Time-Critical Multiresolution Rendering of Large Complex Models

Enrico Gobbetti and Eric Bouvier

CRS4, Center for Advanced Studies, Research and Development in Sardinia, Sesta Strada Ovest, Z.I. Macchiareddu, C.P. 94, I-09010 Uta (CA), Italy,
`http://www.crs4.it/vvr`

Abstract

Very large and geometrically complex scenes, exceeding millions of polygons and hundreds of objects, arise naturally in many areas of interactive computer graphics. Time-critical rendering of such scenes requires the ability to trade visual quality with speed. Previous work has shown that this can be done by representing individual scene components as multiresolution triangle meshes, and performing at each frame a convex constrained optimization to choose the mesh resolutions that maximize image quality while meeting timing constraints. In this paper we demonstrate that the nonlinear optimization problem with linear constraints associated to a large class of quality estimation heuristics is efficiently solved using an active-set strategy. By exploiting the problem structure, Lagrange multipliers estimates and equality constrained problem solutions are computed in linear time. Results show that our algorithms and data structures provide low memory overhead, smooth level-of-detail control, and guarantee, within acceptable limits, a uniform, bounded frame rate even for widely changing viewing conditions. Implementation details are presented along with the results of tests for memory needs, algorithm timing, and efficacy.

Key words: Multiresolution modeling, level-of-detail, adaptive rendering, numerical optimization, time-critical graphics

1 Introduction

Very large and geometrically complex models are common in a number of application domains, including rigid body simulations, computer-aided design, architectural visualizations, flight simulation, and virtual prototyping [4]. These models, exceeding millions of polygons and hundreds of distinct and possibly animated objects, cannot be displayed directly at interactive speeds even on high-end machines. The traditional approach to render them in a time-critical setting is to pre-compute a small number of independent level-of-detail (LOD) representations of each object

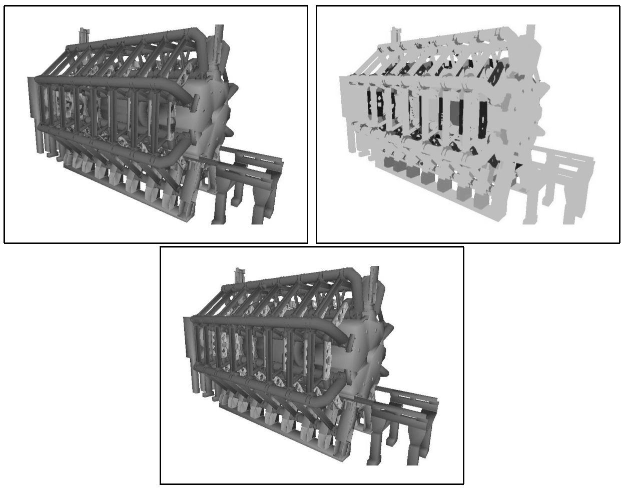
Fig. 1. **Multiresolution rendering example.** These three snapshots illustrate an example rendering of the ATLAS Experiment Pit, part of the LHC facility at CERN. In the bottom image,all objects are rendered at maximum resolution. The top left image is what the viewer actually sees during interactive navigation, with the resolution of each object modified to meet timing constraints. The top right image depicts the resolution chosen for each object, lighter shades representing more detail.

composing the scene, and to switch at run-time between the LODs. This solution has drawbacks both in terms of memory needs and quality of results.

In this paper, we propose to model 3D scenes as collections of multiresolution triangle meshes and to choose the resolution of each mesh by solving a continuous constrained optimization problem, i.e. the maximization of scene quality under timing constraints (see figure 1). Image quality degradation and rendering time are predicted using customizable heuristics. We have recently demonstrated that this solution, that predicts image quality degradation and rendering time using smooth customizable heuristics, leads to significantly higher-quality results over current LOD-based approaches [5]. In this paper, we build upon that work by developing an improved optimization method tailored to a large class of heuristics. Our time-critical multiresolution scene rendering framework improves over current LOD selection methods in the following areas:

- **Ability to meet timing constraints.** In contrast to current static or feedback algorithms, our technique is predictive and aims at guaranteeing a uniform, bounded frame rate even for widely changing viewing conditions; the technique is thus appropriate in a time-critical setting and enables the usage of user-input prediction to reduce perceived lag at the application level;
- **Scene and hardware independence.** Both the image degradation associated to using low resolution meshes and the hardware behavior are modeled by customizable heuristics. This makes it possible to incorporate context-sensitive quality constraints and automatically adapts applications to the specific hardware on which they are running, which is of primary importance for distributed multi-platform graphics applications visualizing a shared model;

- **Low memory overhead.** In contrast to standard LOD approaches, our multiresolution structure is compact while remaining fast enough for high-speed rendering. Without using specific compression techniques the overhead associated to our structure is of about 8% of the single full resolution mesh memory footprint when only position and normal are associated to vertices and becomes smaller if other attributes such as colors and texture coordinates are present;
- **Smooth, measurable image degradation.** Since our multiresolution structure efficiently supports geomorphs, smooth transitions between resolutions are obtained at no cost, without the need to resort to costly blending effects.

The rest of the paper is organized as follows. Section 2 reviews previous work in time-critical multiresolution scene rendering. Section 3 gives an overview of the general ideas of our approach, section 4 concentrates on the image quality degradation and rendering time prediction heuristics, while section 5 discusses the particular problem structure deriving from the selected heuristics and details a customized optimization algorithm. Section 6 describes our multiresolution mesh representation. Section 7 shows some experimental results and discusses advantages and limitations of the method. Finally, section 8 presents conclusions and a view of our future work.

2 Background and related work

2.1 Levels-of-detail for time-critical rendering

Many applications dealing with time-critical graphics include the possibility to store a 3D model in a fixed number of independent resolutions (e.g. OpenInventor [6] and VRML [7]). The main advantages of LODs are the simplicity of implementation and the fact that, as LODs are pre-calculated, the polygons can be organized in the most efficient way (triangle strips, display list), exploiting raw graphics processing speed with retained-mode graphics. The main drawbacks of this technique are related to its memory overhead, which severely limits in practice the number of LODs per object. As an example, representing an object at just the four LODs 100%, 75%, 50%, 25% would cause an overhead of 150% over the single resolution version. The small number of LODs might introduce visual artifacts due to the sudden changes of resolution between differing representations [8] and, more importantly, limits the degrees of freedom of the LOD selection algorithm.

Run-time LOD selection is typically done using static heuristics or feedback algorithms. Static heuristics (e.g. distance-based [7], coverage-based [6], or perceptually motivated [9] LOD mappings) are not adaptive and are therefore inherently unable to produce uniform frame rates, while feedback algorithms, which adaptively vary LOD mappings based on past rendering times (e.g. [10]) suffer of unavoidable overshoot and oscillation problems when the complexity of the environment varies rapidly, e.g. when entering a room in an walkthrough application.

As demonstrated by Funkhouser and Séquin [11], to guarantee bounded frame times, predictive algorithms that optimize LOD selection based on estimates of rendering time and image degradation must be used. Having a *guarantee* on the total maximum lag of the application is a necessary precondition for using prediction techniques for lag reduction [12]. Unfortunately, the combinatorial optimization problem associated to LOD selection is equivalent to a version of the Multiple Choice Knapsack Problem [11,13], which is NP-complete, and approximation algorithms that cannot guarantee optimality have to be used. Current state-of-the-art techniques (Funkhouser and Séquin's greedy algorithm [11] and Mason and Blake's incremental technique [13]) produce a result which is only guaranteed to be half as good as the optimal solution and have a running time depending both on the number of objects and on the number of LODs per object.

We have recently demonstrated that these problems are overcome when using appropriate multiresolution data structures which enable to express predictive LOD selection in the framework of continuous convex constrained optimization [5]. In this paper, we present an improved optimization algorithm that efficiently exploits the problem structure associated to a large class of image degradation and rendering time heuristics.

2.2 Dynamic simplification

An alternative to per-object LOD selection is to dynamically re-tessellate visible objects continuously as the viewing position shifts. As dynamic re-tessellation adds a run-time overhead, this approach is suitable when dealing with very large objects or static environments, when the time gained because of the better simplification is larger than the additional time spent in the selection algorithm. For this reason, this technique has been applied when the entire scene, or most of it, can be seen as a single multiresolution object from which to extract variable resolution representations.

The classic applications of dynamic re-tessellation techniques are in terrain visualization (see [14] for a survey). Hoppe [15] introduced view-dependent simplification of progressive meshes, applying it to the visualization of single large objects. Xia et al. [16,17] discuss the application of progressive meshes to scientific visualization. Luebke and Erikson [18] apply hierarchical dynamic simplification to large polygonal CAD models, by adaptively processing the entire database without first decomposing the environment into individual objects. To support interactive animated environments composed of many objects, this paper focuses on per-object view-independent resolution selection.

3 Overview of the approach

Our approach relies upon a scene description in which objects are represented as multiresolution triangle meshes, i.e. compact data structures able to efficiently provide on demand a triangle mesh approximation of an object with the requested number of faces. At each frame, we aim to find within a fixed short time the triangle mesh representation for each potentially visible object that produces the "best quality" image within the target frame time. This is done in an optimization phase which takes as input the set of potentially visible objects determined by a culling algorithm (e.g. bounding box or occlusion culling) and selects as output the list of triangle meshes to be rendered.

More formally, a triangle mesh is a piecewise linear approximation of a 3D shape that can be denoted by a tuple $M = (K, V, A)$, where K is a simplicial complex specifying the connectivity of the mesh simplices (the adjacency of the vertices, edges, and faces), $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ is the set of vertex positions defining the shape of the mesh in $\Re^3$, and $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m\}$ is the set of attributes associated to the vertices of M. Both the quality of approximation and the rendering complexity are dependent upon the sizes of K and V. A multiresolution representation for a mesh M with $N^{(\mathrm{maxvertex})}$ vertices and $N^{(\mathrm{maxtri})}$ triangles can be seen as a function that takes as input the desired resolution $r \in [0,1] \subset \Re$ and produces as output another mesh $M(r)$ that approximates the same shape with around $\lfloor rN^{(\mathrm{maxtri})} \rfloor$ faces. While the number of triangles is discrete, smooth transition between LODs can be obtained by using geomorphs [15], i.e. by interpolating between neighboring representations.

At each frame, the culling algorithm produces thus a parameterized representation of the visible objects set, which associates to each parameter value $\mathbf{r} \in [0,1]^n$ an actual set of triangle meshes $S(\mathbf{r}) = \{M_1(r_1), M_2(r_2), \ldots, M_n(r_n)\}$. Our goal is to find the optimal parameter vector $\mathbf{r}^\star$ for a given viewing configuration $\mathcal{E}$, which includes all parameters of the environment that influence the computation of the image of an object (e.g. viewpoint and viewing frustum, number and type of active lights). To this end, we define two heuristics for the visible object set: a $cost(\mathcal{E}, S(\mathbf{r}))$ heuristic, which estimates the time required to render a scene containing the visible objects at resolution $\mathbf{r}$, and a $degradation(\mathcal{E}, S(\mathbf{r}))$ heuristic, which estimates the quality degradation of the rendered image due to using lower resolution objects to replace full resolution ones. Even though the complexity of a multiresolution representation is discontinuous, as the number of triangles is discrete, we can safely assume that degradation and cost heuristics are smooth. This simplifying assumption, at the core of our approach, introduces an error which is clearly negligible for sufficiently large values of $N^{(\mathrm{maxtri})}$ with respect to the error intrinsically induced by the use of heuristics.

Using this formalism, our approach to predictive adaptive rendering is stated as follows:

$$\begin{array}{rl} \text{Minimize:} & degradation(\mathcal{E}, S(\mathbf{r})) \\ \text{Subject to:} & cost(\mathcal{E}, S(\mathbf{r})) \leq t^{(\text{desired})} \\ & \mathbf{r} \succeq \mathbf{0} \\ & \mathbf{r} \preceq \mathbf{1} \end{array} \tag{1}$$

where $\succeq$ and $\preceq$ denote componentwise inequality, $\mathbf{0}$ and $\mathbf{1}$ are constant vectors and $t^{(\text{desired})}$ is the target display time.

The difficulty of solving problem 1 depends largely on the nature of the $degradation$ and $cost$ heuristics. As we will see, by using a suitable multiresolution representation, simple forms of these heuristics can be found, leading to efficient solution methods applicable in a time-critical setting.

4 Cost and degradation heuristics

4.1 Cost heuristic

Time-critical renderers are typically running on top of a pipelined graphics hardware implementing a Z-buffer algorithm. Scene display starts with an initialization phase (initial setup, buffer clears), followed by the sequential rendering of each of the meshes, and finishes by a finalization phase (buffer swap). Initialization and finalization time can be considered constants, while, assuming a fast enough CPU and an efficient multiresolution mesh representation, mesh rendering is dominated either by the time to define rendering attributes and process all the triangles, or by the time to fill the covered pixels, depending on where the pipeline bottleneck is. On most hardware configurations, the time to define rendering attributes and process all the triangles is just dictated by the speed of the graphics pipe-line for all operations before rasterization. On very high-end graphics systems, actually fetching triangles from the multiresolution structure may become the dominant phase. In both cases, however, we assume that the cost remains linear in the number of triangles and we thus only need to determine the "speed" of the dominant phase for the prediction of the rendering time. As we will see in the results section, this assumption is verified for the data structure presented in this paper.

In other words, the cost of rendering a multiresolution mesh M at resolution r can be estimated as follows:

$$t^{(\mathrm{setup})} + max \begin{Bmatrix} t^{(\mathrm{tri})} r \cdot N^{(\mathrm{maxtri})}(M) \\ t^{(\mathrm{pix})} N^{(\mathrm{pix})}(M, \mathcal{E}) \end{Bmatrix} \tag{2}$$

where $N^{(\mathrm{maxtri})}(M)$ is the maximum number of triangles for mesh M, $t^{(\mathrm{setup})}$ is the time associated to setup the rendering environment for an object (e.g. material binding time for OpenGL), $t^{(\mathrm{tri})}$ is the time to send a triangle through the pipeline (i.e. the maximum between the time to fetch a triangle from the multiresolution structure and that of processing it without rasterization), $t^{(\mathrm{pix})}$ is the time to fill a pixel, and $N^{(\mathrm{pix})}(M, \mathcal{E})$ is an estimation of the number of pixels covered by mesh M when rendered with a viewing configuration $\mathcal{E}$. From equation 2, we can derive the minimal resolution $r^{(\mathrm{min})}$ under which a reduction in resolution (and therefore possibly in quality) does not reduce rendering time:

$$r^{(\mathrm{min})} = \frac{t^{(\mathrm{pix})} N^{(\mathrm{pix})}(M, \mathcal{E})}{t^{(\mathrm{tri})} N^{(\mathrm{maxtri})}(M)} \tag{3}$$

The cost heuristics can thus be modeled as:

$$\begin{aligned} cost(\mathcal{E}, S(\mathbf{r})) &= t^{(\mathrm{fixed})} + \mathbf{t}^{(\mathrm{max})^{\top}} \mathbf{r} \\ \mathbf{r} &\succeq \mathbf{r}^{(\mathrm{min})} \\ \mathbf{r} &\preceq \mathbf{1} \end{aligned} \tag{4}$$

where $t^{(\mathrm{fixed})} = t^{(\mathrm{init})} + t^{(\mathrm{final})} + \sum_i t_i^{(\mathrm{setup})}$ is the resolution-independent portion of the frame time, $t^{(\mathrm{init})}$ and $t^{(\mathrm{final})}$ are the frame initialization and finalization times, $\mathbf{t}^{(\mathrm{max})}$ is the vector containing the maximum rendering time $t^{(\mathrm{tri})} \cdot N^{(maxtri)}(M)$ for each mesh M, and $\mathbf{r}^{(\mathrm{min})}$ is the vector of minimal resolutions as of equation 3. The constants $t^{(\mathrm{setup})}$, $t^{(\mathrm{tri})}$, $t^{(\mathrm{pix})}$, $t^{(\mathrm{init})}$, and $t^{(\mathrm{final})}$ can be determined by benchmarks in a preprocessing step. As these constants obviously depend on rendering attributes such as shading model, number of light sources, and texturing, we pre-compute their values for each combination of rendering attributes and choose at run-time the appropriate set of values to use for each object.

The cost model presented here assumes an ideal environment in which rendering time is dictated only by rendering operations. In practice, however, process scheduling, page faults, and other direct or indirect blocking kernel calls are out of user control and have an impact on the rendering time. Our current approach to reduce unwanted variations in frame-rate is to add to $t^{(fixed)}$ a worst case estimate of the impact of the system and application environment on the rendering time.

4.2 Image quality degradation heuristic

The $degradation(\mathcal{E}, S(\mathbf{r}))$ heuristic should provide an estimation of the perceptual distance between the image produced by rendering in a viewing configuration $\mathcal{E}$ a scene composed of the multiresolution objects present in S at resolutions $\mathbf{r}$ and the image obtained in the same configuration with all objects at full resolution.

Since local illumination models are used in time-critical applications, global image quality degradation is well approximated by summing per-object quality degradation measures. Visual degradation can thus be modeled via an equation of the form:

$$degradation(\mathcal{E}, S(\mathbf{r})) = \sum_i w_{\text{interest}}(\mathcal{E}, S_i)\varepsilon_{\text{image}}(\mathcal{E}, S_i, r_i) \tag{5}$$

where $w_{\text{interest}}(M, \mathcal{E})$ is a resolution-independent weighting factor measuring the importance of the object M to the user from the viewpoint $\mathcal{E}$ and $\varepsilon_{\text{image}}(\mathcal{E}, M, r)$ is a factor estimating how well the image produced by the mesh at resolution r approximates the image of the mesh at maximum resolution.

4.2.1 Resolution-independent factors

In our implementation, object importance is modeled by:

$$w_{\text{interest}}(\mathcal{E}, M) = w_v(\mathcal{E}, M)w_e(\mathcal{E}, M)w_s(M) \tag{6}$$

where $w_v(\mathcal{E}, M)$ and $w_e(\mathcal{E}, M)$ are factors proportional to the decline in visual acuity with apparent object velocity and distance from the focus point and $w_s(M)$ is a user-definable object importance factor (e.g. to impose higher quality for interactively manipulated objects). Various authors have derived velocity and eccentricity factors from experimental data. We use the models developed by Reddy [19], which are tailored to computer graphics imagery:

$$
\begin{aligned}
w_v(\mathcal{E}, M) &= \begin{cases} 1 & \text{when } v \leq 0.825 \\ -0.463 \log(v) + 0.9615 & \text{otherwise} \end{cases} \\
w_e(\mathcal{E}, M) &= \begin{cases} 1 & \text{when } E \leq 5.79 \\ 7.49/(1 + 0.3E)^2 & \text{otherwise} \end{cases}
\end{aligned} \tag{7}
$$

where v is the apparent velocity of the object's bounding box center and E is the apparent distance of the projected bounding box to the focus point (i.e. to the screen's center or to the 2D cursor position during object manipulation). Both v and E should be measured in visual degrees, taking thus into account the physical display resolution and physical viewing distance. The conversion from pixels to visual degrees is straightforward [19]:

$$
p = 2d \tan \frac{1}{2} \min(\frac{R_x}{W_x}, \frac{R_y}{W_y}) \tag{8}
$$

where p is the number of pixels per visual degree, R_x and R_y are the horizontal and vertical display resolution, W_x and W_y are the physical horizontal and vertical display size, and d is the physical viewing distance.

Similar models for w_{interest}, but with different formulations of w_v and w_e are presented in [11,20,21,5].

4.2.2 Resolution-dependent factors

Accurately measuring $\varepsilon_{\text{image}}(\mathcal{E}, M, r)$ is the most challenging task: measuring image difference using a perceptually meaningful image metric is still an open research subject (e.g. see [22]) and all proposed techniques would require an excessive amount of computation to be carried-out on-line. The main techniques that have been proposed for run-time LOD selection (e.g. [11,20,5]) thus predict visual degradation via an analytic function based on a more or less explicit model of the effects of simplification on the graphics models. A formulation that generalizes a wide class of proposed heuristics is the following:

$$
\varepsilon_{\text{image}}(\mathcal{E}, M, r) = \frac{w_i(\mathcal{E}, M)}{\alpha} r^{-\alpha} \tag{9}
$$

where $w_i(\mathcal{E}, M) > 0$ is a resolution-independent weighting factor and $\alpha > -1$ is an exponent defining the shape of the error curve. The most common choices for these parameters are:

$$\begin{aligned}
[20]:&\{w_i(\mathcal{E}, M) = \frac{N^{(\mathrm{pix})}(\mathcal{E}, M)}{N^{(\mathrm{maxtri})}(M)}, \alpha = 1\} \\
[11]:&\{w_i(\mathcal{E}, M) = \frac{N^{(\mathrm{pix})}(\mathcal{E}, M)}{N^{(\mathrm{maxtri})}(M)^2}, \alpha = 2\} \\
[5]:&\{w_i(\mathcal{E}, M) = N^{(\mathrm{pix})}(\mathcal{E}, M)\sqrt{N^{(\mathrm{maxtri})}(M)}, \alpha = -\frac{1}{2}\}
\end{aligned} \tag{10}$$

These formulations intuitively express image degradation as a function which monotonously increases with the average projected triangle size.

4.2.3 Temporal coherence

The degradation heuristic defined in the previous sections is purely based on image quality and does not depend on quality variation over time. It is possible to explicitely take into account temporal coherence by including in equation 5 an hysteresis factor penalising resolution changes, as done for example in [5,11]. We have found, however, that for complex scenes hysteresis is not necessary, as resolution changes are already distributed on many objects and, because of the resolution-independent factors, already primarily affect the objects on which changes are less noticeable. We therefore do not include hysteresis in our quality degradation heuristic.

5 Optimization algorithm

5.1 Resolution optimization problem

With our cost and degradation heuristics, the resolution optimization problem 1 can now be written as:

$$\min\{f(\mathbf{r}) = \sum_i \frac{c_i}{\alpha} r_i^{-\alpha} : \mathbf{A}\mathbf{r} \preceq \mathbf{b}\} \tag{11}$$

where

$$
\mathbf{A} = \begin{pmatrix} t_1^{(\max)} & t_2^{(\max)} & \dots & t_n^{(\max)} \\ -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} t^{(\text{desired})} - t^{(\text{fixed})} \\ -r_1^{(\min)} \\ -r_2^{(\min)} \\ \vdots \\ -r_n^{(\min)} \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}
$$

$$
\mathbf{c} = \begin{pmatrix} w_v(\mathcal{E}, M_1) w_e(\mathcal{E}, M_1) w_s(M_1) w_i(\mathcal{E}, M_1) \\ w_v(\mathcal{E}, M_2) w_e(\mathcal{E}, M_2) w_s(M_2) w_i(\mathcal{E}, M_2) \\ \vdots \\ w_v(\mathcal{E}, M_n) w_e(\mathcal{E}, M_n) w_s(M_n) w_i(\mathcal{E}, M_n) \end{pmatrix}
$$

As we will see, this particular problem structure leads to an efficient numerical solution techniques.

5.2 Idea of an active set strategy

The first-order necessary conditions for the existence of a minimizer $\mathbf{r}^\star$ of the constrained optimization problem 11 require the existence of Lagrange multipliers λ_i, such that

$$
\begin{aligned} \nabla_r f(\mathbf{r}^\star) + \sum_{i \in \mathcal{A}^\star} \lambda_i^\star \mathbf{a}_i &= 0 \\ \lambda_i^\star &\geq 0 \end{aligned} \tag{12}
$$

where $\mathcal{A}^\star = \{i \in \mathcal{I} : \mathbf{a_i}^T \mathbf{r}^\star = b_i\}$ is the active set at $\mathbf{r}^\star$, $\mathcal{I} = \{1, 2, ..., 2n + 1\}$ is the index set for the inequality constraints, $\mathbf{a}_i$ denotes the i-th row of the constraint matrix $\mathbf{A}$, and $\mathbf{b}_i$ denotes the i-th element of the constraint vector $\mathbf{b}$. This implies that it is possible to search for a solution of the original inequality- constrained optimization problem along the edges and faces of the feasible set by solving a sequence of equality-constrained problems of the form $\min\{f(\mathbf{r}) : \mathbf{a}_i^T\mathbf{r} = \mathbf{b}, i \in \mathcal{W}\}$, iteratively refining the working set $\mathcal{W}$ with the aim of determining the optimal active set $\mathcal{A}^\star$ which satisfies condition 12.

This type of optimization technique, known as ”active-set strategy” has proven very effective for bound constrained or quadratic programming problems [23,24]. The performance of the method depends on how efficient it is to generate an initial feasible solution, to solve equality constrained problems during the iterative refinement phase, and to compute Lagrange multipliers for checking convergence. As we will see, all these sub-problems can be solved with extreme efficiency in the case of resolution optimization.

5.3 Active set strategy for resolution optimization

5.3.1 Initialization phase

The generation of a feasible starting point $\mathbf{r}^{(0)}$ and an initial working set $\mathcal{W}^{(0)}$ for the problem is straightforward, since we have known lower resolution bounds $\mathbf{r}^{(\min)}$, a cost heuristic monotonously increasing with $\mathbf{r}$, and a degradation heuristic which is instead monotonously decreasing.

If the problem is infeasible (i.e. $cost(\mathbf{r}^{(\min)}) > t^{(\text{desired})}$) or if the problem is feasible with all objects at full resolution (i.e. $cost(\mathbf{1}) \leq t^{(\text{desired})}$), the optimization process terminates and appropriate values for $\mathbf{r}^\star$ can be readily returned. In all other cases, we know that the problem has a solution with a rendering time of exactly $cost(\mathbf{r}^\star) = t^{(\text{desired})}$. This means that in the time-constrained case, $1 \in \mathcal{A}^\star$, as constraint number 1 is the timing constraint which is always satisfied at the optimum. We can thus initialize the working set $\mathcal{W}^{(0)} = \{1\}$. The simplest solution to generate a feasible starting point $\mathbf{r}^{(0)}$ is to start searching for a solution from $\mathbf{r}^{(0)} = \mathbf{r}^{(\min)}$, which we know is feasible. An incremental technique that produces a value which is closer to the optimum is to start from the resolution of the objects computed at the previous frame (or from the minimal resolution for newly visible objects) and to iteratively reduce, until the problem becomes feasible, object resolutions by a factor β, starting from the objects with the highest degradation/cost ratio [5].

5.3.2 Iterative refinement

At each iteration k, given a feasible $\mathbf{r}^{(k)}$ and a current working set $\mathcal{W}^{(k)}$, we find a search direction $\mathbf{d}^{(k)}$ by solving the equality constrained problem

$$\min\{f(\mathbf{r}^{(k)} + \mathbf{d}^{(k)}) : \mathbf{a}_i^T(\mathbf{r}^{(k)} + \mathbf{d}^{(k)}) = b_i, i \in \mathcal{W}^{(k)}\} \tag{13}$$

The solution to this problem can be easily computed by introducing the Lagrangian function $\Phi(\mathbf{d}, \nu) = f(\mathbf{r}^{(k)} + \mathbf{d}) + \sum_{i \in \mathcal{W}^{(k)}} \nu_i(\mathbf{a}_i^T(\mathbf{r}^{(k)} + \mathbf{d}) - b_i)$ and solving for $\nabla_{\mathbf{d},\nu}\Phi(\mathbf{d}, \nu) = 0$. Plugging equation 11 into this and solving for $\mathbf{d}$ gives us the following simple analytic formula for $\mathbf{d}^{(k)}$:

$$d_i^{(k)} = \begin{cases} -r_i^{(k)} + \frac{b_{i+1}}{a_{i+1,i}} & \text{if } i \in \mathcal{L}^{(k)} \\ -r_i^{(k)} + \frac{b_{i+n+1}}{a_{i+n+1,i}} & \text{if } i \in \mathcal{U}^{(k)} \\ -r_i^{(k)} + \frac{(b_1 - \sum_{j \in \mathcal{B}^{(k)}} (r_j^{(k)} + d_j^{(k)}) a_{1,j}) c_i}{a_{1,i}^{\frac{1}{\alpha+1}} \sum_{j \in \mathcal{I} \setminus \mathcal{B}^{(k)}} c_j a_{1,j}^{1 - \frac{1}{\alpha+1}}} & \text{otherwise} \end{cases} \tag{14}$$

where $\mathcal{L}^{(k)} = \{i \in \{1, 2, \ldots, n\} : i + 1 \in \mathcal{W}^{(k)}\}$ is the index set of the variables whose value is determined by active lower bound constraints, $\mathcal{U}^{(k)} = \{i \in \{1, 2, \ldots, n\} : i + n + 1 \in \mathcal{W}^{(k)}\}$ is the index set of the variables whose value is determined by active upper bound constraints, and $\mathcal{B}^{(k)} = \mathcal{L}^{(k)} \cup \mathcal{U}^{(k)}$.

Taking a step $\mathbf{d}^{(k)}$ would take us to the minimizer of f on the subspace defined by the current working set, but may violate some new constraints. We thus compute the largest possible step size $\mu^{(k)} \leq 1$ that does not violate any constraint and set $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \mu^{(k)}\mathbf{d}^{(k)}$. The working set is then updated by including in $\mathcal{W}^{(k+1)}$ all constraints active at $\mathbf{r}^{(k+1)}$ and the process is repeated until convergence.

5.3.3 Convergence test

If $\mathbf{d}^{(k)} = 0$, then $\mathbf{r}^{(k)}$ is a feasible solution of the problem which is also the minimizer of f on the subspace defined by $\mathcal{W}^{(k)}$. First order optimality conditions imply that there exist multipliers $\lambda_i^{(k)}$ such that:

$$\nabla_r f(\mathbf{r}^{(k)}) + \sum_{i \in \mathcal{W}^{(k)}} \lambda_i^{(k)} \mathbf{a}_i = \mathbf{0} \tag{15}$$

From 12, we know that if $\lambda_i^{(k)} \geq 0$ for $i \in \mathcal{W}^{(k)}$, then $\mathcal{W}^{(k)} = \mathcal{A}^\star$ and $\mathbf{r}^{(k)}$ is the minimizer for the original problem. In this case, the iterative refinement process can terminate with the solution $\mathbf{r}^\star = \mathbf{r}^{(k)}$. Otherwise, we remove from the working set the constraint with the most negative $\lambda_i^{(k)}$, therefore enabling a further decrease in the value of f in subsequent iterative refinement steps.

The unique set of Lagrange multipliers $\lambda_i^{(k)}$ necessary for the convergence test is easily computed by finding the minimum norm solution of equation 15. Plugging equation 11 into 15 and solving for $\lambda_i^{(k)}$ gives us the following simple analytic formula:

$$\lambda_i^{(k)} = \begin{cases} -\frac{\sum_{j \in \mathcal{I} \setminus \mathcal{B}^{(k)}} a_{1,j} c_j r_j^{-\alpha-1}}{\sum_{j \in \mathcal{I} \setminus \mathcal{B}^{(k)}} a_{1,j}^2} & \text{if } i = 1 \\ \frac{-c_{i-1} r_{i-1}^{\alpha-1} - \lambda_1^{(k)} a_{1,i-1}}{a_{i,i-1}} & \text{if } i-1 \in \mathcal{L}^{(k)} \\ \frac{-c_{i-n-1} r_{i-n-1}^{\alpha-1} - \lambda_1^{(k)} a_{1,i-n-1}}{a_{i,i-n-1}} & \text{if } i-n-1 \in \mathcal{U}^{(k)} \end{cases} \tag{16}$$

5.3.4 Time-critical optimization algorithm

While the presented algorithm is very efficient, requiring only $O(n)$ operations per step, it is possible that for very large scenes the time needed to find the optimal solution could become too high for real-time use. However, as at each frame the algorithm starts from a solution which is close to the one computed at the previous frame and produces a sequence of intermediate solutions $\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \ldots$ monotonously converging to the optimum, we know that acceptable approximate results are available at certain time-deadlines. It is thus possible to create a time-critical version of the algorithm by modifying the termination criterion to not only include the convergence test but also a test for the expiration of the allocated time. This idea is used in algorithm 1. Both the CPU time spent in the optimization and the time that will be spent in rendering the meshes at the resolution suggested by the optimization algorithm are thus parameters of the algorithm and can be externally imposed. These characteristics make it an ideal candidate for an optimization stage in a time-critical rendering pipeline.

5.4 Time-critical rendering pipeline

A time-critical rendering pipeline aims to display an image at certain time-deadlines independently of the complexity of the scene. To reach this goal, we exploit the properties of our algorithm by adaptively controlling at each frame both the time budget allocated to the optimization and the desired display time.

The parameters under system control are the maximum visual feedback lag $t^{(\mathrm{lag})}$, and the fraction of the frame time to devote to optimization. At each frame, we perform the culling, optimization, and display steps in a sequence. The culling step's time may vary and is dependent on the type of algorithm used and on the complexity of the scene as seen from the current viewpoint. Before starting the optimization step, we measure how much of the frame time is still available and allocate in this range the appropriate time budgets for the optimization and display steps. The optimization step is run for the allocated time and its result is then passed to the final

Algorithm 1 Active-set strategy for solving $\min\{f(r) : \mathbf{a}_i^T\mathbf{r} \leq \mathbf{b}, i \in \mathcal{I}\}$

```
**Given:** Starting point $\mathbf{r}^{(0)} : \mathbf{a}_i^T\mathbf{r}^{(0)} \leq \mathbf{b}, i \in \mathcal{I}$
**Given:** Timing constraint $t^{(opt)}$
$k \leftarrow 0$; $done \leftarrow \mathbf{false}$; $stopped \leftarrow \mathbf{false}$
$\mathcal{W}^{(k)} \leftarrow \{i \in \mathcal{I} : \mathbf{a}_i^T\mathbf{r}^{(k)} = \mathbf{b}\}$
**while not** ($done$ **or** $stopped$) **do**
  $\mathbf{d}^{(k)} \leftarrow \arg\min_{\mathbf{d}}\{f(\mathbf{r}^{(k)} + \mathbf{d}) : \mathbf{a}_i^T(\mathbf{r}^{(k)} + \mathbf{d}) = \mathbf{b}, i \in \mathcal{W}^{(k)}\}$
  **if** $\|\mathbf{d}^{(k)}\| = 0$ **then**
    $\mathbf{r}^{(k+1)} \leftarrow \mathbf{r}^{(k)}$
    $\lambda^{(k+1)} \leftarrow \lambda : \nabla f(\mathbf{r}^{(k+1)}) + \sum_{i \in \mathcal{W}^{(k)}} \lambda_i^{(k+1)}\mathbf{a}_i = 0$
    $j \leftarrow \arg\min_i\{\lambda_i^{(k+1)}\}$
    **if** $\lambda_j^{k+1} < 0$ **then**
      $\mathcal{W}^{(k+1)} \leftarrow \mathcal{W}^{(k)} \setminus \{j\}$
    **else**
      $\mathcal{W}^{(k+1)} \leftarrow \mathcal{W}^{(k)}$; $done \leftarrow \mathbf{true}$
    **end if**
  **else**
    $\mu^{(k)} \leftarrow \arg\max_\mu\{\mu = \min(1, \frac{b_i - \mathbf{a}_i^T\mathbf{r}^{(k)}}{\mathbf{a}_i^T\mathbf{d}^{(k)}}) : \mathbf{a}_i^T\mathbf{d}^{(k)} > 0, i \in \mathcal{I} \setminus \mathcal{W}^{(k)}\}$
    $\mathbf{r}^{(k+1)} \leftarrow \mathbf{r}^{(k)} + \mu^{(k)}\mathbf{d}^{(k)}$
    $\mathcal{W}^{(k+1)} \leftarrow \mathcal{W}^{(k)} \cup \{i \in \mathcal{I} \setminus \mathcal{W}^{(k)} : \mathbf{a}_i^T\mathbf{r}^{(k)} = \mathbf{b}\}$
  **end if**
  $k \leftarrow k + 1$; $stopped$ = Time elapsed $> t^{(opt)}$
**end while**
$\mathbf{r}^\star \leftarrow \mathbf{r}^{(k)}$
$\mathcal{W}^\star \leftarrow \mathcal{W}^{(k)}$
**Ensure:** $\mathbf{r}^\star$ is feasible and $f(\mathbf{r}^\star) \leq f(\mathbf{r}^{(0)})$
**Ensure:** $done \Rightarrow \mathbf{r}^\star$ is optimal and $\mathcal{W}^\star$ is the optimal active set
**Ensure:** $done \Rightarrow$ Time elapsed $\leq t^{(opt)}$
**Ensure:** **not** $done \Rightarrow$ Time elapsed $\approx t^{(opt)}$
```

display stage. This time-critical computing approach bounds the maximum visual feedback lag and enables the use of prediction techniques that extrapolate past user input data to future time points for lag reduction [12].

On a single-processor machine, the maximum visual feedback lag also dictates the maximum visual feedback frequency. On a multi-processor machine, visual feedback frequency can be independently controlled using separate threads for each pipeline stage, as in [10].

6 Multiresolution mesh representation

Our optimization approach is independent from the particular data structure used to represent multiresolution meshes. The only requirements are the ability to represent a mesh with an arbitrary number of triangles and to traverse the structure at arbitrary resolutions faster than the graphics pipe-line or, at least, in a time compatible with our linear cost model. An additional requirement for our approach to be practical for large scene databases is data structure compactness.

The Progressive Mesh (PM) [15] representation is a suitable candidate structure. However, the PM representation is compact but cannot be rendered directly, since it has first to be traversed to construct a single resolution mesh structure which is then used for rendering [25]. Managing the dynamic mesh structures associated to each multiresolution representation introduces both time and space overhead in scene rendering application. Experimental results [25] indicate a reconstruction rate of less than 200K triangles/sec on a Pentium Pro 200 Mhz. While this cost can be amortized on more than one frame if the single resolution meshes are cached, this is at the expense of memory. Moreover, exploiting per-object frame-to-frame coherency is only a partial solution for complex scenes, because of the discontinuity in scene complexity caused by objects entering into or exiting from the viewing frustum [11].

In this section, we propose a simple multiresolution triangle mesh structure (TOM: Totally Ordered Mesh) that efficiently supports vertex packing and indexing. The structure is compact, requiring only a small overhead over the single full resolution mesh, and provides fast triangle and vertex traversal rates at any resolution. A similar structure has been independently developed by Guéziec et al. [26] for streaming geometry in VRML.

6.1 TOM: Totally Ordered Mesh

Several algorithms have been recently published that simplify a polygonal mesh by iteratively contracting vertex pairs (e.g. [27,28,15,29–31]). A vertex pair contraction operation, denoted $(v_1, v_2) \rightarrow \tilde{v}$, replaces two vertices (v_1, v_2) with a single target point $\tilde{v}$ to which all the incident edges are linked, and removes any triangles that have degenerated into lines or points. The operation is quite general, and can express both edge-collapse and vertex clustering algorithms. The primary difference between vertex pair contraction algorithms lies in how the particular vertex pairs to be contracted are chosen and in where the new vertices are positioned. We define vertex substitution, denoted $v_1 \rightarrow v_2$, the restricted form of vertex pair contraction where the target point $\tilde{v}$ is constrained to be the second vertex of the pair, v_2 (see figure 2). By iteratively applying vertex substitution, a triangle mesh can be reduced by removing one vertex and possibly some degenerated faces at a time. Recent research results demonstrate that good simplification quality and speed can be obtained using this technique [32].

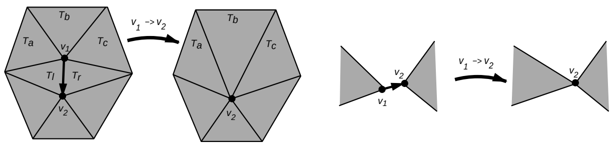

Fig. 2. **Vertex substitution.** On the left, the substitution $v_1 \rightarrow v_2$ removes two triangles, as (v_1, v_2) is an edge. On the right, the substitution $v_1 \rightarrow v_2$ just connects two disjoint mesh regions. In both cases, all the simplification information can be retrieved from data stored at the vertex level in the original vertex list.

6.1.1 Data structure

As iterative vertex substitution does not modify vertex data and does not add new vertices, the only information that has to be stored explicitly is the vertex substitution history of each vertex. A total order can be defined both on the vertex list and on the triangle list based on the contraction resolution. Sorting according to this order after the simplification generates a compact and efficient data structure (see figure 3). By ordering the vertex list, we obtain a packed representation where the active vertices at vertex resolution $r_v = \frac{n}{N^{(\mathrm{maxvertex})}}$ are exactly the first n ones in the vertex array of size $N^{(\mathrm{maxvertex})}$. Moreover, by ordering the triangle list, we have a way to iterate through the triangles that define the mesh at an arbitrary resolution in a time depending only on the number of active triangles and the lengths of the vertex substitution chains.

The memory overhead introduced to store the multiresolution mesh is limited to the space required to store the vertex substitution history associated to vertex pair contraction. We encode a vertex substitution by associating to each vertex the vertex resolution at which the transformation occurs and the reference to the vertex by which it is to be substituted. As vertices are sorted according to their resolution,

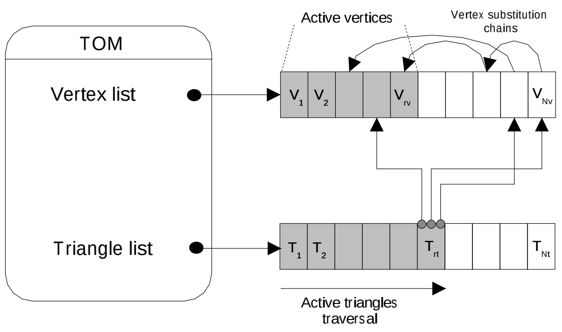

Fig. 3. **TOM data structure.** Multiresolution meshes are stored using a vertex list and a triangle list sorted according to contraction resolution.

only the vertex reference needs to actually be stored, since the vertex resolution is implicit in the vertex index. The minimal vertex resolution of a triangle, i.e. the vertex resolution at which a triangle is removed from the mesh because of the contraction of one of its edges, does not need to be stored, as it can be retrieved in a short time by traversing the substitution chains of its vertices.

With this representation, the space overhead over a single-resolution mesh representation is equal to just one vertex index per vertex. For a typical mesh of $N^{(maxtri)} = 2N^{(\mathrm{maxvertex})}$ triangles, considering to use 32 bits to represent both a vertex index and a floating point number, the overhead associated to the above structure is of about 8% of the single full resolution mesh memory footprint when only position and normal are associated to vertices and becomes smaller if other attributes such as colors and texture coordinates are present.

6.1.2 Mesh traversal

To render a mesh with a specified number of triangles n, we first determine the vertex resolution r_v at which triangle number $n+1$ collapses, then traverse the first triangles following the vertex chains until the active vertices are reached. Triangle traversal stops at the first collapsed triangle, i.e. the first time a triangle has two equal vertex indices. It should be noted that, as the structure is based on vertex substitution, it could be impossible to get an approximation with exactly n triangles, as each substitution deletes one or more faces (typically two for manifold objects). In this case, the traversal algorithm enumerates at most n faces, introducing a negligeable error.

As demonstrated in the results section, the lengths of the substitution chains are limited and relatively independent from the model size. In any case the lenght at full resolution is always so that triangle traversal at full resolution is strictly linear. When resolution decreases, the traversal rate also decreases but slowly, because vertex substitution cannot, by definition, create too long chains for all the vertices. In fact, each vertex substitution $v_1 \rightarrow v_2$ increments by one the depth of all the vertex chains containing vertex v_1 but also keeps unchanged the length of all chains containing vertex v_2.

Smooth transitions between resolutions can be obtained by interpolating vertex data. As only one vertex is substituted at each simplification step, only the attributes for the last active vertex have to be interpolated before the traversal. We currently implement this feature using linear interpolation for all data.

7 Implementation and results

An experimental software library supporting the time-critical multiresolution scene rendering algorithm described in this paper has been implemented and tested on Silicon Graphics IRIX and Windows NT machines. The results presented here were obtained on a Silicon Graphics 320 PC running Windows NT 4.0 and configured with a single 500 MHz Pentium III processor with 512 Kb L2 cache, 256 Mb RAM, and a Cobalt graphics chipset.

7.1 TOM data structure

The triangle meshes used in the evaluation were selected among those available in the public domain to enable comparison with other approaches. The characteristics of these meshes are summarized in table 1. Multiresolution versions have been constructed using a memoryless simplification algorithm similar to the one introduced by Lindstrom and Turk [30], modified for building TOM data structures by incrementally constructing vertex chains during simplification. A detailed description of the simplification technique is beyond the scope of this paper. Details are available elsewhere [33].

Mesh	**Vertices**	**Triangles**	**Site**
bunny	35942	69449	*www-graphics.stanford.edu*
fandisk	6871	12946	*www.research.microsoft.com/˜hoppe*
cow	2915	5804	*www.cs.cmu.edu/˜garland/cow.html*

Table 1
Triangle meshes used in the tests

7.1.1 Memory

Table 2 summarizes the storage requirements of the TOM structure for the test meshes, compared to the original mesh in face-vertex form, to a typical LOD representation with the six levels of details 100%, 50%, 25%, 12%, 6%, 3%, and to a progressive mesh (PM) representation using the memory-resident data structure presented in [25], simplified by removing face-level attributes to make comparison fair. As the PM representation cannot be rendered directly, but has first to be traversed to construct a single resolution mesh structure [25], we provide for this representation the minimum and maximum memory required, corresponding to rendering the mesh at the lowest, respectively highest, possible resolution. All size estimations assume that the mesh contains only normals as vertex attributes, and that 32 bits are used for both integer and floating point data.

Mesh	*bunny*	*fandisk*	*cow*
Face-Vertex	1656 (100%)	313 (100%)	136 (100%)
TOM	1797 (108%)	340 (108%)	148 (108%)
LOD	3521 (213%)	666 (213%)	290 (212%)
PM-0%	1966 (119%)	376 (120%)	159 (117%)
PM-100%	4436 (268%)	840 (269%)	364 (267%)

Table 2
Minimum storage needs for a rendering application. Sizes in Kb, percentages are with respect to the mesh in face-vertex form.

As we can see from table 2, the TOM multiresolution structure is the most compact, with an overhead of only 8% with respect to the single resolution mesh in face-vertex form. The overhead fraction is further reduced when associating more attributes at each vertex (e.g. normal, color, texture coordinates).

7.1.2 Traversal and rendering

As we are focusing on time-critical scene rendering applications, the most important results are relative to the triangle traversal rate through our multiresolution structure. The overhead with respect to traversing a single resolution mesh is only dependent on the lengths of the vertex substitution chains. Figure 4 presents the overhead associated to traversing the *bunny*, *fandisk*, and *cow* meshes at various resolutions. As we can see, the results on different meshes are very similar. The overhead grows slowly with the simplification ratio, and in all cases, never exceeds 75%. It is thus possible to traverse all active triangles while retrieving vertex attributes at a speed sufficient to feed the 3D graphics pipeline even on high-end 3D accelerators.

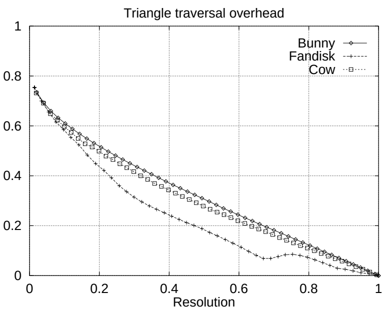

Fig. 4. **Multiresolution traversal overhead.** Ratio of extra vertices traversed to render meshes at multiple resolutions. Resolution expressed as fraction of vertices.

Figure 5 presents the data structure traversal and rendering performance obtained with the multiresolution structure for the *cow* mesh. Similar results were obtained for the other meshes and are not presented here. For the benchmarks we used two directional lights, Gouraud shading, and a viewing configuration compressing the mesh to a 50x50 pixel area, to avoid fill-rate bottlenecks. Our current implementation renders meshes with the strict OpenGL 1.0 core command set as a sequence of independent triangles whose vertices are specified using `glNormal/glVertex` calls. We expect a performance improvement by using the `EXT_vertex_array` extension.

As we can see, even using standard OpenGL, the rendering performance at all resolutions for the multiresolution version is similar to the limit given by rendering a streamlined version of the mesh. In both cases, the meshes are rendered at a constant speed of about 800 KTris/second at all resolutions, showing that our linear cost heuristics is a good approximation of the rendering behavior. Triangle traversal rates on the multiresolution structure, measured by having the traversal routine call empty vertex/normal procedures are well above the rates obtained with the rendering code for all resolutions, since they range from 3.7 MTris/second to over 10 MTris/second. We can therefore expect to have good timing predictions using the linear cost heuristics even on current high-end machines.

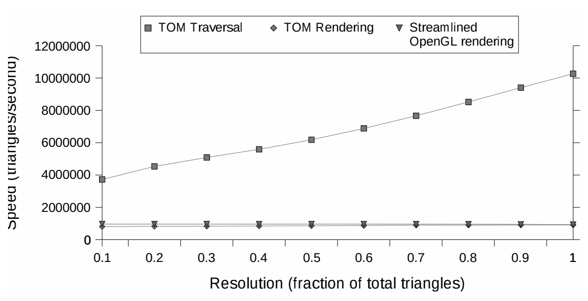

Fig. 5. **Multiresolution traversal and rendering performance.** Timing statistics for the *cow* mesh. Experimental measures on a SGI 320, single 500 MHz Pentium III with 512 Kb L2 cache, 256 Mb RAM, Cobalt graphics. Rendering environment: two directional lights, one material per object, Gouraud shading, independent triangles, one normal per vertex

7.2 Time-critical rendering

To test the behavior of our algorithm, we have written a simple walkthrough application on top of our multiresolution modeling and time-critical rendering libraries. In this application, the culling phase uses a simple bounding box test, the optimization phase uses the algorithm presented in this paper, and rendering is performed in OpenGL, with one positional light, one material per object, Gouraud shading, and one normal and one color per vertex. The application is single-threaded and the high resolution `QueryPerformanceCounter` API is used for all timing measures. In all tests, we have used the parameters $\{w_i(\mathcal{E}, M) = \frac{N^{(\mathrm{pix})}(\mathcal{E},M)}{N^{(\mathrm{maxtri})}(M)}, \alpha = 1\}$ for the degradation heuristic. A videotape demonstrating the system with recordings of live sequences is available [34].

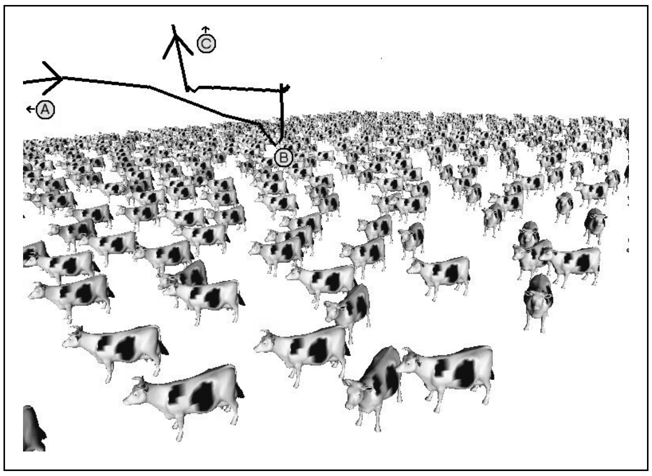

Fig. 6. **Scene walkthrough environment**. Test scene at full resolution contains 529 objects for a total of 3,070,316 triangles. The camera is moving from point A to point C along the path colored in black, always looking towards point B.

7.2.1 Cost model coefficients

The cost model coefficients corresponding to the rendering environment used in the benchmark application where determined experimentally by rendering sample objects with a variety of sizes and LODs. Table 3 summarizes the values used for the tests.

Description	**Cost model coeff.**	**Value**
Initialization/finalization	$t^{(\text{setup})} + t^{(\text{final})}$	$13889\mu s$
Triangle draw	$t^{(\text{tri})}$	$1.064\mu s/tri$
Pixel fill	$t^{(\text{pix})}$	$0.005\mu s/pix$
Material setup	$t^{(\text{init})}$	$49.14\mu s/obj$

Table 3
Cost model coefficient for benchmark application. Experimental measures on a SGI 320, single 500 MHz Pentium III with 512 Kb L2 cache, 256 Mb RAM, Cobalt graphics. Rendering environment: one positional light, one material per object, Gouraud shading, independent triangles, one normal and one color per vertex

7.2.2 Test environment

We have recorded various parameters during the walkthrough of a test scene containing 529 objects for a total of 3,070,316 polygons (see figure 6). The figure was constructed by randomly distributing over the X-Y plane colored replicas of the *cow* mesh. Images were displayed on a 512x512 window.

The scene has been constructed and the camera path has been established so as to include various extreme viewing configurations, representing typical situation: scene exploration, object inspection, sudden changes of interest.

All objects are initially visible from point A and are progressively exiting from the viewing frustum until point B is reached, where the camera is focusing on one objects, the others remaining on the background. At point B, the camera suddenly changes orientation, and a large number of objects becomes immediately visible. Finally, the camera moves back towards point C, always looking towards the scene.

Without resolution adaptation, rendering times on the machine used for the tests varies from 752 to 3320 milliseconds per frame depending on the number of visible objects.

7.2.3 Experimental results and discussion

The number of potentially visible triangles for each observer viewpoint along the test path, as well as the number of triangles actually rendered to meet a display time constraint of $t^{(\mathrm{desired})} = 100ms$ is presented in the bottom diagram of figure 8. The corresponding frame time statistics are presented in the top diagrame of the same figure. The predicted frame time closely matches the actual measured frame time, validating our cost model assumptions.

The actual frame time is maintained below the desired time even in the presence of large variations in visual complexity. Speedups with respect to full resolution rendering range from 7.5x to 33.2x. Even with a relatively large number of objects, we can see that the optimization time remains relatively small compared to the display time. The tests have been performed using a time constraint for the optimization step of $t^{(\mathrm{opt})} = 7ms$. For large portions of the path, optimization has been completed in a time sensibly inferior to the allocated limit (typically $3ms$), producing an optimal solution and leaving more time for other system tasks. In a more elaborate implementation, a feedback algorithm could be used for the adaptation of $t^{(\mathrm{opt})}$.

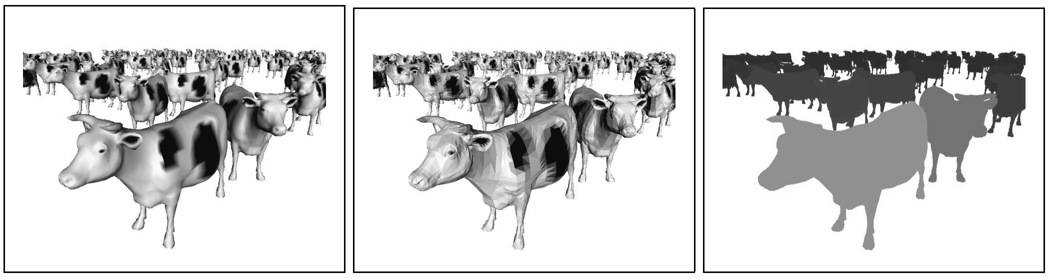

Fig. 7. **Test scene as seen from viewpoint B.** Smooth shading, flat shading, and resolution mapping.

Figure 7 presents the scene as seen from viewpoint B. In the first image, the objects are rendered in smooth shading, as presented to the viewer during interaction. The middle image presents the scene seen from the same viewpoint, with all objects rendered in flat shading to emphasize tessellation details. The last image depicts the resolution chosen for each object, lighter shades representing more detail. In

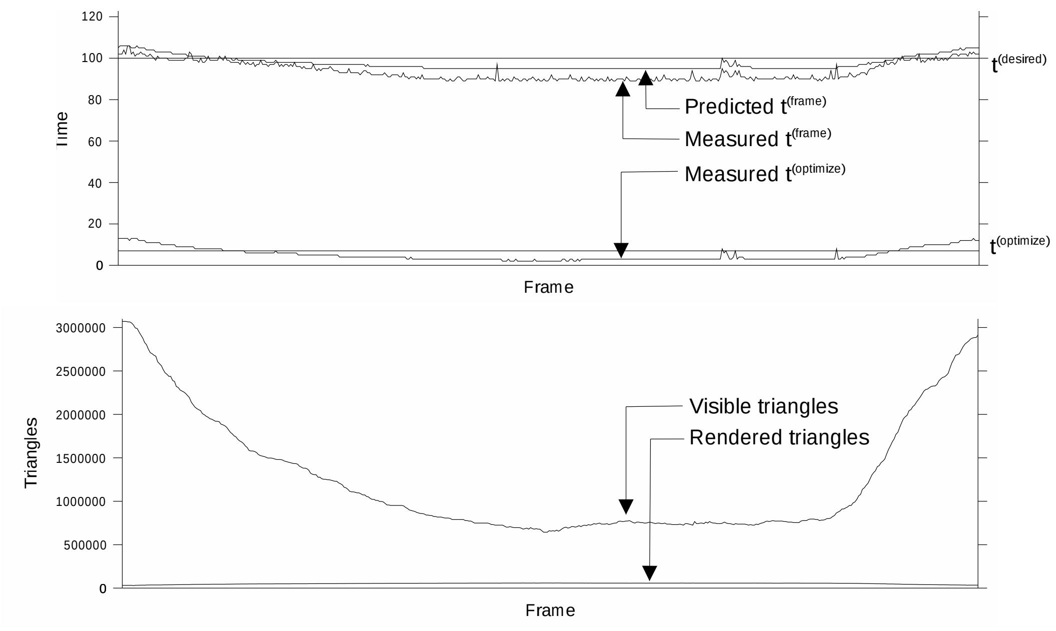

Fig. 8. **Statistics for each observer viewpoint along the test path.** Predicted time closely matches the actual frame time, which is maintained below the targeted $100ms$, even in the presence of wide variations in potential scene complexity.

this particular frame, resolutions have been distributed in the range $[0.376, 0.996]$. The mapping illustrates the effect of the degradation heuristic on the distribution of the polygon budget. As we can see, there is a wide variation between resolutions, but little perceptual differences between levels of detail, which only appear with some evidence in the flat shading figure.

Timing and quality results are significantly better than what can be obtained with the standard discrete LOD approaches.

7.2.4 CAD visualization example

Computer Aided Design (CAD) and architectural models typically contain hundreds of objects and hundreds of thousands to millions of 3D primitives. Our algorithm can be used to display these models at interactive rates. We have tested our experimental walkthrough application on a model of the ATLAS Experiment Pit, a component of the Large Hadron Collider (LHC), a particle accelerator facility of the European Laboratory for Particle Physics (CERN) in Geneva, Switzerland. The full resolution model contains 250K triangles and 985 parts. Figure 9 presents three images of the detector. In the first image, the detector is rendered at full resolution. In the middle image, the resolution of each of the detector's component has been adapted to meet a rendering time constraint of $100ms$ on the machine used for the tests. The overall complexity of the model has consequently been reduced to 110’300 triangles from the original 245’538. The last image depicts the resolution chosen for each object, lighter shades representing more detail. The mapping illustrates that the reduction in detail has been obtained by simplifying the small components in the interior of the detector. Figure 10 shows another frame of the walkthrough, with the user focusing on the details which were reduced in the overall view. Now, construction details in the foreground are clearly visible, and timing constraints are met by reducing the complexity of background objects.

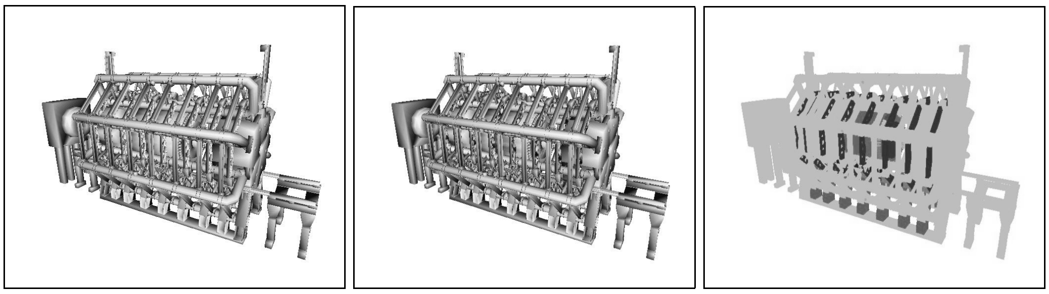

Fig. 9. **ATLAS Experiment Pit, part of the LHC facility at CERN.** Full resolution (245’638 triangles), adaptive resolution (110’300 triangles), and resolution mapping.

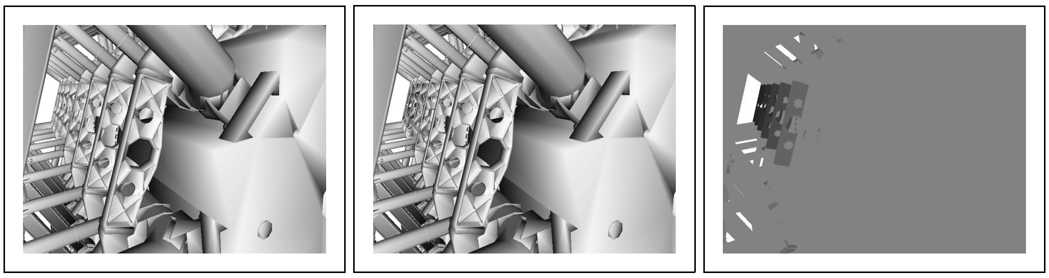

Fig. 10. **ATLAS Experiment Pit detail, part of the LHC facility at CERN.** Full resolution (221’814 triangles), adaptive resolution (116’716 triangles), and resolution mapping.

8 Conclusions and future work

We have described a framework for time-critical rendering of very large and geometrically complex scenes. Our technique relies upon a scene description in which individual scene components are represented as multiresolution triangle meshes. We perform a constrained optimization at each frame to choose the resolution of each potentially visible object that generates the best quality image while meeting timing constraints. Image quality degradation and rendering time are predicted using customizable heuristics.

While our previous work [5] focused on a general solution to the problem, valid as long as the problem remains convex, the present work demonstrates that the particular optimization problem associated to a large class of commonly used heuristics can be solved more efficiently using an active-set strategy, which searches for a solution of the original inequality-constrained optimization problem along the edges and faces of the feasible set by solving a sequence of equality-constrained problems. By exploiting the problem structure, Lagrange multiplier estimates and equality constrained problem solutions are computed in linear time. An important area for future work is the development and validation of perceptually accurate image degradation measures, as well as of approximations leading to simplified numerical solution methods applicable in a time-critical setting.

The applicability of our approach depends on the ability to represent a mesh with an arbitrary number of triangles and to traverse a mesh structure at an arbitrary resolution in a short predictable time. A data structure satisfying these criteria has been described. The current data structure only supports view-independent resolution rendering, which is appropriate for scenes composed of many small scale objects. We are working on extending the data structure to support high-speed variable resolution traversals. In this case, the optimization algorithm will assign a polygon budget to each of the visible objects based on timing constraints, but each object will autonously decide how to distribute this budget at each frame. This approach will improve the visual quality of scenes composed of large objects with linked pieces. Currently, selecting different resolutions for the pieces may lead to cracks in the overall assembly, a problem that can be avoided by treating each group of linked pieces as a single multiresolution object.

Our experimental results demonstrate that the presented algorithms and data structures provide low memory overhead and smooth level-of-detail control, and guarantee, within acceptable limits, a uniform, bounded frame rate even for widely changing viewing conditions. The system enables the handling of scenes totaling millions of polygons and hundreds of independent objects on a standard graphics PC.

We believe that ultimately a time-critical rendering system should combine algorithms such as ours with occlusion culling [35], image based rendering [36] and other rendering acceleration methods [37]. The system should automatically partition the scene, choosing the most appropriate methods to use for different sets of objects based on cost/quality consideration and algorithm constraints.

Acknowledgments

We would like to thank Hugues Hoppe, Michael Garland, and the Stanford graphics group for making benchmark datasets available to the research community. The ATLAS dataset is courtesy of CERN, Geneva, Switzerland. We also greatly appreciate the help received from Marco Agus and Fabio Bettio in taking experimental measures.

This research is partially sponsored by the European project CAVALCADE (ES-

PRIT contract 26285). We also acknowledge the contribution of Sardinian regional authorities.

References

[1] Paolo Cignoni, Enrico Puppo, and Roberto Scopigno. Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer*, 13(5):199–217, 1997. ISSN 0178-2789.

[2] Hugues Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings IEEE Visualization'98*, pages 35–42. IEEE, 1998.

[3] Lichan Hong, Shigeru Muraki, Arie Kaufman, Dirk Bartz, and Taosong He. Virtual voyage: Interactive navigation in the human colon. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 27–34. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[4] Patrice Torguet, Olivier Balet, Enrico Gobbetti, Jean-Pierre Jessel, Jérôme Duchon, and Eric Bouvier. Cavalcade: A system for collaborative prototyping. In Gérard Subsol, editor, *Proc. International Scientific Workshop on Virtual Reality and Prototyping*, pages 161–170, June 1999. Conference held in Laval, France, June 3–4.

[5] Enrico Gobbetti and Eric Bouvier. Time-critical multiresolution scene rendering. In *Proceedings IEEE Visualization*, pages 123–130, Conference held in San Francisco, CA, USA, October 1999. IEEE Computer Society Press.

[6] Open Inventor Architecture Group. *Open Inventor C++ Reference Manual: The Official Reference Document for Open Systems*. Addison-Wesley, Reading, MA, USA, 1994.

[7] VRML 97, International Specification ISO/IEC IS 14772-1, December 1997.

[8] Paul S. Heckbert and Michael Garland. Multiresolution modeling for fast rendering. In *Proc. Graphics Interface '94*, pages 43–50, Banff, Canada, May 1994. Canadian Inf. Proc. Soc. URL: `http://www.cs.cmu.edu/~ph`.

[9] Michael Reddy, Benjamin A. Watson, Neff Walker, and Larry F. Hodges. Managing level of detail in virtual environments: a perceptual framework. *Presence: Teleoperators and Virtual Environments*, 6(6):658–666, 1997.

[10] John Rohlf and James Helman. IRIS performer: A high performance multiprocessing toolkit for real–Time 3D graphics. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 381–395. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[11] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 247–254, August 1993.

[12] Mathias Wloka. Lag in multiprocessor virtual reality. *Presence: Teleoperators and Virtual Environments*, 4(1):50–63, September 1995.

[13] Ashton E. W. Mason and Edwin H. Blake. Automatic hierarchical level of detail optimization in computer animation. *Computer Graphics Forum*, 16(3):191–200, August 1997. Proceedings of Eurographics '97. ISSN 1067-7055.

[14] Leila De Floriani, Paola Marzano, and Enrico Puppo. Hierarchical terrain models: survey and formalization. In *Proceedings SAC'94*, pages 323–327, Phoenix (AR), March 1994.

[15] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96 Proc.*, pages 99–108, Aug. 1996. URL: `http://www.research.microsoft.com/research/graphics/hoppe/papers.html`.

[16] Julie C. Xia and Amitabh Varshney. Dynamic view-dependent simplification for polygonal models. In *IEEE Visualization '96*. IEEE, October 1996. ISBN 0-89791-864-9.

[17] Julie C. Xia, Jihad El-Sana, and Amitabh Varshney. Adaptive Real-Time Level-of-Detail-Based Rendering for Polygonal Models. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):171–183, April 1997.

[18] David Luebke and Carl Erikson. View-dependent simplification of arbitrary polygonal environments. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 199–208. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[19] Michael Reddy. The Development and Evaluation of a Model of Visual Acuity for Computer-Generated Imagery. technical report ECS-CSG-30-97, Dept. of Computer Science, Edinburgh University, February 1997.

[20] Lewis Hitchner. Virtual planetary exploration: A very large virtual environment. *Virtual Reality for Visualisation (IEEE Visualisation)*, pages 148–163, 1993.

[21] Toshikazu Ohshima, Toshiyazu Yamamoto, and Hideyuki Tamura. Gaze-directed adaptive rendering for interacting with virtual space. In *IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 103–110. IEEE, October 1996.

[22] Laszlo Neuman, Kresimir Matkovic, and Werner Purgathofer. Perception based color image difference. *Computer Graphics Forum*, 17(3), September 1998.

[23] Roger Fletcher. *Practical Methods of Optimization. Volume 2: Constrained Optimization*. Wiley, New York, NY, USA, 1981.

[24] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, 1981.

[25] Hugues Hoppe. Efficient implementation of progressive meshes. *Computers and Graphics*, 22(1):27–36, January 1998.

[26] André Guéziec, Gabriel Taubin, Bill Horn, and Francis Lazarus. A framework for streaming geometry in VRML. *IEEE Computer Graphics and Applications*, 19(2):68–78, March/April 1999.

[27] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *SIGGRAPH '93 Proc.*, pages 19–26, Aug. 1993. URL: `http://www.research.microsoft.com/research/graphics/hoppe/papers.html`.

[28] André Guéziec. Surface simplification with variable tolerance. In *Second Annual Intl. Symp. on Medical Robotics and Computer Assisted Surgery (MRCAS '95)*, pages 132–139, November 1995.

[29] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 97 Proc.*, pages 209–216, August 1997. URL: `http://www.cs.cmu.edu/~garland/quadrics`.

[30] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *Proceedings IEEE Visualization'98*, pages 279–286. IEEE, 1998.

[31] Hugues Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *Proceedings IEEE Visualization*, pages 59–66, Conference held in San Francisco, CA, USA, October 1999. IEEE Computer Society Press.

[32] Lief Kobbelt, Swen Campagna, and Hans-Peter Seidel. A general framework for mesh decimation. In Wayne Davis, Kellogg Booth, and Alain Fourier, editors, *Proceedings of the 24th Conference on Graphics Interface (GI-98)*, pages 43–50, San Francisco, June18–20 1998. Morgan Kaufmann Publishers.

[33] Eric Bouvier and Enrico Gobbetti. TOM – totally ordered mesh: a multiresolution triangle mesh structure for time-critical graphics. Technical report, CRS4, Center for Advanced Studies, Research, and Development in Sardinia, Cagliari, Italy, September 1999.

[34] Enrico Gobbetti and Eric Bouvier. Time-critical multiresolution scene rendering. Technical Report VIDEOTAPE 99/84, CRS4, Center for Advanced Studies, Research, and Development in Sardinia, Cagliari, Italy, October 1999.

[35] Hansong Zhang, Dinesh Manocha, Thomas Hudson, and Kenneth E. Hoff III. Visibility culling using hierarchical occlusion maps. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 77–88. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[36] Daniel Aliaga and Anselmo Lastra. Automatic image placement to provide a guaranteed frame rate. In *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, pages 307–316. ACM SIGGRAPH, Addison Wesley, August 1999.

[37] Daniel Aliaga, Jonathan Cohen, Andrew Wilson, Hansong Zhang, Carl Erikson, Kenneth Hoff, Thomas Hudson, Wolfgang Stuerzlinger, Eric Baker, Rui Bastos, Mary Whitton, Fred Brooks, and Dinesh Manocha. A framework for the real-time walkthrough of massive models. Technical Report TR98-013, Department of Computer Science, University of North Carolina - Chapel Hill, March 26 1998.