PROJECT REPORT

Merging of 3D Objects

Submitted by
Ioannis Atsonios∗
Gagan Bansal†

Under the supervision of
Dr. Michael Misha Kazhdhan
Department of Computer Science
Johns Hopkins University

December 10, 2007

∗iatsonios@gmail.com
†gagan.bansal@gmail.com
1 Introduction

Modeling of 3D geometric objects is a difficult and time-consuming task, due to their complex geometric structure. Instead, mesh editing is used to create new 3D objects by compositing the already existing 3D models, so it is crucial that the task should be computationally tractable and efficient. Albeit the method should minimize any visual artifacts that may be created at the stage of merging. This task is an optimization problem that can be encapsulated by a Poisson equation with optimal boundary conditions.

2 Problem Statement

Given a source mesh object and a target mesh object, we want to cut a part of the source mesh and then paste over some specified part of the target mesh. We work with a constrained version of the problem where, as the user specifies two clipped meshes which are aligned in 3D space. The user also marks curves on both the meshes specifying the boundaries on both the meshes. The following illustrates the input to the program:

![Figure 1: Input Meshes](image)

3 Approach

Given two space-aligned meshes and two curves specifying the boundaries over the two meshes, the first task at hand is to find the correspondence between the vertices on both the meshes. There may be a scenario where the number of vertices on the two boundaries is not the same. We solve this problem by introducing new points on the boundary having lesser number of points. Let the boundaries be $B_1$ and $B_2$ corresponding to meshes $M_1$ and $M_2$. 


M2. Let $B1$ have lesser number of points than $B2$, then we do the following:

function addMorePoints($B1$)
begin
  while ($B1$ has lesser points than $B2$)
  begin
    select the longest edge on $B1$
    insert a point at the mid point of this edge
    Add triangles to mesh $M1$ corresponding to new vertex added
    update $B1$
  end
end

Figure 2: Correspondance between two meshes

Once we have same number of points on the boundary, the next task is to find the correspondance between the two boundaries. Let both $B1$ and $B2$ after refinement have $n$ vertices. We start by finding the closest vertex in $B2$ to each vertex of $B1$. So for each of the $n$ vertices we have its closest vertex in $B2$. We take Euclidean distance to find the closest vertex. Note that this works fine because we had earlier assumed that input meshes are spacially aligned. Once we have the closest neighbors for each vertex we define a correspondance by assigning one vertex at a time to its closest vertex and assigning rest of the vertices accordingly. So in total we get a total of $n$ correspondances. For each correspondance we find the error function which is defined as the difference between the distance between two assigned vertices and distance of a that vertex to its closest vertex. We do this over all correspondances and find the correspondance with the least error. This gives us a mapping between the vertices on $B1$ and $B2$.

After we have the correspondance we formulate the linear system. Sup-
pose we wish to solve the system over mesh $M_1$, we take the constraints as vertices on $B_1$ mapped to corresponding vertices on $B_2$. We used uniform weight Laplacians. For each vertex we defined its laplacian as

$$L(v_i) = v_i - \frac{1}{d_i}(\Sigma v_j)$$

where $d_i$ is the degree of vertex $i$ and $j$ iterates over all neighbors of $v_i$. In Matrix form the Laplacian Matrix can be written as

$$L = I - D^{-1}A$$

where $A$ is the adjacency Matrix for the mesh and $D = \text{diag}(d_1, d_2, ..., d_n)$

The linear system is

$$LV = \Delta$$

But instead of this system we solve another system which is $L'V = C$ where $V$ is Matrix corresponding to all vertices of $M_1$ (we are solving the system over $M_1$) The first $n$ entries are the coordinates of corresponding coordinates of boundary vertices of $M_1$ in $M_2$. Let $M_1$ have $m$ vertices. The next $m$ entries of $C$ are the laplacian coordinates of $M_1$. $L'$ is $(m + n) \times m$ matrix. The first $n$ rows of $L'$ have all zeroes except one position corresponding to the boundary vertex. The last $m$ rows is $L$ matrix. So this is an overconstrained system and solved in least squares sense. The corresponding linear system is

$$L'^T L'V = L'^T C$$

We solve this system using our linear solver for $V$ to get the new 3D coordinates of $M_1$

4 Implementation

We implemented the whole procedure in C++ using OpenGl and Glut libraries. The input were two meshes in OFF format (vertices and triangles). The two boundaries were manually specified in the code. More specifically we constructed by using parametric functions (such as ellipsoid-like meshes, spherical-like meshes, random mesh etc) in Matlab and saving them in OFF format. Unfortunately it was very hard task to find input for our program in the specific form (clipped and spatially aligned meshes) that we wanted to be processed by the algorithm, the processing especially for large input (which can have more interest) was almost prohibitive. We made a
library of functions for Matrix operations like the arithmetic operation, inverse of matrix.
We initially looked for some standard packages for solving the linear systems. We found TAUCS package but were unable to link that to our code so instead we wrote our own linear solver for solving $AX = B$ based on LU decomposition using partial pivoting. We tested our results against that of Matlab and the results of our linear solver were quite comparable to that of Matlab.

5 Results

![Merging of two tetrahedrons](image.png)

Figure 3: Merging of two tetrahedrons

The meshes merge well for the limited examples we tested.

6 Future Work

We propose some series of directions of future work of this project:

- To employ highly local operations only in the region of interest (namely boundaries) in order to prevent the propagation of numerical errors of the errors to the residual mesh.
• Thorough treatment of scalability of the algorithm, especially for the case of huge 3d meshes. We conjecture this task is computationally hard and very interesting in terms of theoretical advancement, due to the fact that we have to deal with a very sparse system, with sophisticated ways (possibly) to traverse the graph.
Figure 6: Merging of cylinder (shown in green) with a random mesh. The cylinder deforms

- We would like to get more accustomed with smoothness operations over meshes not only in terms of obtaining aesthetically better results, but more interestingly for the mathematics that dictate that operation.

- We would like to explore the compression of meshes in terms of using differential operations. This idea mostly motivated by the notion of differential compression that is done in 1d signal, we would like to expand to the 3d cases and more interestingly to employ graph operations over meshes, such as decomposition, sparsification (especially if we can employ randomness in that step).

- We would like to explore the instance that we have not aligned boundaries, is very interesting not only practically but also theoretically (what type of metric we can use, what type of operations to surpass the inherent ill posed property that dictates that type of instances).

7 References

[2] Differential coordinates for local mesh morphing and deformation, Marc Alexa
[4] Laplacian Surface Editing, Olga Sorkine et al