Implementation of a Freeform Modelling Tool

Robert Jendersie, Johannes Hauffe

June 18, 2020

1 Introduction

This report aims to explain implementation details of our multi-resolution freeform modelling tool. Both in terms of control metaphor and mathematical foundation, we follow the approach described in [1]. Thus, we focus on aspects which are either different in our implementation or only mentioned briefly in the original paper.
In addition, we give technical background for the occurring *parameters*, many of which are exposed to the user to allow for experimentation and handling of vastly different meshes.

2 Smooth Deformation

Same as in [1], we perform the smooth deformation by minimizing a certain energy functional of the surface with respect to some boundary conditions. For the practical implementation on a triangle mesh we use discretization of the Laplace-Beltrami operator from [2]. For the vertices $P = \begin{bmatrix}p_1, \dots, p_n\end{bmatrix}^T$ of the mesh, it can be written in matrix form as

$$\triangle = M^{-1}L, \tag{1}$$

where M is the diagonal matrix of vertex areas

$$m_{ii} = 2A_{\text{mixed}}(p_i),$$

and L is the sparse symmetric operator of edge weights e_{ij}

$$l_{ij} = l_{ji} = \begin{cases} e_{ij} & i \neq j, \text{edge i-j exists} \\ -\sum_{p_k \in N_1(p_i)} e_{ik} & i = j \\ 0 & \text{else} \end{cases},$$

where $N_1(p)$ is the one-ring of p. To improve numerical robustness for poor triangulations, the edge weights are computed by clamping extreme cot angles to the range $[0 + c, 180^\circ - c]$

$$\begin{aligned} e_{ij} &= \max(0, f(\cot\alpha_{ij}) + f(\cot\beta_{ij})) \\ f(x) &= \max(\cot c, \min(x, \cot(180^\circ - c))). \end{aligned}$$

We use $c = 3^\circ$.
With (1) we can introduce the higher *order* operators $\triangle^k$ where $k = 1$ deforms the surface like a membrane by minimizing the area, $k = 2$ characterizes thin plate surfaces which minimize surface bending and $k = 3$ minimizes curvature variation. At the boundary, this behaviour can be smoothly interpolated pointwise by introducing diagonal matrices D_l

$$\hat{\triangle}^2 = M^{-1}LD_1M^{-1}L \tag{2}$$

,

$$\hat{\triangle}^3 = M^{-1}LD_2M^{-1}LD_1M^{-1}L, \tag{3}$$

with $d_{k_{ii}} = \lambda_k(p_i)$ from [1]. This value can be adjusted by the user as *smoothness* for boundary and handle points. Finally, we can look at solving $\hat{\triangle}^k = 0$. Taking only the rows $\hat{\triangle}^k_{sup} = \begin{bmatrix}L_1 L_2\end{bmatrix}$ acting on the support vertices s and the fixed boundary vertices b we get

$$\begin{bmatrix} L_1 & L_2 \\ 0 & I \end{bmatrix} \begin{bmatrix} s \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix},$$

which leads to the sparse system with non-trivial solution

$$L_1 s = -L_2 b. \tag{4}$$

In all cases L_1 is positive definite and in the context of solving the system (4), both (1) and (2) can be easily made symmetric, since multiplication by M from the left effectively removes the leftmost M^{-1}. For (3) this only works if $D_1 = D_2$, that is, when no interpolation is done. Thus, if applicable, we employ a sparse LDL^T decomposition and fall back to a sparse LU decomposition for the latter case.
Although either decomposition needs to be done just once and solving the system afterwards is relatively fast, introducing precomputed basis functions as described in [1] further improves performance. Instead of picking affinely independent points from the handle h, we always use the orthogonal frame

$$Q \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} h & 1 \end{bmatrix},$$

to find the matrix $Q \in \mathbb{R}^{H \times 4}$ of affine combinations.

3 Detail Preservation

We implement a multi-resolution editing approach based on point wise displacement vectors, as described in [3].

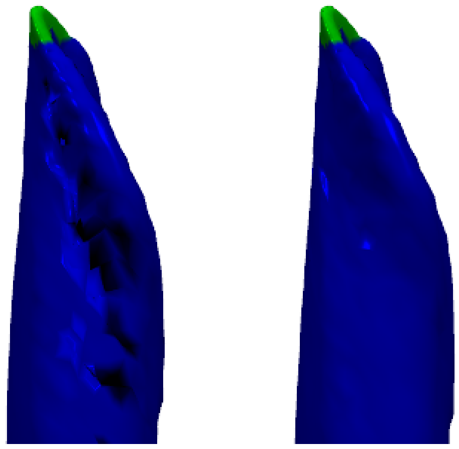

Figure 1: Detail reconstruction using the local frame of each vertex (left) and the frame with the shortest displacement vector in the 4-ring neighbourhood (right).

To preserve high frequency components in the support area, the details need to be first extracted and then reapplied to the modified mesh. We either use the displacement resulting from the initial solution of (4) without changes to the handle, or perform *implicit smoothing* of the form

$$(I - dt\triangle^k)s = s_0, \tag{5}$$

to the support region with initial points s_0. The choice of a reasonable time-step dt varies per mesh and can be adjusted by the user as *strength*. To maintain the full details, the *smoothing order k* should be the same as for the deformation, but in some cases a different order is more robust. Similar to (4), we can integrate the fixed boundary and make the system symmetric to then solve

$$(M_1 - dtL_1)s = M_1 s_0 + dtL_2 b, \tag{6}$$

with the sparse LDL^T decomposition, where M_1 are the area weights of vertices associated with s. Since the Laplace-Beltrami operator (1) is used, the points should only move in normal direction. Thus, the resulting displacement is encoded per vertex in a local frame, defined by its normal and one edge. Optionally, other vertices in the *n-ring* can be considered to find the local frame where the displacement vector has the smallest length. Also proposed in [3], this leads to fewer anomalies in the reconstructed surface, as seen in Figure 1.

4 Results

To evaluate the usability of our tool, we measure the performance for both modifications to the mesh and parameter changes of the operator. Times are measured on a i5-6600k (4 × 3.50GHz) with 16GB DDR4 running Win10 Pro 64bit. The test scenario consists of a mesh with 125k vertices, with a handle region containing 15k and the support region 35k vertices.

Table 1: Measurements for updates to the operator.

parameter	time [ms]
order 1	276.77
order 2	922.87
order 3	2262.19
smoothness	5692.42

Table 2: Measurements for changes to the smoothing.

parameter	time [ms]
details order 1	212.81
details order 2	830.77
details order 3	2168.81
search ring 1	89.59
search ring 5	239.42
search ring 10	725.02

First we consider modifications to the mesh. Updating the support vertices takes just 1.6ms, adding the details another 21.6ms, resulting overall in a smooth framerate for editing.
Parameter changes require recomputing the higher order operator and precomputing (4). Times for this are recorded in Table 1. A higher order significantly increases computation times and so does the switch to the *LU* solver for smoothness. Still, a wait time of up to 6s makes it possible to just try out different configurations during editing.
Similar values are observed for the smoothing-step in Table 2. Considering that the editing already requires a fixed boundary, the extra implicit smoothing is only useful in edge cases where full reconstruction is not feasible due to strongly stretched or compressed areas. Thus, this time can mostly be saved. A larger seach ring however, can be beneficial in both cases. Above a value of 10, few differences where observed and considering the extra cost of around $\frac{1}{3}$ of a third order operator computation in this case, this option can always be used.

References

[1] M. Botsch and L. Kobbelt, "An intuitive framework for real-time freeform modeling," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 630–634, 2004.

[2] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," in *Visualization and mathematics III*, pp. 35–57, Springer, 2003.

[3] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, "Interactive multi-resolution modeling on arbitrary meshes," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 105–114, 1998.