Encoding Meshes in Differential Coordinates

Daniel Cohen-Or Olga Sorkine

School of Computer Science
Tel Aviv University

Abstract

Representing surfaces in local, rather than global, coordinate systems proves to be useful for various geometry processing applications. In particular, we have been investigating surface representations based on differential coordinates, constructed using the Laplacian operator and discrete forms. Unlike global Cartesian coordinates, that only represent the spatial locations of points on the surface, differential coordinates capture the local surface details which greatly affect the shading of the surface and thus its visual appearance. On polygonal meshes, differential coordinates and the discrete mesh Laplacian operator provide an efficient linear surface reconstruction framework suitable for various mesh processing tasks. In this paper we discuss the important properties of differential coordinates and show their applications for surface reconstruction. In particular, we discuss quantization of the differential coordinates, Least-squares meshes and mesh editing.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: differential coordinates, Laplacian, least-squares meshes, mesh editing

1 Introduction

Differential encoding for mesh deformation has been an active research area in recent years. Early forms of differential encoding of meshes have been used for mesh compression. Using predictive schemes for encoding mesh geometry, such as the parallelogram rule, a vertex is encoded as the difference between the predicted position and its actual position. Better schemes are multi-way, that is, the prediction is based on an average of several predictions from different directions. The prediction based on the Laplacian operator takes into account all possible directions and thus yields better prediction than schemes based on a single way or just a few (see Table 2 in [Sorkine et al. 2003]).

The Laplacian operator will be described below. As we shall see, it has been proved to be more than a prediction scheme per se. The Laplacian operator has a nice, well studied mathematical formulation, for which we can apply spectral analysis and understand its behavior. With certain extension, the Laplacian operator allows to define discrete bases for surface geometry that are tailored to represent specific geometric features [Sorkine et al. 2005]. The Laplacian operator has lead to various developments in several applications, like mesh encoding, mesh deformation and mesh manipulation.

2 Mesh encoding

Polygonal meshes and triangular meshes in particular are commonplace and the de-facto standard for representing surfaces in computer graphics. The representation of a mesh $\mathcal{M}$ consists of its geometry and its topology. The geometry is a list of n Cartesian coordinates encoding the position of n vertices. The topology encodes the connectivity among the vertices, which defines the topology of the surface. The mesh represents a 2-manifold in 3D space, typically a closed, watertight model, with possibly a small number of boundaries b and/or handles g. From the Euler equation:

$$V - E + F = 2(1-g) - b = \chi(\mathcal{M}),$$

it can be deduced that when g and b are small,

$$2V = 3E = F.$$

Furthermore, the average degree (or valence) of a vertex is 6. These special properties of the graph connectivity lead to very efficient connectivity encoding techniques [Touma and Gotsman 1998; Alliez and Desbrun 2001; Alliez and Gotsman 2005].

Encoding the geometry remains a challenge [Alliez and Gotsman 2005]. The Cartesian coordinates are typically represented by high-precision floating-point numbers and as such they cannot be compressed by standard methods like LZ or arithmetic coding. Thus, some quantization is always necessary, and quantization necessarily introduces errors and some loss of data.

2.1 Quantization errors

Since quantization errors are inevitable, the question is how to visually or perceptually suppress them. Let us first understand the visual effect of quantizing the Cartesian coordinates of a mesh. Figure 1(a) shows a sphere coarsely tessellated with floating-points coordinates, and (b) shows the same sphere with its coordinates quantized. The two look the same; however, when we apply the same quantization over a finely tessellated sphere (Figure 1(c)), the visual degradation is clearly apparent (see Figure 1(d)). The quantization produces high-frequency errors across the surface, which alters the surface normals, and the finer the tessellation, the greater the relative quantization error.

Since our perception is highly sensitive to shading, the "jaggies" effect is irritating. Thus, only mild quantization of Cartesian coordinates is possible without causing visible artifacts. We claim that the fact that the quantization error is more severe on fine meshes is somewhat disappointing, since the investment in encoding fine meshes is counter-productive. Before we move on, we would like to make a disclaimer: the spheres in Figure 1 are rendered with flat

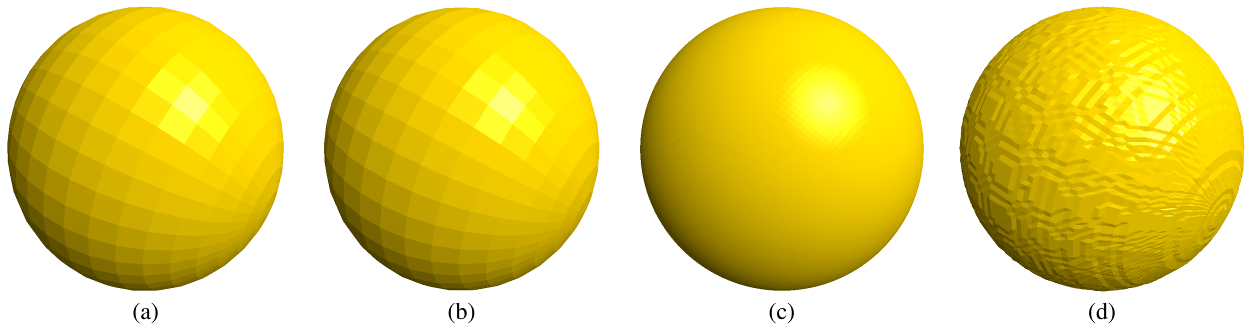


Figure 1: The counter-intuitive effect of Cartesian quantization. Fine-sampled surfaces suffer more than coarse ones. (a) A coarse mesh representation of a sphere with high-precision coordinates. (b) The same mesh with Cartesian coordinates quantized to 8 bits/coordinate. (c) A fine mesh representation of the sphere. (d) Quantization of the fine mesh to 8 bits/coordinate yields a jaggy surface.

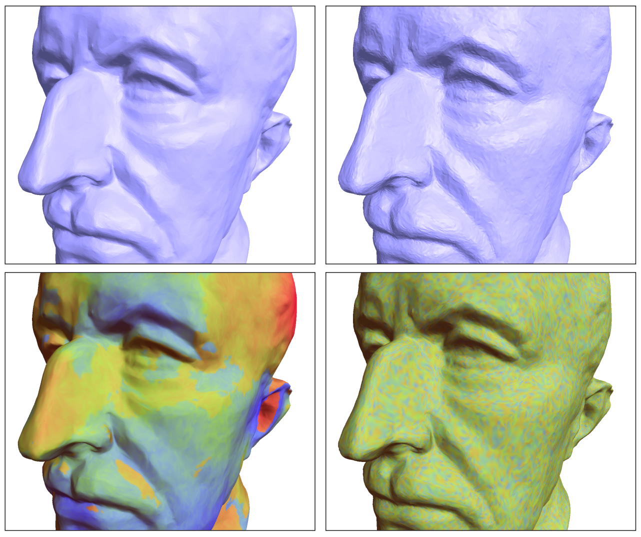


Figure 2: The Laplacian coordinates quantization to 5 bits/coordinate (left) introduces low-frequency errors, whereas Cartesian quantization to 11 bits/coordinate (right) introduces noticeable errors. The upper row shows the quantized model, and the bottom figures use color to visualize the corresponding quantization errors.

shading to emphasize the effect. Smooth shading can alleviate the jaggy appearance to some degree.

As will be discussed below, rather than encoding the geometry with Cartesian coordinates, one can encode the mesh with differential coordinates. Quantizing such differential coordinates can lead to a different visual effect, which we claim to be less visually irritating. Basically, the quantization introduces low frequencies, instead of high ones. The effect is shown in Figure 2. To explain this effect, we first need to introduce the particular differential coordinates used, namely the Laplacian coordinates, and apply some spectral analysis to understand their behavior under quantization.

2.2 The Laplacian operator

Let us define the *relative* or *δ-coordinates* of a vertex $\mathbf{v}_i$ to be the difference between the absolute coordinates of $\mathbf{v}_i$ and the center of mass of its immediate neighbors in the mesh,

$$\delta_i = (\delta_i^{(x)}, \delta_i^{(y)}, \delta_i^{(z)}) = \mathbf{v}_i - \frac{1}{d_i}\sum_{k=1}^{d_i} \mathbf{v}_{i_k} ,$$

where d_i is the number of immediate neighbors of $\mathbf{v}_i$ (the degree or valence of $\mathbf{v}_i$) and $\mathbf{v}_{i_k}$ is $\mathbf{v}_i$'s kth neighbor. The transformation of the vector of absolute Cartesian coordinates to the vector of relative coordinates can be represented in matrix form. Let A be the adjacency (connectivity) matrix of the mesh:

$$A_{ij} = \begin{cases} 1 & i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise,} \end{cases}$$

and let D be the diagonal matrix such that $D_{ii} = d_i$. Then the matrix transforming the absolute coordinates to relative coordinates (scaled by D) is $L = D - A$,

$$L_{ij} = \begin{cases} d_i & i = j \\ -1 & i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

That is, $L\mathbf{x} = D\delta^{(x)}$, $L\mathbf{y} = D\delta^{(y)}$, and $L\mathbf{z} = D\delta^{(z)}$, where $\mathbf{x}$ is a column n-vector containing the x absolute coordinates of all the vertices, and similarly for y and z. Without loss of generality, we now focus on the vectors $\mathbf{x}$ and $\delta = D\delta^{(x)}$.

The matrix L is called the Laplacian of the mesh. The Laplacian is symmetric, singular and positive semi-definite. The rank of L is $n - c$, where c is the number of connected components of the mesh. Since we are dealing with single component meshes, the rank is $n - 1$, which means that we can actually recover $\mathbf{x}$ from δ if we know, in addition to δ, the Cartesian coordinate x_i of one vertex $\mathbf{v}_i$. In the following we call the known vertex an *anchor*. Let us define L^{-1} as the operator that recovers $\mathbf{x}$ from δ. That is, $L^{-1}(\delta) = \mathbf{x}$.

Before moving on, note that the matrix L is defined based on the connectivity information alone, and in that sense it is geometry-oblivious.

2.3 Spectral analysis

Now, the interesting question is what happens if we quantize the δ-coordinates. Can we still go back to $\mathbf{x}$? Intuitively, the answer is yes, but with some small errors. But then, another interesting question is how the reconstruction error behaves. To understand the error behavior we need to apply spectral analysis. However, before that let us take a look at the visual effect. Figure 3(a) shows the

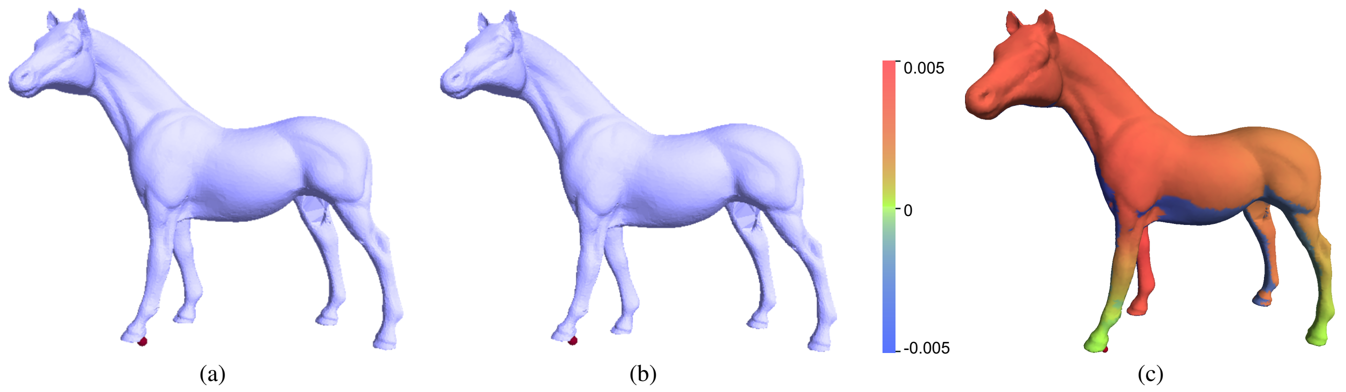


Figure 3: High-pass quantization in action. In (a) the original *Horse* model is displayed, while (b) shows the same model reconstructed from δ-coordinates quantized to 7 bit/coordinate. Evidently, high-pass quantization preserves the local surface details, which makes the quantized *Horse* almost undistinguishable from the original. In (c) we visualize the quantization error using pseudocolor; the error has mainly low-frequency components, which can be observed in the smooth error coloring.

original *Horse* model and 3(b) the *Horse* recovered from quantized δ-coordinates. That is:

$$\mathbf{x}' = L^{-1}(\delta'),$$

where δ' is the quantized version of δ. Since quantization introduces a small error, we can write

$$\delta' = \delta + \varepsilon.$$

Thus,

$$\mathbf{x}' = L^{-1}(\delta + \varepsilon) = \mathbf{x} + L^{-1}(\varepsilon).$$

So the difference between the two horses in Figure 3 is $L^{-1}(\varepsilon)$. The error is not visually noticeable. However, if one were to flip back and forth between the two images, some low-frequency error would be observed. The high frequencies, or the surface details, look the same, but at large, some global, low-frequency distortion is introduced. This suggests that the error vector $L^{-1}(\varepsilon)$ is of low frequency. Let us analyze L and L^{-1}. Note that L is a high-pass filter, that is, it attenuates low frequencies and amplifies high ones. This can also be observed at its eigenvectors. The eigenvectors of L associated with small eigenvalues are smooth (low-frequency), while the eigenvectors associated with high eigenvalues are oscillating (high-frequency). Now let us look at the spectrum of L^{-1}. The eigenvalues of L^{-1} associated with low-frequency eigenvectors are now large and amplify the low frequencies. The eigenvalues of L^{-1} associated with high-frequency eigenvectors are now small and attenuate or actually preserve the high frequencies. In particular, L^{-1} has amplified the low frequencies of ε, and consequently the low frequencies in the reconstruction error of the recovered vector $\mathbf{x}'$.

3 High-pass quantization

The above suggests different means for encoding meshes. Rather than quantizing the global Cartesian coordinates, which leads to high-frequency errors, one can quantize the δ-coordinates and introduce low-frequency errors. Such quantization has been termed High-Pass Quantization [Sorkine et al. 2003] since it preserves the high frequencies. As shown in [Sorkine et al. 2003], the gain in the compression ratio compared to conventional quantization is not necessarily significant, but the visual effect is different. High-pass quantization introduces low-frequency error, to which the human perception is much less sensitive than to high-frequency errors. High-pass quantization has the effect of preserving the details of the surface. This property is desirable for various applications for which the surface details are important, as discussed later.

3.1 Bounding the low-frequency error

Having a low frequency error across the surface suggests that it might be effective to "nail" the model in place by adding more than a single anchor vertex. Indeed, it can be shown that the more anchors we use, the lower the error. Anchors cost additional storage space, however in practice less than 1% of the model vertices need to be anchored for visually good reconstruction. While the addition of anchors is intuitive, it needs to be applied with care. Simply eliminating the anchors from the L matrix and the associated linear system by erasing the rows and the columns of the anchor vertices will produces spikes at the anchored vertices, since when we quantize δ, no smoothness constraints are posed on the anchors. Thus, we keep the Laplacian constraints of all the vertices and add the anchors as additional constraints. By adding anchors, the matrix becomes rectangular, so we solve the system in the least-squares sense:

$$\mathbf{x}' = \underset{\mathbf{x}'}{\operatorname{argmin}} \left(\left\| L\mathbf{x}' - \delta' \right\|^2 + \sum_{i \in C} \left| x'_i - c_i \right|^2 \right),$$

where C is the set of anchors and c_i are their positions. Assume w.l.o.g. that $|C| = k$ and $C = \{1, 2, \ldots, k\}$. The system above can be formulated as

$$\mathbf{x}' = \underset{\mathbf{x}'}{\operatorname{argmin}} \left\| A\mathbf{x}' - \mathbf{b} \right\|^2,$$

where

$$A = \left(\frac{L}{I_{k\times k} \mid 0_{k\times(n-k)}} \right), \quad (1)$$

$$\mathbf{b} = (\delta'_1, \ldots, \delta'_n, c_1, \ldots, c_k)^T. \quad (2)$$

This has the effect that $\mathbf{x}'$ is the solution that best agrees with the δ' constraints and the positional anchors. In [Sorkine et al. 2003] it has been shown that adding a rather small number of anchors significantly improves the visual error, and furthermore that the least-squares solution nicely distributes the error across the surface. That is, the error mainly consists of low frequencies and the details of the surface (the high frequencies) are preserved.

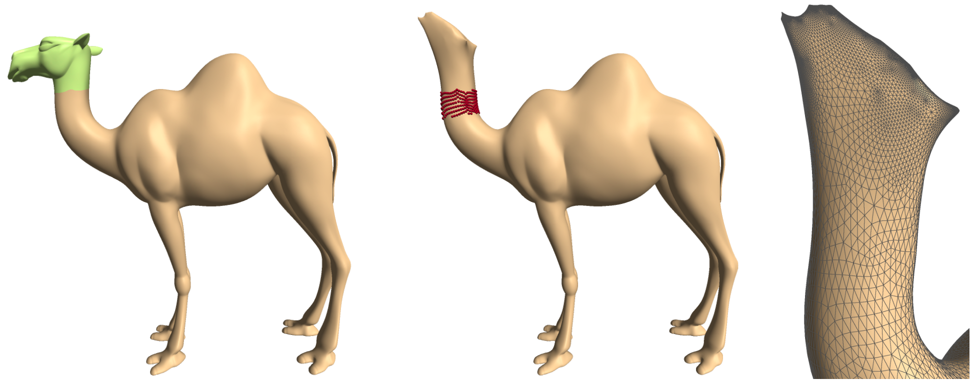

(a) (b) (c)

Figure 4: Reconstructing the geometry of the *Camel*'s head using the original connectivity. We removed the geometry from the head (marked in (a)). (b) The anchor 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 anchor point.

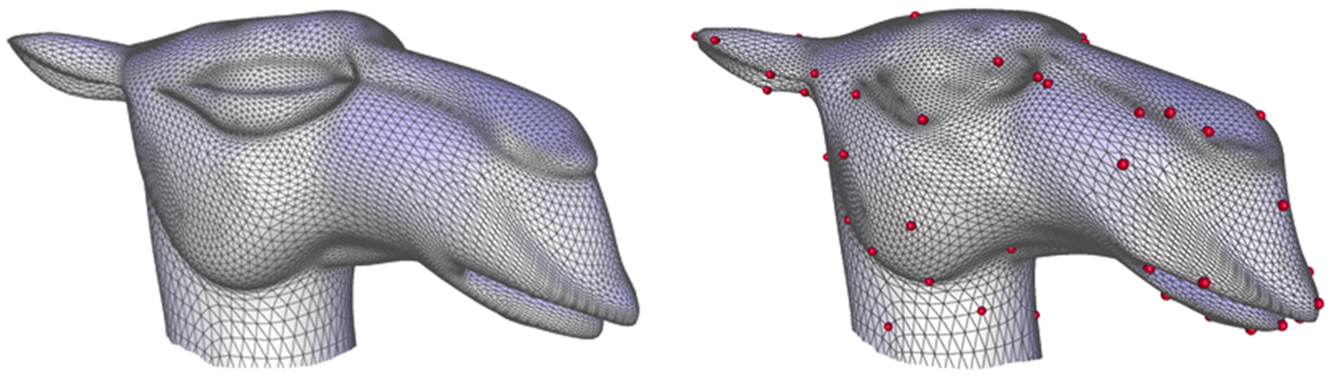

(a) (b)

Figure 5: Approximation using LS-meshes. (a) the original *Camel* head; (b) LS-mesh reconstruction with a few anchors demonstrates the approximation ability of non-pure connectivity meshes.

4 Shape from connectivity

Up to now we have seen the effect of quantizing the δ-coordinates. Yet, an interesting question is how far we can go with the quantization of the δ-coordinates. At the limit, what if we reduce the δ-information to zero bits? When no bits are allocated for the δ-coordinates, the solution of $\mathbf{x}' = L^{-1}(0)$ generates some geometry from nearly the connectivity alone (not entirely alone, because we still have a small number of anchors containing some geometry).

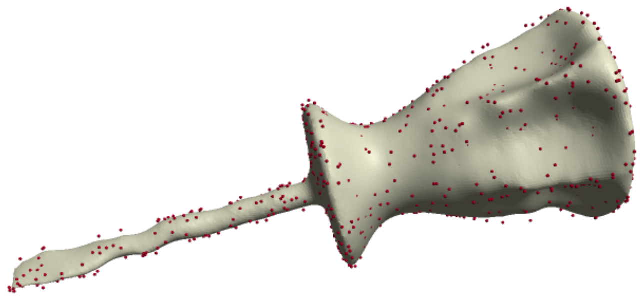
The answer is demonstrated in Figure 4. In this example, the head of the *Camel* is represented with no geometry, that is, differential coordinates with zero bits. The recovered head (Figure 4(c)) seems like a smooth version of the original head. It has some resemblance to the original head, and certainly contains some non-trivial geometry. It should be emphasized that the recovered head is reconstructed from the connectivity alone. The red vertices around the neck of the camel are anchor points which are used as constraints for solving the system $\mathbf{x}' = L^{-1}(0)$.

Solving the above linear least-squares system reconstructs the geometry of the mesh vertices, while approximating the known geometry of the anchors. 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 [Isenburg et al. 2001]. In their work, Isenburg et al. [2001] 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 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.

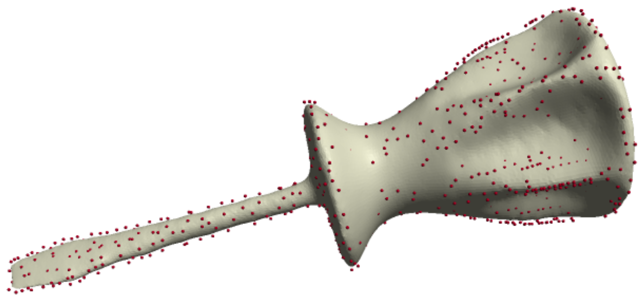
In light of the above, a mesh recovered by the least square solution can be regarded as a non-pure connectivity mesh, where only some of the vertices contain geometric information. In [Sorkine and Cohen-Or 2004] we use the term LS-meshes for meshes that are reconstructed by such least-squares (LS) solution with a small number of anchor points. While the connectivity shapes [Isenburg et al. 2001] strive to satisfy the uniform edge length condition, in an LS-mesh the vertices satisfy flatness and fairness conditions in the least-squares sense.

Figure 5 shows the approximation power of an LS-mesh. Given the connectivity augmented with a rather small set of anchor points, the shape of the *Camel*'s head can be well approximated (see more results in [Sorkine and Cohen-Or 2004]).

A relevant question is where to place the anchors. Figure 6 shows an example of the *Screwdriver* model which consists of 27152 ver-



(a)



(b)

Figure 6: Selecting the anchor points. In (a), 1000 anchor points for the LS-mesh were chosen randomly, which leads to poor reconstruction in some parts of the model. In (b), 1000 anchors were selected using our greedy approach, leading to better approximation.

tices, and on which we placed 1000 anchors points. On the left we see the results of placing the anchors by a random selection, while on the right we see the results of a greedy approach, which places one anchor at a time in the vertex that attained the maximal reconstruction error. One can see that the random selection is inefficient since it does not "predict" the places which will have large reconstruction error. On the other hand, the greedy selection works quite well and concentrates the anchors in strategic regions, such as the edges of the screwdriver, where the reconstruction error is likely to be larger otherwise. One can accelerate and approximate the greedy approach by placing a number of anchors at local error maxima, and then recompute the LS-mesh to obtain the error estimation for the next selection step. An additional anchor selection algorithm is developed in [Chen et al. 2005] based on a theoretical bound of the reconstruction error.

The geometry of an LS-mesh is defined by the following formula:

$$\mathbf{x} = A^*\mathbf{b} \quad (A^* = (A^TA)^{-1}A^T),$$

where the matrix A has the same structure as in Eq. (2) and

$$\mathbf{b} = (0_{1\times n},\, c_1, \ldots, c_k)^T.$$

In [Sorkine et al. 2005] we analyze the reconstructed geometry $\mathbf{x}$ and showed that in fact it is a linear combination of k basis functions that are derived from A^*:

$$\mathbf{x} = c_1\mathbf{f}_1 + c_2\mathbf{f}_2 + \ldots + c_k\mathbf{f}_k,$$

where

$$\mathbf{f}_i = A^* \cdot (0_{1\times n},\, 0_{1\times(i-1)},\, 1,\, 0_{1\times(k-i)})^T.$$

We show that the basis functions $\mathbf{f}_i$ are fairly smooth, and most of their "energy" is concentrated near a single anchor. Unlike the basis functions (the eigenvectors) of the Laplacian, here the basis functions are "geometry-aware". In [Sorkine et al. 2005] we show the effectiveness of having geometry awareness as opposed to having bases which are geometry oblivious: geometry-aware bases can approximate surface features much better, given that the anchor points are located in the vicinity of the features. Furthermore, an eigendecomposition is very computationally-intensive for moderate meshes, and thus computing and storing the (dense) spectral basis matrix is prohibitively expensive; we show that by defining the geometry-aware basis vectors to be the solutions of certain least-squares problems, the reconstruction problem reduces to solving a single sparse linear least-squares problem, which can be solved quickly using a state-of-the-art sparse-matrix factorization algorithms.

5 Laplacian mesh editing

We have seen above that the addition of anchor points makes our linear system over-determined, for which, in general, no exact solution may exist. However, the system is full-rank and thus has a unique solution in the least-squares sense:

$$\mathbf{x} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(\|L\mathbf{x} - \delta\|^2 + \sum_{i\in C} |x_i - c_i|^2 \right),$$

This gave rise to the idea of using this framework for mesh editing. Since the δ-coordinates represent the local details, the above system enables their preservation through various modeling tasks which are defined by manipulating the anchor positions c_i. In [Lipman et al. 2004; Sorkine et al. 2004; Lipman et al. 2005a] we have introduced such Laplacian mesh editing techniques and showed an efficient linear reconstruction based on matrix factorization of sparse matrices (see Figure 7).

Unlike other mesh editing approaches (e.g., [Kobbelt et al. 1998; Botsch and Kobbelt 2004]) our approach explicitly emphasizes the preservation of details. Dealing with surfaces with rich details is becoming more and more pronounced in computer graphics and geometric modeling, as the available surface data becomes increasingly complex and detailed, especially thanks to the proliferation of sub-millimeter accuracy 3D scanners and sophisticated input devices and modeling packages for virtual 3D sculpting.

The main challenge in any mesh manipulation application is to handle non-trivial transformations, i.e., transformations which include rotations (especially large rotations), while preserving as much as possible the visual characteristic of the shape at interactive rates. It has been accepted that deforming shapes *as rigidly as possible* provides plausible results. The key idea is to factor out the rotation from the deformation. Since rotations are rigid transformations, such factorization enables treating the deformation as a pure rotation plus a residual elastic deformation. Cohen-Or et al. [1998] have applied this concept to minimize the global deformation during shape interpolation. Alexa et al. [2000] show how it can be applied locally as a means of treating the volume (area) of a shape as rigidly as possible. Xu et al. [2005] and Summer et al. [2005] have extended these principles to the surface of a shape. However, in the context of shape editing, the problem of factoring out the rotation is significantly harder, since the target shape is not explicitly given. Thus, the factorization and the shape definition have to be solved simultaneously [Sorkine et al. 2004], or otherwise induced from the transformation of the control vertices (anchors).

A more direct approach to factoring rotations is to represent the shape with intrinsic coordinates [Sederberg et al. 1993], or with

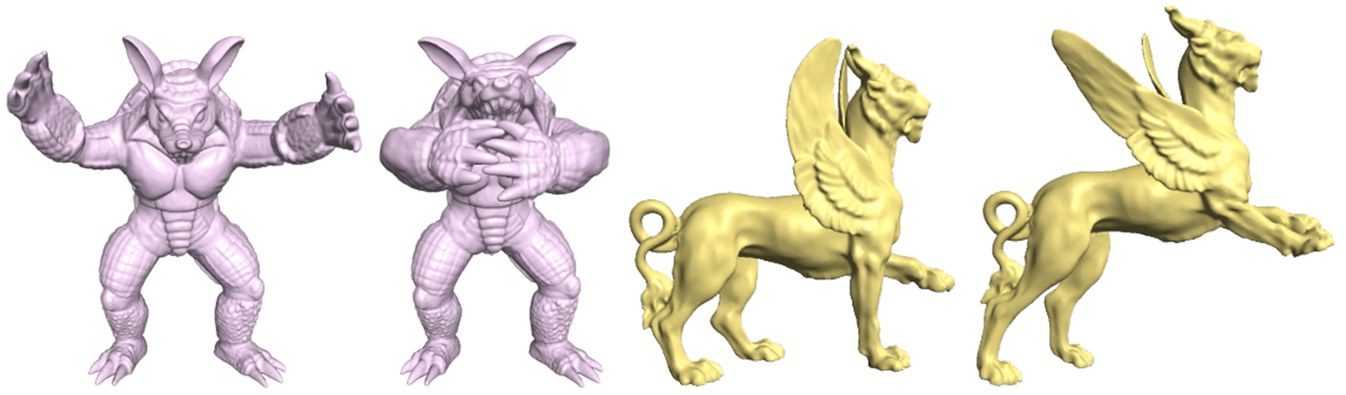


Figure 7: Examples of mesh editing results using Laplacian surface editing [Sorkine et al. 2004]. Note the intuitive preservation of the local surface details.

rotation-invariant coordinates [Lipman et al. 2005b]. With purely rotation-invariant coordinates, the factorization of the rotation is given for free. Lipman et al. [2005b] proved that by representing the mesh vertices within their own local frames, it is possible to uniquely represent a mesh, and that its reconstruction merely requires solving a sequence of two linear systems.

In recent years, many researchers have studied the manipulation of meshes while preserving their surface details [Lipman et al. 2004; Sheffer and Kraevoy 2004; Sorkine et al. 2004; Sumner and Popović 2004; Yu et al. 2004; Fu and Tai 2005; Igarashi et al. 2005; Ji et al. 2005; Lipman et al. 2005b; Nealen et al. 2005; Sumner et al. 2005; Volodine et al. 2005; Xu et al. 2005; Zayer et al. 2005; Zhou et al. 2005; Au et al. 2006]. As shown in these works, this framework has other interesting applications in addition to mesh editing via handle manipulation: the framework can be applied to detail transfer between surfaces, mesh transplanting, sketch-based editing of silhouettes and other curves, generation of suggestive contours and sharp features, pose transfer, example-based editing and mesh smoothing. The common idea in these works is to represent the surface with differential coordinates and directly control these coordinates under some constraints defining various objectives.

The subject of optimal deformation is studied in [Lipman et al. 2006]; this work formulates the deformation in terms of the local frames and shows how to modify them in order to obtain a deformed surface that satisfies the modeling constraints and whose curvature is as close as possible to the original surface. Lipman et al. [2006] also show how to extend the local frames framework to provide deformations that preserve the local volume of the shape. It should be noted that while [Lipman et al. 2005b; Lipman et al. 2006] enable *linear* mesh editing with rotation-invariant representation, and thus allow arbitrarily large rotations, this comes at the price of loosing the ability to set direct positional constraints, since the constraints have to be formulated in terms of the local frames (i.e. orientations and not positions). A non-linear framework allows to perform editing with positional constraints and large rotations, as well as volume preservation, of course at the cost of efficiency and/or convoluted implementation [Botsch et al. 2006; Huang et al. 2006].

6 Epilogue

In contrast to the traditional global Cartesian coordinates, which can only tell the spatial location of each point, a differential surface representation carries information about the local shape of the surface, the size and orientation of local details. Therefore, defining operations on surfaces that strive to preserve such a differential representation, results in detail-preserving operations. The linearity of the processing framework makes it very efficient, and it has become an attractive research direction, as is evident from the amount of recent publications on the subject.

References

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *Proceedings of ACM SIGGRAPH*, ACM Press/Addison-Wesley Publishing Co., 157–164.

ALLIEZ, P., AND DESBRUN, M. 2001. Valence-driven connectivity encoding for 3D meshes. *Computer Graphics Forum 20*, 3, 480–489.

ALLIEZ, P., AND GOTSMAN, C. 2005. Recent advances in compression of 3D meshes. In *Advances in Multiresolution for Geometric Modelling*, Springer-Verlag, N. Dodgson, M. Floater, and M. Sabin, Eds., 3–26.

AU, O. K.-C., TAI, C.-L., LIU, L., AND FU, H. 2006. Dual Laplacian editing for meshes. *IEEE Transactions on Visualization and Computer Graphics 12*, 3, 386–395.

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for real-time freeform modeling. In *Proceedings of ACM SIGGRAPH*, ACM Press, 630–634.

BOTSCH, M., PAULY, M., GROSS, M., AND KOBBELT, L. 2006. PriMo: Coupled prisms for intuitive surface modeling. In *Proceedings of the Eurographics Symposium on Geometry Processing*. To appear.

CHEN, D., COHEN-OR, D., SORKINE, O., AND TOLEDO, S. 2005. Algebraic analysis of high-pass quantization. *ACM Transactions on Graphics 24*, 4, 1259–1282.

COHEN-OR, D., LEVIN, D., AND SOLOMOVICI, A. 1998. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics 17*, 2, 116–141.

FU, H., AND TAI, C.-L. 2005. Mesh editing with affine-invariant Laplacian coordinates. Tech. Rep. HKUST-CS05-01, Hong Kong University of Science and Technology, January.

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 25*, 3. To appear.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 24*, 3, 1134–1141.

ISENBURG, M., GUMHOLD, S., AND GOTSMAN, C. 2001. Connectivity shapes. In *Proceedings of IEEE Visualization*, 135–142.

JI, Z., LIU, L., AND WANG, G. 2005. A global Laplacian smoothing approach with feature preservation. In *Proceedings of the 9th International Conference on Computer Aided Design and Computer Graphics*, IEEE Computer Society, 269–274.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of ACM SIGGRAPH*, ACM Press, 105–114.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., LEVIN, D., RÖSSL, C., AND SEIDEL, H.-P. 2004. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, IEEE Computer Society Press, 181–190.

LIPMAN, Y., SORKINE, O., ALEXA, M., COHEN-OR, D., LEVIN, D., RÖSSL, C., AND SEIDEL, H.-P. 2005. Laplacian framework for interactive mesh editing. *International Journal of Shape Modeling 11*, 1, 43–62.

LIPMAN, Y., SORKINE, O., LEVIN, D., AND COHEN-OR, D. 2005. Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 24*, 3, 479–487.

LIPMAN, Y., COHEN-OR, D., GAL, R., AND LEVIN, D. 2006. Volume and shape preservation via moving frame manipulation. *ACM Transactions on Graphics*. To appear.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 24*, 3, 1142–1147.

SEDERBERG, T. W., GAO, P., WANG, G., AND MU, H. 1993. 2-D shape blending: an intrinsic solution to the vertex path problem. In *Proceedings of ACM SIGGRAPH*, ACM Press, 15–18.

SHEFFER, A., AND KRAEVOY, V. 2004. Pyramid coordinates for morphing and deformation. In *Proceedings of the Second International Symposium on 3DPVT (3D Data Processing, Visualization, and Transmission)*, IEEE Computer Society Press, 68–75.

SORKINE, O., AND COHEN-OR, D. 2004. Least-squares meshes. In *Proceedings of Shape Modeling International*, IEEE Computer Society Press, 191–199.

SORKINE, O., COHEN-OR, D., AND TOLEDO, S. 2003. High-pass quantization for mesh encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry processing*, Eurographics Association, 42–51.

SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry processing*, ACM Press, 179–188.

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.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 23*, 3, 399–405.

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIĆ, J. 2005. Mesh-based inverse kinematics. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 24*, 3, 488–495.

TOUMA, C., AND GOTSMAN, C. 1998. Triangle mesh compression. In *Proceedings of Graphics Interface*, 26–34.

VOLODINE, T., VANDERSTRAETEN, D., AND ROOSE, D. 2005. Smoothing of meshes and point clouds using weighted geometry-aware bases. Tech. Rep. TW451, Department of Computer Science, Katholieke Universiteit Leuven, March.

XU, D., ZHANG, H., WANG, Q., AND BAO, H. 2005. Poisson shape interpolation. In *Proceedings of the ACM Symposium on Solid and Physical Modeling*, ACM Press, 267–274.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with Poisson-based gradient field manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 23*, 3, 644–651.

ZAYER, R., RÖSSL, C., KARNI, Z., AND SEIDEL, H.-P. 2005. Harmonic guidance for surface deformation. In *Computer Graphics Forum (Proceedings of Eurographics)*, Blackwell, Eurographics, 601–609.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Large mesh deformation using the volumetric graph Laplacian. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 24*, 3, 496–503.