A Note on Boundary Constraints for Linear Variational Surface Design

Andrew Nealen
TU Berlin

Olga Sorkine
TU Berlin

Abstract

This note presents a proof for the subspace property of linear variational methods for fair surface design. Specifically, we show that any surface geometry computed from positional constraints which lie on a linear subspace will be forced to lie in this same subspace.

Keywords: Fair Surface Design, Linear Variational Surfaces

1 Introduction

In the process of creating FiberMesh [Nealen et al. 2007], we have implemented both Least-Squares Meshes (LSM) [Sorkine and Cohen-Or 2004; Sorkine et al. 2005] and the Botsch/Kobbelt (BK) framework [Botsch and Kobbelt 2004] for fair surface computation. In 2D MATLAB implementations, this works fine as long as the positional constraints (also known as the *anchors*) are not collinear, as we can see from the examples in Fig. 1.

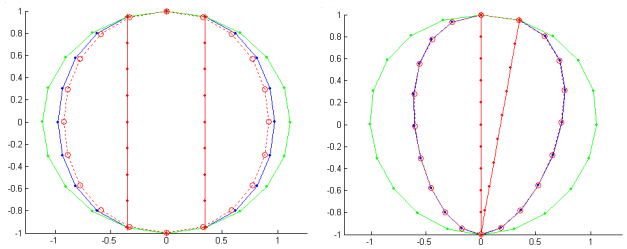

Figure 1: MATLAB *examples where anchor points (red crosses) are not collinear. The red dotted line is the LSM solution, while the red, blue and green solid lines are the BK solutions for* $k = 1$, $k = 2$ *and* $k = 3$ *respectively.*

When the anchors lie in a linear subspace (such as a line in 2D), all other vertices of this shape are forced to lie in this subspace as well (see Fig. 2).

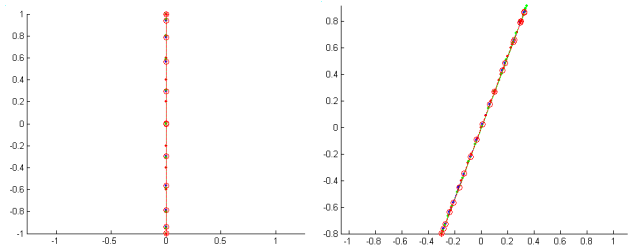

Figure 2: *Degenerate* MATLAB *examples with two (left) or 3 collinear anchors (right). Color coding as in Fig. 1*

Intuitively, this means that all vertices are computed as affine combinations of the anchor vertices (see Fig. 3). In the following, we show that the linear solutions of [Sorkine and Cohen-Or 2004] and [Botsch and Kobbelt 2004] both have this property, which makes them unsuitable for *inflating* a mesh defined by a planar curve, since in a 2D sketching tool we will generally have constraints which lie on a plane (the two anchors in Fig. 2 can be seen as the cross section of a planar 2D sketch). Adding positional constraints perpendicular to the sketch plane works, which is analogous to implementing higher order boundary constraints, yet requires manual placement, which is a nontrivial task in a 3D modeling environment (see Fig. 1 for the cross section analogy).

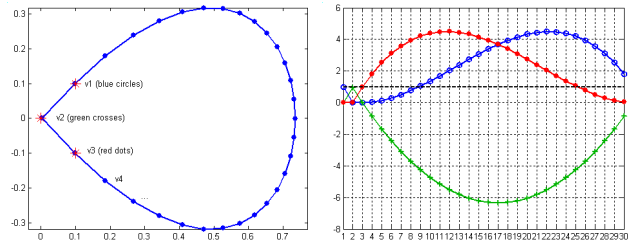

Figure 3: MATLAB *LSM example with three adjacent anchor points. The weights are high, therefore the smoothness constraints cannot be met (right, first three indices). Note that the horizontal dashed line is the sum of the three bases.*

2 Least-squares Meshes

Suppose we have connected mesh topology M with n vertices. Denote by $\mathbf{L}$ the Laplacian matrix with $\mathbf{L}(1,\ldots,1)_n^T = \mathbf{0}_n$, i.e. all rows of $\mathbf{L}$ sum up to zero. Some k vertices are tagged as anchor vertices; w.l.o.g. assume that the anchors are vertices $\{1,2,\ldots,k\}$. In [Sorkine and Cohen-Or 2004] the positions of all vertices are obtained as the solution of the following least-squares problem

$$\begin{aligned}&\underset{\mathbf{x}}{\operatorname{argmin}} \left\| \left(\begin{array}{c|c} \mathbf{L} \\ \hline \mathbf{I}_{k\times k} & \mathbf{0} \end{array} \right) \mathbf{x} - \begin{pmatrix} \mathbf{0}_n \\ \mathbf{x}'_k \end{pmatrix} \right\|^2 \\ &= \underset{\mathbf{x}}{\operatorname{argmin}} \left\| \mathbf{A}\mathbf{x} - \mathbf{b} \right\|^2 , \end{aligned} \tag{1}$$

where $\mathbf{I}_{k\times k}$ is the k by k identity matrix and $\mathbf{x}'_k = (x'_1,\ldots,x'_k)^T$ is the vector of prescribed anchor positions. While the matrix $\mathbf{L}$ has rank $n-1$, the rectangular matrix $\mathbf{A}$ involved in the least-squares problem above has full column rank after adding a single positional constraint (or more). Therefore, we can write the least-squares solution explicitly as

$$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}. \tag{2}$$

Observation 2.1. *The solution to Eqn. 2 is an affine combination of the anchor positions* $\mathbf{x}'_k$.

Proof. First we expose the structure of the vector $\mathbf{A}^T\mathbf{b}$ of length n, which has anchor positions in the first k entries, and zeros elsewhere

$$\mathbf{A}^T\mathbf{b} = \left(\mathbf{L}^T \left| \begin{array}{c} \mathbf{I}_{k\times k} \\ \hline \mathbf{0}_{(n-k)\times k} \end{array} \right. \right) \begin{pmatrix} \mathbf{0}_n \\ \mathbf{x}'_k \end{pmatrix} = \begin{pmatrix} \mathbf{x}'_k \\ \mathbf{0}_{n-k} \end{pmatrix}. \tag{3}$$

It is easy to see that:

$$\mathbf{A}^T\mathbf{A} = \left(\mathbf{L}^T\mathbf{L} + \left(\begin{array}{c|c} \mathbf{I}_{k\times k} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array} \right) \right). \tag{4}$$

Denote $\mathbf{1}_n = (1, \ldots, 1)_n^T$. Since $\mathbf{L}\mathbf{1}_n = \mathbf{0}_n$, also $\mathbf{L}^T\mathbf{L}\mathbf{1}_n = \mathbf{0}_n$. Thus:

$$(\mathbf{A}^T\mathbf{A})\mathbf{1}_n = \left(\mathbf{L}^T\mathbf{L} + \left(\begin{array}{c|c} \mathbf{I}_{k\times k} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right)\right)\mathbf{1}_n = \begin{pmatrix} \mathbf{1}_k \\ \mathbf{0}_{n-k} \end{pmatrix} \quad (5)$$

Let us multiply the last equation by $(\mathbf{A}^T\mathbf{A})^{-1}$ on the left side:

$$(\mathbf{A}^T\mathbf{A})^{-1}(\mathbf{A}^T\mathbf{A})\mathbf{1}_n = (\mathbf{A}^T\mathbf{A})^{-1}\begin{pmatrix} \mathbf{1}_k \\ \mathbf{0}_{n-k} \end{pmatrix}$$

$$\mathbf{1}_n = (\mathbf{A}^T\mathbf{A})^{-1}\begin{pmatrix} \mathbf{1}_k \\ \mathbf{0}_{n-k} \end{pmatrix} \quad (6)$$

The last equality tells us that the sum of the first k elements of each row of $(\mathbf{A}^T\mathbf{A})^{-1}$ is equal to 1. These multiply the first k elements of $\mathbf{A}^T\mathbf{b}$ in Eqn. 2, which shows that the solution $\mathbf{x}$ is an affine combination of the anchors $\mathbf{x}'$.

□

3 Botsch/Kobbelt Bases

In [Botsch and Kobbelt 2004] the anchors are interpolated instead of approximated. Therefore, we are dealing with disjoint sets of vertices (in contrast to [Sorkine and Cohen-Or 2004] where the anchors are a subset of all vertices). We denote the set of k fixed vertices as $\mathbf{F}$, the set of n free vertices as $\mathbf{P}$ and $N = n + k$. Note that the fixed vertices are only required to enforce boundary constraints around the free region, since their positions are known a priori (their smoothness constraints are also dropped from the system, as we will see below).

First, the $N \times N$ Laplacian Matrix $\mathbf{L}$ is raised to the desired power p (where $p = 1$ for the membrane, $p = 2$ for the thin-plate and $p = 3$ for the minimum variation solution). W.l.o.g. assume that the k fixed vertices have indices $1, \ldots, k$ in the system, and the free vertices have indices $k + 1, \ldots, N$, then for the smoothness constraint we have

$$\mathbf{L}^p \begin{pmatrix} \mathbf{F} \\ \mathbf{P} \end{pmatrix} = \mathbf{0}. \quad (7)$$

The smoothness constraints on vertices in $\mathbf{F}$ are removed from the system by omitting the first k rows of $\mathbf{L}^p$, resulting in the $n \times N$ matrix Δ^p. By adding the boundary conditions back to the system, we arrive at the formulation used in [Botsch and Kobbelt 2004]

$$\left(\begin{array}{c|c} \Delta^p \\ \hline \mathbf{I}_{k\times k} & \mathbf{0} \end{array}\right)\begin{pmatrix} \mathbf{F} \\ \mathbf{P} \end{pmatrix}$$

$$= \left(\begin{array}{c|c} \Delta^p_{\mathbf{F}} & \Delta^p_{\mathbf{P}} \\ \hline \mathbf{I}_{k\times k} & \mathbf{0} \end{array}\right)\begin{pmatrix} \mathbf{F} \\ \mathbf{P} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{F} \end{pmatrix}. \quad (8)$$

This is for notational convenience only. Here, we use $\Delta^p_{\mathbf{F}}$ for the upper left $n \times k$ matrix and $\Delta^p_{\mathbf{P}}$ for the upper right $n \times n$ matrix. By dropping the bottom k rows of Eqn. 8 and rearranging we get

$$\mathbf{P} = (\Delta^p_{\mathbf{P}})^{-1}(-\Delta^p_{\mathbf{F}})\,\mathbf{F} = \mathbf{B}_{BK}\mathbf{F}. \quad (9)$$

The $n \times k$ matrix $\mathbf{B}_{BK}$ can be interpreted as the matrix of basis functions. This basis suffers from the subspace property, which is equivalent to saying that the basis vectors (the columns of $\mathbf{B}_{BK}$) sum to 1.

Observation 3.1.

$$(\Delta^p_{\mathbf{P}})^{-1}(-\Delta^p_{\mathbf{F}})\,\mathbf{1}_k = \mathbf{1}_n. \quad (10)$$

Proof. We know that the rows of Δ^p sum to zero, since these are rows of the original Laplacian matrix. This can also be written as

$$(\Delta^p_{\mathbf{F}})\,\mathbf{1}_k + (\Delta^p_{\mathbf{P}})\,\mathbf{1}_n = \mathbf{0}_n. \quad (11)$$

Rearranging and multiplying from the left with $(\Delta^p_{\mathbf{P}})^{-1}$ yields

$$\mathbf{1}_n = (\Delta^p_{\mathbf{P}})^{-1}(-\Delta^p_{\mathbf{F}})\,\mathbf{1}_k \quad (12)$$

Which shows that the bases in [Botsch and Kobbelt 2004] sum to 1 on each mesh vertex.

□

4 Conclusions

To overcome these issues, we have implemented a fast approximation of the full nonlinear solution in our FiberMesh tool [Nealen et al. 2007] (Fig. 4). This system runs at interactive rates for reasonably sized models (up to a few thousand vertices) on a 1GHz pentium laptop. The method furthermore avoids the common problem of *curvature concentrating near the positional constraints* by introducing edge length constraints near the anchors, thereby smoothing out the approximated surface metric.

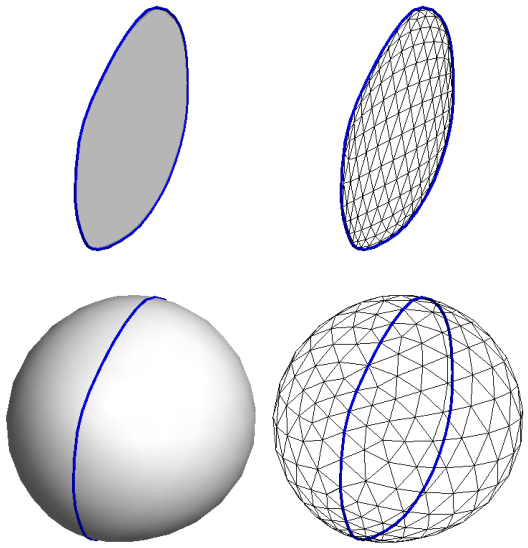

Figure 4: *The results of least-squares meshes (top) and our nonlinear solution (bottom) for positional constraints which lie on a planar curve.*

References

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3, 630–634.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. FiberMesh: Designing freeform surfaces with 3d curves. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 26*, 3. conditionally accepted.

SORKINE, O., AND COHEN-OR, D. 2004. Least-squares meshes. In *Shape Modeling International*, 191–199.

SORKINE, O., COHEN-OR, D., IRONY, D., AND TOLEDO, S. 2005. Geometry-aware bases for shape approximation. *IEEE Transactions on Visualization and Computer Graphics 11*, 2, 171–180.