Computer-Aided Design 79 (2016) 48–59

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

# Interactive inspection of complex multi-object industrial assemblies

O. Argudo, I. Besora, P. Brunet*, C. Creus, P. Hermosilla, I. Navazo, À. Vinacua

*ViRVIG Research Group, Technical University of Catalonia, Spain*[1]

ARTICLE INFO

*Article history:*
Received 10 February 2015
Accepted 22 June 2016

*Keywords:*
Computer graphics
Methodology and techniques

ABSTRACT

The use of virtual prototypes and digital models containing thousands of individual objects is commonplace in complex industrial applications like the cooperative design of huge ships. Designers are interested in selecting and editing specific sets of objects during the interactive inspection sessions. This is however not supported by standard visualization systems for huge models. In this paper we discuss in detail the concept of rendering front in multiresolution trees, their properties and the algorithms that construct the hierarchy and efficiently render it, applied to very complex CAD models, so that the model structure and the identities of objects are preserved. We also propose an algorithm for the interactive inspection of huge models which uses a rendering budget and supports selection of individual objects and sets of objects, displacement of the selected objects and real-time collision detection during these displacements. Our solution – based on the analysis of several existing view-dependent visualization schemes – uses a Hybrid Multiresolution Tree that mixes layers of exact geometry, simplified models and impostors, together with a time-critical, view-dependent algorithm and a Constrained Front. The algorithm has been successfully tested in real industrial environments; the models involved are presented and discussed in the paper.

## 1. Introduction

A number of algorithms for real-time visualization of huge digital 3D models have been proposed. While they are well suited for many applications, they do not meet the present user requirements in some industrial applications. Complex virtual prototypes are essential in many industrial endeavors involving large models, like in the automotive, aeronautic and ship-building industries. Moreover, the high cost of many of these designs, sometimes destined to be built only once, makes the use of physical prototypes unfeasible. Also, these models contain thousands of individual objects. Instead of relying only on standard visualization systems, designers are interested in addressing individual objects and specific sets of objects. We have identified these requirements for inspection applications in industrial design of complex multi-object assemblies:

- The View-Dependent Visualization algorithm should guarantee a certain frame-rate with good image quality.
- The System must support the selection of individual objects and of hierarchies of objects, during the navigation, to access information of these objects or to annotate and modify them.
- Limited scene editing (including displacement of the selected objects) and real-time collision detection during scene editing must be supported.

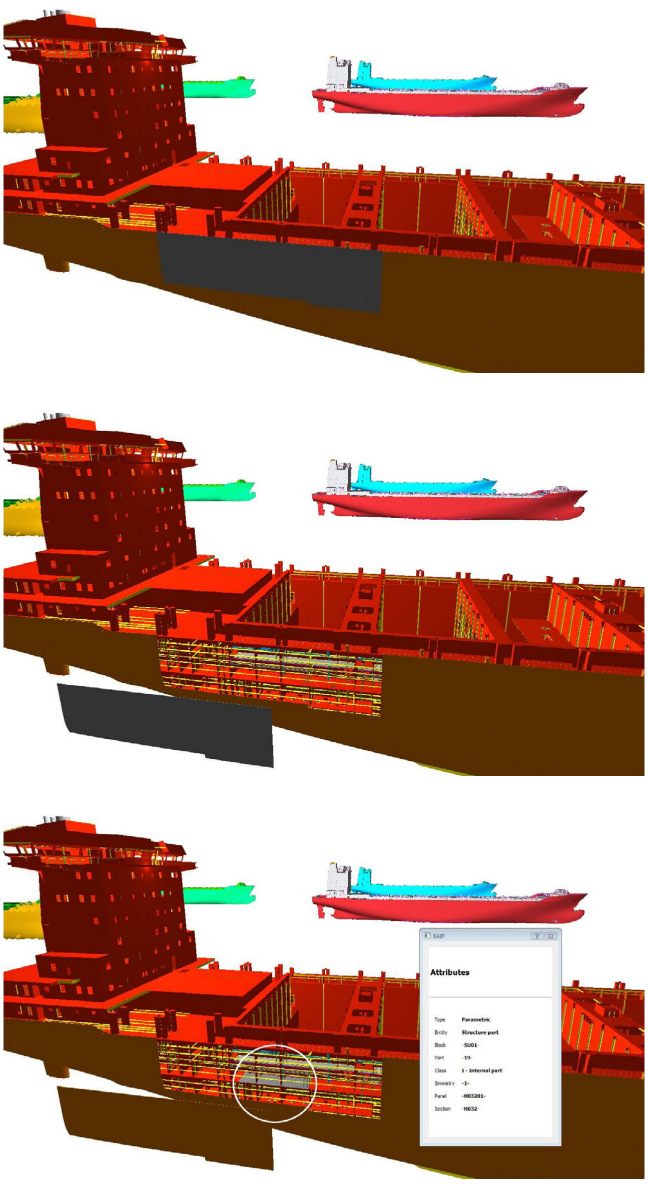
As far as we know, no present algorithm fulfills all of the above requirements. Furthermore, previous solutions often resort to substitutions, modifications or simplifications of the geometry that blur the scene structure and the individual objects. Instead, we are interested in preserving the design intent, and the structure of the design tree of the original CAD model, that is meaningful to the users. We show that we can achieve these goals while maintaining sustained frame rates for very large models. Fig. 1 shows the edition and annotation of such a model. These editions and annotations are saved for later incorporation into the CAD model if appropriate. Building on previous contributions in the literature, we propose a solution that preserves CAD hierarchies and object identities while allowing simple interactions. The main contributions of this paper are:

- A formal discussion of the front concept in multiresolution trees, and the characterization of the properties required for time-critical rendering.
- An object-aware scene simplification and multiresolution scheme that results in a multilayered, multiresolution tree

* Corresponding author.
*E-mail addresses:* oscar.argudo.medrano@gmail.com (O. Argudo), onesvenus@gmail.com (I. Besora), pere@lsi.upc.edu (P. Brunet), crazycraft@gmail.com (C. Creus), pit2500@gmail.com (P. Hermosilla), isabel@lsi.upc.edu (I. Navazo), alvar@lsi.upc.edu (À. Vinacua).

[1] http://virvig.eu.

http://dx.doi.org/10.1016/j.cad.2016.06.005
0010-4485/© 2016 Elsevier Ltd. All rights reserved.



| | |
|---|---|
| Type | Parametric |
| Entity | Structure part |
| Block | -SU01- |
| Part | -19- |
| Class | I - Internal part |
| Simmetry | -1- |
| Panel | -H03201- |
| Section | -H032- |

**Fig. 1.** The model used in the tests. The top image shows a portion of the hull selected by the user in gray. In the middle image the user has moved the selected portion of the hull, partially revealing the complexity of the model inside. The bottom image shows data associated to an object, that the user may annotate. Notice the ships in the background, which are rendered using omnidirectional relief impostors.

with cost and benefit functions per node, which is monotonic by construction. Layers mix geometry and impostors. The layered scheme is based on the results of an evaluation of user perception of several hierarchical representations.

- Support for the selection of individual objects and sets of objects, displacement of the selected objects and real-time collision detection during these displacements while in the interactive navigation, thanks to the object-aware nature of the representation.
- A time-critical, view-dependent visualization algorithm with constrained front update based on a greedy optimization per frame, usable in commodity hardware.

The rest of the paper is organized as follows. Section 2 reviews prior work. After an analysis of front-based rendering algorithms in Section 3, an overview of the algorithm is presented in Section 4, whereas Sections 5 and 6 detail the scene tree generation algorithm and its visualization. Section 7 discusses several results on an example scene, and Section 8 presents the main conclusions and outlines future research directions.

## 2. Previous work

In this section we discuss only a selection of the algorithms most relevant to our work. For an extensive survey, the reader can refer to [1] or [2]. These algorithms are usually based on the generation, in a preprocess step, of a scene data structure which represents the scene model at different levels of detail. During the interactive visualization, a suboptimal set of nodes is computed and rendered at each frame.

Multiresolution Trees are well-known data structures that represent scenes and assemblies at multiple resolutions. Multiresolution Trees can be binary [3], quaternary [4] or octal [5], and based on either a spatial subdivision [6] or a scene subdivision [4]. Binary subdivision structures like Kd-trees have been widely used because of their splitting flexibility [7].

Multiresolution Trees creation algorithms work in two steps. In the first step, the Tree structure is generated in a top-down way by recursively distributing the scene geometry from parent to son nodes. At the end of this first step, the algorithm has distributed the scene geometry among all tree leaf nodes, and leaf nodes have a size not exceeding a predefined value. The second step operates bottom-up by merging node information and simplifying the union of the informations in their son nodes in a way such that all internal nodes have a size again within the chosen limit.

The paper from Funkhouser and Sequin [8] was seminal in this area. Their scene structure consisted of a simple list (or array) of objects, which sufficed for scenes of only moderate complexity. The preprocess consisted in the computation of a 2D array of object representations, storing each of the $N$ objects at $M$ different levels of detail. Standard simplification algorithms were used for this purpose. In the kernel of the visualization scheme, cost and benefit functions were defined and computed for each object and for each of its $M$ LODs. The cost (time to render the atomic object) was considered to be constant, and computed in the preprocess step. Benefit, however, was dynamic, depending on the camera position and on how the object was projected on the viewport. Funkhouser and Sequin further defined a constrained optimization problem per frame: the goal was to maximize the total benefit per frame, with the constraint that the total cost of the rendered primitives did not exceed the rendering budget. For this purpose, they proposed a greedy front update algorithm, applied at each frame. The object with the maximum benefit to cost ratio was refined, while one or more objects having the lowest ratio were coarsened to keep the total cost below the budget. Based on this work, Gobbetti and Bouvier [9] proposed a solution for this optimization using Lagrange multipliers.

The term *view-dependent visualization algorithms* was coined to refer to algorithms based on a hierarchy of objects (the multiresolution tree) and a dynamic rendering front that adapts itself during the visualization. View-dependent algorithms include FarVoxels, LayeredPointClouds, TetraPuzzles, Quick-VDR and others. We briefly review these algorithms in the next paragraphs. In them, the front update is based on a suitable benefit function, but in all these algorithms no information about the frame computing and rendering time is taken into account.

Far Voxels [3] uses hybrid multiresolution Kd-trees, with triangle strips of the original scene model in leaf nodes and approximates volume representations in internal tree nodes. These nodes are discretized into a fixed number of around 16K voxels. Voxels contain parameterized direction-dependent material models, generated by sampling the geometry in the node along rays emanating from 256K viewpoints around it. The rendering algorithm uses