View-Dependent Textured Splatting for Rendering Live Scenes

David T. Guinnip Shuhua Lai Ruigang Yang
University of Kentucky

Abstract

This sketch presents a novel approach for rendering low resolution point clouds with multiple high resolution textures–the type of data commonly generated by real-time vision systems. The low precision, noisy, and sometimes incomplete nature of such data sets is not suitable for existing point-based rendering techniques that are designed to work with high precision and high density point clouds.

Our new algorithm–*View-dependent Textured Splatting* (VDTS)–combines traditional splatting with a view-dependent texturing strategy to increase rendering quality of low resolution data sets with high resolution textures. VDTS requires no pre-processing, addresses texture visibility and anti-aliasing on the fly, and can be efficiently accelerated by commodity graphics hardware. Therefore it is well suited for real-time rendering of dynamic scenes online.

1 The Approach

Given a point cloud that approximates the scene geometry and a number of images (textures) taken from known locations (reference views), our VDTS algorithm aims to synthesize new images from a user-driven view point. It proceeds by first generating an oriented rectangle (quad) for each point primitive. Then, textures are projected onto these quads and blended together in a view-dependent fashion similar to [Debevec et al. 1998; Buehler et al. 2001]. Finally, these textured quads are rendered from the desired view point through the standard graphics pipeline.

From a higher level, VDTS resembles a classic textured splatting algorithm. However, the use of projective texture mapping requires the visibility of textures to be solved correctly. Existing algorithms either ignore this problem (as in [Buehler et al. 2001]), which leads to rendering artifacts around occlusion boundaries; or solve it in a pre-processing step (as in [Debevec et al. 1998; Matusik et al. 2002]), which limits their feasibility for real-time dynamic data sets. VDTS instead solves the visibility problem on the fly to provide better rendering results for dynamic data sets. Some details are presented in the following paragraphs.

Real-Time Visibility Analysis VDTS performs visibility analysis in real-time using a shadow mapping technique [Segal et al. 1992], which is accelerated by modern graphics hardware. A per-fragment test is performed at rendering time to compare depth buffer values of the reference view(s) and rendered view. Comparing our visibility test to shadow mapping, an "invisible" fragment is equivalent to an "in-shadow" fragment.

When working with a low resolution point cloud, each point needs to be turned into a fairly large quad to avoid holes in the rendered image. This can cause rendering artifacts around surface boundaries. Figure 2 (Left) illustrates the problem. P and Q are two points sampled from the object surface. They are turned into two quads (still denoted as P and Q) at rendering time. A region of Q extends beyond the object surface, creating an inaccurate silhouette. P, in the meantime, partially occludes the surface behind it, creating some ghosting effects because textures are blended on the wrong surface. Both problems are visible in Figure 2 (middle). While inaccurate silhouettes can be removed by segmenting input images to render only the foreground objects, ghosting is difficult to remove with the low-resolution point cloud. However, when a high resolution depth map is available from each reference view (such as data sets converted from CG models), both types of artifacts can be removed by setting a maximum deviation κ between the depth of each rendered fragment and the projected depth of any contributing source view. If the difference exceeds κ, the fragment's depth will be set behind the far clipping plane, removing it from the rendered view. We call this technique *visibility clipping*. It can be efficiently implemented in a fragment program. Figure 2 (Right) illustrates the benefit of visibility clipping.

Figure 1: Comparison of rendering methods. **(Left)** Point splatting (one color per point); **(Right)** VDTS rendering. Reconstruction contains 6103 point samples with 8 source views.

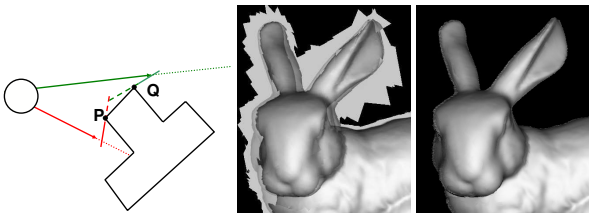

Figure 2: VDTS rendering with toggled visibility clipping (VC). **(Left)** A low resolution point cloud leads to an inaccurate surface representation; **(Middle)** VC disabled; **(Right)** VC enabled. Model from Stanford 3D Scanning Repository.

2 Real-time System

To demonstrate the effectiveness of VDTS for real-time rendering of dynamic scenes, we have implemented a complete on-line system for scene acquisition and rendering. It consists of a pair of stereo cameras at XGA resolution and a 3.2Ghz PC. From captured XGA images, we construct a depth map at quarter resolution (256×192) using stereo vision techniques. The depth map and the stereo images are then used by VDTS for rendering. Although not demonstrated, it is trivial for VDTS to handle multiple depth maps (reconstructed from different camera pairs). Mesh-based techniques, however, will have to deal with the very challenging 3D triangulation problem. In term of performance, VDTS runs well over 30 frames per second while the reconstruction takes about 40-60 ms per image pair. They can run asynchronously.

References

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S., AND COHEN, M. 2001. Unstructured Lumigraph Rendering. In *Proceedings of SIGGRAPH 2001*, 43–54.

DEBEVEC, P. E., BORSHUKOV, G., AND YU, Y. 1998. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. In *9th Eurographics Rendering Workshop*, 427–433.

MATUSIK, W., PFISTER, H., NGAN, A., BEARDSLEY, P., ZIEGLER, R., AND MCMILLAN, L. 2002. Image-Based 3D Photography using Opacity Hulls . In *Proceedings of SIGGRAPH 2002*.

SEGAL, M., KOROBKIN, C., VAN WIDENFELT, R., FORAN, J., AND HAEBERLI, P. 1992. Fast Shadows and Lighting Effects Using Texture Mapping. In *Proceedings of SIGGRAPH 1992*, 249–252.