jansen, "Scheduling Level of Detail with Guaranteed Quality and Cost," *Proc. Web3D Conf.,* pp. 43-51, Feb. 2002, http://www.cg.its.tudelft.nl/~wouter/publications/publ.html.

[54] W. Pasman, A. van der Schaaf, R. Lagendijk, and F.W. Jansen, "Accurate Overlaying for Mobile Augmented Reality," *Computers & Graphics,* vol. 23, no. 6, pp. 875-881, 1999, http://www.cg.its.tudelft.nl/~wouter/publications/publ.html.

[55] W. Pasman, "Low Latency Rendering," http://www.ubicom.tudelft.nl/project/P2/P2.3/P2.3.1.ASP, 1999.

[56] W. Pasman and F.W. Jansen, "Realistic Low-Latency Mobile AR Rendering," *Proc. Int'l Symp Virtual and Augmented Architecture (VAA01),* pp. 81-92, June 2001, www.cg.its.tudelft.nl/~wouter/publications/publ.html.
[57] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle, "View-Based Rendering: Visualizing Real Objects from Scanned Range and Color Data," *Rendering Techniques '97 (Proc. Eighth Eurographics Workshop Rendering),* pp. 23-34, 1997.
[58] M. Garland, "QSlim 2.0 [Computer Software]," Univ. of Illinois at Urbana-Champaign, UIUC Computer Graphics Lab, http://graphics.cs.uiuc.edu/~garland/software/qslim.html, 1999.
[59] M. Ramasubramanian, S.N. Pattanaik, and D.P. Greenberg, "A Perceptually Based Physical Error Metric for Realistic Image Synthesis," *Proc. SIGGRAPH '99,* pp. 73-82, 1999.
[60] M. Regan and R. Pose, "Priority Rendering with a Virtual Address Recalculation Pipeline," *Proc. SIGGRAPH '94, Computer Graphics, Ann. Conf. Series,* pp. 155-162, July 1994.
[61] T. Riegel, "Coding of PANORAMA 3-D Sequences," http://uranus.ee.auth.gr/~alex/panorama, 1998.
[62] J. Rossignac and P. Borrel, "Multi-Resolution 3D Approximations for Rendering Complex Scenes," *Modeling in Computer Graphics: Methods and Applications,* B. Falcidieno and T.L. Kunii, eds., pp. 455-465, Berlin: Springer-Verlag, 1993.
[63] J. Rossignac, "Edgebreaker: Connectivity Compression for Triangle Meshes," *IEEE Trans. Visualization and Computer Graphics,* vol. 5, no. 1, pp. 47-61, Jan.-Mar. 1999.
[64] G. Schaufler and M. Priglinger, "Efficient Displacement Mapping by Image Warping," *Proc. 10th EUROGRAPHICS Workshop Rendering,* pp. 175-186, 2000, http://graphics.lcs.mit.edu/~gs/papers.
[65] G. Schaufler and W. Stürzlinger, "A Three Dimensional Image Cache for Virtual Reality," *EUROGRAPHICS '96 Proc.,* vol. 15, no. 3, pp. 227-236, Aug. 1996, http://www.gup.uni-linz.ac.at/~gs/research/icache.
[66] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen, "Decimation of Triangle Meshes," *Proc. SIGGRAPH '92,* pp. 65-70, 1992, ftp://ftp.cs.cmu.edu/afs/cs/project/anim/ph/paper/multi97/release/schroede/deci.pdf.
[67] J. Shade, D. Lischinski, D.H. Salesin, T. DeRose, and J. Snyder, "Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments," *Computer Graphics Proc. (SIGGRAPH '96),* pp. 75-83, 1996.
[68] J. Shade, S. Gortler, L. He, and R. Szeliski, "Layered Depth Images," *Proc. 25th Ann. Conf. Computer Graphics (SIGGRAPH '98),* pp. 231- 242, 1998.
[69] F.X. Sillion, G. Drettakis, and B. Bodelet, "Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery," *Proc. Eurographics '97,* pp. C207-C218, Sept. 1997, http://www-imagis.imag.fr/Membres/Francois.Sillion/Papers/Index.html.
[70] Stanford 3D Scanning Repository, "Happy Buddha [Computer file],"Dept. of Computer Science, Stanford Univ., 1996, http://www-graphics.stanford.edu/data/3Dscanrep.
[71] J. Torborg and J.T. Kajiya, "Talisman: Commodity Realtime 3D Graphics for the PC," *Computer Graphics Proc., Proc. SIGGRAPH '96,* pp. 353-363, 1996. www.research.microsoft.com/SIGGRAPH96/96/Talisman.
[72] UbiCom, "Ubiquitous Communications: Aiming at a New Generation Systems and Applications for Personal Communication,"interdisciplinary program (DIOC), Delft Univ. of Technology, 2001, http://www.ubicom.tudelft.nl.
[73] Web3D Consortium, "VRML97 International Standard ISO/IEC 14772-1:1997," 2002, http://www.web3d.org/vrml/spec.htm.
[74] R.C. Veltkamp, "Shape Matching: Similarity Measures and Algorithms," 2001, ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-2001/2001-03.pdf.
[75] B. Watson, A. Friedman, and A. McGaffey, "Using Naming Tme to Evaluate Quality Predictors for Model Simplification," *Proc. ACM Computer Human Interaction (CHI '00),* pp. 113-120, 2000.
[76] B.A. Watson, A. Friedman, and A. McGaffey, "Measuring and Predicting Visual Fidelity," *Proc. SIGGRAPH 2001, Computer Graphics Proc., Ann. Conf. Series,* pp. 213-220, Aug. 2001, http://www.cs.nwu.edu/~watsonb/school/publications.html.
[77] G. Wolberg, *Digital Image Warping.* Los Alamitos, Calif.: IEEE CS Press, 1990.
[78] J.C. Xia and A. Varshney, "Dynamic View-Dependent Simplification for Polygonal Models," *Proc. Conf. Visualization '96,* pp. 327-334, 1996, ftp://ftp.cs.sunysb.edu/pub/varshney/papers/av_vd_vis.pdf and http://www.cs.umd.edu/~varshney.
[79] H. Yee, S. Pattanaik, and D.P. Greenberg, "Spatiotemporal Sensitivity and Visual Attention for Efficient Rendering of Dynamic Environments," *ACM Trans. Graphics,* vol. 20, no. 1, pp. 39-65, Jan. 2001.

**Wouter Pasman** received the Master's degree in computer science from the University of Amsterdam, The Netherlands, in 1991. From 1993-1997, he did research on 3D x-ray luggage inspection at Delft University of Technology, for which he received the PhD degree in industrial design engineering in 1997. Between 1997 and 2001, he researched real-time rendering for mobile augmented reality as part of the UbiCom project at Delft University of Technology. Currently, he is doing research in the areas of multimodal interaction and user-context modeling. His research interests include augmented reality, quality of service handling, 3D computer graphics, human factors, and multimodal interaction.

**F.W. Jansen** obtained the PhD degree in 1987 for work on constructive solid geometry and spatial subdivision techniques for ray tracing. He is a professor of computer graphics in the Department of Information Technology and Systems at Delft University of Technology. In 1988, he worked for a year at the IBM Watson Research Center on CSG display algorithms for graphics hardware. His research interests include parallel ray tracing, realtime rendering, augmented reality, and 3D interaction. He is a member of the IEEE and the IEEE Computer Society