DetailPreserving 2D Laplacian Editing

Ribard Alexandre*
ENSIMAG - MMIS

Stage IRL effectué dans l’équipe
Imagine - Inria-LJK
Tuteur: Stefanie Hahmann
2014

Figure 1: 2D mesh editing operations using : Laplacian curve editing and detail preservation. Free form deformations 3 details preserved (a-b), multiple details (c-d)

Abstract

Curve editing operations commonly require geometric details of the curve to be preserved as much as possible. However, that geometric detail is an intrinsic property of a curve. Consequently, curve editing is best performed by operating over an intrinsic curve representation. The Laplacian of a 2D mesh provides such a representation. Each vertex is thus encoded relatively to its neighborhood. The Laplacian of the mesh is enhanced to be invariant to locally linearized rigid transformations and scaling. Based on this Laplacian representation, useful editing operations can be developed such as interactive free-form deformation in a region of interest based on the transformation of a handle. The main computation involved in all operations is the solution of a sparse linear system, which can be done at interactive rates. Based on the Laplacian Editing method of Sorkine et al. [1] we have developed a feature-aware 2S shape editing method. We add to the standard elastic deformation (compression, stretching) the possibility to keep some user-specified shape feature preserved. They are allowed to naturally follow the prescribed deformation but should not change in size and shape. The problem is solved by modifying Sorkine’s least-squares formulation. We demonstrate the effectiveness of our approach in several examples, showing that the editing operations both change the global shape of the curve while respecting structural geometric detail and faithfully respecting a specified detail’s structural geometric nature.

Keywords: curve editing, detail preservation, Laplacian editing

1 Introduction

Preserving details is an important issue in shape editing. Many previous mesh editing methods are based on an elastic deformation model and thus tend to uniformly distribute the distortion in a least-squares sense over the entire deformation region. Therefore the details are generally not well preserved since they undergo the same relative deformation as the global shape.

The present work is inspired by a recently published work of Dekkers and Kobbelt [2] which extend the image-based technique seam carving [3] to 3D meshes. The general idea in [3] is to separate the shape into regions which are allowed to deform elastically and regions which should deform rigidly and gradually define the elasticity of the in-between regions. Even though we don’t use the technique of computing seams, our general goal is similar. We aim to develop a 2D shape editing technique which combines elastic shape parts with rigid shape parts. To this end we extend the rotation invariant Laplacian Editing approach of Sorkine et al. [1] with a simple technique which allows to apply only rigid deformations to the user-specified details on a 2D curve.

The rest of the report is organized as follows. In Sections ??, ?? and ?? we present three existing 2D shape editing methods. Based of the Laplacian Edition method, we present our approach in Section ?? and show some results in Section ?? before discussing the limitations in Section ?? and concluding the report.

*e-mail: alexandre.ribard@ensimag.grenoble-inp.fr
2 Free Form Deformation

Free Form Deformations as introduced by Sederberg [?] for 3D objects can also be applied to 2D shapes. A 2D Free Form Deformation (FFD) is a map from \( \mathbb{R}^2 \rightarrow \mathbb{R}^2 \). In the way that it defines a new position for every point in a given rectangular section.

![Image](image1.png)

**Figure 2:** Two FFD examples: the original model (left) is deformed using a 2D Free Form Deformation.

FFD is a three step process:

1. Calculate the relative coordinates of every point to be deformed to the grid.
2. Deform the grid by moving control points. Note: Control points can be interactively moved by the user or automatically generated by an animation routine. For instance, if the user wants to deform the shape of a teapot, he could drag one of the control points farther away to stretch the teapot. Most FFD tools allow the user to interactively change the control points and immediately see the deformation of the object inside the grid.
3. Compute deformed positions of every point. Since every point is mapped relatively to the original grid. We map their newly deformed coordinates from the deformed grid.

We have computed an example in Figure ??.

![Image](image2.png)

**Figure 3:** Deformation of the crocodile model (a) with FFD. First we calculate the position of the model (a) relatively to the grid (in red). The user then specifies a deformation which changes the shape of the grid. This gives us the deformed grid in figure (b). Using the mapping function used in calculating the relative coordinates, we get the deformed coordinates of the model’s vertexes. This gives us the deformed model in (b).

3 As-Rigid-As-Possible Deformation

The 2D As-Rigid-As-Possible deformation method of Igarashi et al. [?] is an interactive system that lets a user move and deform a two-dimensional shape without manually establishing a skeleton or freeform deformation (FFD) domain beforehand.

It first supposes that the boundary of the model can be represented as a simple closed polygon. As-Rigid-As-Possible Deformation first generates a triangulated mesh inside the boundary.

![Image](image3.png)

**Figure 4:** Overview of the system. The system first triangulates the original shape.

The user manipulates the shape by indicating handles on the shape and then interactively moves the handles (Figure 5). The user clicks on the shape to place handles and drags the handles to move them.

![Image](image4.png)

**Figure 5:** The user can interactively move specified handles.

Then, for the given handle configuration, the system first generates an intermediate result by minimizing scale-independent distortion (Figure 4.a). The system then fits triangles from the rest shape to corresponding triangles in the intermediate result (Figure 4.b). The system generates a final result (Figure 4.c) by minimizing the difference between the fitted triangles and the corresponding triangles.

This method thus consists of three steps:

▷ Step one: scale-free construction. Step one generates an intermediate result by minimizing an error function that allows rotation and uniform scaling.

▷ Step two: scale adjustment. This step takes the intermediate result from step one (the xy-coordinates of all vertices) as input and returns the final result (updated xy-coordinates of the free vertices) by adjusting the scale of the triangles in the mesh.

▷ Step three: Polishing. This step minimizes the difference between the fitted triangles and
In our 2D case, the formula (1) becomes:

\[ \delta_i = v_i - \frac{1}{d_i} \sum_{j \in N_i} v_j = L'(v_i) \]  \hspace{1cm} (1)

\( d_i \) = number of vertices neighboring \( v_i \) (=2 in our case)
\( N_i \) = all of \( v_i \)'s neighboring vertices.

These coordinates, also called Laplacian coordinates, encode information concerning the relative position between a vertex and its neighbors. Firstly by using those relative coordinates, one has an easy access to local information of the curve at each vertex.

The idea is to use this information and minimize it’s variation when easy access to local information of the curve at each vertex.

\[ \delta_{1x} \delta_{1y} \]
\[ . \]
\[ \delta_{nx} \delta_{ny} \]
\[ v_{1x} v_{1y} \]
\[ . \]
\[ v_{nx} v_{ny} \]
\[ = \Delta = LV \]
\[ V \]

\( L = I - D^{-1}A \)
\( D = \text{Diag}(d_i) = \text{Diag}(2) = 2I \) (in our case)
\( A(i,j) = \begin{cases} 1 & \text{if } i = j \\ 1 & \text{if } i \text{ and } j \text{ are neighbors} \\ 0 & \text{else} \end{cases} \)

The goal of Laplacian Editing is to minimize a certain energy which graduates the ‘irregularity’ of the new curve compared to the old one. Since the Laplacian coordinates \( \Delta(V) \) contain the intrinsic property of the particular details of the polygon curve \( V \), the idea is to minimize the difference between the old Laplacian coordinates and the new ones, i.e. minimize

\[ E(V') = \sum_{i=1}^{n} \|\delta_i - L'(v'_i)\| + \sum \|v'_i - u_i\| \]  \hspace{1cm} (4)

where \( u_i \) denotes the absolute coordinates of the anchors fixed by the user.

By solving this minimization problem, we will get new absolute coordinates of the vertices.

▷ The second term of the equation is here to impose that the handles remain fixed at the given position.

▷ With this constraint in place, the rest of the vertices’ coordinates are found by solving the minimization problem.

Even if this offers pleasant results, several problems do remain.

Most and foremost : the Laplacian coordinates are only translation invariant. Thus the minimization problem will yield proper results (global and local detail preservation) only if the deformation specified by the user is a translation.

In order to solve this major issue, Sorkine et al. [2] propose rotation and scale invariant Laplacian coordinates. They implicitly compute an appropriate transformation \( T_i \) per vertex \( v_j \) based on the eventual new (but wrong) configuration of vertices \( V' \).

▷ Since the Laplacian coordinates are sensible to scaling and rotations, the previous described method will not be able to transform the model accordingly. \( \Rightarrow \) detail will thus be lost.

So the newly transformed vertices \( V' \) are wrongly transformed \( (v'_{i,\text{ideal}} = v'_i + \text{err}_{i}) \).

So the idea is to compensate for that error (\( \text{err}_{i} \)) directly in the system above.

We thus try to find a certain transformation \( T_i \) (which is unknown and depends on how \( v'_i \) is wrongly transformed), which when applied to \( v_i \) in the system and will compensate for the bad transformation.

\[ \Rightarrow \]

\[ E(V') = \sum_{i=1}^{n} \|T_i(V')\delta_i - L'(v'_i)\| + \sum \|v'_i - u_i\| \]  \hspace{1cm} (5)

Note that both \( T_i(V') \) and \( V' \) are unknown.
So one needs to find \( T_i(V') \) in order to solve the minimization problem (6).
One can observe that, if \( T_i \)’s coefficients are a linear function in \( V' \), then solving for \( V' \) in (6) implies finding \( T_i \) since \( E(V') \) is simply a quadratic function in \( V' \). It would simply require that we add a term to (4).

The idea for defining \( T_i \) as a linear function of \( V' \) is to derive it from \( v_i \) and its neighbors transformation into \( v_i' \) (\( i \)).

\[
\text{i.e:} \quad \min_{T_i} \left( \|T_i v_i - v_i'\| + \sum_{j \in N_i} \|T_i v_j - v_j'\| \right) \quad (6)
\]

where the transformations \( T_i \) are restricted to translations, rotations and dilations, i.e.

\[
T_i = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} E & 0 & 0 \\ 0 & F & 0 \\ 0 & 0 & 1 \end{pmatrix}
\]

\[
= \begin{pmatrix} a & w & t_x \\ -w & a & t_y \\ 0 & 0 & 1 \end{pmatrix}.
\quad (7)
\]

Given the matrix in equation (7), we can write down the linear dependency (cf Eq (6)). We thus want to minimize:

\[
\|A_i \begin{pmatrix} a \\ w \\ t_x \\ t_y \end{pmatrix} - b_i \| \quad (8)
\]

Where \( A_i \) contains the position \( v_i \) and it’s neighbors. And \( b_i \) contains \( v_i' \) and its neighbors.

\[
A_i = \begin{pmatrix} x_{i-1} & y_{i-1} & 1 & 0 \\ x_i & y_i & 1 & 0 \\ x_{i+1} & y_{i+1} & 1 & 0 \\ x_{i-1} & y_{i-1} & 0 & 1 \\ x_i & y_i & 0 & 1 \\ x_{i+1} & y_{i+1} & 0 & 1 \end{pmatrix}
\]

\[
b_i = \begin{pmatrix} x_{i-1}' \\ y_{i-1}' \\ x_i' \\ y_i' \\ x_{i+1}' \\ y_{i+1}' \end{pmatrix}
\quad (9)
\]

The linear least-squares problem in (8) is solved by:

\[
\begin{pmatrix} a & w \\ t_x & t_y \end{pmatrix} = (A_i^T A_i)^{-1} A_i^T b_i \quad (10)
\]

\[
= A_i^{-1} b_i \text{ is } A_i \text{ is invertible}
\]

So \( T_i \)’s coefficients are a linear function of \( b_i \) is \( A_i \) is directly known from \( V \).

### 4.2 Implementation

A linear system has to be solved in order to find the correctly transformed vertices \( V' \).

- We first compute the matrix for the linear system.

\[
L' = \begin{pmatrix} L & 0 \\ 0 & L \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (11)
\]

\( L \) being defined at (3).

The last line is here to force the handles to maintained their specified positions.

\( L' \) is thus a \((n + n + nb\text{handles}) \times (n + n)\) matrix.

- We then compute \( A_i^{-1} \) (or it’s pseudo inverse)

\[
A_{inv} \quad \text{where}
\begin{align*}
(AA_{inv} A) & = A \\
(AA_{inv} A)^T & = AA_{inv} \\
(\delta x)^T & = (T_{\Delta x})
\end{align*}
\quad (12)
\]

- To add the \( \|T_i(V')\delta_i - \mathcal{L}(v'_i)\| \) constraint, we first calculate:

\[
T_{\Delta} = A_{inv} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = (T_{\Delta x} x_{anchor} \quad y_{anchor}) \quad (13)
\]

Then we update \( L' \):

\[
L'' = \begin{pmatrix} L - T_{\Delta x} & -T_{\Delta y} \\ -T_{\Delta y} & L - T_{\Delta x} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (14)
\]

- We then can then solve the least-square problem by inverting \( L'' \):

\[
L'' V' = \begin{pmatrix} 0 \\ . \\ . \\ x_{anchor} \\ y_{anchor} \\ 0 \\ . \\ . \\ 0 \end{pmatrix} \quad (15)
\]

### 5 Detail Preservation

Using Laplacian Curve Editing, I have developed an easy way a user can specify that a certain detail be explicitly preserved.

1. The user first loads the 2D model he wants to edit.
2. He then selects a zone of interest. This denotes the static and dynamic part of the model.
   \( \Rightarrow \) the static part of the curve will not be deformed.
3. The user then specifies a handle.
   \( \Rightarrow \) it will be moved by the user
   \( \Rightarrow \) Using Laplacian curve editing, the rest of the model is going to be morphed to both, fit with the specified handle position and preserve local and global detail of the model.
4. Before moving the handle the user selects the details to be explicitly preserved.
5. The model is then deformed.
⇒ Naturally (plausible) looking deformations with rotation, scale and translation invariant as well as explicit preservation of the specified details.

For example:

![Figure 7: The 'virus' model is loaded. The black * denote the boundaries between the static and dynamic part of the model. Here only the right side of the model is going to be deformed. The red * is the specified handle. The green part of the model (identified as the curve between the blue points) are the details which are to be explicitly preserved.](image)

**Figure 7:** The 'virus' model is loaded. The black * denote the boundaries between the static and dynamic part of the model. Here only the right side of the model is going to be deformed. The red * is the specified handle. The green part of the model (identified as the curve between the blue points) are the details which are to be explicitly preserved.

5.1 Work

**Motivation:** 2D Laplacian Curve Editing produce visually pleasing results. However, one cannot specifically preserve a certain detail on the model. We have developed a method which, using Laplacian Curve Editing discussed above, enable a user to preserve specific details of the mesh.

We then demonstrate the effectiveness of our approach in several examples, showing that the editing operations both change the global shape of the mesh while respecting structural geometric detail and faithfully respecting a specified detail’s structural geometric nature.

Laplacian Curve Editing is really effective at preserving local and global detail. Thus is we deform a model, the local and global aspect of the curve is going to be preserved by the editing process.

Now consider a specific detail.
A detail on the model is fully described by 3 factors:
1. Its geometric aspect.
2. Its orientation relatively to the global curve.
3. Its relative scale.

Laplacian Curve Editing deforms the whole curve uniformly with no head towards the relative scale of the detail.
We notice that, at the end of the Laplacian deformation:
- Its geometric aspect is preserved.
- Its relative orientation is equally preserved.

However, its scale is not.
⇒ We thus need to impose that the relative scale of the detail remains the same throughout the deformation.

**Implementation:**

The idea is specify in the system that the scale of the specified detail is exactly preserved.
We assimilate the detail to a triangle cf Figure 8.
We then specify that the scale of this triangle remains constant throughout the deformation.

![Figure 8: In green, the part denoting the details of the cloud model which are to be preserved. As we can see, the triangle abc globally denotes the general scale of the detail. [a,b] ≈ detail\_width and [m,c] ≈ detail\_height. If the three points: a,b,c remain at their relative distance from one other throughout the deformation, the scale of the detail will be preserved.](image)

We first compute:

\[
\begin{align*}
\|a - b\| &= \text{dist}(a,b) = t_{\text{width}} \\
\|m - c\| &= \text{dist}(m,c) = t_{\text{height}}
\end{align*}
\]

(16)

In order to impose that these distance remain constant throughout the deformation, for each specified details we add 2 lines to the linear system discussed above.

\[
L'' = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & x_{\text{anchor}} \\
0 & 0 & 0 & 0 & 0 & y_{\text{anchor}} \\
0 & 0 & 0 & 0 & 0 & t_{\text{width}} \\
0 & 0 & 0 & 0 & 0 & t_{\text{height}}
\end{pmatrix}
\]

(17)

The first line is to compute \(d_w = |x_a - x_b| + |y_b - y_a|\). To then impose that \(d_w = t_{\text{width}}\), we modify eq (15.). The second line is for \(d_h = |x_m - x_c| + |y_m - y_c| = t_{\text{height}}\).

Eq (15.) then becomes:

\[
L'''V' = \begin{pmatrix}
0 \\
. \\
. \\
. \\
. \\
0
\end{pmatrix}
\]

(18)

We then reiterate that process for every specified details.
User Interface:

Since the user has specified to us what detail he wants preserved, we can easily identify the width and height of the detail. Or we simply ask the user to draw two lines in order to identify them.

For every model, we use only a couple of triangles at the right place to preserve the desired detail. We could use automatic methods to identify specific details in a zone of interest. We could automatically determine the best distance to conserve in order that the proportions of the detail remain constant. This would save the user some time. For example with the crocodile (Figure 12.), specifying that every teeth is preserved could be tedious.

6 Results

Using Matlab, we have implemented this method. The user first loads a specific model. The handles as well as the details are then specified. We then perform 2D Laplacian Deformations coupled with the detail-preservation technique discussed above.

Figure 9: Figure (a) : regular Laplacian editing. Figure (b) : with detail preservation. Figure (a) and (b): the green curve denotes the initial model. The size and shape of the detail are preserved. The details also translated and rotated naturally while conserving their initial shape. All of this following the prescribed deformation of the red handle.

Figure 10: Multiple details can be added and preserved. Figure (a) : initial model. Figure (b) : Laplacian Editing with detail preservation. Figure (a): the green curve denotes the initial model.
Figure 11: Figure (a) : regular Laplacian editing. Figure (b) : with detail preservation. Figure (a) and (b): the green curve denotes the initial model. The size and shape of the detail are preserved. The details also translated and rotated naturally while conserving their initial shape. Note with (b): the detail conserve their relative orientation to their neighbors as well as keeping their initial size.

Figure 12: Green curve: original model. Respectively: with detail preservation (c) and none (b). We have identified and conserved 3 details in figure (c). The lower and upper jaw as well as the space in between. We thus impose that the height and length of the jaw (upper and lower) is conserved. We do the same for the space in between. The jaw remains the same size (compared to stretching (a)). What is more, the teeth remain the same size and conserve their respective orientation.
7 Limitations and Future Work

There is however one issue concerning this method.

If the number of constraints (used to preserved specified details) is too important (ie: there isn’t enough constraint-free points to deform accordingly to the underlying Laplacian Editing), this method produces unpleasing results cf Figure 14.

- Seam carving methods could be used to counteract this problem (ie: artificially adding points on the fly to the model to obtain a correct deformation).
- We could also deform the model first then use a 'relaxation' technique to re-scale the detail correctly.

Our experiences suggests that it is necessary to maintain an equilibrium between the number of constrained points by our detail-conservation method and the one free to deformed according to the Laplacian deformation. Finding a satisfying solution for this issue could be a path to explore.
Acknowledgements

I would like to express my gratitude towards Stefanie Hahmann. The *Crocodile, Sin Circle, Unicorn, Virus* and *Seahorse* models are courtesy of Stefanie Hahmann.