Computer-Aided Design 58 (2015) 132–140

Contents lists available at ScienceDirect

Computer-Aided Design

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

Efficient direct rendering of deforming surfaces via shared subdivision trees☆

Fuchang Liu [a,c,*], Tobias Martin [b], Sai-Kit Yeung [c], Markus Gross [b]

[a] Hangzhou Normal University, Hangzhou, China
[b] ETH Zurich, Switzerland
[c] Singapore University of Technology and Design, Singapore

HIGHLIGHTS

- We present Shared Subdivision Trees (SST) to rasterize implicit surfaces on GPUs.
- We address the problem of efficiently rendering implicit surfaces which undergo a nonlinear deformation throughout the rendering process.
- We map Shared Subdivision Trees well to parallel computing platforms such as CUDA.

ARTICLE INFO

Keywords:
Isosurface visualization
GPU rendering
Computational geometry and object modeling

ABSTRACT

In this paper, we present a subdivision-based approach to rasterize implicit surfaces embedded in volumetric Bézier patches undergoing a nonlinear deformation. Subdividing a given patch into simpler patches to perform the surface rasterization task is numerically robust, and allows guaranteeing visual accuracy even in the presence of geometric degeneracies. However, due to its memory requirements and slow convergence rates, subdivision is challenging to be used in an interactive environment. Unlike previous methods employing subdivision, our approach is based on the idea where for a given patch only one subdivision tree is maintained and shared among pixels. Furthermore, as the geometry of the object changes from frame to frame, a flexible data structure is proposed to manage the geometrically varying Bézier patches. The resulting algorithm is general and maps well to parallel computing platforms such as CUDA. We demonstrate on a variety of representative graphics and visualization examples that our GPU scheme scales well and achieves up to real-time performance on consumer-level graphics cards by guaranteeing visual accuracy.

1. Introduction

Deforming *B*-spline volumes with embedded scalar fields frequently occur in a variety of computer graphics and engineering applications. For instance, in free-form deformation [1] an implicit surface of a scalar field is deformed by deforming the geometry of its associated *B*-spline bounding volume. *B*-spline volumes are also a fundamental primitive in isogeometric analysis [2] where they are used to represent the geometry of a physical object. Physical analysis is applied directly to the *B*-spline volume representation, where the analysis result is represented as an associated attribute. Depending on the simulation scenario, the geometry of the representation may undergo shape changes. For instance, an elastic body deforms when external forces are applied, where stress is an attribute of the deforming object.

In this paper, we address the problem of efficiently rendering implicit surfaces which undergo a nonlinear deformation throughout the rendering process. The deformation is performed on a volumetric representation, which can be converted into a set of Bézier volumes. While the topology of the deforming surface may remain the same throughout the animation, its scale may change nonuniformly from frame to frame. Extraction- and sampling-based methods are not only challenged by the changing surface properties and the dynamic volumetric deformations, but also by the reconstruction of all the features present in the implicit surface (for instance, see thin features in Fig. 1).

☆ This paper has been recommended for acceptance by Dr. Vadim Shapiro.

* Correspondence to: Digital Media and HCI Research Center, Hangzhou Institute of Service Engineering, Hangzhou Normal University, Yuhang Qu Haishu Road 58, Hangzhou 311121, China.

E-mail addresses: liufububai@gmail.com, liufu@ewha.ac.kr (F. Liu), martint@inf.ethz.ch (T. Martin), saikit@sutd.edu.sg (S.-K. Yeung), grossm@inf.ethz.ch (M. Gross).

http://dx.doi.org/10.1016/j.cad.2014.08.005
0010-4485/

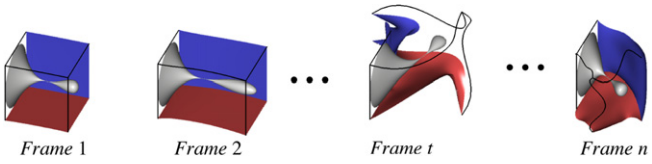

Fig. 1. Deformation sequence of an isosurface of a tri-quintic algebraic function. Methods to extract the implicit surface, such as Marching Cubes, are challenged because of the nonlinear distortions and thin surface features.

Given this scenario, subdividing the Bézier patches into simpler patches is key. Traditionally, a subdivision-based approach builds a subdivision tree for each pixel, where sub-patches in the tree are kept and only subdivided further if they potentially contain a piece of the surface overlapping with the pixel. In the limit, the leafs of the subdivision tree constitute to that piece of the surface, passing through the pixel. However, subdivision is computationally expensive and only converges linearly to a solution. Therefore, instead of subdividing the patch to pixel size, it is only subdivided until all intersections can be determined using the Newton–Raphson method. Then, the local subdivision tree is discarded. In order to reduce the size of local sub-division trees, a pre-subdivision stage [3] is employed: before rendering takes place, the patch is first subdivided into a set of simpler patches. The main drawback with this strategy is that a hierarchical data structure has to be maintained, and has to be rebuilt whenever the geometry changes. This poses additional challenges to map such a scheme efficiently to GPU. Furthermore, the hierarchy is view independent, i.e., it may consist of too many (or too few) levels, and also patches which are occluded from the current view.

Main contribution: we present a novel concept for GPU, called *Shared Subdivision Trees* (SST), to rasterize implicit surfaces represented by multiple Bézier patches undergoing a nonlinear deformation during rendering. Conceptually, as illustrated in Fig. 2, for a given patch, a single subdivision tree is maintained which is shared among pixels. A patch in the subdivision tree is only subdivided further if requested by a screen pixel, which eliminates the redundant subdivision work. The proposed method can be seen as moving the pre-subdivision stage into the rendering stage, where the subdivision tree of a given patch is built by exploiting the high parallelism of current GPUs. The visibility problem is solved by a conventional sweep and prune method which allows to handle datasets as they occur in practice. We demonstrate on a variety of representative examples that our scheme is computationally efficient and yields interactive and for some examples even real-time frame rates. In addition, we verify that our scheme scales well with respect to memory requirement and rendering speed.

The outline of this paper is as follows. After discussing the related work in Section 2, the mathematical framework used for this work is introduced in Section 3. The proposed algorithm is described in Section 4 and its implementation is discussed in Section 5. Then, three applications and associated studies are presented in Section 6.1. Finally, we evaluate the efficiency of our proposed method in Section 6.2, and conclude the paper in Section 7.

2. Previous work

Direct rendering on uniform grids: there is a vast body of work to directly render implicit surfaces of volumetric scalar fields. A variety of highly efficient methods exist when the scalar field in world space can be described by a trivariate or piecewise trivariate polynomial. Methods in this category date back to the work by Rockwood [4] which computes univariate contours from a Bézier volume. Roots of these contours correspond to points on the isosurface. A related approach presented in [5] converts the algebraic function along the ray into Bernstein form to efficiently and robustly determine all intersections between the ray and the implicit surface. Knoll et al. [6] present an approach to render implicit surfaces of algebraic functions using interval arithmetic achieving real-time frame rates. Approaches falling into the same category, but which are based on sampled volume data are [7–9]. More recently, Liu et al. [10] present an isosurface rasterization approach exploiting cache coherency to further speed up rendering.

Due to the polynomial nature of the scalar field, the scalar field along a viewing ray can be represented in closed form. This property results in a lower memory footprint making it easier to solve the problem on the GPU. However, in our scenario, the volume embedding the scalar field undergoes a nonlinear deformation (Fig. 1 and for more examples Figs. 7 and 8). In this case, the scalar field consists of a highly nonlinear term which makes it impossible to express the scalar function along the ray analytically. Because of that, it is unclear how to extend the methods above to also work efficiently in this scenario. In this paper, we present an efficient rasterization method which robustly renders isosurfaces embedded in deformed objects.

Extraction-based methods: among the first methods to render scalar data embedded in deformed volumes is [11]. The method is based on an isosurface sampling approach similar to [12]: points are iteratively projected onto the isosurface and the surface is rendered using a point-based rendering system such as [13]. High visual accuracy can be achieved following this strategy. However, determining a point sampling which guarantees visual accuracy is difficult. These methods generally tend to oversample the implicit surface in order to reconstruct thin or smaller features as the one shown in Fig. 2. Extraction based methods face similar problems. Such an approach first extracts the implicit surface using a method such as Marching Cubes [14], or Marching Tetrahedra [15]. Then, the extracted triangle mesh approximating the smooth implicit surface is rendered. While extraction can be executed very efficiently, the smooth representation first has to be discretized into a linear format. This requires the sampling of the volumetric patch. Efficiently generating a sampling such that the extracted triangle mesh is accurate up to image resolution is an open problem. Note that, all these challenges are amplified when the implicit surface undergoes a nonlinear deformation every frame.

Ray-sampling-based methods: given a ray passing through a pixel and a deformed volumetric patch with embedded scalar field, an intersection of the ray with the implicit surface is computed in two steps: (1) determine the entry and exit point of the ray into the patch; and (2) perform root finding on these bounds to identify where the ray intersects the implicit surface. Since the scalar function cannot be written in closed form, as discussed above, the latter step requires sampling of the scalar field along the ray, where for each sample, the inverse function has to be evaluated using a numerical method. For instance, [16,17] adaptively sample this function to compute a polynomial interpolant based on a Legendre basis which can be arbitrarily close to the solution. Similarly, [18] present a GPU ray-caster using a frequency based adaptive sampling approach to account for high variations along the ray. To achieve interactive frame rates, the method stores the volumetric patches in a grid.

Ray-sampling methods, such as the ones discussed above, assume that the mapping between the reference element to the deformed patch is bijective. However, this is often not the case in practice. For instance, in physically based animation [19], patches undergoing a nonlinear deformation may self-intersect or even invert. This results in zero Jacobians, where at these locations the mapping is not bijective and therefore, a numerical method such as Newton–Raphson to compute the inverse cannot be used. This type of data presents severe stability and convergence issues for the rendering approaches mentioned above. Furthermore, boundaries