Assignment Five: 4D Convex Hull Construction

David Gu

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

gu@cs.stonybrook.edu

December 25, 2020
The input to the 4D convex hull algorithm is a set of 3D points

\[ P = \{ p_1, p_2, \ldots, p_n \} \]

The output is the 3D Delaunay triangulation of the point set \( P \).

**Input**

The input points are randomly generated within the unit sphere.

**Output**

The Delaunay triangulation is represented as a tetrahedral mesh, using Dart data structure to store.
**Convex Hull**

**Algorithm Pipeline**

- Lift each point \( p_i = (x_i, y_i, z_i) \) to the parabola,
  \[ q_i = (x_i, y_i, z_i, w_i), \quad w_i = \frac{1}{2}(x_i^2 + y_i^2 + z_i^2), \]

- Pick four points to form two tetrahedra with opposite orientations, and glue them to form a topological ball, and assign the ball as the initial convex hull of \( \{q_i\} \), denoted as \( C \);

- Select a point \( q_l \), which is as far as possible from the current \( C \);

- For each face on the hull \( C \), test the visibility with respect to \( q_l \);

- Remove all the visible tetra from \( C \);

- For each face \([q_i, q_j, q_k] \) on the contour (the surface separating the visible and invisible parts of \( C \)), connect the face with the point \( q_l \) to form a triangle \([p_i, p_j, p_k] \), add the face to \( C \);

- Repeat step 2 through 5, until all the points have been processed.
Visibility Testing

Given a tetrahedron \([p_i, p_j, p_k, p_l]\) and the new point \(p_m\), the visibility testing is equivalent to compute the volume of the 5-simple \([p_i, p_j, p_k, p_l, p_m]\), which is given by

\[
\begin{vmatrix}
    x_i - x_m & y_i - y_m & z_i - z_m & w_i - w_m \\
    x_j - x_m & y_j - y_m & z_j - z_m & w_j - w_m \\
    x_k - x_m & y_k - y_m & z_k - z_m & w_k - w_m \\
    x_l - x_m & y_l - y_m & z_l - z_m & w_l - w_m \\
\end{vmatrix}
\]

and check whether the volume is positive or not.
Figure: Convex hull computation process.
Figure: Convex hull computation process.
Instruction
Dependencies

1. ‘DartLib’, a general purpose 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 mesh;
- convex_hull/include, the header files for convex_hull;
- convex_hull/src, the source files for convex_hull;
- CMakeLists.txt, CMake configuration file;
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_1_skeleton
3. mkdir build
4. cd build
5. cmake ..
6. open OTHomework.sln inside the build directory.
Finish your code in your IDE

- You need to modify the file: ConvexHull4D.cpp;
- search for comments “insert your code”
- Modify functions:
  1. `CConvexHull4D :: volume_sign(const CTetTest, const CPoint4p)`
  2. `CConvexHull4D :: _init(std :: vector < CPoint4 > sites)`
  3. `CTetMesh :: create_tet(const CDArray < int > test)`
  4. `CTetMesh :: remove_tet(const int tet_idx)`
Finish your code in your IDE

Try your best to improve the efficiency.