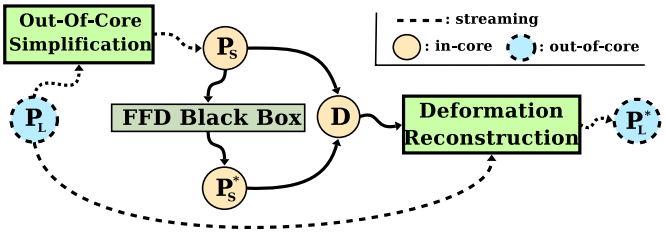
Scalable Freeform Deformation

Tamy Boubekeur* Olga Sorkine + Christophe Schlick*
* : LaBRI - INRIA - University of Bordeaux + : TU Berlin



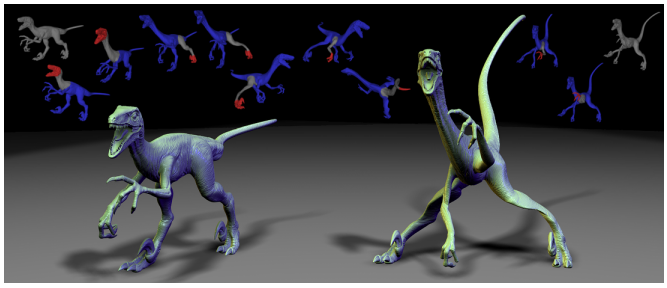


Figure 1: Left: *Our FFD system with two out-of-core bracketing steps.* **Right:** *Interactive FFD on the Raptor (8M triangles.)*

1 Introduction

Freeform deformation techniques are powerful and flexible tools for interactive 3D shape editing. However, while interactivity is the key constraint for the usability of such tools, it cannot be maintained when the complexity of either the 3D model or the applied deformation exceeds a given workstation-dependent threshold. In this work, we solve this scalability problem by introducing a streaming system based on a sampling-reconstruction approach. First a fast *out-of-core adaptive simplification* algorithm is performed in a pre-processing step, for quick generation, of a simplified version of the model. The resulting model can then be submitted to arbitrary FFD tools, as its reduced size ensures interactive response. Second, a post-processing step performs a *feature-preserving deformation reconstruction* that applies to the original model the deformation undergone by its simplified version. Both bracketing steps share a *streaming* and *point-based* basis, making them fully scalable and compatible both with point-clouds and non-manifold meshes. Our system also offers a generic out-of-core multi-scale layer to FFD tools, since the two bracketing steps remain available for partial upsampling during the interactive session. Arbitrarily large 3D models can thus be interactively edited with most FFD tools, opening the use of advanced deformation metaphors to models ranging from million to billion samples. Our system also enables offers to work on models that fit in memory but exceed the capabilities of a given FFD tool.

2 Out-of-core Adaptive Simplification

As a pre-process, we propose a new efficient adaptive out-of-core downsampling, performed on the original large object P_L during a streaming process, to get a simplified version P_S that will preserve interactivity. As the input may be arbitrary large, our algorithm uses *spatial finalization* [Isenburg et al. 2006] to maintain a low memory footprint during sampling. This leads to the construction of a coarse grid, where cells are valid (all elements present in memory) only for a partial time during this streaming. This allows us to use a *volume-surface tree* [Boubekeur et al. 2006] for efficient adaptive sampling of the enclosed geometry. The partitioning generated by the volume surface tree creates a point sampling of the area, optionally equiped with re-indexed triangles if provided in the input stream. The simplified model P_S, obtained with a small in-core memory fooprint, is then submitted to an arbitrary interactive FFD tool, considered as a black box providing a deformation P_S^* of P_S.

3 Streaming Deformation

The goal of the out-of-core streaming post-process is to efficiently and accurately transfer the deformation defined on P_S during interactive FFD to the original gigantic object P_L, to get the final deformed object P_L^*. Our algorithm is purely local (each sample processed independently) which makes it compatible with geometric streaming, crucial in the context of large objects.

We propose to extract from each original sample p of P_L a local displacement according to P_S: $p = p' + d \cdot n$, where p' is the projection of p onto an average plane H, with normal n defined by locally filtering P_S with an Hermitian kernel, and d the signed distance from p to H. Note that using a scalar displacement in the normal direction makes it rotation-invariant. In fact, P_S^* already captures the deformation of the low frequency part of P_L, so we deform this low frequency part in the spirit of Laplacian deformation, by using intrinsic coordinates of p'. In contrast with prior approaches, we propose a faster encoding, based on a fixed 3-neighborhood. First we select 3 samples of P_S in the local neighborhood of p using a kD-Tree, by forming the smallest as-equilateral-as possible triangle $T(p) = \{q_i, q_j, q_k\}$, both for symmetry and accuracy. This selection allows to use *projected barycentric coordinates* [Heidrich 2005] $B(p) = \{b_i, b_j, b_k\}$ for encoding p' relatively to these samples, avoiding least-squares fitting frequently used in previous local coordinates systems. Finally, the deformation is transfered by:

$$p'^* = T'^*(p) \cdot B'(p)^\top .$$

We also extract a scaling factor to compute d^* according to local area modification on P_S, obtaining:

$$\forall p \in P_L \qquad p^* = D(p) = T'^*(p) \cdot B'(p)^\top + d^* \cdot n^* .$$

This point-based deformation transferred from P_S^* to P_L is fast and smooth thanks to the Hermitian filtering of the projection plane, which provides a good trade-off for quickly processing the huge number of samples (up to a billion in our tests) streamed from P_L.

References

BOUBEKEUR, T., HEIDRICH, W., GRANIER, X., AND SCHLICK, C. 2006. Volume-surface trees. *Computer Graphics Forum v25*, n3, p399–406.

HEIDRICH, W. 2005. Computing the barycentric coordinates of a projected point. *Journal of Graphics Tools v10*, n3, p9–12.

ISENBURG, M., LIU, Y., SHEWCHUK, J., AND SNOEYINK, J. 2006. Streaming computation of delaunay triangulations. *ACM Trans.Graph. v25*, n3, p1049–1056.

Figure 2: *The interactive smooth deformation of this model (28M triangles) has been done in less than 5 minutes, including pre and post streaming process, with our system.*