CS 6491 Project 4 - Hamiltonian Path Animation

Ana Huamán Quispe

I. Problem Statement

- Given a manifold mesh, perform a Hamiltonian traversal by using the Laced Ring algorithm [1] discussed in class.
- Animate the path creation procedure.

II. Hamiltonian Path

A. *Laced Ring Algorithm*

In order to create a path that passed through all the vertices of a given manifold mesh, we implemented a Hamiltonian path using the Laced Ring algorithm. The basic algorithm is shown in Procedure 1. As it can be seen, it consists of a greedy invasion of the mesh triangles starting from a random seed corner s. The path generated is edge-connected.

Procedure `ringExpander(s)`

---
1   $c = s$;
2   $c.n.v.m = c.p.v.m = true$;
3 **repeat**
4   **if** $!c.v.m$ **then**
5        $c.v.m = c.t.m = true$;
6   **else if** $!c.t.m$ **then**
7        $c = c.o$;
8     $c = c.r$;
9 **until** $c! = s.o$;

B. *Implementation Details*

- We used the standard Corner Table data structure to save the mesh information. However, we added extra vectors to save new data available:
  - Marked triangles: Triangles that belonged to the Hamiltonian path
  - Triangle type: A vector that stored the type of each Marked Triangle (T_0, T_1 or T_2)

III. Animation

A. *Branch-based path representation*

The Hamiltonian path obtained in the previous section is a tree of triangles with multiple branches. In order to animate this path as a volumetric curve, we calculated and stored the baricenter of each Marked triangle (triangles on the path). Our base approach consists on simply forming a curve that passes through these points.

The first problem, of course, comes with the fact that the path branchs off at some triangles, hence the curve cannot be obtained by simply concatenating all the baricenters. To overcome this, we noticed 2 important things:

- The Hamiltonian path, obtained with the LR algorithm, grows branches on the points corresponding to triangles of the type T_0, that is, triangles whose edges are shared with other path triangles
- Starting at a triangle T_0 on the Hamiltonian path obtained with LR, we reach the end triangle of a branch if and only if this end triangle is of type T_2 (has 2 shared edges with no-path triangles)

With these observations in mind, our approach for animating will express the Hamiltonian path as a set of branches which start at baricenters of triangles of type T_0 and end in corresponding baricenters of triangles T_2:

- For all baricenters of the Marked triangles
  - If the corresponding triangle is of type T_0, follow the Hamiltonian path (stored in Marked Triangles); adding the traversed baricenter points in a vector p. This will represent a new branch of our path.
  - End the traverse, once the corresponding triangle is of type T_2
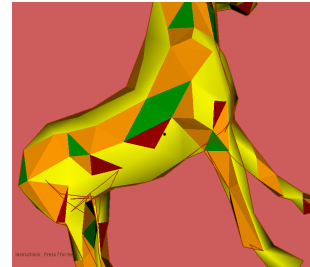 - Store p as a branch

Fig. 1. Example of triangles T_0 and T_2 (depicted in green and red respectively)

B. *Visualization*

The visualization of our Hamiltonian path consists on showing all the branches found in the previous step. For this purpose, we use the Frenet-Serret formulas for each branch (which is after all a curve). We depict the branches as curves that have a circular section at each baricenter point. These sections are located in the plane formed by the Normal and Binormal vector of each corresponding point. Then, we only join these sections and a volume is formed.

For this section we reused part of the base code given in class, which was modified to include visualizing the different branches and the animation procedure.

IV. Results

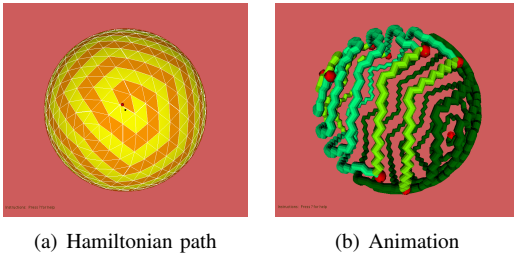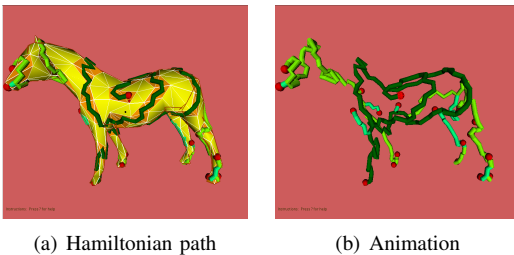Results of our approach using 3 different kind of meshes are shown in the following figures:

(a) Hamiltonian path   (b) Animation

Fig. 2. Results with sphereSmooth.vts

(a) Hamiltonian path   (b) Animation

Fig. 3. Results with horse.vts

(a) Hamiltonian path (bunny front view)   (b) Animation
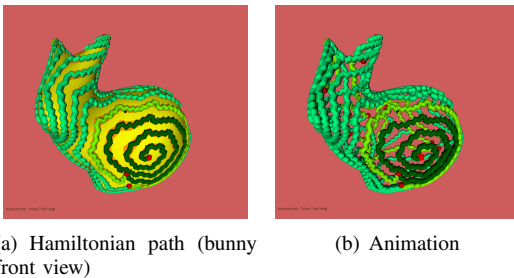
Fig. 4. Results with bunny.vts (front view)

(a) Hamiltonian path (bunny back view)   (b) Animation
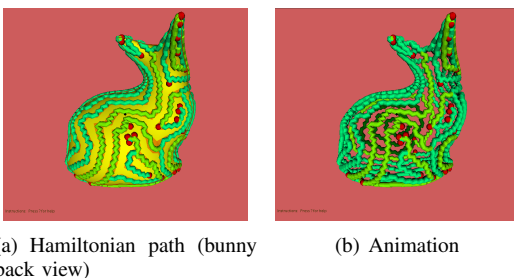
Fig. 5. Results with bunny.vts (back view)

References

[1] T. Gurung, M. Luffel, P. Lindstrom, and J. Rossignac, "Lr: compact connectivity representation for triangle meshes," in *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4. ACM, 2011, p. 67.