Mesh Ensemble Motion Graphs

Doug L. James Christopher D. Twigg Andrew Cove
Carnegie Mellon University

Robert Y. Wang
Massachusetts Institute of Technology

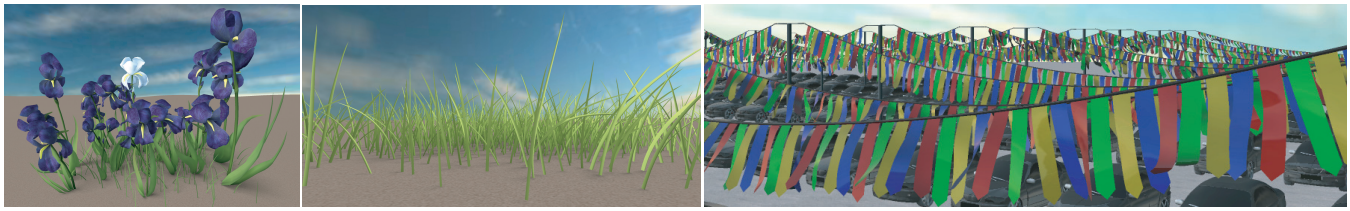

Figure 1: *Mesh ensemble motion graph examples synthesized in real time and without nonphysical interpenetrations. All examples have several thousand compressed degrees of freedom, and resulted from expensive thin-shell finite element simulations with robust contact handling.*

Abstract

We describe a technique for using space-time cuts to smoothly transition between stochastic mesh animation clips while subject to physical noninterpenetration constraints. These transitions are used to construct *Mesh Ensemble Motion Graphs* for interactive data-driven animation of high-dimensional mesh animation datasets, such as those arising from expensive physical simulations of deformable objects blowing in the wind (see Figure 1). We formulate the transition computation as an integer programming problem, and use a novel randomized algorithm to compute transitions subject to noninterpenetration constraints.

1 Asynchronous Mesh Transitions

Compressed mesh animations are attractive for real-time hardware-accelerated playback of complex mesh deformation phenomena that is otherwise too expensive to compute on the fly [James and Twigg 2005]. Mesh ensembles with numerous distributed mesh elements, such as a garden of flowers or thousands of colliding flags driven by stochastic wind forces, provide a challenge for data-driven animation given the intrinsically high-dimensional dynamic phenomena. However, such examples are appealing candidates for interactive data-driven animation given the high cost of simulation using robust contact handling [Bridson et al. 2002]. Motion graph techniques [Kovar et al. 2002; Lee et al. 2002] provide the basic abstraction for data-driven animation, yet simply splicing together motion clips (to loop or transition within the dataset) leads to transition artifacts, since in high dimensional state spaces it is exceedingly rare to find close transitions, and synchronized transitions introduce undesirable errors with strong spatial correlation.

To overcome these limitations, we allow asychonous transitions whereby each mesh subgroup can transition at a different frame so as to minimize its transition error (see Figure 2). In practice this is easily achieved by minimizing a separable objective function $\Phi(\vec{\tau})$ that measures the smoothness of the transition as a function of each of the G mesh groups' transition offset times, $\vec{\tau} \in \mathbb{Z}^G$. Given a transition $\vec{\tau}$, we exploit tree-structured scene kinematics to reconstruct the scene shape during the asynchronous transition.

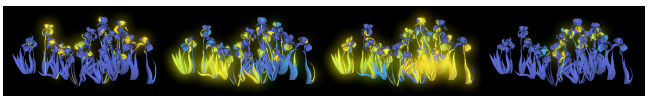

Figure 2: **Asynchronous transitions** *allow scene components to transition at slightly different times–yellow indicates mesh groups undergoing transitions–and thus avoid transition artifacts by (i) reducing total transition error, and (ii) diffusing group transition errors in both space and time.*

2 Noninterpenetration Constraint Programming

Although asynchronous mesh ensemble transitions are conceptually similar to space-time cut techniques used for video-based rendering [Kwatra et al. 2003], mesh animations have the added complexity of a 3D embedding for which not all shape configurations are feasible due to interpenetrations. Unfortunately, allowing mesh groups to transition any time they desire can lead to low-quality animations with substantial and unsightly interpenetration artifacts. We address this by minimizing the transition cost function for $\vec{\tau}$ subject to geometric noninterpenetration constraints between all ensemble mesh groups. We first compute the unconstrained minimum and detect unsatisfied noninterpenetration constraints. Next we apply an iterative local search strategy to randomly sample components of $\vec{\tau}$ associated with unsatisfied noninterpenetration constraints. When the list of known interpenetration constraints has been emptied, new unsatisfied constraints are computed by performing collision detection on randomly sampled frames from the transition. Similar to other randomized constraint programming techniques [Selman et al. 1996], we either eventually find a feasible $\vec{\tau}$ that eliminates all existing interpenetrations, or we time-out and investigate other transition opportunities. Deformable collision detection is optimized by exploiting the compressed shape representation [James and Twigg 2005], as well as sparse functional dependencies of the noninterpenetration constraints on $\vec{\tau}$. Thus, in a fraction of the time required to compute the original high-quality robust-contact physically based animations, we can compute noninterpenetrating asynchronous transitions to build Mesh Ensemble Motion Graphs for interactive animation of complex high-dimensional, deformation phenomena.

References

Bridson, R., Fedkiw, R. P., and Anderson, J. 2002. Robust Treatment of Collisions, Contact, and Friction for Cloth Animation. *ACM Transactions on Graphics 21*, 3 (July), 594–603.

James, D. L., and Twigg, C. D. 2005. Skinning Mesh Animations. *ACM Transactions on Graphics 24*, 3 (Aug.), 399–407.

Kovar, L., Gleicher, M., and Pighin, F. 2002. Motion Graphs. *ACM Transactions on Graphics 21*, 3 (July), 473–482.

Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. 2003. GraphCut Textures: Image and Video Synthesis Using Graph Cuts. *ACM Transactions on Graphics 22*, 3 (July), 277–286.

Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., and Pollard, N. S. 2002. Interactive Control of Avatars Animated With Human Motion Data. *ACM Transactions on Graphics 21*, 3 (July), 491–500.

Selman, B., Kautz, H., and Cohen, B. 1996. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 26. American Mathematical Society, 521–532.