Fast Hexahedral Mesh Extraction from Locally Injective Integer-Grid Maps

Tobias Kohler

October 5, 2023
Motivation

Tet-Mesh

- easier to generate

Hex-Mesh

- nicer numerical features
**Motivation**

Tet-Mesh  

- easier to generate

Hex-Mesh  

- nicer numerical features
Integer-Grid Map (IGM)

Image Source: (Lyon et al. 2016)

**Remark**

parametrization $f$ is per cell and transition $\tau$ is per half-face.
Robust HexEx (Lyon et al. 2016)

- Extracts a hex-mesh from a given tet-mesh and an IGM.
  1. Preprocessing - extract $\tau$ from $f$ and resolve floating-point inaccuracies.
Robust HexEx (Lyon et al. 2016)

- Extracts a hex-mesh from a given tet-mesh and an IGM.
  1. Preprocessing - extract $\tau$ from $f$ and resolve floating-point inaccuracies.
  2. Geometry Extraction - extract a hex-vertex for each intersection of the parametrization with the integer-grid.
Robust HexEx (Lyon et al. 2016)

- Extracts a hex-mesh from a given tet-mesh and an IGM.
  - 1. Preprocessing - extract $\tau$ from $f$ and resolve floating-point inaccuracies.
  - 2. Geometry Extraction - extract a hex-vertex for each intersection of the parametrization with the integer-grid.
  - 3. Topology Extraction - enumerate darts and trace their connections through the parametrization.
Robust HexEx (Lyon et al. 2016)

- Extracts a hex-mesh from a given tet-mesh and an IGM.
  1. Preprocessing - extract $\tau$ from $f$ and resolve floating-point inaccuracies.
  2. Geometry Extraction - extract a hex-vertex for each intersection of the parametrization with the integer-grid.
  3. Topology Extraction - enumerate darts and trace their connections through the parametrization.
  4. Postprocessing - resolve problems due to flipped or degenerate cells.
Robust HexEx (Lyon et al. 2016)

Darts (Kraemer et al. 2014)

\[ 6 \cdot 4 \cdot 2 \cdot |C| = 48 |C| \text{ darts} \]
\[ 4 \cdot 48 |C| = 192 |C| \text{ connections} \]
Degenerate Cells

No local injectivity!
Robust HexEx - Room for Improvements

- General: Degenerate and flipped tets
Robust HexEx - Room for Improvements

- General: Degenerate and flipped tets
- Geometry Extraction: All points in the bounding box are tested
Robust HexEx - Room for Improvements

- General: Degenerate and flipped tets
- Geometry Extraction: All points in the bounding box are tested
- Topology Extraction: Inefficient data structure
Robust HexEx - Room for Improvements

- General: Degenerate and flipped tets
- Geometry Extraction: All points in the bounding box are tested
- Topology Extraction: Inefficient data structure
- General: Potentially long lists are searched in linearly
From Robust HexEx to Fast HexEx

Goal: Make it faster!
Fast HexEx (Ours)

Extracts a hex-mesh from a given tet-mesh and an IGM. Local injectivity is required.

1. Preprocessing - unchanged
2. Geometry Extraction - using rasterization
3. Topology Extraction - with propellers
4. Postprocessing
Exact Predicates (Shewchuk 1996)

All geometric tests like ONLINESEGMENT, INTRIANGLE, ... build on ORI2D and ORI3D.
Hex-Vertex Extraction
Hex-Vertex Extraction

Duplicates
Hex-Vertex Extraction

**Algorithm**

Vertex Extraction on Edges (High-Level)

1: for $e \in E$ do
2:     pick any cell $c \in C$ s.t. $e \sim c$
3:     $Z \leftarrow F_c(e) \cap \mathbb{Z}^3$
4:     if $Z \neq \emptyset$ then
5:         $G \leftarrow G \cup \{e\}$
6:     for $z \in Z$ do
7:         generate hex-vertex with generator $e$, geometric embedding $f_c^{-1}(z)$
8:     and parameter $z$ in the chart of $c$

Goal: Find $F_c(x) \cap \mathbb{Z}^3$ efficiently.
The Easy Case - Vertices

\[ \{(x, y, z)\} \cap \mathbb{Z}^3 \]
Hex-Vertex Extraction - Candidates

- Which points to test against the exact predicates?
Hex-Vertex Extraction - Candidates

- All of $\mathbb{Z}^3 \Rightarrow$ infeasible
Hex-Vertex Extraction - Candidates

- All of $\mathbb{Z}^3 \Rightarrow$ infeasible
- Bounding box $\Rightarrow$ inefficient
Hex-Vertex Extraction - Candidates

- All of $\mathbb{Z}^3 \Rightarrow$ infeasible
- Bounding box $\Rightarrow$ inefficient
- Rasterize element $\Rightarrow$ keep exact predicate calls to a minimum
Rasterization - Top-Down
For simplicity and efficiency:

\[ S_x \geq S_y \geq S_z \]

Rasterization axis \( r := \arg \min_i (S_i > 0) \)
Rasterizing Cells/Tetrahedra

$$A_z \geq B_z \geq C_z \geq D_z$$
Rasterizing Faces/Triangles

\[ A_r \geq B_r \geq C_r \]
Rasterizing Quads

\[ A_r, B_r, C_r \geq D_r \Rightarrow 3 \text{ cases} \]
Rasterizing Edges/Line Segments

Trivial case: $S_y = S_z = 0$
Floating-Point Inaccuracies
Hex-Vertex Extraction

- Result: List of generators, each with list of isolated hex-vertices
Hex-Vertex Extraction

- Result: List of generators, each with list of isolated hex-vertices

- Missing: Connectivity (hex-edges, -faces, -cells)
Propellers

$2|E|$ propellers
$2|E| + 8|F|$ connections
Local Topology per Generator
Local Topology in Cells
Local Topology - Propeller Roots
Local Topology - Rotating from Roots to Blades
Regular Hex-Edges

Valence 4 inner regular edge
Singular Hex-Edges

Valence 3 inner singularity edge

Valence 5 inner singularity edge
Tracing

“Connecting the dots”
The Final Step - Hex-Cells

3 propellers make up a hex-corner, 8 hex-corners define a hex-cell.
The Algorithm

1: preprocessing()
2: for $x \in V \cup E \cup F \cup C$ do
3:     extractHexVertices($x$)
4: for $g \in G \setminus C$ do
5:     enumeratePropellerRoots($g$)
6: for $p \in \mathcal{P}_g$ do
7:     connectPropellerBlades($p$)
8: for $v_h \in V_h$ do
9:     $g \leftarrow$ generator of $v_h$
10: for $p \in \mathcal{P}_g$ do
11:     connectPropellerOpposite($v_h$, $p$)
12: extractHexCells()
Improvements to Robust HexEx

- Rasterization instead of bounding box check
Improvements to Robust HexEx

- Rasterization instead of bounding box check
- Propellers instead of darts
Improvements to Robust HexEx

- Rasterization instead of bounding box check
- Propellers instead of darts
- Local topology is stored per generator instead of per hex-vertex
Improvements to Robust HexEx

- Rasterization instead of bounding box check
- Propellers instead of darts
- Local topology is stored per generator instead of per hex-vertex
- Hash-maps for potentially large collections instead of lists
HexMe dataset (Beaufort et al. 2022)
A Coarse Example

\[ \frac{H}{T} = 2\% \]

\[ \text{Fast} \quad = 59\% \]

\[ \text{Robust} \]
A Fine Example

\[
\frac{H}{T} = 1175\%
\]

Fast \quad Robust = 3\%

Fast Hexahedral Mesh Extraction from Locally Injective Integer-Grid Maps

Tobias Kohler
Timings

Fast Hexahedral Mesh Extraction from Locally Injectable Integer-Grid Maps

Tobias Kohler
Scaling Comparison

Robust HexEx

Fast HexEx

Time [s]

Preprocessing
Vertex Extraction
Local Connections
Tracing Connections
Postprocessing
Cell Extraction

Parametrization Scaling Factor

Preprocessing
Vertex Extraction
Local Connections
Tracing Connections
Postprocessing
Cell Extraction

Parametrization Scaling Factor

Fast Hexahedral Mesh Extraction from Locally Injective Integer-Grid Maps

Tobias Kohler
Potential Improvements

Operate on entire blocks of cells.
Potential Improvements

Parallelization (Vertex Extraction, Local Topology, ...)

Thank You

Questions?
References


Lyon, Max, David Bommes, and Leif Kobbelt (July 2016). “HexEx: Robust Hexahedral Mesh Extraction”. In: ACM Trans. Graph. 35.4. ISSN: 0730-0301. DOI: 10.1145/2897824.2925976. URL: https://doi.org/10.1145/2897824.2925976.