Project: flows on 3D surfaces

Daw Lara, Pignotti Simone, Revuz Anselme

March 29, 2018

Introduction

The goal of the project was to implement different kind of flows for 3D surfaces using the programming framework Processing, and to compare the results obtained on multiple surfaces encoded as PLY files. We have implemented the harmonic flow, the harmonic flow divided by the area, the mean curvature flow and its squared variant. We also made the computations compatible with open surfaces, for which the boundary vertices should remain fixed.

Documentation

To implement the interface, we used the Processing library G4P. To install it, open Processing IDE (tested on v3.3.6), go to the "Sketch" menu, click on "Import Library...", and finally "Add Library...". In the window which appears, search "g4p" in the field "Filter" and install "G4P" (tested on v4.1.4 of such library).

The main menu is composed by three elements:

- the "NEW" button, allowing to open a new window;
- the "Start/Stop flow" button, allowing to start or stop the flow on every window;
- a list of clickable buttons for opening each window's parameter menu.

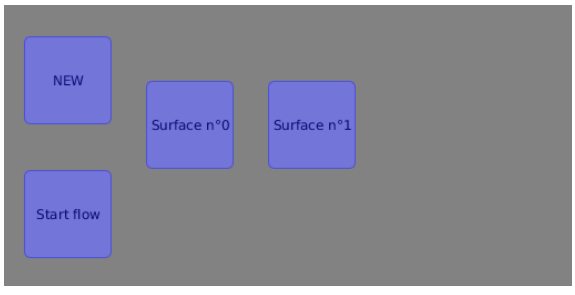

Figure 1: Main menu

When clicking on the button associated to a window, a new menu appears allowing to modify the following parameters:

- the flow to apply, encoded by an integer, namely:
  1. mean curvature followed by volume renormalization;
  2. mean curvature with projection on the volume constraints;
  3. squared mean curvature followed by volume renormalization;
  4. squared mean curvature with projection on the volume constraints;

5. harmonic followed by volume renormalization;
6. harmonic with projection on the volume constraints;
7. harmonic followed by division by the area and volume renormalization;
8. harmonic with division by the area and projection on the volume constraints;

- the value of "tau", a float $0 < \tau \leq 1$ representing the time scale applied to the flow;
- the PLY file to use, with examples provided in the data directory;
- the activity of the flow on the current window, allowing to stop the flow on it even if "Start flow" was pressed in the main menu.

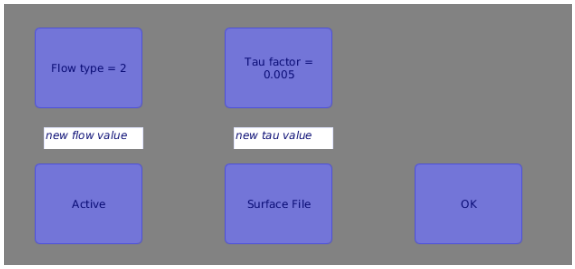

Figure 2: Parameters of each window

At creation, a window opens the cube surface, with no flow applied and $\tau = 0.005$. Once created, the windows cannot be closed. To disable/enable the flow on a window, click on the "Active/Inactive" button.

The value of τ is parsed as a float, so it is expected to be a floating point number in the standard notation.

The button "OK" allows to go back to the main menu, but every modification is immediately applied, even without pressing "OK".

When a window is selected, the keys 'A' and 'Z' allows to zoom in and out, respectively.

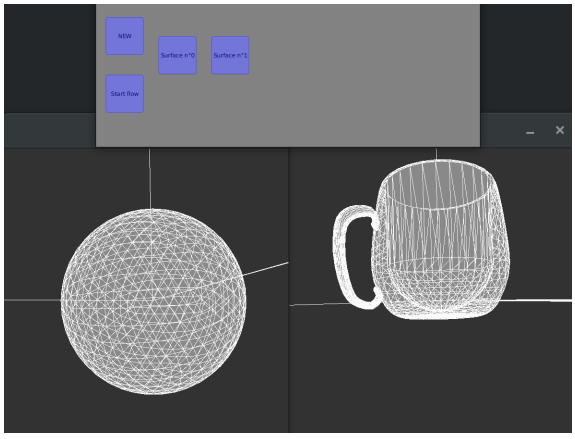

Figure 3: Multiple windows synchronously applying different flows on different surfaces

Implementation of the flows

In all the following descriptions, the flow for boundary vertices is set to 0.

Harmonic Flow: For each vertex p_i, compute the set of adjacent vertices N_i. For each $p_j \in N_i$, add $\overrightarrow{p_i p_j}$ to $\overrightarrow{H_i}$, and divide by the cardinality of N_i.

$$\overrightarrow{H_i} = \frac{1}{|N_i|} \sum_{p_j \in N_i} \overrightarrow{p_i p_j}$$

Harmonic Area Flow: For each vertex p_i, compute the set of adjacent vertices N_i. First, calculate the harmonic flow $\overrightarrow{H_i}$ as above, then divide by the area calculated in the following way:

$$A_i = \frac{1}{2} \sum_{j=1}^{|N_i|-1} ||\overrightarrow{N_i[0]N_i[j]} \times \overrightarrow{N_i[j]N_i[j+1]}||$$

Finally the flow will yield:

$$\overrightarrow{H_i} = \frac{1}{A_i} \left(\frac{1}{|N_i|} \sum_{p_j \in N_i} \overrightarrow{p_i p_j} \right)$$

Mean Curvature Flow: Let us define the notation used for each vertex:

- q is the point for which the MCF is being computed;
- p_i is the predecessor of q on face j, the successor of q on face $(j+1)$ and therefore the shared vertex of faces $(j, j+1)$;
- p_{i-1} is the successor of q on face j;
- p_{i+i} is the predecessor of q on face $(j+1)$;
- $\overrightarrow{m_i}$ is the cross product of the edge (p_i, p_{i+1}) by the normal to the plane of (p_i, q) and (p_{i+1}, q).

Then the following steps are applied:

- find the first incident face f and the predecessor and the successor of q in f;
- iterate over faces containing q in the right order (by finding the face containing q and its predecessor;
- compute the MCF with the formula:

$$\overrightarrow{MCF_i} = -\frac{1}{2} \sum_{f \in F : q \in f} \overrightarrow{m_i}$$

Mean Curvature Flow (cotan version): This version has been implemented but is not runnable from the main menu. The other version yields more stable results and is therefore the default choice.

Let us keep the same notation as for the other version, and introduce:

$$\alpha_B = \widehat{\overrightarrow{qp_{i-1}}\ \overrightarrow{p_{i-1}p_i}}$$

$$\alpha_A = \widehat{\overrightarrow{p_i p_{i+1}}\ \overrightarrow{p_{i+1}q}}$$

where B stands for "before" and A for "after". $\overrightarrow{m_i}$ in this case is the edge (p_i, q). Then:

- find the first incident face f and the predecessor and the successor of q in f;
- Iterate over the faces in the right order by finding the face containing q and its predecessor;

Finally, the MCF is given by:

$$\overrightarrow{MCF_i} = \frac{1}{2} \sum_{f \in F : q \in f} (\cot \alpha_B + \cot \alpha_A) \times \overrightarrow{m_i}$$

Projection on the volume constraints: Furthermore, we implemented a method to project any flow on the volume constraints of the surface and conserve the volume "a priori", even for open surfaces. Of course it is still possible to renormalize "a posteriori" by using the ratio between the initial volume of the surface and the one computed after having applied the flow.
Step 1: **Gradient**
For each vertex p_i, iterate over the faces to which it belongs and:

- find the position of p_i in this face;
- iterate over all the vertices other than p_i two by two and calculate their cross product $g_f = \overrightarrow{p_l} \times \overrightarrow{p_k}$.

The gradient of p_i is given by $\overrightarrow{G_i} = \sum_{f \in F : p_i \in f} g_f$.
Step 2: **Projection**
For each vertex p_i, the flow F_i will be updated to:

$$\hat{F_i} = \overrightarrow{F_i} - \frac{\overrightarrow{G_i} \cdot \overrightarrow{F_i}}{||\overrightarrow{G_i}||} \overrightarrow{G_i}$$

Examples

Meaningful examples are given in the form of GIF animations for some of the surfaces. They can be found in the gifs directory of the repository. They all use the MCF with different values of τ on different surfaces:

- a sphere with boundary vertices around a pole, with $\tau = 0.01$;
- a dodecahedron with $\tau = 0.01$;
- a custom surface composed by cubes and half-cubes, with $\tau = 0.01$;
- a mug with $\tau = 0.005$.

This is a list of recommended values of τ for a selection of examples. To run the squared variant of the MCF, a generally satisfying practice is to divide the recommended value by a factor of 100, while the HF variant divided by the area requires a multiplication by 100. While these values work best with the projection on the volume constraint, the renormalization versions of all flows shall behave correctly with the same values, and only minor adjustments are needed.

Filename	MCF	HF
boundarySphere	0.05	0.1
sphere10-20	0.05	0.1
dodecahedron	0.01	*
newsurf	0.001	0.01
mug	0.005	0.01

In most of the examples, the time scale should be decreased after some time to allow the vertices to stabilize their positions. This values can be used as reference to launch the flow, but further adjustments are needed in most of the cases for optimal performance.

Conclusions

The mean curvature flow and its squared variant yield the best results, but unfortunately none of the implementations manage to keep the original topology of all surfaces. On complex surfaces, like the mug example, the results are not as good as expected (no "smoothing" effect), but still the computations are numerically stable enough not to end up in geometrically ill behaviours, unless the value of τ is too high.

The biggest improvement was brought by the implementation of the projection on the volume constraints, which improved the stability of all flows. A possible improvement is a method to check that vertices do not cross other faces, but this would add a huge complexity to the calculations, time and space-wise.