Visual Informatics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Visual Informatics

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

# Prediction-based load balancing and resolution tuning for interactive volume raycasting

Valentin Bruder *, Steffen Frey, Thomas Ertl

*University of Stuttgart, Germany*

ARTICLE INFO

*Article history:*
Received 11 August 2017
Accepted 4 September 2017
Available online xxxx

*Keywords:*
Volume raycasting
Performance prediction
Load balancing

ABSTRACT

We present an integrated approach for real-time performance prediction of volume raycasting that we employ for load balancing and sampling resolution tuning. In volume rendering, the usage of acceleration techniques such as empty space skipping and early ray termination, among others, can cause significant variations in rendering performance when users adjust the camera configuration or transfer function. These variations in rendering times may result in unpleasant effects such as jerky motions or abruptly reduced responsiveness during interactive exploration. To avoid those effects, we propose an integrated approach to adapt rendering parameters according to performance needs. We assess performance-relevant data on-the-fly, for which we propose a novel technique to estimate the impact of early ray termination. On the basis of this data, we introduce a hybrid model, to achieve accurate predictions with minimal computational footprint. Our hybrid model incorporates aspects from analytical performance modeling and machine learning, with the goal to combine their respective strengths. We show the applicability of our prediction model for two different use cases: (1) to dynamically steer the sampling density in object and/or image space and (2) to dynamically distribute the workload among several different parallel computing devices. Our approach allows to reliably meet performance requirements such as a user-defined frame rate, even in the case of sudden large changes to the transfer function or the camera orientation.



## 1. Introduction

Volume visualization is a widely used tool for visualization of measured and simulated data in numerous different areas such as physics, engineering, biology and many more. By enabling users of visualization applications to dynamically interact with the volume data, additional insight beyond the initial focus may be gained. Thereby, classic user interactions are adjustments to the transfer function (which maps density values to color) as well as changes to the camera configuration (e.g., rotation and zooming). There are typically two main factors that contribute to a satisfying user experience during interactive exploration of volume data sets: low response times and a high rendering quality. While the latter can be achieved by employing a high sampling of the data set, low latencies and high frame rates are crucial for response times. In the context of the recent revive of virtual reality for scientific applications (Laha et al., 2012), maintaining high and stable frame rates as well as low latencies gains even more importance. In those applications, variable frame rates often tend to cause unpleasant side effects, such as cybersickness, for many users.

To be able to gain interactive frame rates for volume visualizations on workstations, GPUs are often used to accelerate the computation and rendering. Besides the hardware used for computation, interactively changed parameters (i.e., transfer function and camera configuration) have a significant impact on rendering performance. In order to accomplish constant interactivity, those variations in performance need to be accounted for, especially in challenging cases with significant changes between frames (e.g., switching to a different transfer function). One way of absorbing such effects is to adapt the sampling density in object or image space. However, in the case of an interactive application, the basis for this adaption has to be some kind of assessment of how the performance will evolve in upcoming frames (after potentially big changes) in order to avoid unpleasantly long response times or jerky motions.

Predicting performance of volume rendering on parallel hardware is a challenging task because of the involved complexity. Numerous factors have a significant, non-obvious impact on performance. For instance, this includes the hardware employed for

* Corresponding author.
*E-mail address:* valentin.bruder@visus.uni-stuttgart.de (V. Bruder).
Peer review under responsibility of Zhejiang University and Zhejiang University Press.

http://dx.doi.org/10.1016/j.visinf.2017.09.001

parallel computation, as well as the specific algorithm and parameter configuration that may be changed during runtime.

We propose our method to dynamically predict performance of a volume raycasting application that uses popular acceleration techniques. To show the usability of our technique, we present two use cases that are based on our frame time prediction. In the first one, we use the predictions to dynamically adjust the sampling rate of the volume rendering process to reliably meet a user-defined frame target (i.e., interactive frame rates). Thereby, we can adjust the sampling rate in ray space (integration step size along rays) as well as in image space (image resolution, i.e. the number of rays). As a second use case, we dynamically distribute the computational load among multiple heterogeneous GPUs and balance this load according to our predicted frame execution times.

In the following section, we give an overview on related work (Section 2), afterwards we discuss what we consider to be the main contributions of our work.

- We present our general approach for performance prediction and the tuning of sampling rate in image and ray space (Section 3). It is based on the following components:
- assessing performance-critical numbers of raycasting acceleration techniques, including the impact of early ray termination (ERT) and empty space skipping (Section 4);
- predicting on the fly the execution time of upcoming frames using a hybrid performance model (Section 5);
- and balancing the computational load among multiple devices in real-time as well as steering rendering quality towards a user-defined frame rate (Section 6).

To the best of our knowledge, on-line prediction of volume rendering performance has not been published before our conference paper (Bruder et al., 2016). This work is an extended version of that paper. In detail, the extensions compared to our conference paper are:

- load balancing between different GPUs as an additional use case,
- resolution adjustment in image space, also combined with tuning in ray space,
- and minor improvements and additions, such as local illumination.

We present and discuss results in Section 7 and conclude our work in Section 8.

## 2. Related work

*Volume visualization and frame rate adaption.* Volume visualization has been a core subject in scientific visualization research for several decades. In recent times, raycasting has turned out to be one of the mostly used techniques, with its parallel nature supporting GPU and distributed implementations (Engel et al., 2006). Salama et al. (2009) give an overview on basic volume rendering techniques, thereby focusing on illumination and acceleration techniques that we use as well.

Many works have focused on distributed volume rendering, due to the computational requirements posed by high resolution data sets. The current state of the art in GPU techniques for interactive large-scale volume visualization is discussed by Beyer et al. (2015). Especially for distributed rendering, load balancing plays an important role (Ma et al., 1994; Marchesin et al., 2006). In this context, Fogal et al. (2010) discuss and investigate different algorithms for load balancing in their work, while Müller et al. (2006) demonstrate that zooming on parts of volume data sets critically impairs load balance during distributed rendering. To counter this effect, they dynamically reorganize the data distribution in their cluster. The decision on when to move data to another node is based on a simple cost function and the actual load of the previous frame. While such cost functions as basis for load balancing typically work well in the case of gradual changes, sudden changes (e.g. due to a rapidly adjusted transfer function) cannot be handled adequately, inducing significant load-imbalance and performance drops.

Rendering systems typically fix either image quality or frame rate during user interaction. There is some work on techniques designed to keep stable frame rates for image-based rendering, including what we have done as well as one application of our prediction model. Shen and Johnson (1994), Qu et al. (2000) and others re-use pixel values from previous frames and use that to achieve stable frame rates. Wong and Wang (2014) have the same goal for real-time rendering applications but use an open-loop approach of the image generation process underpinned by estimations of its constituents. Using artificial neural networks and fuzzy models, as well as detailed descriptions of distinct rendering processes, they relate inputs and outputs in a non-linear model. In contrast, Woolley et al. (2003) take a more simple approach by using metrics, based on image space distances to steer progressive raytracing. Frey et al. (2014) use a progressive approach to steer the volume visualization process, thereby focusing on resource management, response times and sampling errors. Compared to our approach however, none of those techniques adapt the frame-rate-based on an on-the-fly prediction of the execution time.

*Performance prediction.* There is a large amount of research in the area of application performance prediction and modeling for parallel architectures. However, it is mostly limited to the fields of system architecture and high-performance computing, whereas the research in (interactive) visual computing, which has its own characteristics and challenges, is comparably sparse. Various different approaches have been proposed for performance modeling, including performance skeletons (Sodhi et al., 2008), regression (Barnes et al., 2008), genetic algorithms (Tikir et al., 2007), and machine learning (Lee et al., 2007). Those approaches primarily target performance prediction in large-scale (HPC) systems. However, visual computing applications have different characteristics than those systems in that they typically rely heavily on interaction. The data that is being used for performance modeling typically stems from either specific hardware characteristics, such as (parallel) computational operations per second and memory bandwidth; or from empirical measurements, such as frame execution times and performance counters. Using the latter combined with an analytical model has been defined as a "semi-empirical" model (Hoefler et al., 2011). In our approach, we employ a machine learning model to learn from execution time measurements and combine this with an analytical model, based on known properties of the volume raycasting algorithm. Therefore, we consider it to be such a semi-empirical model.

There exist various off-line performance modeling tools for GPGPU, which has many similarities to GPU volume rendering. An overview of the landscape is given by Madougou et al. (2016). Amarís et al. (2016) compare different machine learning models, namely linear regression, support vector machines and random forests with a BSP-based analytical model for the task of GPU execution time prediction.

In contrast, work on real-time rendering or scientific visualization incorporating real-time performance prediction is comparably sparse. The proposed techniques mainly focus either on performance models for the visualization pipeline (Bowman et al., 2004) or on object-order rendering algorithms (Wimmer and Wonka, 2003; Tack et al., 2004). Ganestam and Doggett (2012) perform auto tuning for interactive ray tracing, thereby using an analytical GPU architecture model as a basis. Compared to our work, their approach mainly focuses on ray-tracing and while their model incorporated caching effects to some degree, other hardware-level

Please cite this article in press as: Bruder V., et al., Prediction-based load balancing and resolution tuning for interactive volume raycasting. Visual Informatics (2017), http://dx.doi.org/10.1016/j.visinf.2017.09.001.