3D Polyhedron Reconstruction and Analysis System - User Documentation

Himangi Parekh

October 28, 2024

Contents

1 Introduction **2**

2 Downloading Libraries and Compiling Instructions **2**

3 Installing Visual Studio Code **2**
3.1 Windows . . . 3
3.2 macOS . . . 3
3.3 Linux . . . 3

4 Installing Standard C++ Libraries **3**
4.1 Installing a C++ Compiler . . . 3
4.1.1 Windows . . . 3
4.1.2 macOS . . . 4
4.1.3 Linux . . . 4

5 Installing Eigen Library **4**
5.1 Windows . . . 4
5.2 macOS and Linux . . . 4

6 Installing SDL2 Library **4**
6.1 Windows . . . 5
6.2 macOS . . . 5
6.3 Linux . . . 5

7 Locating Eigen and SDL2 Directories **5**
7.1 Windows . . . 5
7.2 macOS (Homebrew) . . . 5
7.3 Linux . . . 5

8 Configuring the Makefile **6**
8.1 Example for macOS (Homebrew) . . . 6
8.2 Example for Linux . . . 6

9 Sample Inputs and Outputs **6**

10 Limitations and Scope **9**
10.1 Limitations 9

1 Introduction

This system is designed to reconstruct 3D polyhedra from 2D projections, perform various geometric calculations, and apply transformations. The main components of the system include:

- Input processing and 3D reconstruction
- Geometric calculations (surface area, volume, center of mass)
- Moment of inertia calculation
- Orthographic projections (onto any plane) (equivalent to slicing)
- Transformations (rotation, translation, scaling, reflection)
- Validation of input and reconstruction
- 3D Dynamic Projection (control through cursor)

The system aims to provide a comprehensive tool for engineers and designers working with 3D polyhedra with "holes".

Note that the scientific definition of the polyhedron is from ScienceDirect: "A polyhedron is a three-dimensional solid bounded by a finite number of polygons called faces." The bounded nature of the polyhedron implies it is closed. https://www.sciencedirect.com/topics/mathematics/polyhedron

2 Downloading Libraries and Compiling Instructions

3 Installing Visual Studio Code

Visual Studio Code (VS Code) is a popular, lightweight code editor. The installation instructions below are organized by operating system.

3.1 Windows

1. Open your web browser and go to `https://code.visualstudio.com/`.
2. Click on the `Download for Windows` button.
3. Once the download is complete, open the installer file and follow the on-screen instructions.
4. Launch Visual Studio Code once the installation is complete.

3.2 macOS

1. Open your web browser and go to `https://code.visualstudio.com/`.
2. Click on the `Download for macOS` button.
3. Open the downloaded `.dmg` file and drag the Visual Studio Code icon into the `Applications` folder.
4. Launch Visual Studio Code from your Applications folder.

3.3 Linux

1. Open your terminal.
2. Use the following command to download and install Visual Studio Code:

```
    sudo snap install code --classic

```

3. Once the installation is complete, you can open VS Code by typing:

```
    code

```

4 Installing Standard C++ Libraries

The libraries listed below, such as `iomanip`, `cmath`, and others, are typically included in most C++ compilers by default. You may not need to install them separately. Make sure to install a C++ compiler such as `gcc` (GNU Compiler Collection).

4.1 Installing a C++ Compiler

4.1.1 Windows

1. Install `MinGW` by downloading it from `https://sourceforge.net/projects/mingw/`.
2. Select `gcc` and other required tools during installation.

4.1.2 macOS

1. Open your terminal.
2. Install Xcode Command Line Tools by running:

```
    xcode-select --install

```

4.1.3 Linux

1. Open your terminal.
2. Use the following command to install `gcc`:

```
    sudo apt-get install build-essential

```

5 Installing Eigen Library

Eigen is a C++ template library for linear algebra.

5.1 Windows

1. Visit `https://eigen.tuxfamily.org/dox/GettingStarted.html`.
2. Download the latest version of Eigen.
3. Extract the downloaded folder, and move it to a location like `C:\Eigen`.

5.2 macOS and Linux

1. Open the terminal.
2. Use `Homebrew` (macOS) or `apt` (Linux) to install Eigen:

```
    # macOS
    brew install eigen
    # Linux
    sudo apt-get install libeigen3-dev

```

6 Installing SDL2 Library

SDL2 is a library that allows for the creation of cross-platform applications.

6.1 Windows

1. Visit `https://www.libsdl.org/download-2.0.php`.
2. Download the latest SDL2 development library for Visual C++.
3. Extract the contents and move them to a folder such as `C:\SDL2`.

6.2 macOS

1. Open the terminal.
2. Use `Homebrew` to install SDL2:

```
    brew install sdl2

```

6.3 Linux

1. Open the terminal.
2. Install SDL2 using the following command:

```
    sudo apt-get install libsdl2-dev

```

7 Locating Eigen and SDL2 Directories

After installing the libraries, locate the installation paths. Below are the typical installation locations:

7.1 Windows

- Eigen: `C:\Eigen`
- SDL2: `C:\SDL2`

7.2 macOS (Homebrew)

- Eigen: `/opt/homebrew/include/eigen3`
- SDL2: `/opt/homebrew/Cellar/sdl2/<version>/include`

7.3 Linux

- Eigen: `/usr/include/eigen3`
- SDL2: `/usr/include/SDL2`

8 Configuring the Makefile

To set up the include and library paths in the `Makefile`, update the `INCLUDE` and `LIB` variables as shown below.

```
# Include and library paths
INCLUDE = -I /path/to/eigen3 -I/path/to/sdl2/include
LIB = -L/path/to/sdl2/lib -lSDL2
```

Replace `/path/to/eigen3` and `/path/to/sdl2/include` with the actual paths for your system.

8.1 Example for macOS (Homebrew)

If installed via Homebrew, you can use the following configuration:

```
# Include and library paths
INCLUDE = -I/opt/homebrew/include/eigen3 -I/opt/homebrew/Cellar/sdl2/2.30.8/include
LIB = -L/opt/homebrew/Cellar/sdl2/2.30.8/lib -lSDL2
```

8.2 Example for Linux

```
# Include and library paths
INCLUDE = -I/usr/include/eigen3 -I/usr/include/SDL2
LIB = -L/usr/lib -lSDL2
```

9 Sample Inputs and Outputs

Polyhedron Input

```
Enter the number of vertices in the outer polyhedron: 8
Enter the number of faces in the outer polyhedron: 6
Enter the 2D coordinates for the vertices in the outer polyhedron:

// Outer cube vertices in 2D projections
Vertex 1 (xy): 1.0 1.0
Vertex 1 (xz): 1.0 1.0

Vertex 2 (xy): -1.0 1.0
Vertex 2 (xz): -1.0 1.0

Vertex 3 (xy): -1.0 -1.0
Vertex 3 (xz): -1.0 -1.0

Vertex 4 (xy): 1.0 -1.0
Vertex 4 (xz): 1.0 -1.0
```

```
Vertex 5 (xy): 1.0 1.0
Vertex 5 (xz): 1.0 -1.0

Vertex 6 (xy): -1.0 1.0
Vertex 6 (xz): -1.0 -1.0

Vertex 7 (xy): -1.0 -1.0
Vertex 7 (xz): -1.0 1.0

Vertex 8 (xy): 1.0 -1.0
Vertex 8 (xz): 1.0 1.0

// Define faces for the outer cube
Enter the number of edges in face 1 of the outer polyhedron: 4
Enter vertices for edge 1 in face 1 of the outer polyhedron (format: v1 v2): 1 2
Enter vertices for edge 2 in face 1 of the outer polyhedron (format: v1 v2): 2 3
Enter vertices for edge 3 in face 1 of the outer polyhedron (format: v1 v2): 3 4
Enter vertices for edge 4 in face 1 of the outer polyhedron (format: v1 v2): 4 1

Enter the number of edges in face 2 of the outer polyhedron: 4
Enter vertices for edge 1 in face 2 of the outer polyhedron (format: v1 v2): 5 6
Enter vertices for edge 2 in face 2 of the outer polyhedron (format: v1 v2): 6 7
Enter vertices for edge 3 in face 2 of the outer polyhedron (format: v1 v2): 7 8
Enter vertices for edge 4 in face 2 of the outer polyhedron (format: v1 v2): 8 5

Enter the number of edges in face 3 of the outer polyhedron: 4
Enter vertices for edge 1 in face 3 of the outer polyhedron (format: v1 v2): 1 5
Enter vertices for edge 2 in face 3 of the outer polyhedron (format: v1 v2): 5 8
Enter vertices for edge 3 in face 3 of the outer polyhedron (format: v1 v2): 8 4
Enter vertices for edge 4 in face 3 of the outer polyhedron (format: v1 v2): 4 1

Enter the number of edges in face 4 of the outer polyhedron: 4
Enter vertices for edge 1 in face 4 of the outer polyhedron (format: v1 v2): 2 6
Enter vertices for edge 2 in face 4 of the outer polyhedron (format: v1 v2): 6 7
Enter vertices for edge 3 in face 4 of the outer polyhedron (format: v1 v2): 7 3
Enter vertices for edge 4 in face 4 of the outer polyhedron (format: v1 v2): 3 2

Enter the number of edges in face 5 of the outer polyhedron: 4
Enter vertices for edge 1 in face 5 of the outer polyhedron (format: v1 v2): 1 2
Enter vertices for edge 2 in face 5 of the outer polyhedron (format: v1 v2): 2 6
Enter vertices for edge 3 in face 5 of the outer polyhedron (format: v1 v2): 6 5
Enter vertices for edge 4 in face 5 of the outer polyhedron (format: v1 v2): 5 1

Enter the number of edges in face 6 of the outer polyhedron: 4
Enter vertices for edge 1 in face 6 of the outer polyhedron (format: v1 v2): 3 4
```

```
Enter vertices for edge 2 in face 6 of the outer polyhedron (format: v1 v2): 4 8
Enter vertices for edge 3 in face 6 of the outer polyhedron (format: v1 v2): 8 7
Enter vertices for edge 4 in face 6 of the outer polyhedron (format: v1 v2): 7 3

Enter the number of internal hole polyhedrons within the outer polyhedron (0 if none): 1

Entering internal hole polyhedron 1 of the outer polyhedron
Enter the number of vertices in the internal hole 1 polyhedron: 8
Enter the number of faces in the internal hole 1 polyhedron: 6
Enter the 2D coordinates for the vertices in the internal hole 1 polyhedron:

// Inner cube vertices in 2D projections
Vertex 1 (xy): 0.5 0.5
Vertex 1 (xz): 0.5 0.5

Vertex 2 (xy): -0.5 0.5
Vertex 2 (xz): -0.5 0.5

Vertex 3 (xy): -0.5 -0.5
Vertex 3 (xz): -0.5 -0.5

Vertex 4 (xy): 0.5 -0.5
Vertex 4 (xz): 0.5 -0.5

Vertex 5 (xy): 0.5 0.5
Vertex 5 (xz): 0.5 -0.5

Vertex 6 (xy): -0.5 0.5
Vertex 6 (xz): -0.5 -0.5

Vertex 7 (xy): -0.5 -0.5
Vertex 7 (xz): -0.5 0.5

Vertex 8 (xy): 0.5 -0.5
Vertex 8 (xz): 0.5 0.5

// Define faces for the inner cube
Enter the number of edges in face 1 of the internal hole 1 polyhedron: 4
Enter vertices for edge 1 in face 1 of the internal hole 1 polyhedron (format: v1 v2): 1 2
Enter vertices for edge 2 in face 1 of the internal hole 1 polyhedron (format: v1 v2): 2 3
Enter vertices for edge 3 in face 1 of the internal hole 1 polyhedron (format: v1 v2): 3 4
Enter vertices for edge 4 in face 1 of the internal hole 1 polyhedron (format: v1 v2): 4 1

Enter the number of edges in face 2 of the internal hole 1 polyhedron: 4
Enter vertices for edge 1 in face 2 of the internal hole 1 polyhedron (format: v1 v2): 5 6
Enter vertices for edge 2 in face 2 of the internal hole 1 polyhedron (format: v1 v2): 6 7
```

```
Enter vertices for edge 3 in face 2 of the internal hole 1 polyhedron (format: v1 v2): 7 8
Enter vertices for edge 4 in face 2 of the internal hole 1 polyhedron (format: v1 v2): 8 5

Enter the number of edges in face 3 of the internal hole 1 polyhedron: 4
Enter vertices for edge 1 in face 3 of the internal hole 1 polyhedron (format: v1 v2): 1 5
Enter vertices for edge 2 in face 3 of the internal hole 1 polyhedron (format: v1 v2): 5 8
Enter vertices for edge 3 in face 3 of the internal hole 1 polyhedron (format: v1 v2): 8 4
Enter vertices for edge 4 in face 3 of the internal hole 1 polyhedron (format: v1 v2): 4 1

Enter the number of edges in face 4 of the internal hole 1 polyhedron: 4
Enter vertices for edge 1 in face 4 of the internal hole 1 polyhedron (format: v1 v2): 2 6
Enter vertices for edge 2 in face 4 of the internal hole 1 polyhedron (format: v1 v2): 6 7
Enter vertices for edge 3 in face 4 of the internal hole 1 polyhedron (format: v1 v2): 7 3
Enter vertices for edge 4 in face 4 of the internal hole 1 polyhedron (format: v1 v2): 3 2

Enter the number of edges in face 5 of the internal hole 1 polyhedron: 4
Enter vertices for edge 1 in face 5 of the internal hole 1 polyhedron (format: v1 v2): 1 2
Enter vertices for edge 2 in face 5 of the internal hole 1 polyhedron (format: v1 v2): 2 6
Enter vertices for edge 3 in face 5 of the internal hole 1 polyhedron (format: v1 v2): 6 5
Enter vertices for edge 4 in face 5 of the internal hole 1 polyhedron (format: v1 v2): 5 1

Enter the number of edges in face 6 of the internal hole 1 polyhedron: 4
Enter vertices for edge 1 in face 6 of the internal hole 1 polyhedron (format: v1 v2): 3 4
Enter vertices for edge 2 in face 6 of the internal hole 1 polyhedron (format: v1 v2): 4 8
Enter vertices for edge 3 in face 6 of the internal hole 1 polyhedron (format: v1 v2): 8 7
Enter vertices for edge 4 in face 6 of the internal hole 1 polyhedron (format: v1 v2): 7 3
```

Command Line Interface

```
Select a task to perform:
1. Calculate Surface Area
2. Calculate Volume
3. Calculate Center of Mass
4. Transform Polyhedron
6. Isometric View
7. Orthographic Projection onto Custom Plane
8. Calculate Moment of Inertia
9. Exit
Enter your choice:
```

10 Limitations and Scope

10.1 Limitations

The system has the following limitations and constraints:

- **Input format**: The system requires 2D projections on XY and XZ planes as input. It cannot handle other types of input such as 3D scans or single 2D projections.
- **Convex polyhedra**: The current implementation assumes convex polyhedra. Concave polyhedra may produce incorrect results for some calculations (e.g., volume).
- **Precision**: Floating-point arithmetic is used, which may lead to small inaccuracies in calculations due to rounding errors.
- **Performance**: The system is not optimized for very large polyhedra with thousands of vertices or faces. Performance may degrade for complex shapes.
- **Visualization**: The orthographic projection visualization is basic and does not include features like hidden line removal or shading.
- **File I/O**: The system cannot export results, and it does not currently support importing polyhedra from standard 3D file formats (e.g., OBJ, STL).
- **User interface**: The system uses a command-line interface, which may be less intuitive for users accustomed to graphical interfaces.
- **Error handling**: While basic input validation is implemented, the system may not gracefully handle all possible error cases or invalid inputs.
- **Coordinate system**: The system assumes a right-handed coordinate system. It may not correctly handle or convert between different coordinate systems.
- **Units**: The system does not explicitly handle units. Users must ensure consistent units are used for input and interpret output accordingly.
- **Topology changes**: The system does not support operations that would change the topology of the polyhedron (e.g., boolean operations, subdivision).

These limitations should be considered when using the system in an engineering drawing context. For many basic polyhedra and standard operations, the system should provide accurate and useful results. However, for more complex scenarios or specialized requirements, additional development or integration with more advanced 3D modeling software may be necessary.