On Mesh Editing, Manifold Learning, and Diffusion Wavelets

Raif M. Rustamov

Drew University, Madison NJ 07940, USA
rrustamov@drew.edu

Abstract. We spell out a formal equivalence between the naive Laplacian editing and semi-supervised learning by bi-Laplacian Regularized Least Squares. This allows us to write the solution to Laplacian mesh editing in a closed form, based on which we introduce the Generalized Linear Editing (GLE). GLE has both naive Laplacian editing and gradient based editing as special cases. GLE allows using diffusion wavelets for mesh editing. We present preliminary experiments, and shortly discuss connections to segmentation.

1 Introduction

A remarkable similarity exists between semi-supervised manifold learning and mesh editing: both seek to extrapolate data attached at some points to the whole manifold.

Given a set of labeled samples, extrapolating labels throughout the entire sample space is the task of semi-supervised learning. The qualification "manifold" is added if the samples are assumed to belong to a manifold embedded into a high-dimensional space – attaching labels is equivalent to defining a function on this manifold.

Editing a mesh involves determining the new locations of vertices given new locations of some of the vertices – the handles. The displacement vectors – the differences between new and old vertex positions – can be considered to define a function on the mesh. Thus, given the values of this function at the handles we are trying to extrapolate to the whole mesh – a task that would otherwise qualify as semi-supervised learning. If rotations at handles are also given, propagating them throughout the mesh is again an instance of semi-supervised learning.

Does this similarity of the two fields extend beyond the objectives sought? Laplacian based approaches to mesh editing start by extracting the surface's *differential coordinates*, and then reconstruct the surface by imposing the handle constraints and requiring that the differential coordinates are preserved as much as possible. The differential coordinates capture the local detail, so the more they are preserved, the more the shape is preserved. When viewed from this angle, Laplacian mesh editing seems to bear no resemblance to the methods of semi-supervised learning.

We show, however, that there exists a formal equivalence between the naive (linear) Laplacian editing and semi-supervised learning by bi-Laplacian Regularized Least Squares (Section 4). This allows us to write the solution to Laplacian mesh editing in a "closed" form (Section 5). Based on this closed form we introduce the **G**eneralized **L**inear **E**diting (GLE) which has both naive Laplacian editing and gradient based editing as special cases. GLE allows using diffusion wavelets for mesh editing (Section 6). Preliminary experiments are presented (Section 7), and connections to segmentation are discussed (Section 8).

Our contributions are threefold: clearly spelling out the formal relationship between manifold learning and mesh editing, introducing GLE, and using diffusion wavelets for mesh editing. Although naive Laplacian editing is not practically useful, it becomes so when embedded into nonlinear schemes such as [1]. Therefore, understanding the properties of the naive Laplacian editing better, and investigating its potential improvements may have considerable consequences on the state of art.

2 Related work

Immense attention has been received by direct surface manipulation methods based on differential representations. The first examples were Poisson mesh editing [2] and Laplacian coordinates [3, 4]. As we learn from [5], it was gradient domain image manipulation that inspired these approaches: specifically, the Poisson surface editing of [6] motivated Poisson mesh editing. These initial methods are now customarily referred as naive or linear – they optimize each of x, y, z coordinates separately, and so do not allow dealing with handle rotations. An excellent survey of these and other linear techniques is [5].

Remarkably, differential representations come with an elegant interpretation of being local surface descriptors. For example, Laplacian coordinates are the components of the mean-curvature normal. The interpretation offers a strong intuition about how to deal with rotations, and led to new techniques a few examples of which are [7–9]; a great survey is [10]. The interpretation also inspired the development of other intrinsic coordinates: pyramid coordinates [11], and rotation invariant coordinates [12].

In the light of all these developments, one would legitimately question whether there is a need to study any further the linear methods, generalize or alter them. The answer is best given by an example: in a recent paper [1], the naive Laplacian editing is embedded into an iterative scheme to obtain a non-linear method. The algorithm is guaranteed to converge and is remarkably easy to implement. The method compares very favorably with PriMo [13], a state-of-the-art non-linear technique. This makes us believe that understanding the properties of the naive methods better, and investigating their potential improvements will have considerable consequences on the state of the art.

To the best of our knowledge, the connections to the semi-supervised learning have never emerged, perhaps due to the sheer beauty of the interpretation that differential coordinates came with. Yet we must mention the inspiring

study of Lévy [14], where among other things he explains how Laplace-Beltrami eigenfunctions can be used to manipulate shapes. He proposes pose transfer by exchanging expansion coefficients of the coordinate functions in terms of Laplace-Beltrami eigenfunctions. In addition, in [15] geometry filtering is performed by modifying the expansion coefficients. Another work that uses the Laplace-Beltrami expansion coefficients is [16], where the size of linear systems associated with Laplacian editing are significantly reduced to achieve interactive computational speed for manipulating large meshes. All of these works can be interpreted as an approach to editing where the control is vested into these expansion coefficients, yet they discuss the connections to neither manifold learning nor to Laplacian editing. In a different context, let us also mention the paper [17] which studies mesh smoothing in the light of Laplacian eigenbasis and regularization.

Using different function bases to manipulate meshes would not be surprising to the space deformation community. Just to mention a few, in [18–21] Radial Basis Functions (RBF) are used to infer the space deformation that would satisfy given constraints. There are many bases to choose from and, thus, flexibility to decide based on time/quality constraints. However, to the best of our knowledge, the idea to use different bases to generalize/modify a direct manipulation method such as Laplacian editing is novel. Let us add that diffusion wavelets have found only very limited use in digital geometry processing – the only work that we are aware of is [22] where diffusion wavelets are used for mesh compression.

As for manifold learning, we can only mention some key papers that are directly relevant. Laplacian based regularization was introduced in [23], where semi-supervised learning is reduced to minimization of an expression which contains a penalty term to enforce the labels of labeled samples, and terms to ensure "continuity" of labeling – "close" points get similar labels. The latter are called the *regularization* terms; they ensure that learning results in a sufficiently smooth function. As shown in [23], the Laplacian – the manifold's Laplace-Beltrami operator – makes a good regularization term for a variety of learning applications. Diffusion maps, diffusion wavelets, diffusion wavelet packets and other related concepts were introduced by R. Coifman and coworkers, and the definitive reference is the special issue [24].

3 Manifold learning

We will avoid describing the most general setting for manifold learning, rather our exposition will be geared towards surfaces in space – we will make assumptions and introduce notations most appropriate for our purposes. We will concentrate on versions of Laplacian regularized least squares learning, a concept introduced in [23], heavily borrowing from it and from paper [25] with some notational modifications.

We start with a connected surface $\mathcal{S}$. Some of the points on the surface, say $\{\mathbf{p}_i\}_{i=1}^{l}$ have been labeled – real numbers $\{d_i\}_{i=1}^{l}$ have been assigned to them.

Semi-supervised learning seeks a function f^* that minimizes

$$\sum_{i=1}^{l} (f(\mathbf{p}_i) - d_i)^2 + \beta \|f\|_E^2 + \gamma \|f\|_I^2. \tag{1}$$

The first term tries to enforce the function to take values prescribed at labeled vertices. The last two terms are called regularization terms, and they aim to make the function smooth in *extrinsic* and *intrinsic* senses. To clarify, extrinsic smoothness could mean that function takes close values at points that are close in Euclidean space; intrinsic smoothness would mean close function values for points that are geodetically close. We will be mostly interested in the first and last terms, dropping the middle terms of (1) in what follows.

Belkin et. al. [23] propose to use the surface integral of function's gradient as the intrinsic regularization term, which can be rewritten using Green's theorem as

$$\|f\|_I^2 = \int_{\mathcal{S}} f \Delta f,$$

where Δ is the Laplace-Beltrami operator of the surface. They also note that iterated Laplacians Δ^k and linear combinations of these can be used in this expression to yield other examples of intrinsic regularization terms; thus for a self-adjoint linear differential operator L on the surface, we have the regularization term

$$\|f\|_I^2 = \int_{\mathcal{S}} f L f.$$

The natural realm for discussing the minimization problem (1) is an appropriately chosen Reproducing Kernel Hilbert Space. We avoid these details, and only point out to a few consequences. Let us denote by μ_i and $e_i : \mathcal{S} \to R$ the eigenvalues and the eigenfunctions of the linear operator L. These satisfy $Le_i = \mu_i e_i$. Since L is self-adjoint, the eigenfunctions constitute an orthogonal basis for $L_2(\mathcal{S})$. For a function $f = \sum_i c_i e_i$, the intrinsic regularizator easily evaluates to

$$\|f\|_I^2 = \sum_i \mu_i c_i^2.$$

Now, the *kernel function*

$$K(\mathbf{p}, \mathbf{q}) = \sum_i \frac{e_i(\mathbf{p}) e_i(\mathbf{q})}{\mu_i}$$

can be defined (note the similarity with the formula for matrix pseudoinverses); the important point is that the solution of the semi-supervised learning problem (1) can be written as

$$f^*(\mathbf{q}) = \sum_{i=1}^{l} a_i K(\mathbf{p}_i, \mathbf{q}).$$

Notice that the summation is over labeled point's indices. Numbers a_i are the entries of the vector $\boldsymbol{a}$ which solves the equation

$$(\gamma I_l + K')\boldsymbol{a} = \boldsymbol{d}. \tag{2}$$

Here I_l is the $l \times l$ identity matrix, K' is an $l \times l$ matrix with $K'_{ij} = K(\mathbf{p}_i, \mathbf{p}_j)$, and $\boldsymbol{d}$ is the vector whose i-th entry is the label d_i.

4 Laplacian editing reinterpreted

Given a surface mesh, mesh editing allows a user to modify it by specifying new positions for some surface points; these user constrained points are called *handles*. Laplacian mesh editing is based on extracting the surface's *differential coordinates*, and then reconstructing the surface by imposing the user constraints and requiring that the differential coordinates are preserved as much as possible; for a recent survey we refer the reader to [5]. We will show that the naive or linear version of Laplacian editing, the version where handle rotations are not propogated, is equivalent to semi-supervised learning with the bi-Laplacian as the regularizator.

Consider projection onto x-coordinate$\pi_x : \mathcal{S} \to R$, i.e. the function whose value $\pi_x(\mathbf{p})$ is the x-coordinate of the surface point $\mathbf{p}$. This function, will be assumed to satisfy $\int_{\mathcal{S}} \pi_x = 0$, which in all cases can be achieved by translating the origin to the center of mass of the surface. The differential coordinate $\delta_x(\mathbf{p})$ is defined by

$$\delta_x(\mathbf{p}) = \Delta\pi_x(\mathbf{p}),$$

where Δ is the Laplace-Beltrami operator of the surface. In a similar way one defines $\delta_y(\mathbf{p})$ and $\delta_z(\mathbf{p})$. As an aside, the differential coordinates are precisely the components of the mean curvature normal.

The edited surface $\mathcal{S}'$ is in one to one correspondence with the original surface, so we will use the same letter to denote both a point on $\mathcal{S}$ and $\mathcal{S}'$. Also, $\mathcal{S}'$ has its own function π'_x. Suppose that the handles are the points $\{\mathbf{p}_i\}_{i=1}^l$, and the user has specified new x-coordinates for these points as $\{x'_i\}_{i=1}^l$. Soft constraint Laplacian mesh editing constructs the new surface by minimizing the expression

$$\sum_{i=1}^{l} (\pi'_x(\mathbf{p}_i) - x'_i)^2 + \gamma \int_{\mathcal{S}} (\Delta\pi'_x - \delta_x)^2.$$

We are able to treat the x-coordinate separately, since in Laplacian editing mesh coordinates are treated independently.

Now we will rewrite the problem. Let us introduce the function $f = \pi'_x - \pi_x$, the difference between the old and new x-coordinates. Along with the corresponding functions for the other two coordinates, this function completely determines the new surface in terms of the old. At the handle points $\mathbf{p}_i$ the user has specified the sought values of f, which we denote by $d_i = x'_i - \pi_x(\mathbf{p}_i)$. Note that

$$\Delta\pi'_x - \delta_x = \Delta\pi'_x - \Delta\pi_x = \Delta f.$$

The equality

$$\int_{\mathcal{S}} (\Delta f)^2 = \int_{\mathcal{S}} f \Delta^2 f$$

follows from Green's theorem; this changes our optimization problem into

$$\sum_{i=1}^{l} (f(\mathbf{p}_i) - d_i)^2 + \gamma \int_{\mathcal{S}} f \Delta^2 f.$$

This is precisely equation (1) of semi-supervised learning with $L = \Delta^2$. Notice that, the same derivation is valid for surfaces with boundary, if f assumed to satisfy the Neumann boundary condition. With little thought the reader can see that this assumption is implicit in Laplacian editing.

In a similar vein, let us show that gradient based editing of [2] is equivalent to semi-supervised learning with Laplacian as regularizator; this is precisely Laplacian Regularized Least Squares presented and studied in [23]. In fact, gradient based editing seeks to minimize the expression

$$\sum_{i=1}^{l} (\pi'_x(\mathbf{p}_i) - x'_i)^2 + \gamma \int_{\mathcal{S}} (\nabla \pi'_x - \nabla \pi_x)^2,$$

where ∇ is the gradient. In other words, the reconstruction aims to preserve the gradients of mesh coordinate functions – "gradient coordinates" $\nabla \pi_x$. Using our notation and applying Green's theorem this can be rewritten as

$$\sum_{i=1}^{l} (f(\mathbf{p}_i) - d_i)^2 + \gamma \int_{\mathcal{S}} f \Delta f.$$

Clearly, this is the semi-supervised learning objective with $L = \Delta$.

This proves our claim that *formally*, Laplacian mesh editing is an instance of semi-supervised manifold learning. The formulas make clear that the preservation of differential coordinates is equivalent to intrinsic regularization. This equivalence has an intuitive appeal: regularization forces the displacement function f to be "smooth", this in turn allows to protect the local surface detail – the original goal behind the preservation of differential coordinates.

5 Consequences

After establishing the relationship to manifold learning, we can start importing knowledge from the field. Let us discuss two such examples – they will motivate a generalization of Laplacian and gradient editing.

First, we can write an explicit formula for the solution of the Laplacian and gradient editing. In fact, consider the eigenvalues λ_i and eigenfunctions ϕ_i of the Laplace-Beltrami operator. The eigenvalues are non-negative and constitute a discrete set; we put them into non-decreasing order

$$\lambda_0 = 0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_i < \ldots$$

Note that $\lambda = 0$ is always a simple eigenvalue because the surface is assumed to be connected. The appropriately normalized eigenfunction corresponding to λ_i will be denoted by ϕ_i. Notice that the bi-Laplacian has the same eigenfunctions ϕ_i, with corresponding eigenvalues λ_i^2.

Now the kernel function corresponding to Laplacian editing is given by

$$K(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{\infty} \frac{\phi_i(\mathbf{p})\phi_i(\mathbf{q})}{\lambda_i^2}$$

and the solution to the mesh editing problem is

$$f^*(\mathbf{q}) = \sum_{i=1}^{l} a_i K(\mathbf{p}_i, \mathbf{q}),$$

where a_i are found by solving the system (2). A similar formula is valid for gradient based editing – one replaces λ_i^2 by λ_i in the definition of the kernel function.

Second, there is an interpretation of regularized manifold learning schemes which translates into a remarkable reading of Laplacian editing. To explain, we follow [25] and write the minimization in bi-Laplacian regularized Least squares in terms of eigenvalues and eigenfunctions of Laplace-Beltrami operator.

Remembering that the eigenfunctions ϕ_i constitute a basis for $L_2(\mathcal{S})$, we can expand any function as $f = \sum_i c_i \phi_i$. When this expansion is plugged into the bi-Laplacian regularizator in the objective function, the expression to minimize reduces to

$$\sum_{i=1}^{l} (f(\mathbf{p}_i) - d_i)^2 + \gamma \sum_i \lambda_i^2 c_i^2, \tag{3}$$

while similar formula for Laplacian based regularization would contain λ_i instead of λ_i^2. Thereby, we are trying to device a function by combining ϕ_i so that this combination takes prescribed values in the least squares sense, but we harshly penalize for the use of high frequency eigenfunctions – eigenfunctions corresponding to larger eigenvalues.

Thus, in some sense, Laplacian based semi-supervised learning assumes that Laplacian eigenfunctions constitute the preferred "learning basis". Preferred, because they provide the smoothest basis for $L_2(\mathcal{S})$ in the sense explained in [25]. Higher penalty for high frequency eigenfunctions makes sure that the solution is smooth enough. This fits very well with Occam's razor – lower frequency eigenfunctions are "simpler". For example, under general conditions, the first eigenfunction ϕ_1 changes its sign only once, and has only one minimum and one maximum.

In the context of mesh editing, we should perhaps talk about the "motion basis" of an object. The motion basis would contain (linearly independent) displacement functions for natural articulations of the object, ordered from the simplest to the most complex. One could device a measure of complexity for linear combinations of motion basis functions. The objective of mesh editing would be to find the simplest such combination that satisfies the user constraints.

From this perspective, the fundamental assumption of Laplacian editing is that Laplacian eigenfunctions constitute a good motion basis. Clearly, the difference between Laplacian and gradient based editing is in the penalty for using high frequency eigenfunctions – Laplacian editing is harsher in this respect. Of course, using higher iterated Laplacians in the regularizator further increases the penalty for using high frequencies. An important question emerges – do we have to base our regularization on Laplacian or its iterates? *Can we set the penalties manually for each eigenfunction?*

Clearly such a modification of Laplacian editing, namely choosing the penalties manually, can be obtained by simply changing the formula for the kernel into

$$K(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{\infty} \frac{\phi_i(\mathbf{p})\phi_i(\mathbf{q})}{\mu_i}.$$

We speculate that if the user agrees to set $\mu_i = \infty$ for all $i > r$, then this formula can be efficiently implemented to allow the user to adjust the remaining penalty weights μ_i interactively. Setting infinite penalties is equivalent to truncating the sum in the formula for the kernel function – effectively keeping only r low frequency eigenfunctions.

Remember that in discrete setting, the Laplacian and the kernel function are $n \times n$ matrices, where n is the number of mesh vertices in the region of interest. We will assume that one computes r eigenvectors in the preprocessing step. Of course, computing the eigenvectors of a matrix is a considerable burden; yet if r is in the range of a few hundreds, that many eigenvectors of the Laplacian for a mesh as large as 250K vertices can be evaluated in about 15-20 minutes [15]. During the interactive session, editing will involve solving an $l \times l$ linear system (2), where l is the number of handles, and evaluating the linear combination of l columns of the kernel matrix – the columns associated with the handle vertices. Of course, we should not pre-compute and store the whole $n \times n$ kernel matrix in the main memory. Instead, storing only the r eigenvectors will reduce the space requirement from $O(n^2)$ to $O(nr)$. Now notice that the time complexity of the interactive part – obtaining the new vertex coordinates after the user modifies the penalty weights – is linear in the number of vertices, $O(nl + nr)$. Also, the new coordinates can be evaluated in parallel. Let us emphasize that we have not implemented this approach, and we do not know how the omission of high frequency eigenvectors will influence the result of editing.

6 GLE and diffusion wavelets

Once again, do we have to base our regularization on Laplacian or its iterates? *Can we use a different basis?* These questions lead us to propose the following Generalized Linear Editing, GLE for short. Similar to naive Laplacian editing, GLE does not handle frame rotations, and the displacements of each coordinate are evaluated independently. Thus, for brevity, we concentrate on obtaining the x coordinates only; the other two coordinates require the same steps, perhaps

with different bases and weights if wanted. At the cost of self-repetition, let us clearly outline the algorithm:

1. Pick an orthonormal basis e_i of $L_2(\mathcal{S})$; here $\mathcal{S}$ is the edited surface.
2. Assign penalty weights μ_i to each of the basis functions e_i.
3. Define $K(\mathbf{p}, \mathbf{q}) = \sum_i e_i(\mathbf{p}) e_i(\mathbf{q})/\mu_i$.
4. Let l handle vertices $\mathbf{p}_i, i = 1, ..., l$ and their x-coordinate displacements $d_i, i = 1, ..., l$ be given. Construct vector $\boldsymbol{d}$ whose ith entry is d_i. Solve the equation
$$(\gamma I_l + K')\boldsymbol{a} = \boldsymbol{d}$$
for $\boldsymbol{a}$, where $K'_{ij} = K(\mathbf{p}_i, \mathbf{p}_j)$ is an $l \times l$ matrix. Here γ is a user set positive parameter – the larger is γ the less is pressure to satisfy the displacement constraints, the more weight is given to the regularization term.
5. Compute the new x coordinate $\mathbf{q}'_x$ of the point $\mathbf{q}$ using the formula $\mathbf{q}'_x = \mathbf{q}_x + \sum_{i=1}^{l} a_i K(\mathbf{p}_i, \mathbf{q})$.

When dealing with some region of interest instead of the whole mesh, one will require the boundary to stay constant. This can be achieved in GLE by using basis functions that are supported on the region of interest.

One also may wish to enforce the handle constraints with different weights. Suppose that weights w_i are associated with handle vertices $\mathbf{p}_i, i = 1, ..., l$. One then can construct the $l \times l$ diagonal matrix W with $W_{ii} = 1/w_i$; now in step 4, one replaces the equation for $\boldsymbol{a}$ by $(\gamma W + K')\boldsymbol{a} = \boldsymbol{d}$.

Notice that both the Laplacian and gradient based mesh editing are examples of GLE. To provide further examples, let us use *diffusion wavelets* [26] as a basis. The following discussion will be geared towards implementation, providing the discrete counterparts of the steps above.

Let P be a set of points on a mesh. Clearly, any real-valued function on P can be recorded as a vector in $R^{|P|}$. Then, a diffusion operator T (that linearly acts on these vectors) is a matrix that satisfies certain properties, most notably its eigenvalues are in the $[0, 1]$ interval. In the implementation of diffusion wavelets that we worked with [27], among a few choices for T we used the following: vertices of the mesh are considered as a point cloud, and for each point, one finds a fixed number of nearest neighbors and assigns the corresponding matrix entry to be $\exp(-\Delta * dist^2)$, where $dist$ is the distance to the point in question. This choice on our part was forced by the fact that the diffusion wavelet code was written by the manifold learning community and was geared toward working with point clouds.

Diffusion wavelet construction uses the diadic powers of T to construct subspaces V_i and W_i of $R^{|P|}$ together with their orthogonal bases; the subscript of a subspace is called its *level*. These subspaces satisfy $R^{|P|} = W_0 \oplus V_0$, $V_0 = W_1 \oplus V_1$, $V_1 = W_2 \oplus V_2$, and so on. In fact, in the implementation we worked with, $W_0 = \{\mathbf{0}\}$, and $V_0 = R^{|P|}$ with delta-function basis. To obtain the basis of subspace V_{i+1}, one acts by the diadic power of diffusion operator T^{2^i} on the basis of V_i, and applies some thresholding and a special locality preserving orthogonalization. An important property of these bases is that different levels represent

different *scales* – together they provide a multi-scale basis of $R^{|P|}$. For example, the basis vectors in V_0 represent the smallest scale possible – their support is just one vertex. Meanwhile, the basis vectors of subspaces of high level have global nature, because they are obtained by many repetitive applications of the diffusion operator – in a sense, the function values are diffused throughout the mesh.

Fixing some maximum level m that one desires to work with, one can write

$$R^{|P|} = W_0 \oplus W_1 \oplus W_2 \oplus \cdots \oplus W_m \oplus V_m.$$

In addition, gathering the bases of the subspaces appearing in this equality, one obtains an orthogonal basis of $R^{|P|}$. It is this basis that we will use for editing. Let us construct the $|P| \times |P|$ matrix E which contains the basis vectors as its columns. Next, we need penalty weights $\mu_k, k = 1, ..., |P|$. Since functions in higher levels are smother, we have made a design decision to set $\mu_k = 1/l^\alpha$, where l is the level from which the basis vector $e_k$comes from, and α is a positive user set parameter, the larger it is the more high-level functions are preferred. Very similar results are obtained by setting $\mu_k = 1/\beta^l$, with $\beta > 1$.

Consider the matrix M which has $M_{kk} = 1/\mu_k$, and has zeros everywhere else. Notice that the discrete counterpart of the kernel function $K(\mathbf{p}, \mathbf{q})$ is a $|P| \times |P|$ matrix which we denote by K as well; clearly, $K = EME^T$. Implementing the rest of the steps should be straightforward.

7 Results and Discussion

We provide preliminary results of GLE using diffusion wavelets. Our starting surface is the unit sphere; the mesh is obtained through platonic subdivision using trimesh2 library. The number of vertices is $|P| = 482$. We choose $m = 12$ because V_{12} is one-dimensional, and so no more basis vectors coming from W_i's are produced. Choosing a smaller value of m will lead to less smooth edits, because smoother basis functions belong to higher levels. Figure 1 depicts the result of applying the following editing constraints: the south pole stays the same, the north pole's z-coordinate increases by 1. For comparison, the result of Laplacian editing using cotangent weights is shown in the same figure.

Diffusion wavelets work with more complicated constraints as well. The constraints for Figure 2 are as follows: the vertices with $|z| < 0.1$ are moved closer to the origin by setting $x \to 0.5x$, $y \to 0.5y$, and the vertices with $|z| > 0.7$ are kept same.

Our preliminary implementation was done in MATLAB, and it uses the diffusion wavelet code provided on Mauro Maggioni's website [27]. To give an idea about timing: computation of diffusion wavelet basis for $|P| = 482$ took about 9 seconds, and editing took about 0.5 seconds. However, if the number of vertices is doubled, the time spent to compute the diffusion wavelet basis increases to about 90 seconds, while edit time goes up to 6 seconds. The experiments were run on T7200 @ 2.0GHz laptop with 1GB main memory, running Windows XP Professional.

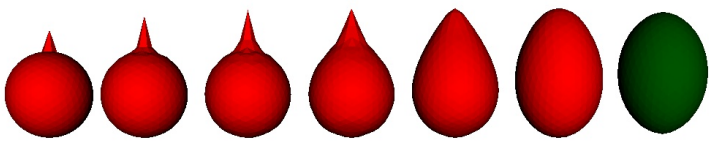


Fig. 1: Diffusion wavelet deformations of the sphere are in red. The parameters are as follows: $m = 12$, and $\alpha = 0, 1, 2, 4, 8, 16$ in respective order. The result of Laplacian editing with the same constraints is given last in green.

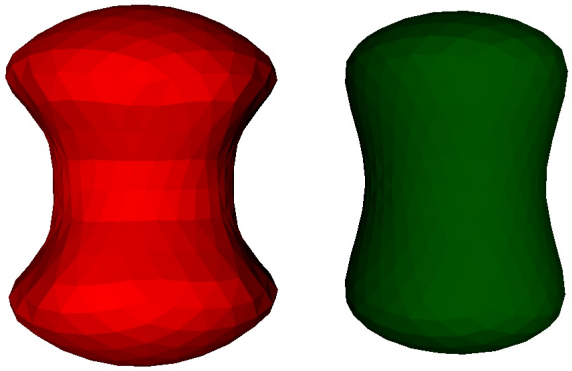
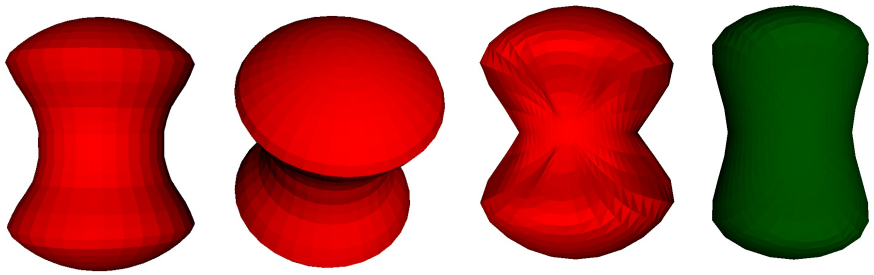


Fig. 2: Diffusion wavelet deformation of the sphere is in red; $m = 12$ and $\alpha = 16$ were used. The result of Laplacian editing with the same constraints is in green.

7.1 Discussion

First, going back to the experiment of Figure 1, notice that by increasing α we decrease the penalty of using high level basis vectors, this effectively suppresses the use of lower level vectors, and results in smoother, more global edits. Thus, the choice of this parameter can be tuned by whether less smooth local or more smooth global edits are wanted. Such scale control in Laplacian editing can only be achieved by varying the region of interest. This is due to the global nature of Laplacian eigenfunctions – their support is the whole mesh. This results in a "butterfly effect" – one may intend to make a little bump on the sphere, but would end up with an ellipsoid.

Second, the reader would have noted our use of uniform sphere sampling. With the implementation of diffusion wavelets that we used, we feel that, at least theoretically, uniformity is necessary. Indeed, the computed diffusion wavelet basis vectors are orthogonal with respect to the standard inner product on $R^{|P|}$ – an inner product that does not correspond to anything geometrically meaningful unless the mesh vertices are distributed uniformly over the area of the surface. However, in practice non-uniform sampling leads to mixed results. For example Figure 3 a) shows a very satisfying result of diffusion wavelet editing on a sphere mesh obtained using longitude-latitude triangulation. Figure 3 b) shows what happens if constraints include both of the poles of this non-uniformly sampled sphere. It is certainly not very satisfying when compared to Laplacian editing.



(a) (b)

Fig. 3: a) Diffusion wavelet deformation of longitude-latitude triangulated sphere. The parameter values $m = 12$ and $\alpha = 16$ were used. Same constraints as for Figure 2. b) Same as in a) except that the "squishing" circle passes through the poles. The result of Laplacian editing with this constraint is in green.

Let us conclude by noting that one could in principle circumvent this issue by first sampling a uniform point cloud on the surface, evaluating the diffusion

wavelets, and then interpolating their values from the point cloud to the mesh vertices.

8 Connections to segmentation

Motion based segmentation [28–30] inputs an animation sequence of a boundary mesh, and clusters together points that move in accord to produce a segmentation. Now one can imagine feeding to such an algorithm "animations" generated by applying Laplacian editing or GLE to the mesh – this would give a segmentation associated with a given mesh editing approach.

Do we need to explicitly produce these animations in order to get the associated segmentation? Looking at the solution of GLE,

$$f^*(\mathbf{q}) = \sum_{i=1}^{l} a_i K(\mathbf{p}_i, \mathbf{q}),$$

we notice that the value of $K(\mathbf{p}_i, \mathbf{q})$ measures how much point $\mathbf{q}$ is influenced by editing the handle at $\mathbf{p}_i$. Consequently, within the GLE framework, the magnitude of the kernel function $K(\mathbf{p}, \mathbf{q})$ is a measure of how much the points $\mathbf{p}$ and $\mathbf{q}$ move in accord – making the kernel function a natural measure of similarity for motion based segmentation. This is not a new approach to segmentation: for example, [31] spells out similar ideas and uses the Green's function of Laplace-Beltrami operator – the kernel function of gradient-based editing – to obtain pose-invariant segmentation of meshes.

Interesting is the manifold learning perspective on this association of editing and segmentation. Indeed, there is a similarity between mesh segmentation and unsupervised manifold learning. In mesh segmentation the objective is to assign a discrete set of labels, segment identifiers, to the mesh vertices. This is similar to defining an integer valued function on a point cloud in a meaningful way – an instance of unsupervised manifold learning. But for each supervised method based on regularization, one can device a corresponding unsupervised method by simply using the same regularization term. In this case one would search for a function $f : \mathcal{S} \to R$ satisfying appropriate conditions – such as being a fuzzy membership function that clusters points into even clusters – that minimizes $\|f\|_I^2$. Thus, it is not surprising to have a segmentation scheme canonically associated with an editing method.

We conclude by noticing that the connection between mesh editing and segmentation can be helpful when designing bases for GLE, assuming that a basis that produces better segmentations should result in better editing.

9 Summary and future work

We have stated the formal relationship between manifold learning and naive Laplacian and gradient-based editing. As a result, we were able to introduce a

generalized editing approach that allows the use of diffusion wavelets for mesh editing. We believe that diffusion wavelets are a very promising alternative to Laplacian eigenfunctions because due to their multi-scale nature. More experiments will be required to verify this; including experiments where handle rotations are propagated either using the existing approaches or through some new approach based once more on diffusion wavelets. Replacing the naive Laplacian editing by GLE in iterative methods such as [1] would provide another venue for experimentation. Another very important direction for future research is devising diffusion wavelets for triangle meshes rather than point clouds and resolving the issue of non-uniform sampling. Faster algorithms to evaluate such wavelets would be indispensable for practical applications.

On the intriguing side, we would like to investigate whether the equivalence extends to non-linear mesh editing approaches as well, and if so, to exploit these equivalences in the opposite direction – see if the insights gained from mesh editing can help in the realm of manifold learning.

References

1. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. (2007) 109–116
2. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with Poisson-based gradient field manipulation. In: TOG(SIGGRAPH). (2004) 644–651
3. Alexa, M.: Differential coordinates for local mesh morphing and deformation. The Visual Computer **19**(2-3) (2003) 105–114
4. Sorkine, O., Cohen-Or, D., Toledo, S.: High-pass quantization for mesh encoding. In: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, Eurographics Association (2003) 42–51
5. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE Transactions on Visualization and Computer Graphics (2007) To appear.
6. Prez, P., Gangnet, M., Blake, A.: Poisson image editing. TOG(SIGGRAPH) **22**(3) (2003) 313–318
7. Sorkine, O., Lipman, Y., Cohen-Or, D., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, ACM Press (2004) 179–188
8. Zayer, R., Rössl, C., Karni, Z., Seidel, H.P.: Harmonic guidance for surface deformation. Comput. Graph. Forum **24**(3) (2005) 601–609
9. Lipman, Y., Cohen-Or, D., Gal, R., Levin, D.: Volume and shape preservation via moving frame manipulation. ACM Trans. Graph. **26**(1) (2007) 5
10. Sorkine, O.: Differential representations for mesh processing. Computer Graphics Forum **25**(4) (2006) 789–807
11. Sheffer, A., Kraevoy, V.: Pyramid coordinates for morphing and deformation. In: 3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium, Washington, DC, USA, IEEE Computer Society (2004) 68–75
12. Lipman, Y., Sorkine, O., Levin, D., Cohen-Or, D.: Linear rotation-invariant coordinates for meshes. ACM Trans. Graph. **24**(3) (2005) 479–487

13. Botsch, M., Pauly, M., Gross, M., Kobbelt, L.: Primo: coupled prisms for intuitive surface modeling. In: SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association (2006) 11–20
14. Lévy, B.: Laplace-Beltrami eigenfunctions: Towards an algorithm that understands geometry. In: Shape Modeling International. (2006)
15. Vallet, B., Lvy, B.: Spectral geometry processing with manifold harmonics. Computer Graphics Forum (Proceedings Eurographics) (2008)
16. Rong, G., Cao, Y., Guo, X.: Spectral mesh deformation. Vis. Comput. **24**(7) (2008) 787–796
17. Volodine, T., Vanderstraeten, D., Roose, D.: Smoothing of meshes and point clouds using weighted geometry-aware bases. In: GMP. (2006) 687–693
18. Borrel, P., Rappoport, A.: Simple constrained deformations for geometric modeling and interactive design. ACM Trans. Graph. **13**(2) (1994) 137–155
19. Turk, G., O'Brien, J.F.: Modelling with implicit surfaces that interpolate. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Courses, New York, NY, USA, ACM (2005) 21
20. Reuter, P., Tobor, I., Schlick, C., Dedieu, S.: Point-based modelling and rendering using radial basis functions. In: GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia, New York, NY, USA, ACM (2003) 111–118
21. Botsch, M., Kobbelt, L.: Real-time shape editing using radial basis functions. Comput. Graph. Forum **24**(3) (2005) 611–621
22. Mahadevan, S.: Adaptive mesh compression in 3d computer graphics using multiscale manifold learning. In: ICML '07: Proceedings of the 24th international conference on Machine learning, New York, NY, USA, ACM (2007) 585–592
23. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. J. Mach. Learn. Res. **7** (2006) 2399–2434
24. Coifman, R.R.: Special issue on diffusion maps. Appl. Comput. Harmon. Anal. **21**(1) (2006) 3
25. Belkin, M., Niyogi, P.: Semi-supervised learning on riemannian manifolds. Mach. Learn. **56**(1-3) (2004) 209–239
26. Coifman, R.R., Maggioni, M.: Diffusion wavelets. Appl. Comput. Harmon. Anal. **21**(1) (2006) 53–94
27. Maggioni, M.: Diffusion wavelet code. http://www.math.duke.edu/ mauro/DWCode.zip
28. Lee, T.Y., Lin, P.H., Yan, S.U., Lin, C.H.: Mesh decomposition using motion information from animation sequences: Animating geometrical models. Comput. Animat. Virtual Worlds **16**(3-4) (2005) 519–529
29. Sattler, M., Sarlette, R., Klein, R.: Simple and efficient compression of animation sequences. In: SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, ACM Press (2005) 209–217
30. Günther, J., Friedrich, H., Wald, I., Seidel, H.P., Slusallek, P.: Ray tracing animated scenes using motion decomposition. Computer Graphics Forum **25**(3) (September 2006) 517–525 (Proceedings of Eurographics).
31. Rustamov, R.M.: Laplace-beltrami eigenfunctions for deformation invariant shape representation. In: Symposium on Geometry Processing. (2007)