A Multi-Touch Interface for 3D Mesh Animation

Duygu Ceylan
Bilkent University
dceylan@cs.bilkent.edu.tr

Tolga Çapın
Bilkent University
tcapin@cs.bilkent.edu.tr

Abstract

We present a mesh animation system targeted for novice users, which utilizes the multi-touch interaction technique and direct manipulation principles. Our solution is based on manipulating meaningful parts of a model simultaneously to produce a sense of movement. For this purpose, we use differential mesh representation and mesh segmentation methods. Animation examples demonstrate that the proposed solution presents a usable interface by giving the users direct control over the system.

Keywords: Laplacian mesh editing, mesh segmentation, volume preserving mesh animation, multi-touch interaction

1 Introduction

Today computer animation is used in many applications, such as computer games, special effects, and virtual environments. The variety of these applications has given rise to the development of commercial tools for modeling and animation, such as Autodesk Maya [1]. However, these tools require expert knowledge or prior training about the animation techniques such as keyframing, and they are tedious to use for non-expert users. Various traditional methods in computer animation production, such as procedural animation or motion capture, are also not accessible by these users. Thus, there is a need for animation tools designed for novice users with no technical or design background.

This paper addresses this problem by creating an expressive interface for animating arbitrary 3D mesh models. The user animates the input 3D mesh *interactively* by simultaneously manipulating - picking and moving - different parts of the mesh directly on display. The main idea is to use the interaction modality of multi-touch display for input to give the users the feeling of directly manipulating objects by hand.

Our framework segments an input 3D model into parts and enables these parts to be manipulated simultaneously. We use dynamic Laplacian differential representation of the input mesh. Differential representations aim to encode shape features of a model and preserve them during manipulation, making them popular for realistic and shape-preserving editing operations [2, 3].

Our work is inspired by sketch-based interfaces proposed for 3D animation, mainly for creation of virtual character motion [4]. Interfaces, which combine the power of direct manipulation interfaces with traditional keyframe methods have also been proposed [5]. Our method differs from these interfaces in that we assume that our target users cannot create keyframe-based animations, and that the input objects can be deformed arbitrarily, as well as repositioned in the scene. Furthermore, our method does not assume an articulated skeleton structure within the input model because not all objects have a skeleton structure, such as jellies or similar deformable models. Moreover, animation effects such as squash and stretch are difficult to achieve with the skeleton-based representation.

The ease of use of our interface has the possibility to create new application areas for 3D animation, such as rapid prototyping in early stages of animation, educational or entertainment use for non-professionals and children, and real-

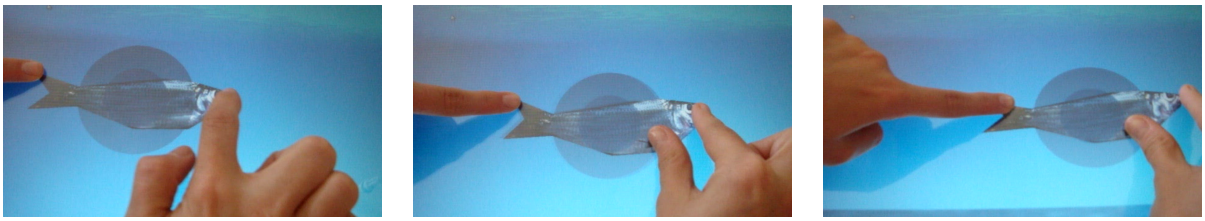


Figure 1: Overview of the system.

time free-form avatar control in shared virtual environments or 3D games.

2 Overview of the System

Our animation system consists of three subsystems. The first subsystem processes a 3D mesh by partitioning and computing the differential representation of it. The second subsystem focuses on animation operations, where as the third subsystem is responsible for interpreting the multi-touch input.

Once a 3D model is loaded, users can interactively manipulate different parts of it by dynamically selecting a *handle* at the desired locations of the model. The part of the model that will be affected from the editing operations is computed automatically. By taking advantage of the multi-point feature of the multi-touch screen, users can select many handles at different locations of the model simultaneously. During the deformation process, the model can also be globally transformed via a transformation widget.

3 Mesh Processing Subsystem

The mesh processing subsystem partitions the input mesh into meaningful parts, and creates the differential representation of these subparts.

The mesh partitioning algorithm is based on the visual saliency of the parts of a model as proposed in the work of Antini et al. [6]. The first step of this algorithm identifies different regions of vertices on a mesh called nodes, based on geodesic distances between them. The second step analyzes these created regions, and merges different nodes according to their adjacency and curvature features. Figure 2 shows an example model partitioned with this method.

Once the input mesh is partitioned, the Laplacian representation of each subpart is defined.

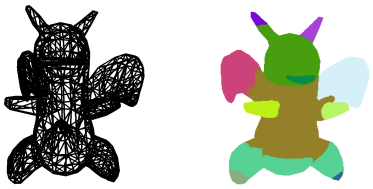


Figure 2: Partitioning of a dragon model

The logic behind the Laplacian framework is the fact that the manipulation of the mesh is meant to animate large features, while keeping the small details locally unchanged. These details are encoded by Laplacian coordinates and preserved during manipulation operations as much as possible. Therefore, the manipulation of the input mesh is carried over Laplacian coordinates of the vertices and the global coordinates are reconstructed at the end. The reconstruction of global coordinates is accomplished by solving a linear system [2]. This linearity makes the approach very efficient and is the main reason for using Laplacian representation in our work.

The main practical problem with Laplacian coordinates is that they are not invariant to scaling and rotation operations and a scheme should be developed to compensate for this deficiency. In our work, we follow the approach proposed by Sorkine et al. [2], which is based on implicit computation of a transformation for each vertex. Based on the reconstructed positions of the vertices, a similarity transformation that transforms the 1-ring neighborhood of a vertex to its new location is computed for each vertex.

4 Animation Subsystem

In our system, a 3D model is animated by touching desired regions of it to dynamically define *handles*, which are vertices that can be moved freely by the user. The movement of the han-

dles is propagated to the rest of the mesh such that the modification is intuitive and resembles animating an object made of soft material.

When a handle is chosen, the mesh subpart containing it is determined as the region of interest (ROI). All the vertices in this part of the mesh except the handle vertex are assumed to be unconstrained freely moving vertices. However, the transition between the ROI and the rest of the model should be smooth, meaning that there must be a set of boundary vertices that remain fixed and constrain the deformation process (Figure 3). In our system, this set is composed of vertices that are the first-order neighbors of the vertices in the ROI and not belonging to the same part of the mesh. Both the handle and the boundary vertex positions are used as constraints while solving the Laplacian system to reconstruct the global coordinates.

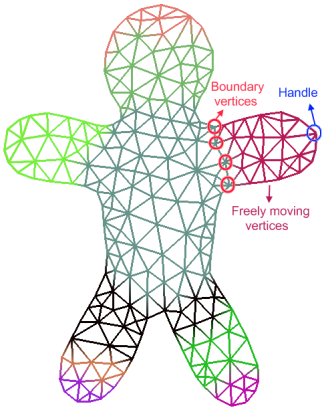
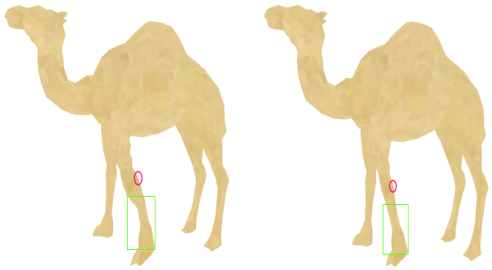


Figure 3: ROI Specification

Preservation of volume during manipulation and animation of objects is one of the fundamental principles in animation. It is expected that no matter how squashed or stretched out a particular object gets, its volume remains constant. On the other hand, during Laplacian manipulation sessions, loss of volume can be observed due to large deformations. To overcome this problem, we include a volume preservation component in our system that extends the approach presented by Funck et al. [7].

The logic behind the volume preservation operations is to define a displacement vector field for the mesh. This field defines in which direction and how strong each vertex should be moved after a manipulation action to preserve the volume. As manipulation actions are applied to the subparts of a mesh, preserving the volume in these parts results in a global volume preservation. Therefore, we define displacement fields separately for each mesh part, pointing away from the mesh. Transformations computed at the end of each Laplacian cycle are applied to the sum of corresponding vertex-vector field pairs. The difference between the results and the original deformed vertices is used to determine how much each vertex should be moved in the direction of its vector field. Figure 4 shows an example model manipulated by the Laplacian framework with and without applying volume preservation.



(a) (b)

Figure 4: Model manipulated (a) with and (b) without volume preservation. Red circles denote the handle positions.

5 User Interface

A notable characteristic of our system is its applicability to multi-touch screens. Multi-touch environments have begun to gain popularity in a variety of applications due to the potential advantages they present in user interaction. The multi-touch interface used in our system is Stantum Technologies' Multi-Touch Development Kit 12.1-D [8].

The most significant benefit of using a multi-touch environment in our work is the ability to manipulate several regions of a model simultaneously to create the illusion of movement. Local manipulation actions are applied by directly touching the desired regions of the model. Meanwhile, the user may also want to globally rotate or translate a model. For this purpose, we have designed a 3D transformation widget, which is an extension of Arcball [9].

The Arcball technique assumes that an object is enclosed in a sphere and the user rotates this

object by moving the input device on the circle, which is the projection of the sphere on the screen. In our design, we define an additional outer circle around this projection circle, creating the widget shown in Figure 5. Cursor movements between the inner and outer circles are mapped to a translation component represented by the vector joining the last point on the inner circle the cursor has passed through to the final cursor position.

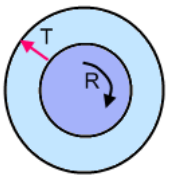


Figure 5: Extended arcball widget used to specify rotations (R) and translations (T)

6 Results and Discussion

The accompanying video demonstrates the real-time animation results created using our system. Figure 1 shows snapshots from this video. The current system provides expressive results by giving the users direct control over the system. However, there are several ways in which we want to improve it. To begin with, skeletal-based animation methods can be implemented to make a comparison with the current free-form mesh animation method and other skeletal-based approaches. The system can be extended with the development of other multi-input interaction methods to increase user control. Finally, a detailed user survey can be conducted to measure how fast users are able to create new animations and what their preferences are when compared with other commercial tools or approaches.

7 Conclusion

In this work we have presented a system for 3D mesh animation with multi-touch interaction. Our system automatically segments an input 3D model into meaningful parts, avoiding complex interaction for manual partitioning. Our Laplacian-based approach for representation of the mesh allows natural manipulation of these parts. Finally, multi-touch interaction methods enable an easy-to-use interface.

References

[1] Autodesk. Autodesk - 2d and 3d design and engineering software for architecture, manifacturing, and digit, 2009. `http://usa.autodesk.com/`.

[2] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.P. Seidel. Laplacian surface editing. *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188, 2004.

[3] T. Igarashi, T. Moscovish, and J.F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Computer Graphics*, 24(3):1134–1141, 2004.

[4] Matthew Thorne, David Burke, and Michiel van de Panne. Motion doodles: an interface for sketching character motion. In *Proceedings of ACM SIGGRAPH 2004*, pages 424–431, New York, NY, USA, 2004. ACM.

[5] T. Igarashi, T. Moscovich, and J. F. Hughes. Spatial keyframing for performance-driven animation. In *Proceedings of SCA '05*, pages 107–115, New York, NY, USA, 2005. ACM.

[6] G. Antini, S. Berretti, and P. Del Bimbo, A.and Pala. 3d mesh partitioning for retrieval by parts applications. *Proceedings of IEEE ICME*, pages 1210–1231, 2005.

[7] W. von Funck, H. Theisel, and H.-P. Seidel. Volume preserving mesh skinning. *Proceedings of Vision Modeling Visualization*, 2008.

[8] Stantum. Stantum, unlimited multitouch, 2010. `http://www.stantum.com/en/`.

[9] K. Shoemake. Arcball: A user interface for specifying three-dimensional orientation using a mouse. *Proceedings of Graphics Interface*, pages 151–156, 1992.