

# A Fast GPU Particle System Approach for Isocontouring on hp-Adaptive Finite Element Meshes

C. A. Pagot<sup>1</sup> J. E. Vollrath<sup>2</sup> J. L. D. Comba<sup>1</sup> D. Weiskopf<sup>2</sup>

<sup>1</sup>Instituto de Informática, Federal University of Rio Grande do Sul, Brazil

<sup>2</sup>Visualization Research Center (VISUS), University of Stuttgart, Germany

## ABSTRACT

The hp-adaptive finite element (FE) method is a discretization scheme which is increasingly finding application in numerical solvers. Most existent visualization tools, however, are based on linear graphic primitives and corresponding low order interpolation schemes. Consequently, pragmatic approaches in such a setting follow a resampling strategy that may introduce a loss of information, prohibitive increase of memory consumption, or additional complex data structures which are likewise difficult to handle for visualization. In this work we advocate the use of a GPU-based particle system for isocontouring of scalar quantities defined on hp-adaptive FE meshes. We present the status of our ongoing research which shows that such an approach fulfills three important requirements: i) preservation of the original data, ii) control over memory consumption and iii) interactive exploration.

**Index Terms:** I.3.1 [Computer Graphics]: Hardware Architecture - Graphics processors—Parallel processing; I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types

## 1 INTRODUCTION

Hp-adaptive FE discretization methods can be employed in scientific and engineering applications, such as discontinuous Galerkin formulations, ranging from aeroacoustics to hydrodynamics. By allowing the elements in the simulation to be refined, and the polynomial order within the element domain to be increased, these methods are able to handle complex geometries while offering good convergence properties. The visualization of the data produced by this method is essential for scientists to understand the simulations. Isocontouring methods such as Marching Cubes [3] are not directly applicable to high order data. Linear graphic primitives and low order interpolants available in visualization systems require elaborate resampling schemes to represent such data with sufficient accuracy and tolerable memory consumption.

In this work we propose a simple and effective particle system approach to interactively and accurately visualize isocontours of scalar quantities defined on hp-adaptive FE meshes. We focus our attention to arbitrary polyhedral elements and polynomials of degree up to 6, given analytically on a monomial basis in object space. Our solution has the following characteristics: a compact memory layout for polynomial coefficients; an efficient framework for polynomial evaluation based on a greedy approach to the multivariate Horner scheme; fast isocontouring formulated as a highly parallelized root finding algorithm in the CUDA [6] programming language; computational efficiency since the computation of the isocontour is view-independent for a given isovalue, as opposed i.e. to raycasting methods.

## 2 RELATED WORK

Remacle *et al.* [8] proposed an adaptive resampling scheme with guaranteed error bounds for the visualization of higher order finite elements and thus effectively transform the problem into one of adaptive mesh refinement (AMR). Figueiredo *et al.* [2] presented two physically based methods to generate triangle meshes from implicit surfaces. The first approach is based on a mass-string system while the second is based on a particle system. Witkin and Heckbert [10] developed an iterative technique, based on particle systems, to model and visualize implicit surfaces. When used as a modeling tool, the points are used as handles that deform the implicit surface. When used as a visualization tool, the particles are projected onto the surface and rendering is performed via splatting. The size and distribution of the splats over the surface take curvature into account and are controlled through several parameters.

Meyer *et al.* [4] presented a particle system based method to sample high order surfaces generated by FE simulations. Particle projection is based in the same procedure presented by Witkin and Heckbert. On the other hand, their approach is based on data sets where the high order functions must be evaluated in a reference space, which involves expensive mapping operations. In this algorithm the projection and repulsion of particles are interleaved until the system converges. Despite its effectiveness, the procedure is not interactive, taking several minutes to project thousands of points. Nelson and Kirby [5] proposed an isocontour raytracing solution for spectral/hp-adaptive FE which projects a polynomial onto the viewing ray of each pixel in the image plane with a subsequent root finding along each ray. To the best of our knowledge, Kooten *et al.* [9] presented the first interactive implicit surface visualization method based on particle systems that runs entirely on a GPU. Points are projected through the same procedure described by Witkin and Heckbert, but neighbor search required for inter-particle repulsion is accelerated using a spatial hash structure.

## 3 A GPU-BASED PARTICLE SYSTEM APPROACH

The hp-adaptive FE data used in this work originates from discontinuous Galerkin (DG) simulations in which the solution is given for each element based on its geometry (which can have an arbitrary polyhedral shape) and polynomials. Matters are simplified by the fact that polynomials are defined in object space. Therefore, an intricate mapping between object and reference space is unnecessary. We assume polynomials to be given on a monomial basis of the form:

$$P(\mathbf{x}) = \sum_{i+j+k < n} c_{i,j,k} x^i y^j z^k \quad (1)$$

with  $i, j, k \in \mathbb{N}_0$ , the maximum order  $n$ , the coefficients  $c_{i,j,k}$  and the object space coordinates  $x, y, z$ . Given  $n$ , which is known beforehand and constant for one scalar quantity per element, we define an ordering rule for the list of coefficients  $C_l$  by the following algorithm:

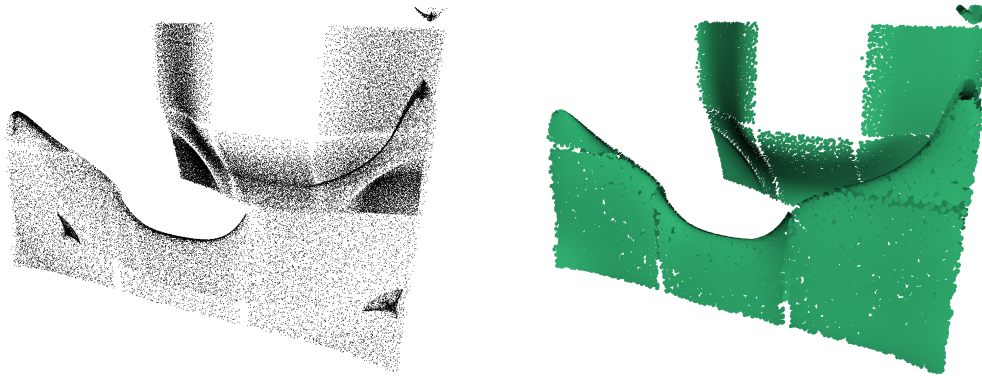


Figure 1: Point density appears to be higher in areas of high surface curvature (left). Rendering with discs and surface lighting (right).

```

l = 0
for k = n - 1..0 do
  for j = n - k - 1..0 do
    for i = n - k - j - 1..0 do
      Cl = ci,j,k
      l += 1
    end
  end
end
end

```

**Algorithm 1:** Ordering rule for polynomial coefficients

A straightforward strategy is to evaluate Equation 1 in the same fashion as Algorithm 1, computing the exponents of  $x, y, z$  explicitly. Instead, our solution uses the greedy multivariate Horner scheme of Ceberio and Kreinovich [1] which we have implemented in C++ and CUDA. Empirical results indicate that using this evaluation scheme, the resulting number of multiplications is equal to the number of summations, which we believe to be the theoretical optimum.

Isocontouring is formulated as a root finding problem with initially uniformly distributed particles per element. Elements are processed successively whereas particles of one element are processed in parallel by spawning a CUDA thread per particle solving  $P(\mathbf{x}) - I = 0$  by successive Newton-Raphson iterations for a given isovalue  $I$ . Since this requires  $\nabla P$ , derivatives in  $x, y$  and  $z$  direction are computed analytically in a preprocessing step. Coefficients of  $P, P_x, P_y$  and  $P_z$  are interleaved and stored in a vector of `float4` which resides in constant global device memory. This interleaving results in a minor memory overhead since the degrees of derivatives and  $P$  differ by one. However, the benefit of this layout is that it allows for a 16 byte stride which enables quick coalesced memory access on the GPU. A particle is modeled with a `struct` containing the position in space and a flag describing its state (three floats and one int). Particles can thus be arranged in a vector with 16 byte stride, which again allows for efficient memory access. Depending on the size of the data it is either possible to stream coefficients and particles to the GPU per element or to store the dataset entirely in device memory. Measurements show that with all data residing locally on device memory, the proposed framework is capable of performing 95 million Newton-Raphson iterations per second on an NVIDIA GeForce 8800 GTX card. Therefore, for a 10 million particle system and 10 Newton-Raphson iterations on average until convergence, the computation roughly lasts a second.

Many particle-based isocontouring methods employ inter-particle repulsion in order to distribute particles evenly on the surface, which is a time consuming task. In this work we propose to intentionally omit the repulsion step based on two observations: First, the projection of particles onto the isocontour can be done quickly on a GPU, as already observed by Kooten *et al.* [9]. Second, it appears that gradient based methods already distribute par-

ticles onto the isocontour with curvature-dependent density, as observed by Figueiredo *et al.* [2], but not further explored. Figure 1 shows some preliminary results.

#### 4 DISCUSSION AND FUTURE WORK

The presented approach is subject of ongoing research which we have decided to share with the community before a subsequent detailed publication. In the immediate future we plan to further explore both the use of local surface curvature and parallelized inter-particle repulsion, for which an n-body approach along the lines of Lyland [7] appears promising.

#### ACKNOWLEDGEMENTS

The authors wish to thank Christoph Altmann and Claus-Dieter Munz of the Institut für Aerodynamik und Gasdynamik (IAG) at the University of Stuttgart for providing the simulation data.

#### REFERENCES

- [1] M. Ceberio and V. Kreinovich. Greedy Algorithms for Optimizing Multivariate Horner Schemes. *SIGSAM Bull.*, 38(1):8–15, 2004.
- [2] L. H. de Figueiredo, J. de Miranda Gomes, D. Terzopoulos, and L. Velho. Physically-based Methods for Polygonization of Implicit Surfaces. In *Proc. Graphics Interface*, pages 250–257, 1992.
- [3] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proc. ACM SIGGRAPH*, volume 21, pages 163–169, New York, NY, USA, 1987.
- [4] M. Meyer, B. Nelson, R. Kirby, and R. Whitaker. Particle Systems for Efficient and Accurate High-Order Finite Element Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1015–1026, Sept.-Oct. 2007.
- [5] B. Nelson, B. Nelson, and R. Kirby. Ray-tracing Polymorphic Multidomain Spectral/hp Elements for Isosurface Rendering. *IEEE Transactions On Visualization And Computer Graphics*, 12(1):114–125, 2006.
- [6] NVIDIA Corporation. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 1.1*. NVIDIA Corporation, 2007.
- [7] L. Nyland, M. Harris, and J. Prins. Fast N-Body Simulation with CUDA. In *GPU GEMS 3*, chapter 31, pages 677–695. Addison-Wesley, 2007.
- [8] B. Remacle, N. Chevaugneon, . Marchandise, and C. Geuzaine. Efficient Visualization of High Order Finite Elements. *International Journal for Numerical Methods in Engineering*, 69(4):750–771, 2006.
- [9] K. van Kooten, G. van den Bergen, and A. Telea. Point-Based Visualization of Metaballs on a GPU. In *GPU GEMS 3*, chapter 7. Addison-Wesley, 2007.
- [10] A. P. Witkin and P. S. Heckbert. Using Particles to Sample and Control Implicit Surfaces. In *Proc. ACM SIGGRAPH*, pages 269–277, 1994.