Assignment Seven: Persistent Homology for Handle and Tunnel Loops

David Gu

Yau Mathematics Science Center
Tsinghua University
Computer Science Department
Stony Brook University

gu@cs.stonybrook.edu

August 29, 2020
Figure: Handle and tunnel loops of the amphora model.
Figure: Null homotopy detection.
Figure: Birkhoff curve shortening.
Figure: Handle and tunnel loops.
Figure: Handle and tunnel loops.
Figure: Interior and exterior volumes.
Volumetric Mesh Generation

The input is an oriented closed triangular mesh, use Dr. Hang Si’s tetgen to generate the interior and exterior volumetric mesh.

**Interior Volume**

Use Tetgen to generate the interior tetrahedral mesh inside the mesh $M$, denoted as $I_M$.

**Exterior Volume**

Construct a sphere enclosing the input mesh $M$, use Tetgen to generate a tetrahedral mesh between the sphere and the mesh $M$. Add the infinity point $\infty$, connect $\infty$ with each triangle face on the sphere to form a tetrahedron, denoted as $O_M$. 
Filtration Generation

Interior Volume

Extract the boundary surface of the interior volume $M = \partial I_M$; Sort all the vertices, edges, faces of $M$, 

$$\sigma_0^1, \sigma_0^2, \cdots, \sigma_0^{n_0}, \sigma_1^1, \sigma_1^2, \cdots, \sigma_1^{n_1}, \sigma_2^1, \sigma_2^2, \cdots, \sigma_2^{n_2}.$$ 

After that insert the interior vertices, edges, faces and tetrahedra of $I_M \setminus M$, 

$$\tau_0^1, \tau_0^2, \cdots, \tau_0^{m_0}, \tau_1^1, \tau_1^2, \cdots, \tau_1^{m_1}, \tau_2^1, \tau_2^2, \cdots, \tau_2^{m_2}, \tau_3^1, \tau_3^2, \cdots, \tau_3^{m_3}.$$
Extract the boundary surface of the exterior volume $M = \partial O_M$; Sort all the vertices, edges, faces of $M$, 

$$\sigma_0^1, \sigma_0^2, \ldots, \sigma_{n_0}^0, \sigma_1^1, \sigma_1^2, \ldots, \sigma_{n_1}^1, \sigma_2^1, \sigma_2^2, \ldots, \sigma_{n_2}^1.$$ 

After that insert the interior vertices, edges, faces and tetrahedra of $O_M \setminus M$, 

$$\tau_0^1, \tau_0^2, \ldots, \tau_{m_0}^0, \tau_1^1, \tau_1^2, \ldots, \tau_{m_1}^1, \tau_2^1, \tau_2^2, \ldots, \tau_{m_2}^1, \tau_3^1, \tau_3^2, \ldots, \tau_{m_3}^3.$$
Pair Algorithm

Pair(σ)

1. $c = \partial_p \sigma$

2. $τ$ is the youngest positive $(p - 1)$-simplex in $c$.

3. while $τ$ is paired and $c$ is not empty do

4. find $(τ, d)$, $d$ is the $p$-simplex paired with $τ$;

5. $c \leftarrow \partial_p d + c$

6. Update $τ$ to be the youngest positive $(p - 1)$-simplex in $c$

7. end while

8. if $c$ is not empty then

9. $σ$ is negative $p$-simplex and paired with $τ$

10. else

11. $σ$ is a positive $p$-simplex

12. endif
The simplices on the surface $M$ are added into the filtration in any arbitrary order. Since $H_1(M)$ is of rank $2g$, the algorithm Pair generates $2g$ number of unpaired positive edges.

The simplices up to dimension 2 in $I$ are added into the filtration. Since $H_1(I)$ of rank $g$, half of $2g$ positive edges generated in step 1 get paired with the negative triangles in $I$. Each pair corresponds to a killed loop, these $g$ loops are handle loops.

Or the simplices up to dimension 2 in $O$ are added into the filtration. Since $H_1(O)$ of rank $g$, half of $2g$ positive edges generated in step 2 get paired with the negative triangles in $O$. Each pair corresponds to a killed loop, these $g$ loops are tunnel loops.
Null Homological Cycle Detection

Input: a graph $G$ on the mesh $M$ labeled as sharp edges;
Output: remove null homological cycles

1. Build a spanning tree $T$ of $G$, $G \setminus T = \{e_1, e_2, \ldots, e_k\}$;
2. Construct cycles $c_i = T \cup e_i$, $i = 1, 2, \ldots, k$;
3. Compute the persistent homology of the mesh $M$;
4. for each cycle $c_i$ find the unpaired youngest generator; if one can not find the generator, then $c_i$ is null homologous.
Birkhoff curve shortening

Diagram showing a network of nodes and edges labeled with indices from 0 to 7, representing a conformal mapping or a curve evolution process in a computational geometry context.
Birkhoff curve shortening
Birkhoff curve shortening

Input: a loop $c$ on $M$ labeled as sharp edges;
Output: a shortened cycle homotopic to $c$;

1. Sort the vertices of $c$ as $v_0, v_1, \ldots, v_{n-1}$;
2. Find the shortest path between $v_0$ and $v_{n/3}$, and replace the sequence of edges between $v_0$ and $v_{n/3}$;
3. Find the shortest path between $v_{n/3}$ and $v_{2n/3}$, and replace the sequence of edges between $v_{n/3}$ and $v_{2n/3}$;
4. Find the shortest path between $v_{2n/3}$ and $v_0$, and replace the sequence of edges between $v_{2n/3}$ and $v_0$;
5. Cyclically shift the vertex sequence, and repeat step 2 through step 4.
Instruction
1. ‘DartLib’, a volumetric mesh library based on Dart data structure.
2. ‘freeglut’, a free-software/open-source alternative to the OpenGL Utility Toolkit (GLUT) library.
Directory Structure

- 3rdparty/DartLib, header files for volumetric mesh;
- HandleTunnelLoop/include, the header files for handle-tunnel loop computation;
- data, Some data models and batch scripts;
- CMakeLists.txt, CMake configuration file;
- resources, snapshot for circular slit mapping results;
Before you start, read README.md carefully, then go through the following procedures, step by step.

1. Install [CMake](https://cmake.org/download/).
2. Download the source code of the C++ framework.
3. Configure and generate the project for Visual Studio.
4. Open the .sln using Visual Studio, and compile the solution.
5. Finish your code in your IDE.
6. Run the executable program.
Configure and generate the project

1. open a command window
2. cd Assignment_7_skeleton
3. mkdir build
4. cd build
5. cmake ..
6. open CCGHomework.sln inside the build directory.
Finish your code in your IDE

- You need to modify the file: HandleTunnelLoop.cpp;
- search for comments “insert your code”
- Modify functions:
  1. `CHandleTunnelLoop :: _pair(std :: set < M :: CVertex* > &vertices)`
  2. `CHandleTunnelLoop :: _pair(std :: set < M :: CEdge* > &edges)`
  3. `CHandleTunnelLoop :: _pair(std :: set < M :: CFace* > &faces)`
  4. `CHandleTunnelLoop :: _mark_loop(M :: CFace * killer)`
Modify assignment one, CutGraph, to implement the algorithms for null homologous cycle detection and Birkhoff curve shortening.