Differential Point Rendering

Aravind Kalaiah Amitabh Varshney
University of Maryland[1]

Abstract. We present a novel point rendering primitive, called *Differential Point* (DP), that captures the local differential geometry in the vicinity of a sampled point. This is a more general point representation that, for the cost of a few additional bytes, packs much more information per point than the traditional point-based models. This information is used to efficiently render the surface as a collection of local neighborhoods. The advantages to this representation are manyfold: (1) it delivers a significant reduction in the number of point primitives that represent a surface (2) it achieves robust hardware accelerated per-pixel shading – even with no connectivity information (3) it offers a novel point-based simplification technique that has a convenient and intuitive interface for the user to efficiently resolve the speed versus quality tradeoff. The number of primitives being equal, DPs produce a much better quality of rendering than a pure splat-based approach. Visual appearances being similar, DPs are about two times faster and require about 75% less disk space in comparison to splatting primitives.

1 Introduction

Point-based rendering schemes [5, 12, 13, 17, 19, 21] have evolved as a viable alternative to triangle-based representations. They offer many benefits over triangle-based models: (1) efficiency in modeling and rendering complex environments, (2) hierarchical organization to efficiently control frame-rates and visual quality, and (3) zero-connectivity for efficient streaming for remote rendering [20].

Current point primitives store only limited information about their immediate locality, such as normal, bounding ball [19], and tangent plane disk [17]. These primitives are then rasterized with flat shading and hence such representations require very high sampling to obtain a good rendering quality. In other words, the rendering algorithm dictates the sampling frequency of the modeling stage. This is clearly undesirable as it may prescribe very high sampling even in areas of low spatial frequency, causing two significant drawbacks: (1) slower rendering speed due to higher rendering computation and related CPU-memory bus activity, and (2) large disk and memory utilization.

In this work we propose an approach of storing local differential geometric information with every point. This information gives a good approximation of the surface distribution in the vicinity of each sampled point which is then used for rendering the point and its approximated vicinity. The extent of the approximated vicinity is determined by the curvature characteristics of the surface: points in a flat or a low curvature region approximate larger vicinities. Our approach has many benefits to offer:

[1]Graphics and Visual Informatics Laboratory, Department of Computer Science and UMIACS, University of Maryland, College Park, MD - 20742, USA, {ark,varshney}@cs.umd.edu

S. J. Gortler et al. (eds.), *Rendering Techniques 2001*

1. **Rendering**: Our approach achieves high rendering quality with per-pixel shading. We reduce the number of rendering primitives and push more computation into each primitive. This reduces the CPU-memory bus bandwidth and the overall amount of computation resulting in a significant speed-up. As the processor-memory speed gap increases we expect this method to get even more attractive.
2. **Storage**: The reduction in the number of primitives more than compensates for the extra bytes of information stored with each point primitive. This leads to an overall reduction in the storage requirements. This reduction also benefits faster streaming of information over the network.
3. **Generality**: The information stored with our point primitives is sufficient to derive (directly or indirectly) the requisite information for prior point primitives. Our work is primarily a focus on the efficiency of per-point rendering computation. It can potentially be used in conjunction with larger point-based organization structures - hierarchical [1, 17, 19] or otherwise [13, 21].

In the following sections, we first mention some related works and then outline the terminology from the differential geometry literature that will be used to describe our approach. This is followed by a discussion of differential points. We then describe our rendering algorithm and compare it with some splatting schemes. We conclude the paper with a discussion of this approach and its potential extensions.

2 Related Work

Classical differential geometry gives us a mathematical model for understanding the surface variation at a point. Surface curvatures can be accurately computed for parametric surfaces and can also be estimated from discrete sampled representations.Taubin [22] estimates curvature at a mesh vertex by using the eigenvalues of an approximation matrix constructed using the incident edges. Desbrun et al. [3] define discrete operators (normal, curvature, etc.) of differential geometry using Voronoi cells and finite-element/finite-volume methods.

Various hierarchical organization schemes have been used that develop lower frequency versions of the original set of point samples [1, 17, 19]. We use a simplification process that prunes an initial set of points to greatly reduce the redundancy in surface representation. Turk [23] uses a point placement approach with the point density being controlled by the local curvature properties of the surface. Witkin and Heckbert [24] use physical properties of a particle system to place points on an implicit surface.

Image-assisted organization of points [5, 13, 21] are efficient at three-dimensional transformations as they use the implicitness of relationship among pixels to achieve fast incremental computations. They are also efficient at representing complex real-world environments. The multiresolution organizations [1, 17, 19] use their hierarchical structure to achieve block culling, to control depth traversals to respect the image-quality or frame-rate constraints, and for efficient streaming of large datasets across the network [20].

Darsa et al. [2], Mark et al. [14] and Pulli et al. [18] use a triangle mesh overlaid on the image sampling plane for rendering. It can be followed by a screen space compositing process. However, such systems can be expensive in computation and storage if high resolutions are desired. Levoy and Whitted [12] introduced points as a rendering primitive. It has been used by Shade et al. [21] and Grossman and Dally [5] for synthetic environments. However, raw point primitives suffer from aliasing problems and holes. Lischinski and Rappoport [13] raytrace a point dataset. Oliveira et al. [15] use