Least-squares Meshes

Olga Sorkine
Tel Aviv University
sorkine@tau.ac.il

Daniel Cohen-Or
Tel Aviv University
dcor@tau.ac.il

Abstract

In this paper we introduce Least-squares Meshes: meshes with a prescribed connectivity that approximate a set of control points in a least-squares sense. The given mesh consists of a planar graph with arbitrary connectivity and a sparse set of control points with geometry. The geometry of the mesh is reconstructed by solving a sparse linear system. The linear system not only defines a surface that approximates the given control points, but it also distributes the vertices over the surface in a fair way. That is, each vertex lies as close as possible to the center of gravity of its immediate neighbors. The Least-squares Meshes (LS-meshes) are a visually smooth and fair approximation of the given control points. We show that the connectivity of the mesh contains geometric information that affects the shape of the reconstructed surface. Finally, we discuss the applicability of LS-meshes to approximation of given surfaces, smooth completion and mesh editing.

1 Introduction

This paper introduces Least-squares Meshes: meshes that are constructed from a given connectivity and approximate a set of control points in a least-squares sense. Given a planar graph with arbitrary connectivity and a sparse set of control points with geometry, we reconstruct the geometry of the rest of the mesh vertices by solving a sparse linear system. The linear system not only defines a surface that approximates the given control points, but it also distributes the vertices over the surface in a fair way. That is, each vertex lies as close as possible to the center of gravity of its immediate neighbors. The LS-meshes are a visually smooth and fair approximation of the given control points.

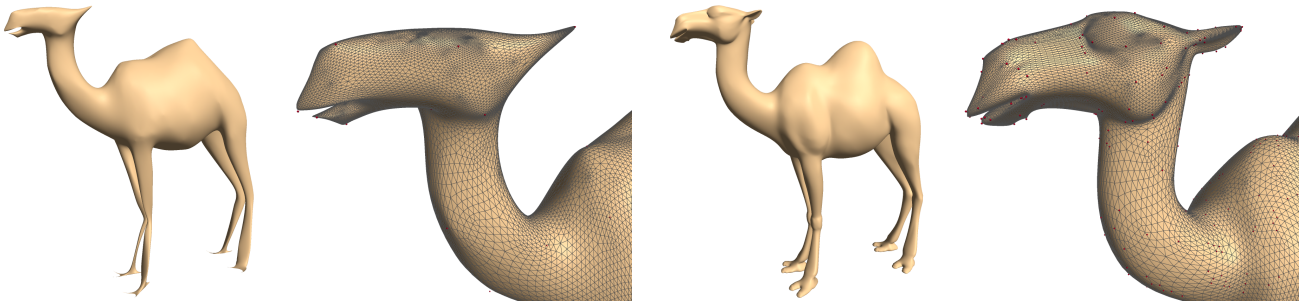
This paper also shows that by carefully selecting the control points, an LS-mesh can effectively approximate a given mesh. Figure 1 displays LS-meshes with the connectivity and control points of the camel model. Figure 2(a) shows a horse model consisting of 20K vertices. In 2(b), the same connectivity is used to approximate a subset of 1000 vertices of the model in 2(a). A close-up view of the LS-mesh is shown in 2(d) to demonstrate the fairing effect of the LS-approximation.

Common scattered data approximation techniques [7] receive an unstructured data set of points as input and fit a continuous surface that approximates (or interpolates) the points, while satisfying some desired conditions, such as smoothness properties. For visualization and further processing purposes, the continuous surface is then often sampled and triangulated. When implicit functions are fitted to approximate a set of points [3, 15, 17], they define a level set of a trivariate function from which the surface needs to be extracted. In contrast, an LS-mesh directly fits a prescribed mesh over a surface that approximates the points.

Our work is close to the convex interpolation method of Floater [6], where a given mesh is laid over a plane. As we shall describe in the following section, our method and Floater's method both distribute the mesh vertices fairly, while satisfying constraints given as control points. However, Floater's control points are *hard* constraints, while our control points are *soft* and satisfied in the least-squares sense. Floater's method reconstructs the geometry of the mesh in the plane, while our method reconstructs an arbitrary surface is 3D. In particular, LS-meshes can be constructed from arbitrary connectivity graphs, and generate shapes with genus higher than zero and surfaces which contain boundaries.

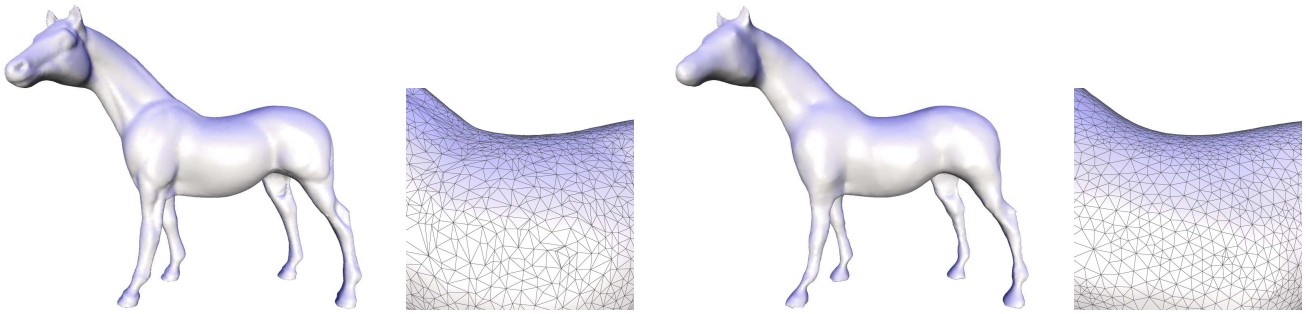
Another related area of work is the variational subdivision schemes [11, 12], which can be used to solve the scattered data interpolation problem. Kobbelt [12] applies Gauss-Seidel iterations to minimize the thin plate energy of the surface [14]. These iterations act as a smoothing filter on the mesh. Interleaving the Gauss-Seidel iteration steps with mesh subdivision steps generates a smooth surface from an initial sparse mesh.

The rest of the paper is organized as follows. The next section reviews the mathematical background needed to construct an LS-mesh. In Section 3 we discuss the properties of LS-meshes. Different strategies to obtain LS-meshes that approximate a given shape are shown in Section 4. Section 5 addresses the algorithmic issues of solving the linear



(a) (b) (c) (d)

Figure 1. LS-mesh: a mesh constructed from a given connectivity graph and a sparse set of control points with geometry. In this example the connectivity is taken from the camel mesh. In (a) the LS-mesh is constructed with 100 control points and in (c) with 2000. The connectivity graph contains 39074 vertices (without any geometric information). (b) and (d) show close-ups on the head; the control points are marked by small dots.



(a) (b) (c) (d)

Figure 2. (a) The original horse model, 19851 vertices; (b) close-up on the original connectivity; (c) LS-mesh of the horse model with 1K control vertices; (d) close-up on the LS-mesh connectivity.

least-squares system. We discuss the results and applications in Section 6 and conclude in Section 7.

2 Overview

Let $G = (V, E)$ be the given mesh graph, where $V = \{1, 2, ..., n\}$ is the set of vertex indices and E is the set of edges. We denote by $\mathbf{v}_i$ the (unknown) location of vertex i in space. The following equation defines a fairness and smoothness condition for vertex $\mathbf{v}_i$ (similar to [6]):

$$\mathbf{v}_i - \frac{1}{d_i} \sum_{j:(i,j)\in E} \mathbf{v}_j = 0, \qquad (1)$$

where d_i is the valence of vertex i. If the above equation is satisfied, the vertex i lies in the center of gravity of its immediate neighbors.

Tutte [18] has shown that if we assume the points $\mathbf{v}_i$ to reside in the 2D plane, the above system defines a valid embedding of a planar graph onto a plane, provided that the boundary vertices of the graph are set to lie on the boundary of a convex polygon (and thus the boundary vertices and their corresponding equations are eliminated from the system). Floater [6] extended this result for systems that require the vertices to lie in any convex combination of their neighbors. Here, we would like to solve this system for 3D meshes, thus $\mathbf{v}_i$'s are assumed to be in $\mathbb{R}^3$. The linear system can be written in matrix form:

$$L\mathbf{x} = 0, \;\; L\mathbf{y} = 0, \;\; L\mathbf{z} = 0,$$

where $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are the n×1 vectors containing the x, y and z coordinates of the n vertices and L is the following n×n matrix:

$$L_{ij} = \begin{cases} 1 & i = j \\ -\frac{1}{d_i} & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

The matrix L is known as the *Laplacian* of the mesh [5, 10, 16]. The rank of L is $n - k$ where k is the

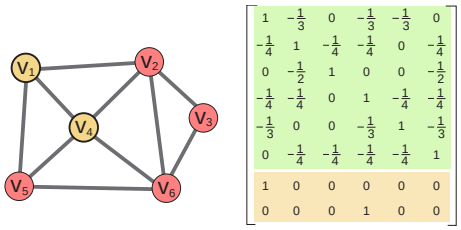


Figure 3. A small example of a graph and its corresponding matrix. The dark vertices are set as control points.

number of connected components in the graph G. Therefore, assuming that our mesh graph is connected, the rank of L is $n-1$, and there is a one-dimensional subspace of solutions for the system spanned by the vector $(1,1,...,1)^T$.

Without any geometric information given, the solution of the system is not interesting. Providing the 3D location for some m *control vertices* allows to get a non-trivial and unique solution. We add the equations of the control vertices:

$$\mathbf{v}_s = (x_s, y_s, z_s), \quad s \in C, \tag{2}$$

where $C = \{s_1, s_2, ..., s_m\}$ is the set of indices of the control vertices. Our system then becomes rectangular ((n+m)×n):

$$A\mathbf{x} = \mathbf{b},$$

where

$$A = \begin{pmatrix} L \\ F \end{pmatrix}, \quad F_{ij} = \begin{cases} 1 & j = s_i \in C \\ 0 & otherwise \end{cases}$$

$$b_k = \begin{cases} 0 & k \le n \\ x_{s_{k-n}} & n < k \le n+m \end{cases}$$

See Figure 3 for an example of a mesh with control vertices and the corresponding matrix. Adding at least one row for a control vertex to our system makes the A matrix full-rank. The system is solved in least-squares sense, i.e. we find $\mathbf{x}$ that minimizes

$$\|A\mathbf{x} - \mathbf{b}\|^2 = \|L\mathbf{x}\|^2 + \sum_{s \in C} |x_s - \mathbf{v}_s^{(x)}|^2. \tag{3}$$

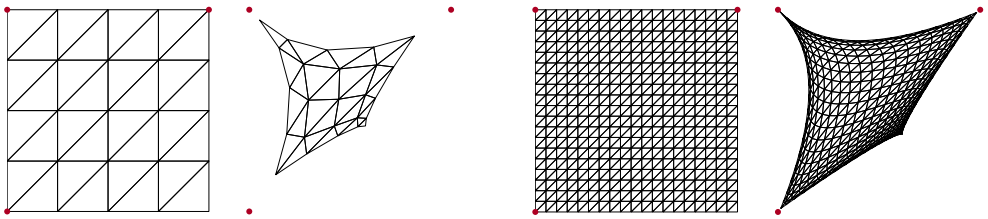
The unique analytical solution is $\mathbf{x} = \left(A^T A\right)^{-1} A^T \mathbf{b}$ since A has full rank. Note that element (i,j) of the matrix $A^T A$ does not vanish only if the graph distance between i and j is at most two. This implies that $A^T A$ is sparse (although not as sparse as A that only accounts for first-order neighborhoods).

Note that the control vertices play a conceptually similar role to the boundary vertices in Tutte's graph embedding solution. They constrain the system by augmenting the connectivity with some geometric information and allow to obtain an interesting solution. However, the major difference between Tutte's and Floater's boundary conditions and our control vertices is the approach to solving the system: while they force the constrained vertices to lie in the exact prescribed location, thus eliminating them from the system, we solve the system in least-squares sense, and the constraints are thus approximated, rather than interpolated. We keep the smoothness and fairness equations for the control vertices, and the resulting solution tends to respect these conditions at the constrained vertices, as well as at the free ones.

To obtain a better understanding and intuition for the claims above, take a look at Figure 4, which shows a simple 2D example. A 5×5 triangulated grid is shown in (a), where three of its vertices are used as control points. The geometric position of the rest of the grid points is meaningless and is only used to illustrate the given connectivity. An LS-mesh constructed from this connectivity with the three control points is shown in the second column. Note that the LS-mesh does not interpolate the three control points, and the vertices are not placed exactly at the average position of their neighbors. The error comes from the fact that there is no solution that can fully satisfy all the 25+3 constraints. When there are more elements in the mesh (see the 20×20 triangulated grid in (b)) the mesh is more flexible and the error is better distributed among the constraints. In particular, the control points are better satisfied.

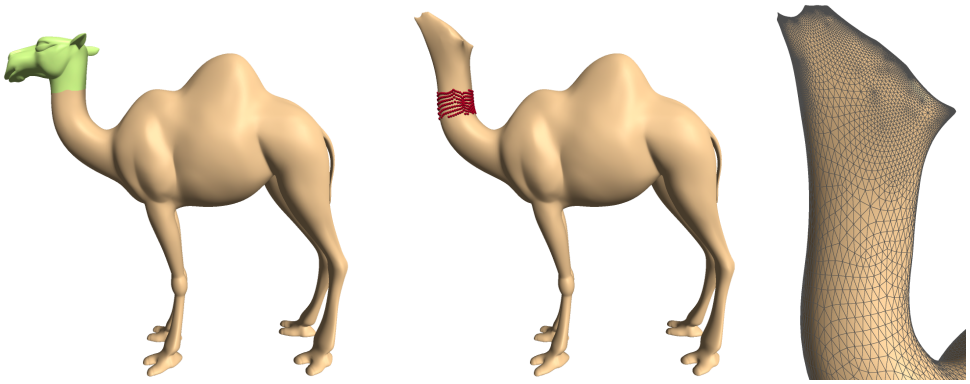
3 Connectivity Meshes

A 3D mesh consists of its connectivity information and the geometry, i.e., the 3D coordinates of the mesh vertices. We will refer to a mesh that has no geometry information as a connectivity mesh. A connectivity mesh has no shape. It can be generated from an arbitrary mesh by removing its geometry information. In our work we deal with what we can call an augmented connectivity mesh, where only a subset of the mesh vertices contains geometric information. Our linear least-squares system reconstructs the geometry of the mesh vertices while approximating the known geometry of the subset, and positions each vertex in the mean geometry of its immediate neighbors. Since the reconstruction system also accounts for the given connectivity of the mesh, it yields a shape which is close to the notion of connectivity shapes [9]. In their work, Isenburg et al. showed that a pure connectivity mesh has some natural shape, assuming that all the edges of the mesh are of equal length. They employ an iterative optimization process which minimizes an energy functional that inflates the mesh towards a smooth shape where the edges are close to uniform length. The optimization process is non-linear and requires to interleave some regularization steps so that the reconstructed shape



(a) 5×5 grid (b) 20×20 grid

Figure 4. LS-meshes of triangulated grids. The first and third columns visualize the connectivity of the given graphs; the geometric position of the grid vertices is meaningless. Three control points at the corners of the grid are used to generate the LS-meshes, their positions are marked by small dots. Note that since the LS-mesh satisfies the constraints in least-squares sense, some error is present: neither the control points are interpolated nor the fairness condition is precisely satisfied.



(a) (b) (c)

Figure 5. Reconstructing the geometry of the camel's head using the original connectivity. We removed the geometry from the head (marked in (a)). (b) The control points around the "hole" are marked by small spheres. (c) Close-up on the reconstructed geometry. Note that the connectivity of the head contains some information that induces non-trivial shape, without using a single control point.

bears some resemblance to the original mesh. However, the key contribution of their work is that it shows that a pure connectivity mesh contains some non-trivial geometric information.

In light of the above, an LS-mesh can be regarded as a non-pure connectivity mesh, where only some of its vertices contain geometric information. If the given connectivity mesh is meant to reconstruct a given shape, these geometry vertices act as control points assisting the reconstruction process to yield a better approximated shape. While the connectivity shapes strive to satisfy the uniform edge-length condition, in an LS-mesh the vertices satisfy flatness and fairness conditions in a least-squares sense. It is interesting to note that the smoothness term used as regularization in the optimization process of connectivity shapes [9] is in fact the Laplacian of the mesh.

When a small set of control points is used, the LS-mesh is reminiscent of the connectivity shapes in the sense that LS-mesh as well draws some non-trivial geometric information from the connectivity alone. This is demonstrated in Figure 7, where the legs on the left are reconstructed from a very sparse set of control points, and and in Figure 5, where the head of the camel model is reconstructed from the connectivity graph, using only the ring of vertices around the neck as control points.

4 Selecting the control points

LS-meshes can approximate a given mesh. A set of control points needs to be chosen in order to bring the surface of the LS-mesh close to the original mesh and minimize the geometric error. Intuitively, the control points should be

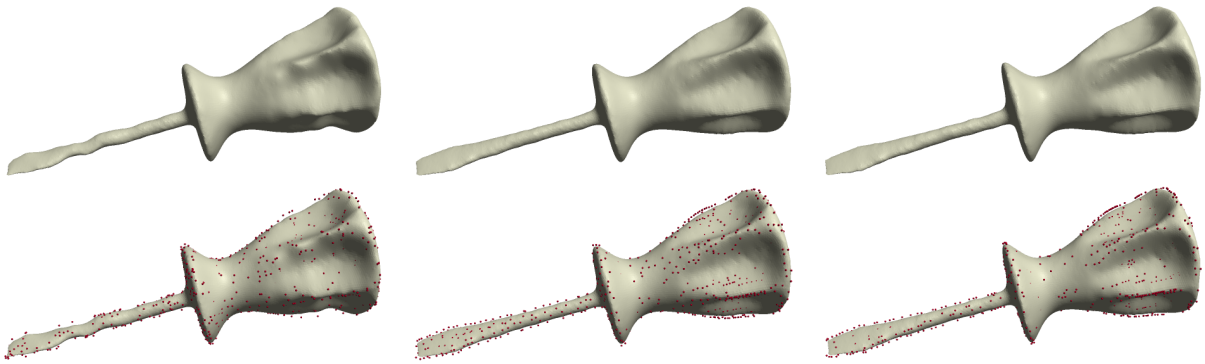


Figure 6. Different approaches to selecting the control points, applied to the screwdriver mesh (27152 vertices, 1000 control points). The bottom row shows the locations of the control points by small dots. In the left column, random selection was used. The middle column displays the greedy approach, which places one control point at a time in the vertex that attained the maximal error. In the right column the combined approach was used, where in each of 31 steps we found 31 local error maxima and placed the control points there, and then recomputed the LS-mesh to obtain the error estimation for the next selection step. One can see that random selection is inefficient since it does not "predict" places which will have large reconstruction error. On the other hand, both greedy selection and the combined approach work quite well and concentrate the control points in strategic regions, such as the edges of the screwdriver, where the reconstruction error is likely to be larger otherwise.

places in "strategic" locations on the surface, such as feature points, the tips of extruding parts and places where geometric detail is present. We have tested several strategies to select the control points:

- Random selection. This is a fast method, however if the original mesh is highly irregular and high-frequency details are present, random sampling might miss them out, which results in an inefficient approximation (see Figure 6(a)).

- One-by-one greedy selection. This method chooses one control point in each step by computing the LS-mesh induced by the current set of control points and placing the new control point at the vertex whose location in the LS-mesh has maximal error compared to its location in the original mesh. This method is slower since it requires solving the least-squares system in each step, but it is successful at identifying the "important" control points (such as points on features and details) and decreasing the approximation error.

- Combined local maxima method. With this strategy, we compute the LS-mesh every K steps and mark the vertex with maximal error as a control point, like in the greedy method. In the $K-1$ steps in between, we select control points by computing local maxima of the error. That is, we traverse the mesh in breadth-first order, starting from the last selected control point (that attains the global maximum of the error) and mark the vertices around it as "forbidden", as long as the error decreases. Then, we find the vertex with the maximal error among the vertices that were not marked. The process repeats $K-1$ times, and then we compute the new LS-mesh. This approach is faster than the greedy method since less solves are required, and it imitates the greedy method by fairly distributing the control points.

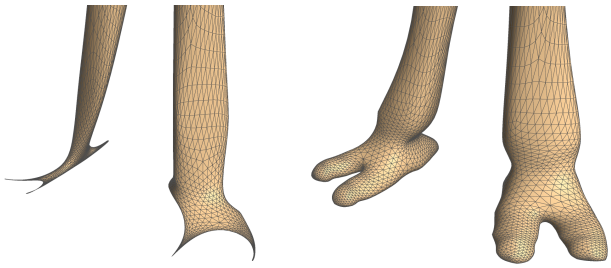
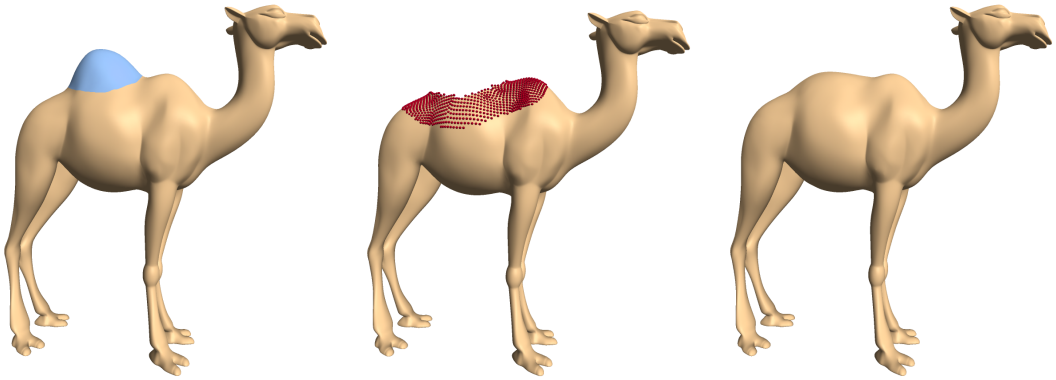


Figure 7. Close-up on the legs of the camel LS-mesh computed using 100 control points (left) and 2000 control points (right).

An example comparing the three approaches in shown in Figure 6. While the greedy approach seems to achieve the best distribution of the control points and hence the smallest geometric error, the combined approach is more practical in making the tradeoff between the approximation error and computation time.



(a) (b) (c)

Figure 8. Reconstructing the geometry of a hole using LS-meshes. (a) shows the original camel model, and the region of the hole is marked. We "removed" the geometry from the vertices of the hump and reconstructed it using the control points (marked by small spheres in (b)) and the connectivity of the hump. The reconstructed model is shown in (c).

Model	# vertices	Factor	Solve	Total
Eight	2,718	0.085	0.004	0.097
Horse	19,851	0.900	0.032	0.996
Screwdriver	27,152	1.646	0.068	1.850
Camel	39,074	2.096	0.073	2.315

Table 1. Running times (sec.) of solving the linear least-squares systems for the different models. *Factor* denotes the time spent on the factorization of the normal equations. *Solve* is the time to solve for one mesh function ($\mathbf{x}$, $\mathbf{y}$ or $\mathbf{z}$). The last column shows the total time to compute the LS-mesh.

5 Solving the system

LS-meshes require to solve a sparse linear least-squares system to reconstruct the geometry, i.e. to minimize $\|A\mathbf{x} - \mathbf{b}\|$. We use a direct method for solving the normal equations $A^T A\mathbf{x} = A^T\mathbf{b}$. A factorization of the coefficient matrix $A^T A = R^T R$ is found, where R is an upper triangular matrix. Then $\mathbf{x}$ (and $\mathbf{y}, \mathbf{z}$) is found by solving two triangular linear systems $R^T R\mathbf{x} = A^T\mathbf{b}$, that is $R^T\tilde{\mathbf{x}} = A^T\mathbf{b}$ and $R\mathbf{x} = \tilde{\mathbf{x}}$.

Table 1 records the solution times for the models used in our experiments on a 2.4 GHz Pentium 4 computer. The table shows the time to decompose the coefficient matrix of the normal equations into its triangular factors, the solution time for one mesh function ($\mathbf{x}$, $\mathbf{y}$ or $\mathbf{z}$ vectors) and the total solution time. The direct method is quite fast for moderately large meshes. Most of the time is spent on computing the factorization, while the time of the solving is negligible.

6 Discussion

As discussed above, LS-meshes can approximate shapes. As the number of control points gets smaller, the LS-mesh gets closer and closer to being a pure connectivity shape. Figure 11 shows a series of LS-meshes approximating different models with increasing number of control points. When the amount of control points is really small, the LS-mesh is distorted and bears almost no similarities to the original shape. However, as the number of control points increases, the reconstruction quickly gets closer to the original shape. By giving higher weight to the control points constraints, the LS-mesh can become closer to being interpolatory. The weights are incorporated by modifying equation (2) to the following:

$$w_s\mathbf{v}_s = w_s(x_s, y_s, z_s).$$

Thus, the energy minimized by the LS system has now the form

$$\|A\mathbf{x} - \mathbf{b}\|^2 = \|L\mathbf{x}\|^2 + \sum_{s\in C} w_s^2 |x_s - \mathbf{v}_s^{(x)}|^2.$$

As one can see in the 2D example in Figure 9, when the weight of the control points increases, the LS-mesh better approximates them at the expense of the fairness constraints.

LS-meshes can be potentially utilized for filling holes in surfaces. It is possible to adopt a framework similar to the one proposed by Lévy [13]. Lévy conformally embeds the mesh in the 2D plane, triangulates the hole in the parameter domain, and then reconstructs the geometry of the newly

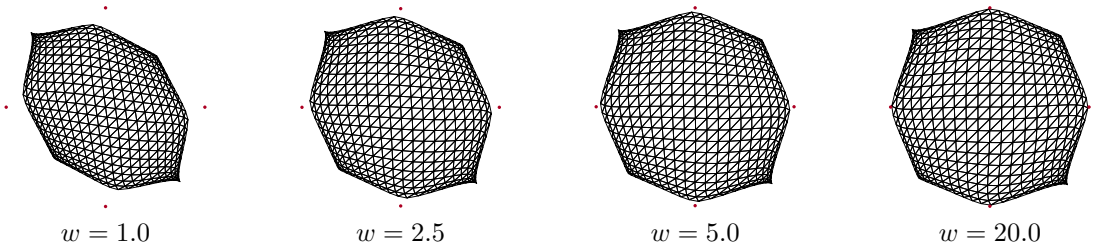


Figure 9. LS-meshes constructed with varying the weight of the control points constraints. The location of the control points is marked by small circles. As the weight increases, the LS-mesh becomes closer to interpolatory, at the expense of compromising the fairness conditions.

introduced vertices by minimizing a discrete curvature criterion. By solving our linear system, where the known vertices around the hole serve as control points, the geometry of the unknown vertices is reconstructed. See Figure 8, where the vertices around the hump were taken as control points, and the hump was reconstructed by solving the system on the reduced mesh (which included the control points and the connectivity of the hump). We do not need a parameterization of the mesh but just a triangulation of the hole. The resulting shape depends on the number of vertices used in the triangulation and the quality of the meshing. For instance, when taking the head of the camel and "removing" the geometry from it, and then reconstructing it by solving our system, we arrive at the non-trivial shape shown in Figure 5.

Another interesting area of application is shape modeling and editing. By manipulating the control points, a new LS-mesh is immediately implied. However, unlike in subdivision methods [19], here the connectivity is readily given and better fits to the subject geometry. This suggests that LS-meshes can be effective for editing existing shapes. See for example the horse in Figure 10, where we moved the control points of its head, resulting in an edited version of the same mesh. The reconstruction of the LS-mesh is very fast, assuming that the factorization of the normal equations matrix of the LS system is pre-computed, which needs to be done only once per LS-mesh. Then, to obtain the edited LS-mesh we only need to solve the system by back-substitution, which is very efficient, as discussed in Section 5 (in particular, refer to the timings in the *Solve* column of Table 1).

It is quite tempting to investigate the potential of LS-meshes for geometry compression and progressive transmission of 3D meshes. Previous work on progressive mesh compression employed construction of hierarchy for both the geometry and connectivity of the mesh [1, 8, 20]. Recent methods have concentrated on progressive representation of the geometry only [4, 10]. In that sense, LS-meshes are somewhat related to the Spectral Compression method [10], since in both cases, the connectivity of mesh

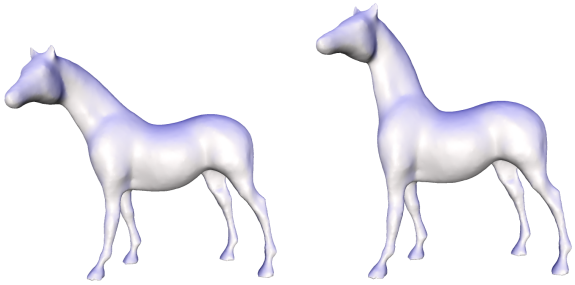


Figure 10. An example of editing the LS-mesh. On the left the original LS-mesh is shown; on the right the resulting LS-mesh after moving the control points on the head.

is assumed to be decoded before the geometry is decoded. Roughly speaking, the spectral method succeeds to reconstruct large models with a visually pleasing appearance using a few thousands of spectral coefficients. In terms of file size, this is equivalent to a few thousands of control points (see Figure 2(c) for the horse model approximated with 1000 control points). A fair comparison would require to agree on a visual metric and to draw distortions curves. While such a comparison is beyond the scope of this paper, we would like to emphasize that spectral methods tend to be optimal [2]. However, a progressive transmission of a triangular mesh based on LS-meshes can be advantageous in terms of its computational cost. The spectral method requires computing the eigenvectors of the mesh Laplacian matrix, which for large meshes is an extremely costly computation. In contrast, the solution of linear least-squares system is fairly simple and direct.

7 Conclusion

LS-meshes are conceptually simple and easy to implement. They provide means to reconstruct smooth meshes from a given connectivity mesh and a (possibly small) num-

ber of control points. The geometry of the mesh is reconstructed by solving a sparse linear system. LS-meshes can deal with high-genus shapes and surfaces which contain boundaries.

It is important to note that the uniform coefficients in the smoothness condition (Eq. 1) depend solely on the connectivity of the mesh. Other forms of discretization can be used if the mesh geometry is known. For example, Kobbelt [12] locally fits quadratic polynomials to the mesh to approximate second order derivatives. Then, the smoothness condition is expressed by non-uniform weights that vary over the mesh depending on the surface geometry and the characteristics of the meshing. In our ongoing research we are looking into various discretizations schemes of the smoothness condition, such as averaging of higher-order neighborhoods.

We feel that this work opens up a lot of new and interesting research directions, which we hope others will join us in exploring. These include:

- Understanding the behavior of the LS solution and the connection to subdivision surfaces. We refer the reader again to Figure 5, where the shape of the camel's head is reconstructed solely from the connectivity. Applying a subdivision scheme, starting from some coarse triangulation of the "chopped" neck, would reconstruct only a smooth cap. The machinery that recovers geometric information from the connectivity needs further exploration.

- Expanding the domain of LS-meshes to shapes with sharp features. Currently, LS-meshes handle only smooth surfaces. We would like to incorporate sharp edges, preferably in similar global LS framework.

- Analyzing the smoothness and the approximation properties of LS-meshes.

- Exploring the potential of LS-meshes for progressive transmission of meshes along the lines discussed above.

Acknowledgements

We would like to thank Sivan Toledo for insightful discussions and Christian Rössl for proofreading. This work was supported in part by grants from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities), by the Israeli Ministry of Science, by the German Israel Foundation (GIF) and by the EU research project 'Multiresolution in Geometric Modelling (MINGLE)' under grant HPRN-CT-1999-00117. The screwdriver mesh is courtesy of Cyberware.

References

[1] P. Alliez and M. Desbrun. Progressive compression for lossless transmission of triangle meshes. In *Proceedings of ACM SIGGRAPH 2001*, pages 198–205, 2001.

[2] M. Ben-Chen and C. Gotsman. On the optimality of spectral compression of meshes. Preprint, available online from *http://www.cs.technion.ac.il/∼gotsman/publications.html*, 2003.

[3] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

[4] P. H. Chou and T. H. Meng. Vertex data compression through vector quantization. *IEEE Transactions on Visualization and Computer Graphics*, 8(4):373–382, 2002.

[5] M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. Journal*, 23:298–305, 1973.

[6] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

[7] R. Franke and G. M. Nielson. Scattered data interpolation and applications: A tutorial and survey. In H. Hagen and D. Roller, editors, *Geometric Modelling, Methods and Applications*, pages 131–160. Springer Verlag, 1991.

[8] H. Hoppe. Progressive meshes. In *Proceedings of ACM SIGGRAPH 96*, pages 99–108, August 1996.

[9] M. Isenburg, S. Gumhold, and C. Gotsman. Connectivity shapes. In *Proceedings of IEEE Visualization 2001*, pages 135–142, 2001.

[10] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *Proceedings of ACM SIGGRAPH 2000*, pages 279–286, July 2000.

[11] L. Kobbelt. A variational approach to subdivision. *Computer Aided Geometric Design*, 13:743–761, 1996.

[12] L. Kobbelt. Discrete fairing and variational subdivision for freeform surface design. *The Visual Computer*, 16(3-4):142–158, 2000.

[13] B. Lévy. Dual domain extrapolation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, 22(3):364–369, 2003.

[14] H. P. Moreton and C. H. Séquin. Functional optimization for fair surface design. In *Proceedings of ACM SIGGRAPH 92*, pages 167–176. ACM Press, 1992.

[15] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, 22(3):463–470, 2003.

[16] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH 95*, pages 351–358, 1995.

[17] G. Turk and J. F. O'Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873, Oct. 2002.

[18] W. T. Tutte. How to draw a graph. *Proc. London Mathematical Society*, 13:743–768, 1963.

[19] J. Warren and H. Weimer. *Subdivision Methods for Geometric Design: A Constructive Approach*. Morgan Kaufmann Publishers Inc., 2001.

[20] J. C. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. In *Proceedings of IEEE Visualization '96*, pages 327–334., 1996.

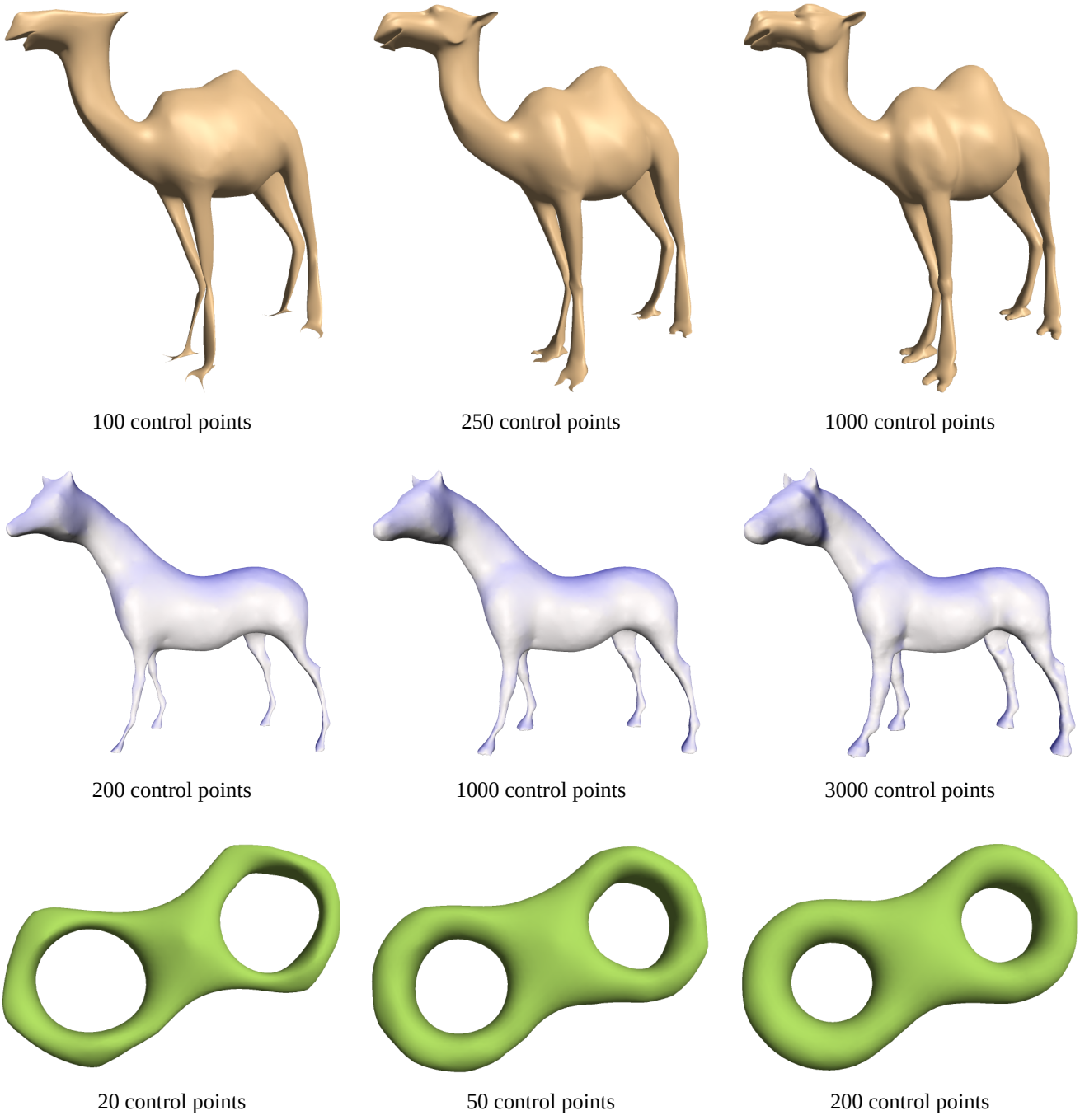


Figure 11. Examples of different LS-meshes. Each row displays LS-meshes computed using the same connectivity graph and a varying number of control points. Note that LS-meshes can have arbitrary topology, including genus greater than zero.