Outer surface extraction for complex 3D building models

Yifang Zhao

Mentor #1: Liangliang Nan
Mentor #2: Hugo Ledoux
Mentor #3: Ravi Peters

TU Delft

Outline

- Introduction
- Related works
- Methodology
- Results & discussion
- Conclusions & future works

Outline

- Introduction
- Related works
- Methodology
- Results & discussion
- Conclusions & future works

Motivation

- 3D building models frequently used in urban applications

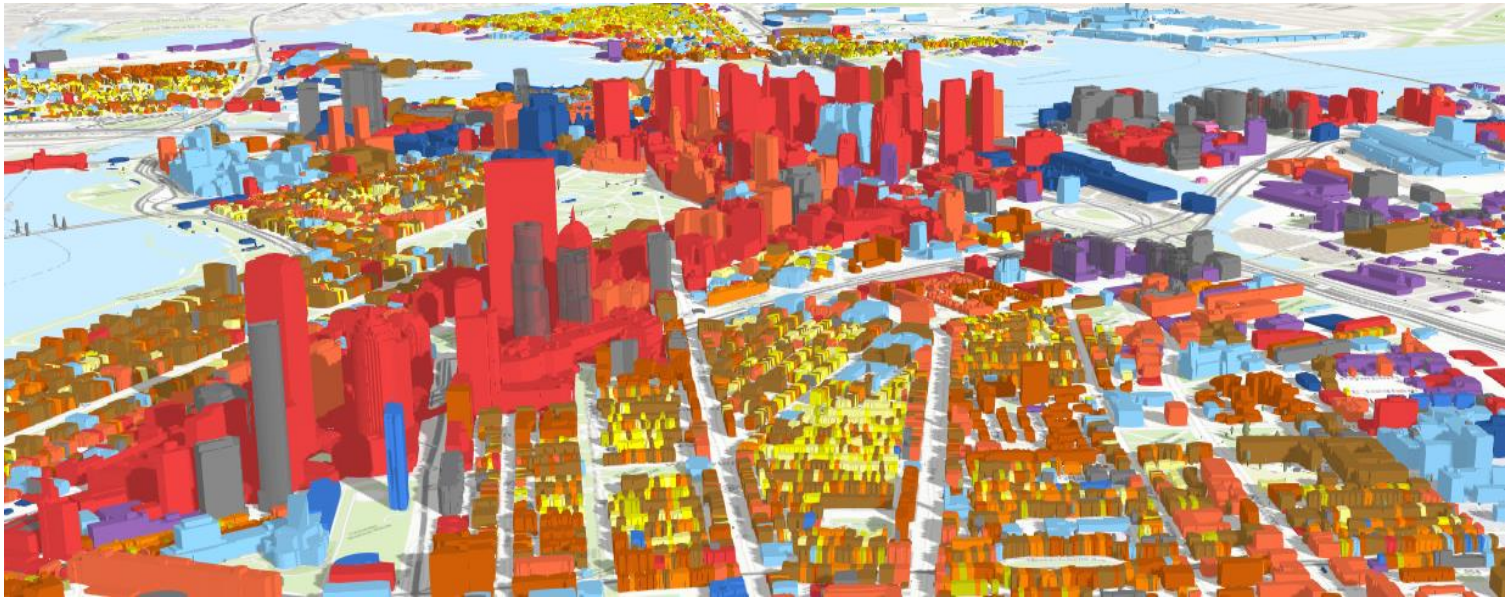


Figure 1. “A 3D Model of Boston for Better Urban Planning”. Retrieved May 19, 2020, from https://modelur.eu/a-3d-model-of-boston-available-for-public-use/

Motivation

- 3D building models frequently used in urban applications

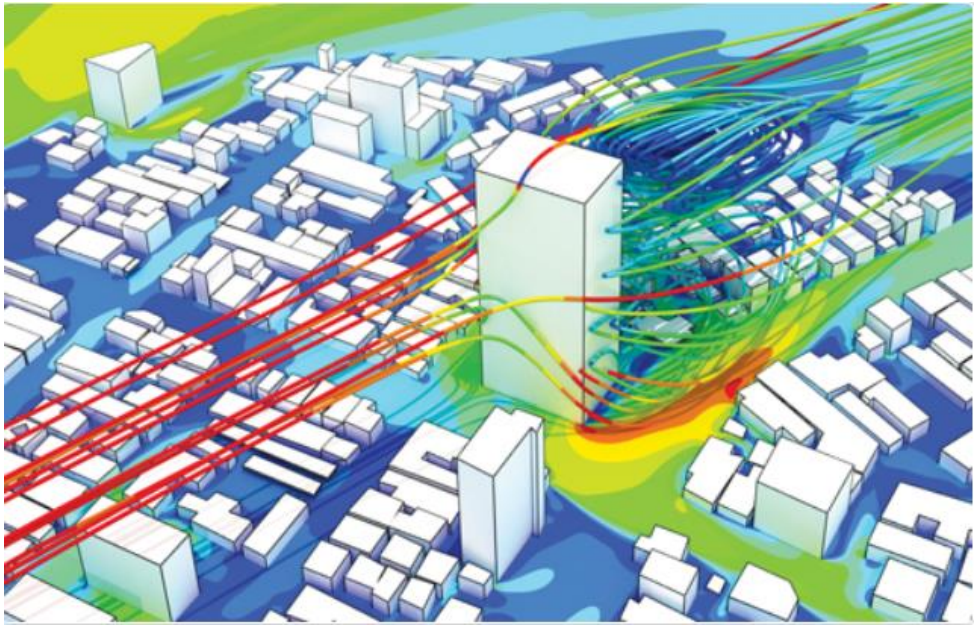


Figure 2. Wind simulation. Retrieved January 14, 2020, from https://www.simscale.com/

Motivation

- 3D building models frequently used in urban applications

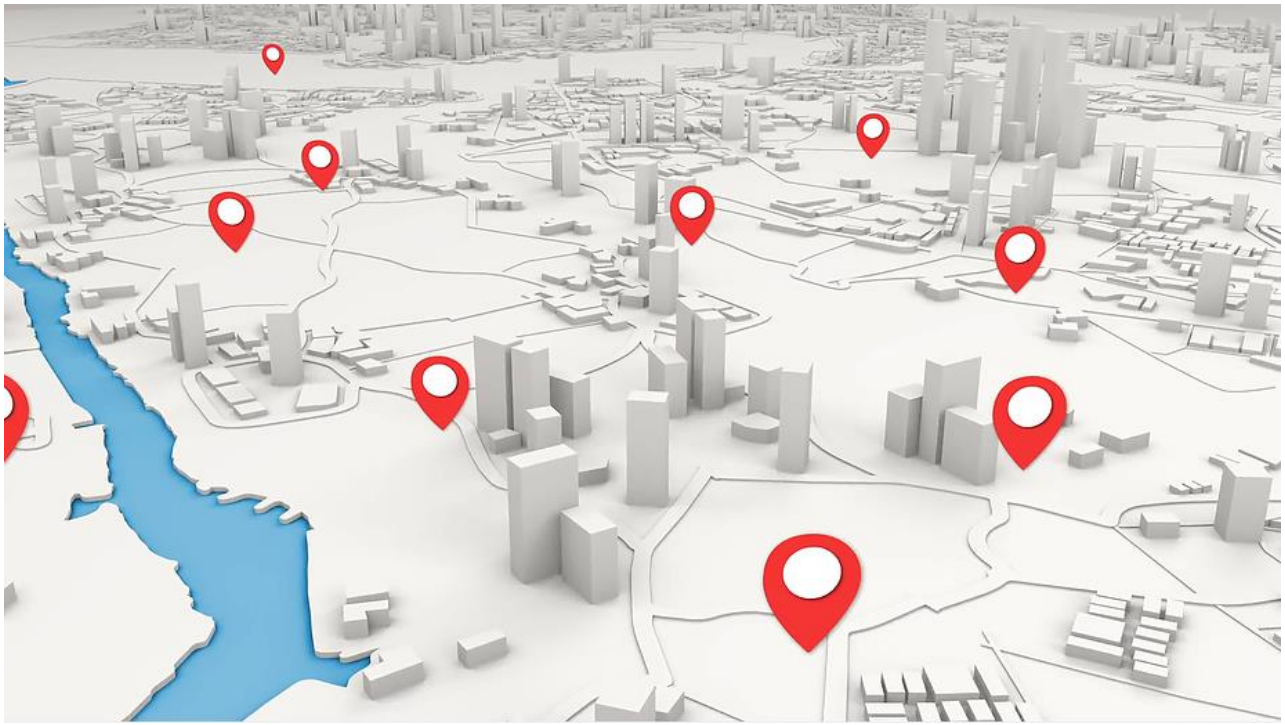


Figure 3. Urban navigation. Retrieved June 30, 2020, from https://www.treistek.com/post/3d-city-modeling-in-navigational-applications

Motivation

- 3D building models frequently used in urban applications
- The input required to be closed and manifold

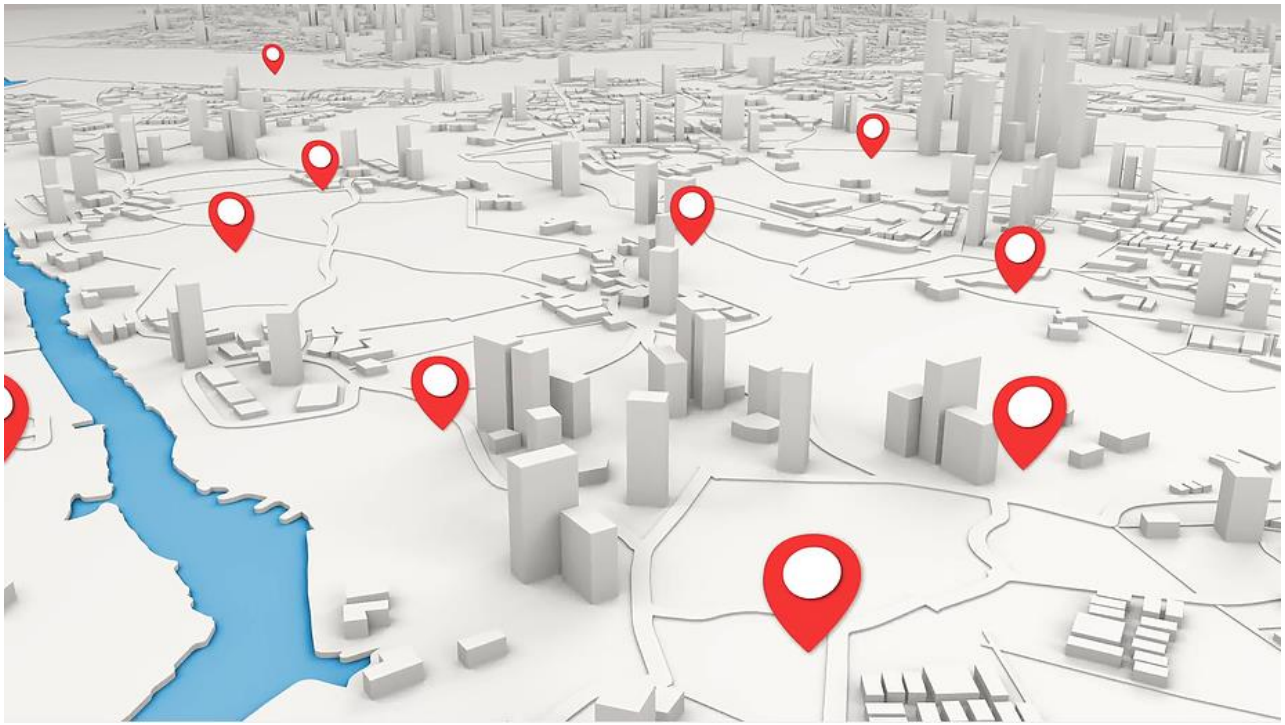


Figure 3. Urban navigation. Retrieved June 30, 2020, from https://www.treistek.com/post/3d-city-modeling-in-navigational-applications

Motivation

- 3D building models frequently used in urban applications
- The input required to be closed and manifold
- Visually plausible but defective

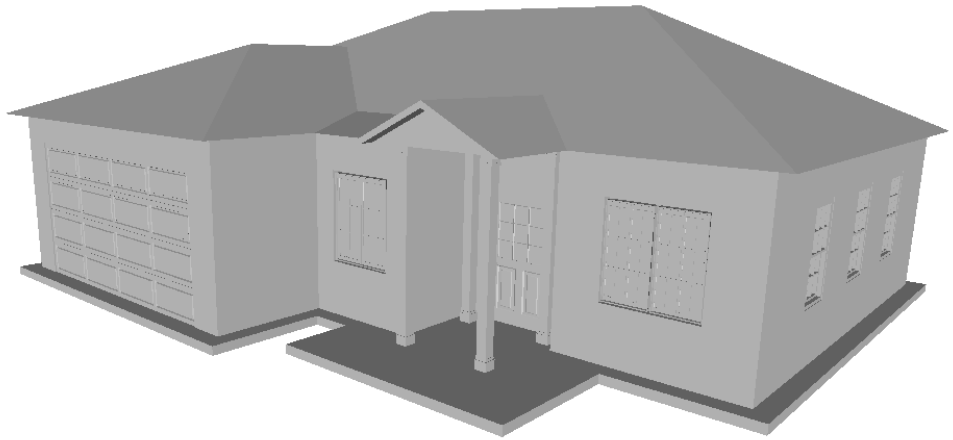


Figure 4. Artifacts of a building model from *Fan and Wonka (2016)* . Green: boundary edges. Red: self-intersecting faces. Rendered in MeshLab.

Motivation

- 3D building models frequently used in urban applications
- The input required to be closed and manifold
- Visually plausible but defective

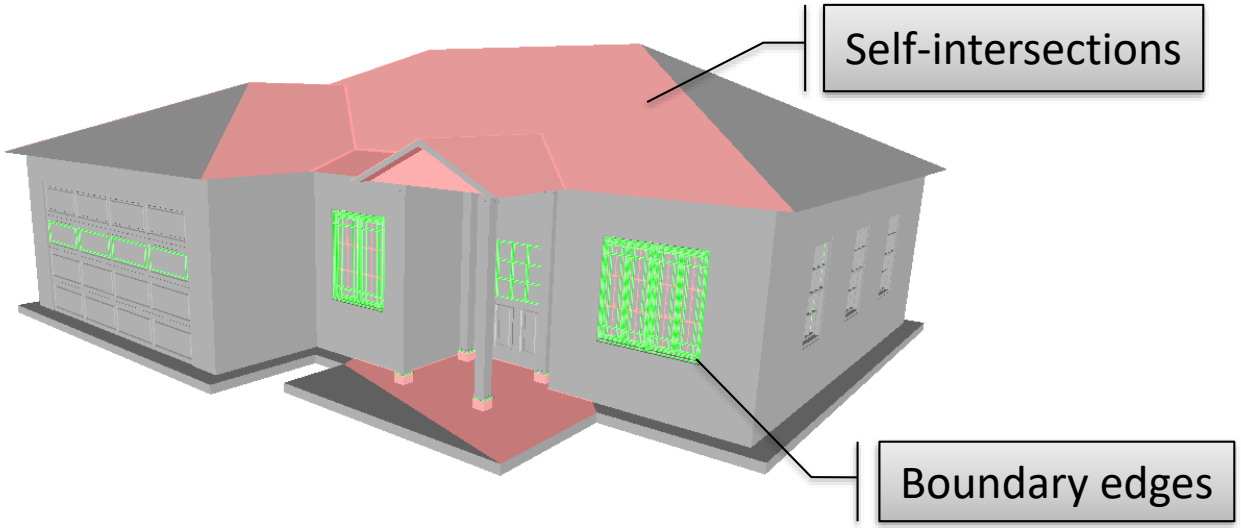


Figure 4. Artifacts of a building model from *Fan and Wonka (2016)* . Green: boundary edges. Red: self-intersecting faces. Rendered in MeshLab.

Motivation

- 3D building models frequently used in urban applications
- The input required to be closed and manifold
- Visually plausible but defective

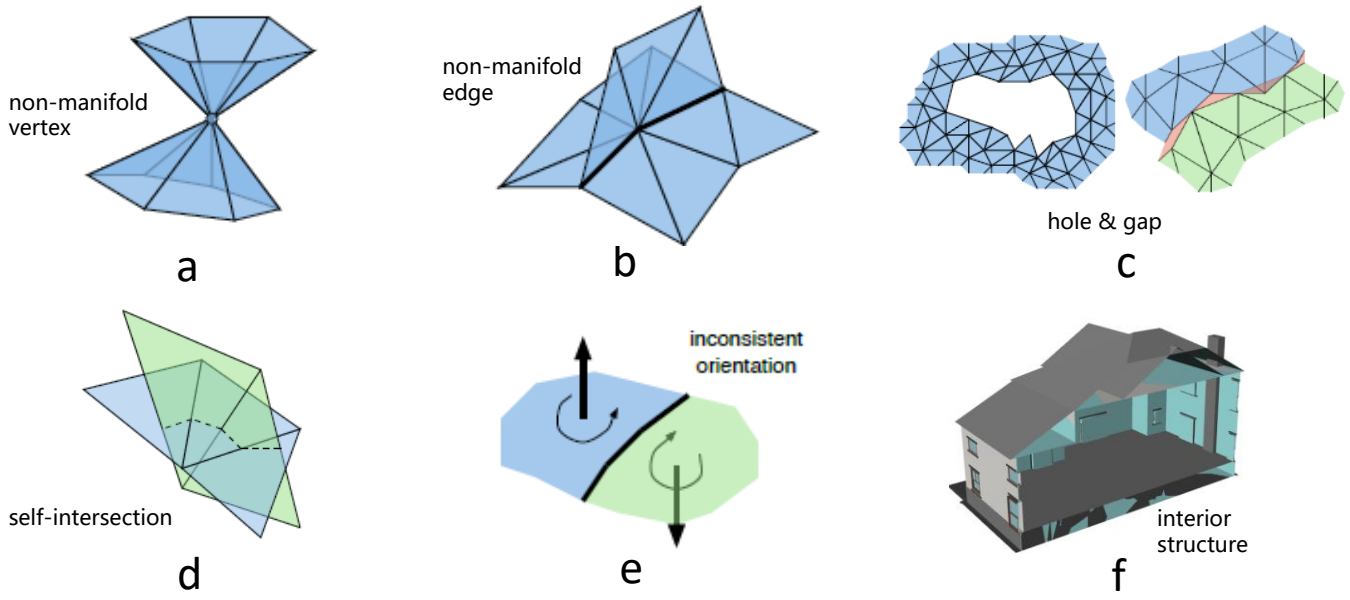


Figure 5. (a)-(e): Artifact chart. Adapted from *Geometric Modeling Based on Polygonal Meshes,* by M. Botsch et al., 2007. (f): Building model from *Fan and Wonka (2016)* . Rendered in *Mapple*.

Research question

*How can we accurately extract a **watertight** and **manifold** outer surface from an error-ridden 3D building model?*

Outline

- Introduction
- Related works
- Methodology
- Results & discussion
- Conclusions & future works

Related works

Two main categories exist w.r.t outer surface extraction:

- **Surface-oriented method**
  - Local.
  - Direct.
  - Manual.
  - One-to-one.
- **Volumetric method**
  - Global.
  - Indirect.
  - Automatic.
  - All-in-one.
  - Conditioned.

Surface-oriented method

Consistent Normal Orientation for Polygonal Meshes

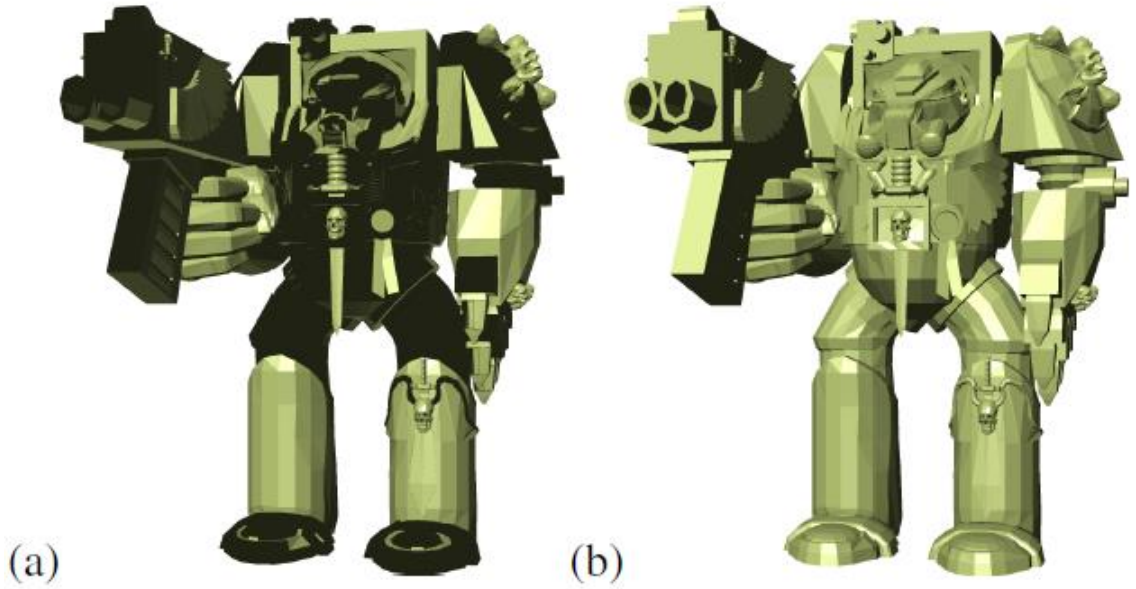


Figure 6. "(a) A model consisting of 2398 separate surface patches with inconsistent orientation; due to the lighting, the faces oriented 'backwards' are rendered dark. (b) Same model after applying our algorithm (with exactly the same lighting)". Reprinted from *Consistent Normal Orientation for Polygonal Meshes*, by Borodin, P. et al., 2004.

Surface-oriented method

Filling Holes in Meshes

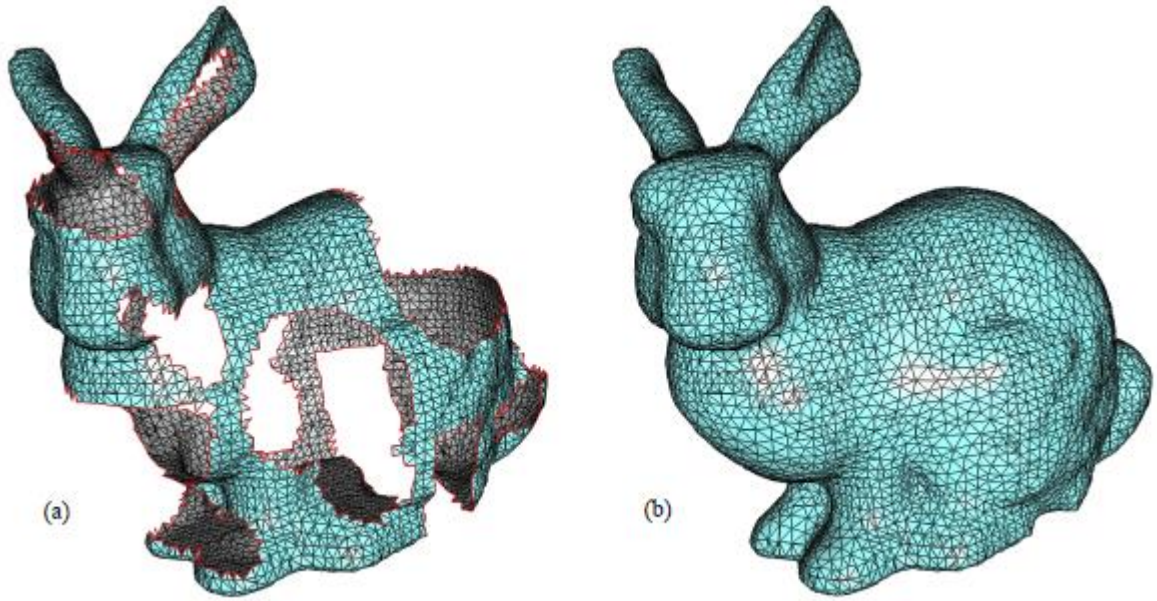


Figure 7. “(a) Mutilated Stanford bunny. (b) After hole triangulation, meshing and fairing”. Reprinted from *Filling Holes in Meshes*, by Liepa, P., 2003.

Related works

Two main categories exist w.r.t model repair:

- **Surface-oriented method**
  - Local.
  - Direct.
  - Manual.
  - One-to-one.
- **Volumetric method**
  - Global.
  - Indirect.
  - Automatic.
  - All-in-one.
  - Conditioned.

Volumetric methods

Consistent Solid and Boundary Representations from Arbitrary Polygonal Data

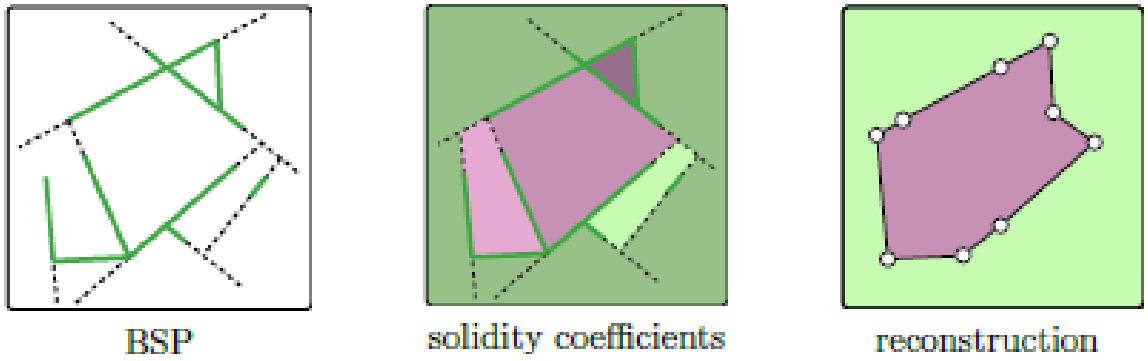


Figure 8. Illustration of Murali's Method. Adapted from *Geometric Modeling Based on Polygonal Meshes,* by M. Botsch et al., 2007.

Volumetric methods

Tetrahedral Meshing in the Wild

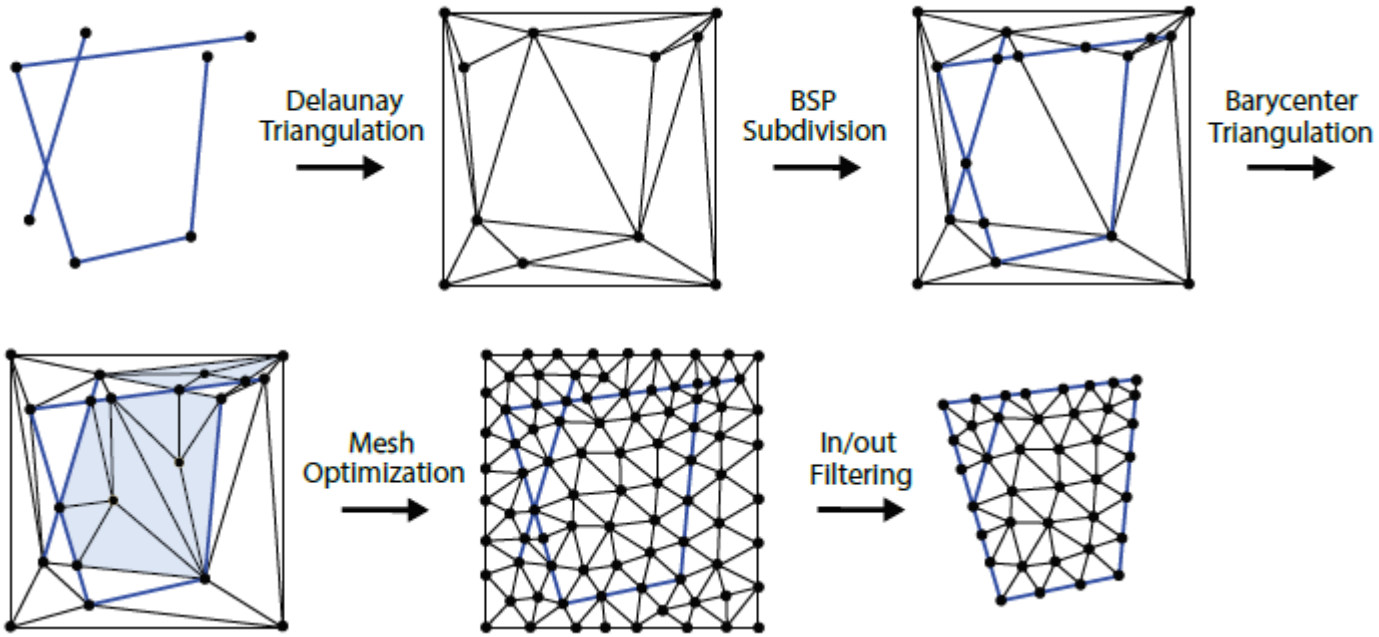
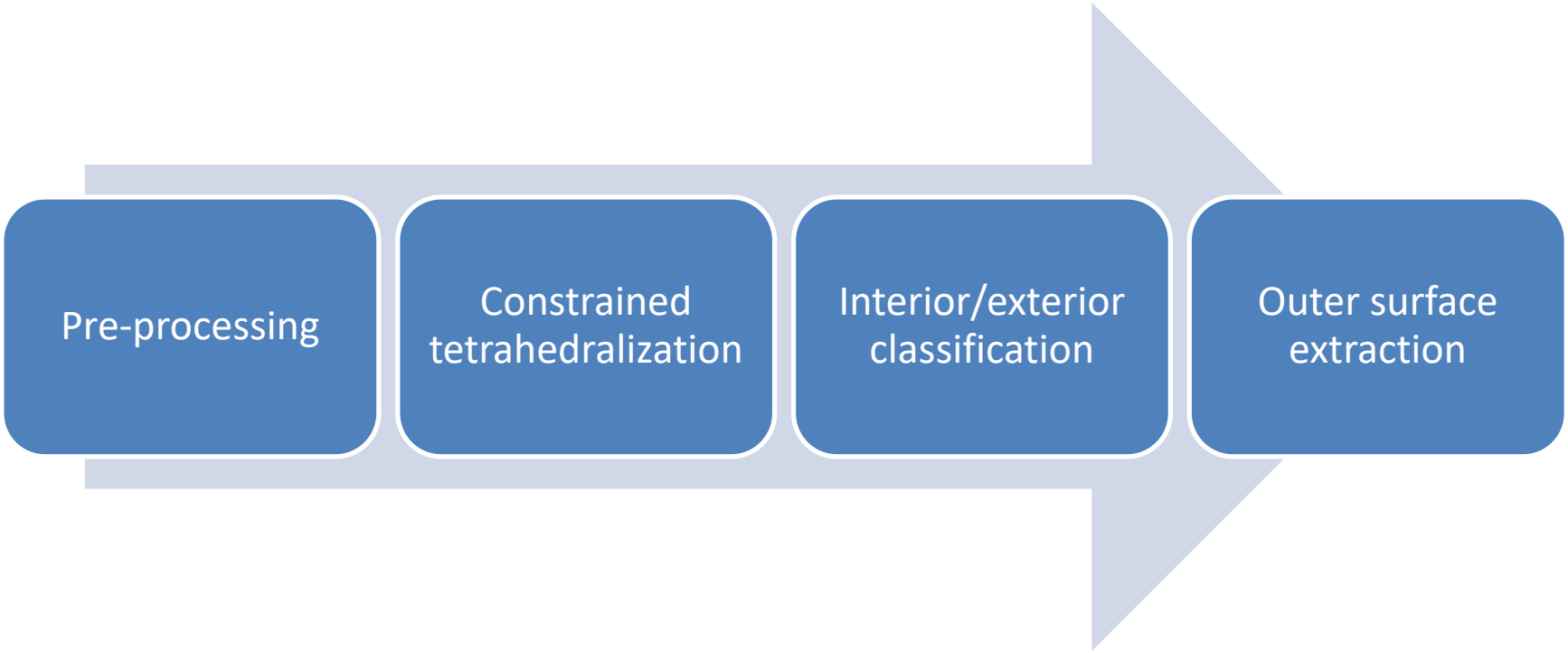


Figure 9. A diagram illustrating the pipeline of TetWild in 2D. Reprinted from *Tetrahedral Meshing in the Wild,* by Hu, Y. et al., 2018.

Outline

- Introduction
- Related works
- Methodology
- Results & discussion
- Conclusions & future works

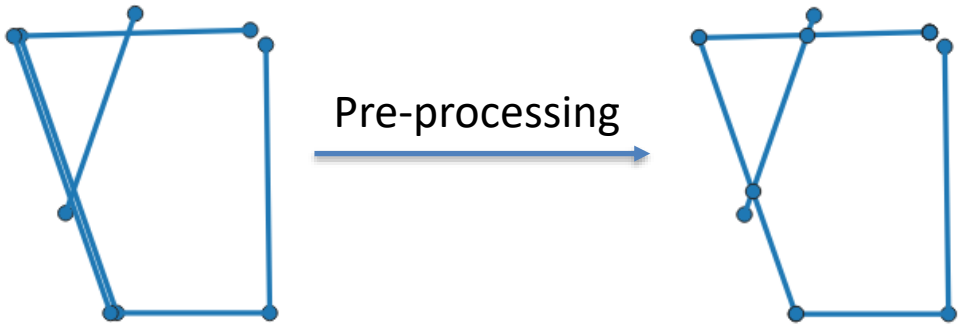
Pipeline


Pre-processing
Constrained tetrahedralization
Interior/exterior classification
Outer surface extraction

1. Pre-processing

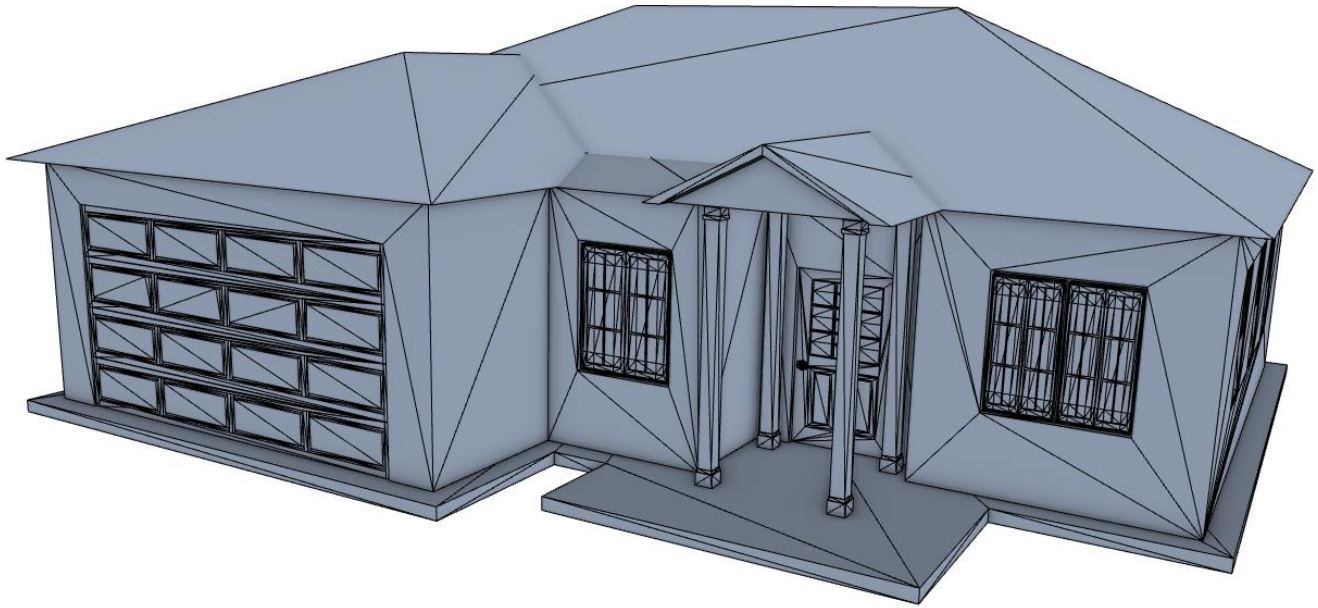
Remove duplications and self-intersections

- Input: Original model
- Method: Duplication removal + Remeshing[1]
- Output: Pre-processed model



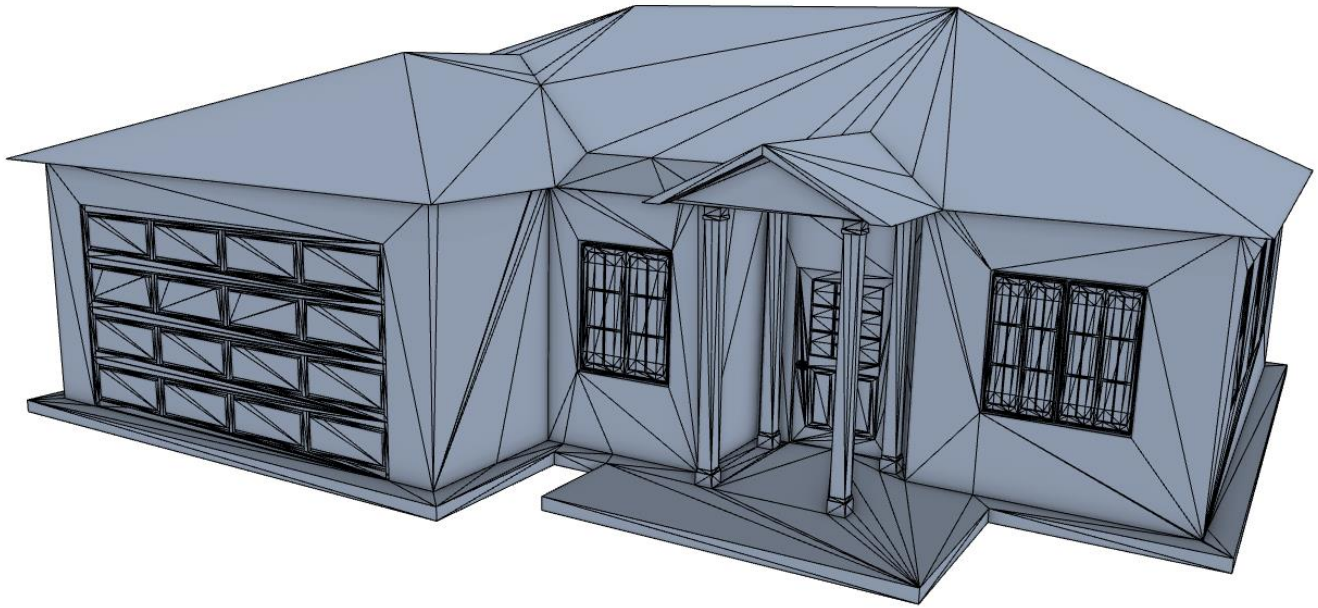
[1] Code migrated from Easy3D.

1. Pre-processing



Original model

1. Pre-processing

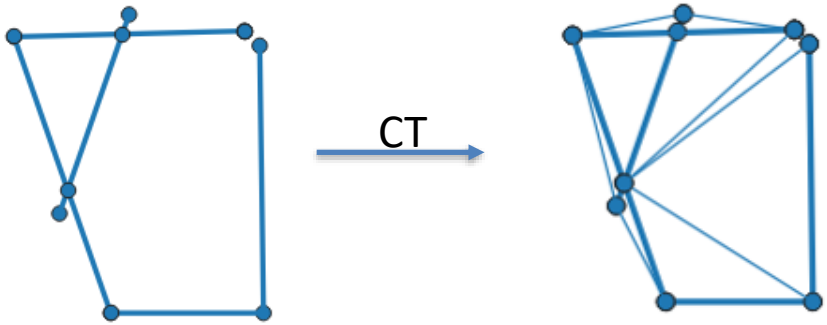


Pre-processed model

2. Constrained tetrahedralization (CT)

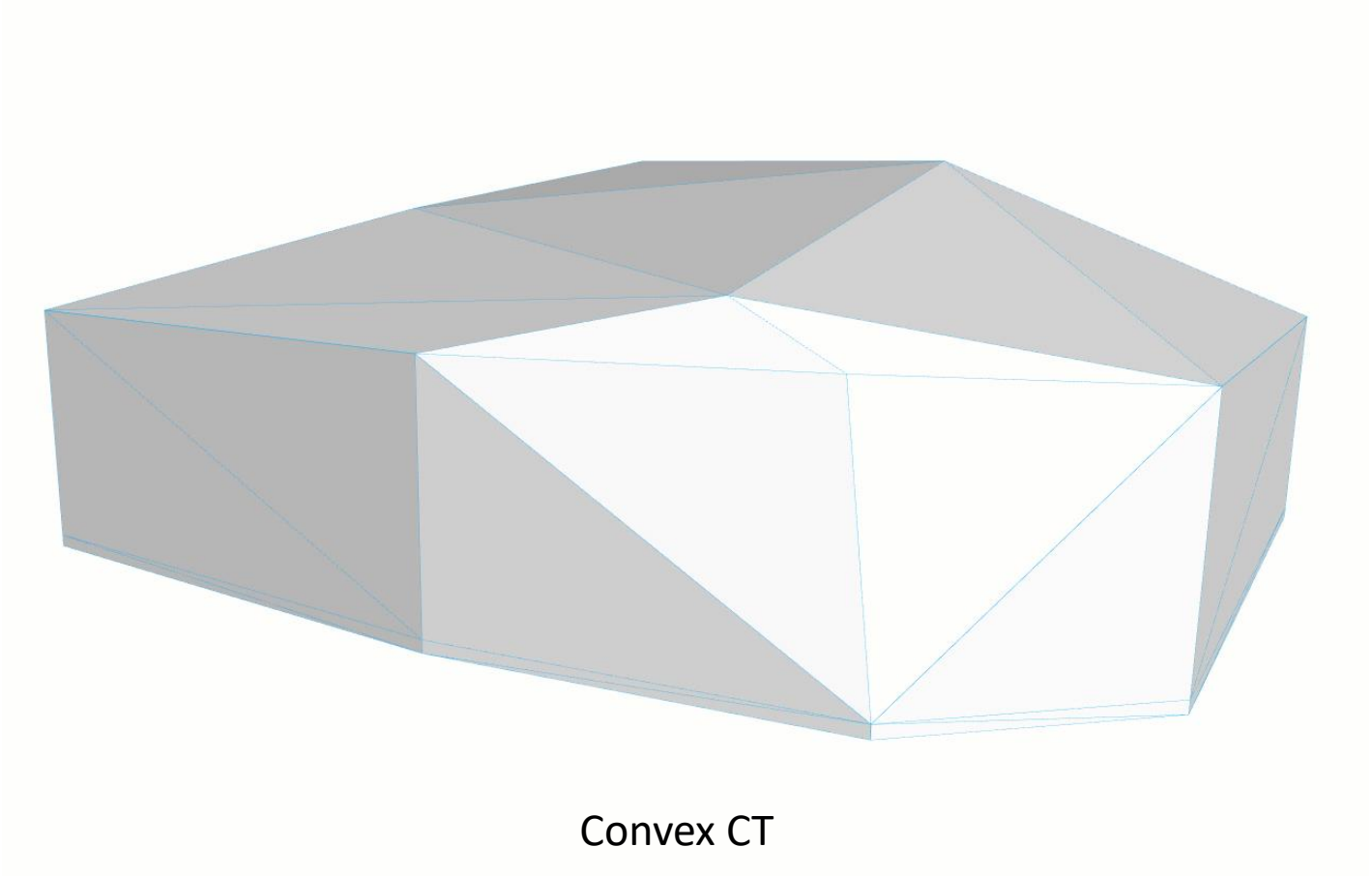
Tetrahedralize the model domain

- Input: Pre-processed model
- Method: Constrained tetrahedralization[1]
- Output: Convex CT



[1] Code migrated from TetGen.

2. Constrained tetrahedralization (CT)

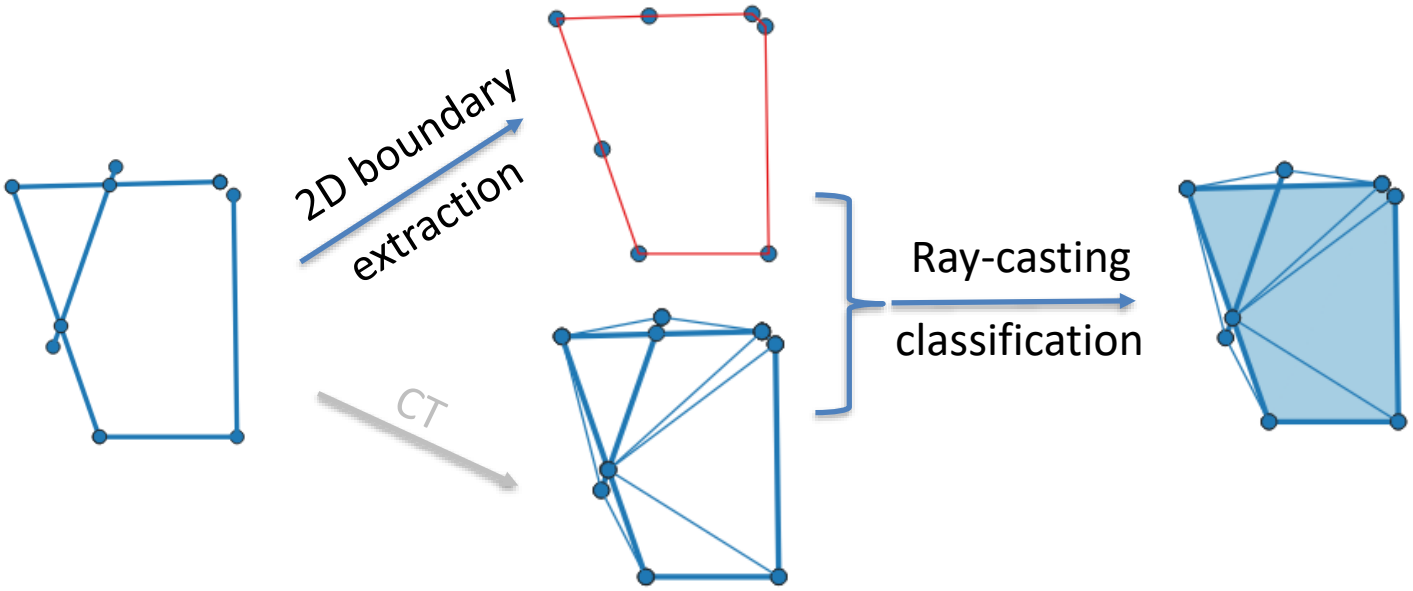


Convex CT

3. Interior/exterior classification

Classify tetrahedra as being interior or exterior

- Input: Convex CT
- Method: Ray-casting classification based on extracted 2D boundary[1]
- Output: Classified CT



[1] Builds on the method proposed by N. Tzounakos in "Robust Interior-Exterior Classification for 3D Models".

3.1 Robust classification based on ray-casting

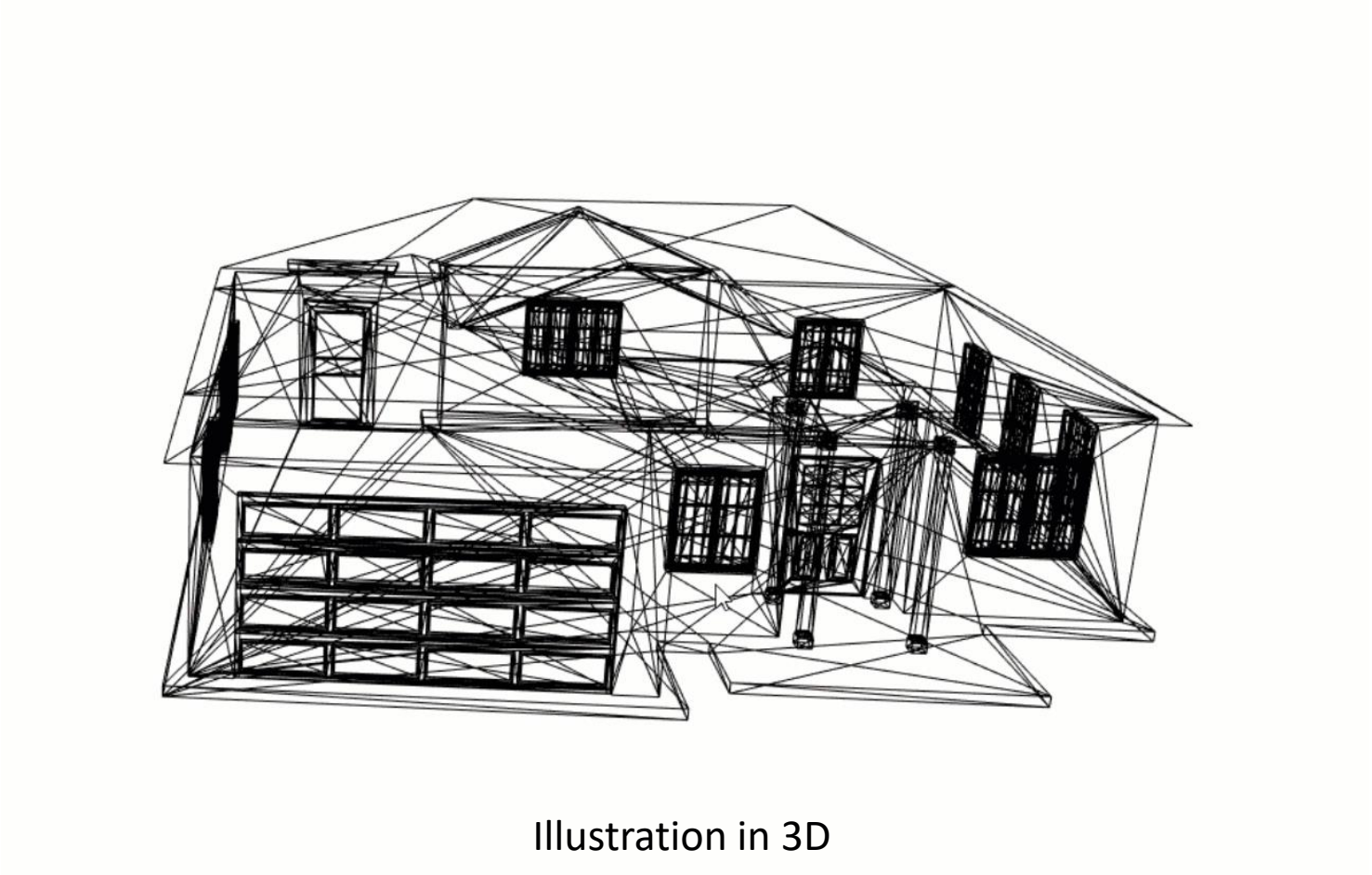
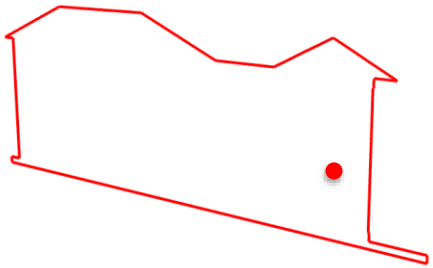
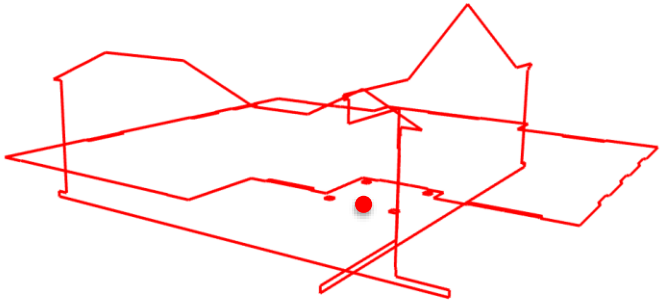


Illustration in 3D

3.1 Robust classification based on ray-casting



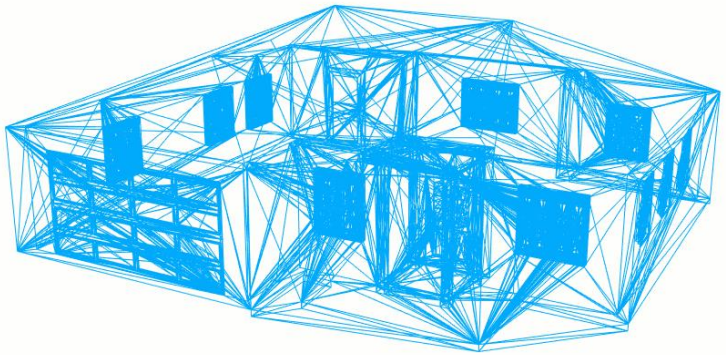
One cross-section doesn't work



Three cross-sections

A point is classified as being interior only if it is so for all three cross-sections.

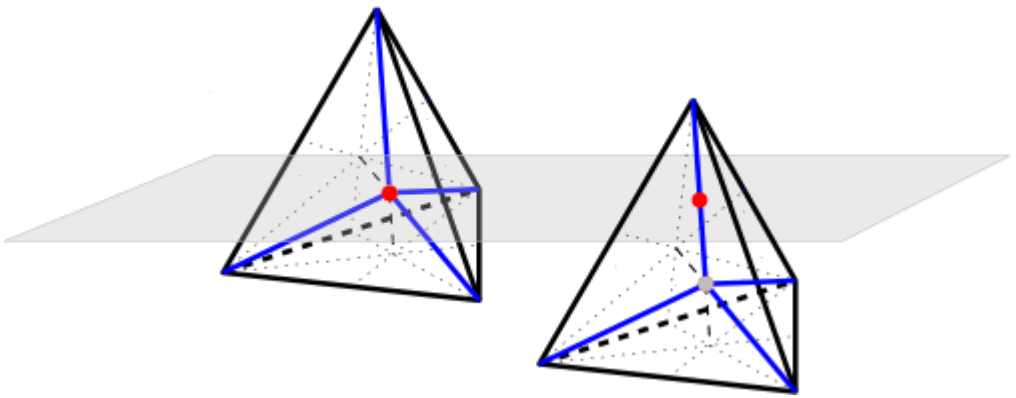
3.2 Grouping of tetrahedra



- Based on spatial connectivity
- One special group consists of tetrahedra connecting to the outer space (see left)
- Other groups are protected by the model

Therefore, we only have to perform classification for this group. In this way, we've greatly reduced the number of tetrahedra that need to be classified.

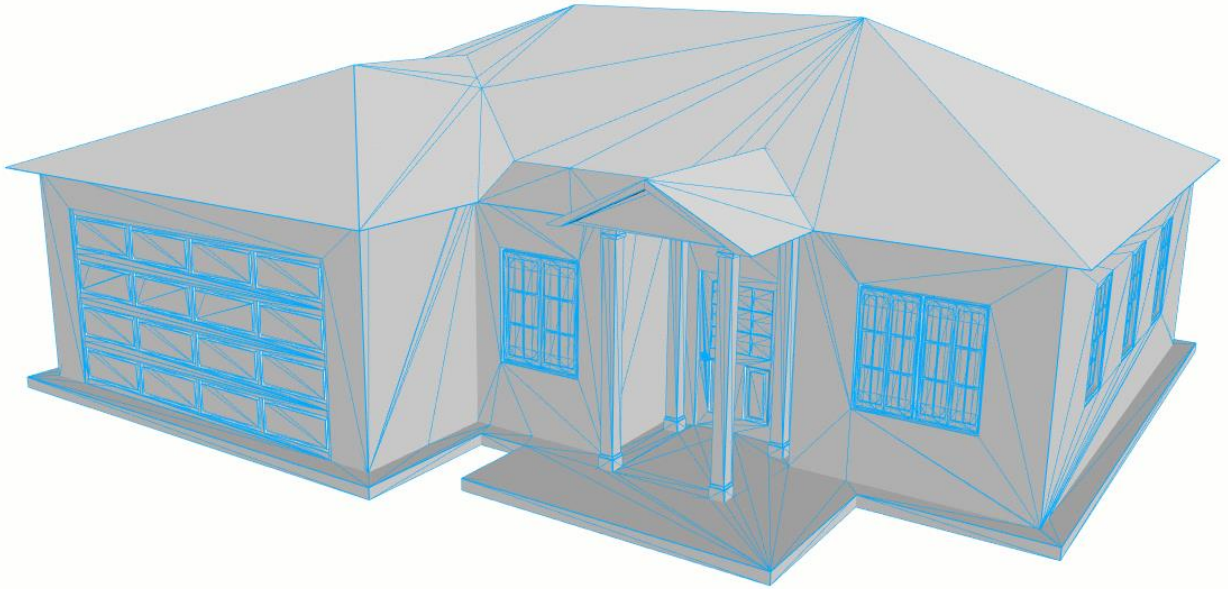
3.3 Reuse of 2D boundaries



- Most time-consuming part during classification
- Store the pre-constructed 2D boundaries
- Reuse if applicable

Initially, the cross-section passes through the centroid of a tetrahedron, and this centroid represents the tetrahedron during classification.
In cases of reusing, a new representative point (rendered red) must be constructed.

3.4 Classified CT

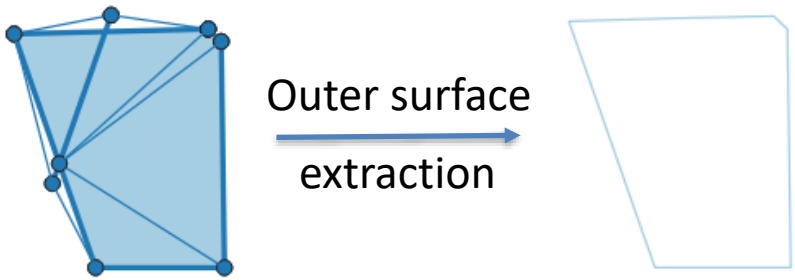


Classified CT

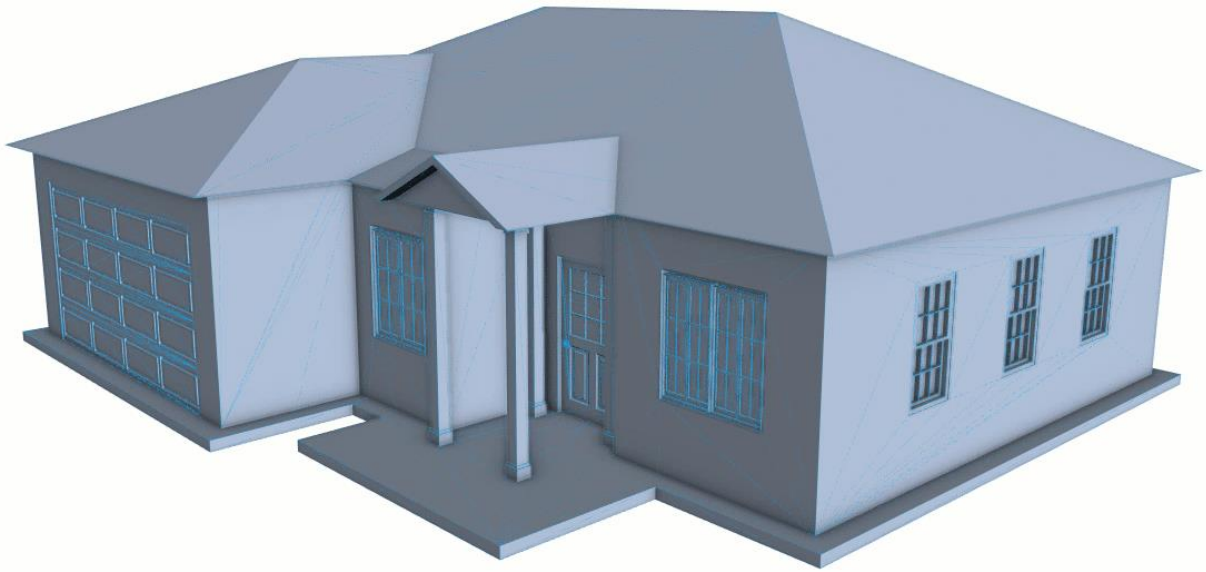
4. Outer surface extraction

Extract triangles incident to interior and exterior tetrahedra

- Input: Classified CT
- Method: Non-manifoldness removal
- Output: Watertight and manifold triangle mesh



4. Outer surface extraction



Outer surface

Outline

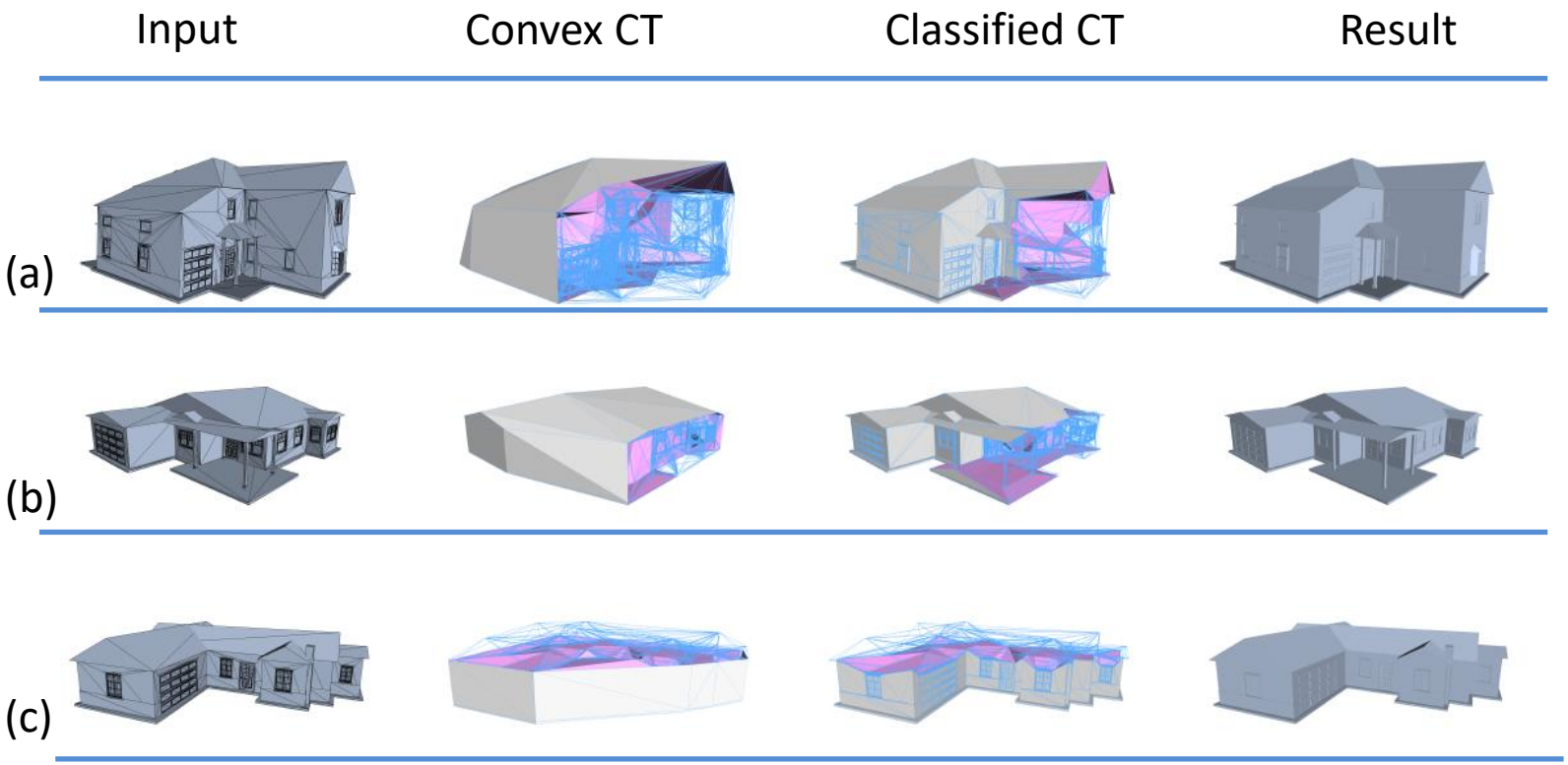
- Introduction
- Related works
- Methodology
- Results & discussion
- Conclusions & future works

Testing models

	f	v	degeneracy	duplication	self-int	NM e	NM v	BD e
task_1_x	11262	7970	607	908	807	1405	2	845
task_2_x	13386	8698	770	1710	638	2071	1	779
task_4_x	28759	28002	355	12990	6086	1148	141	20100
task-1-x	16724	27188	426	90	1745	342	64	9665
task-2-x	9048	7520	283	672	90	514	4	409
task-7-x	7120	6300	209	52	421	241	34	394
task-8-x	15759	10394	867	2034	259	2462	1	918
task-11-x	5751	4856	218	236	170	343	13	382
task-12-x	13058	15198	228	98	2170	394	44	3862
task-16-x	6640	4882	273	194	37	495	0	264
task-17-x2	12730	14918	647	36	2194	172	4	3291
task-19-x	6435	5950	316	184	137	272	25	793
task-20-x	6721	5834	256	96	114	227	31	425
task-21-x	9267	7307	351	714	251	665	1	508

Table 1: Statistics of input errors. f: number of triangles. v: number of vertices. degeneracy: number of denegerate triangles. duplication: number of duplicated triangles. self-int: number of self-intersecting triangles. NM e: number of non-manifold edge. NM v: number of non-manifold vertices. BD e: number of boundary edges.

Results

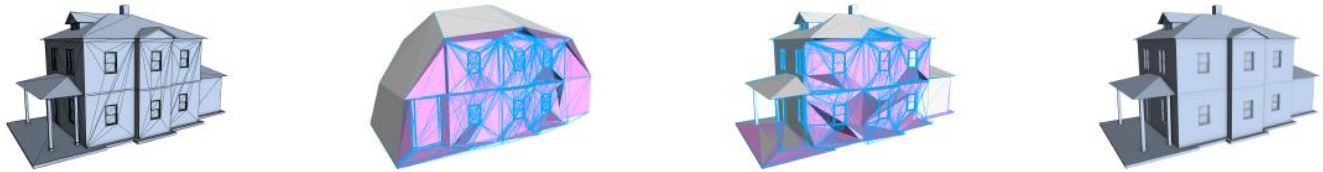


Results

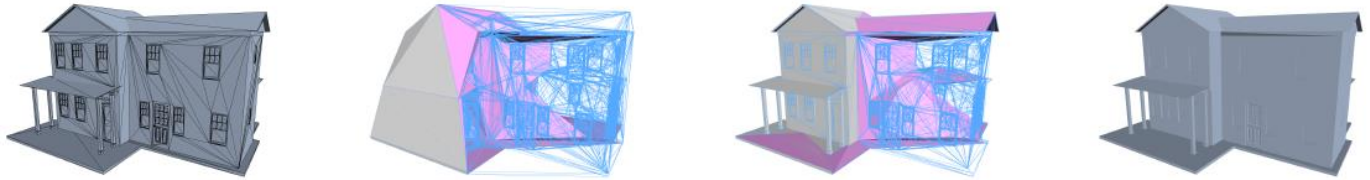
(d)

(e)

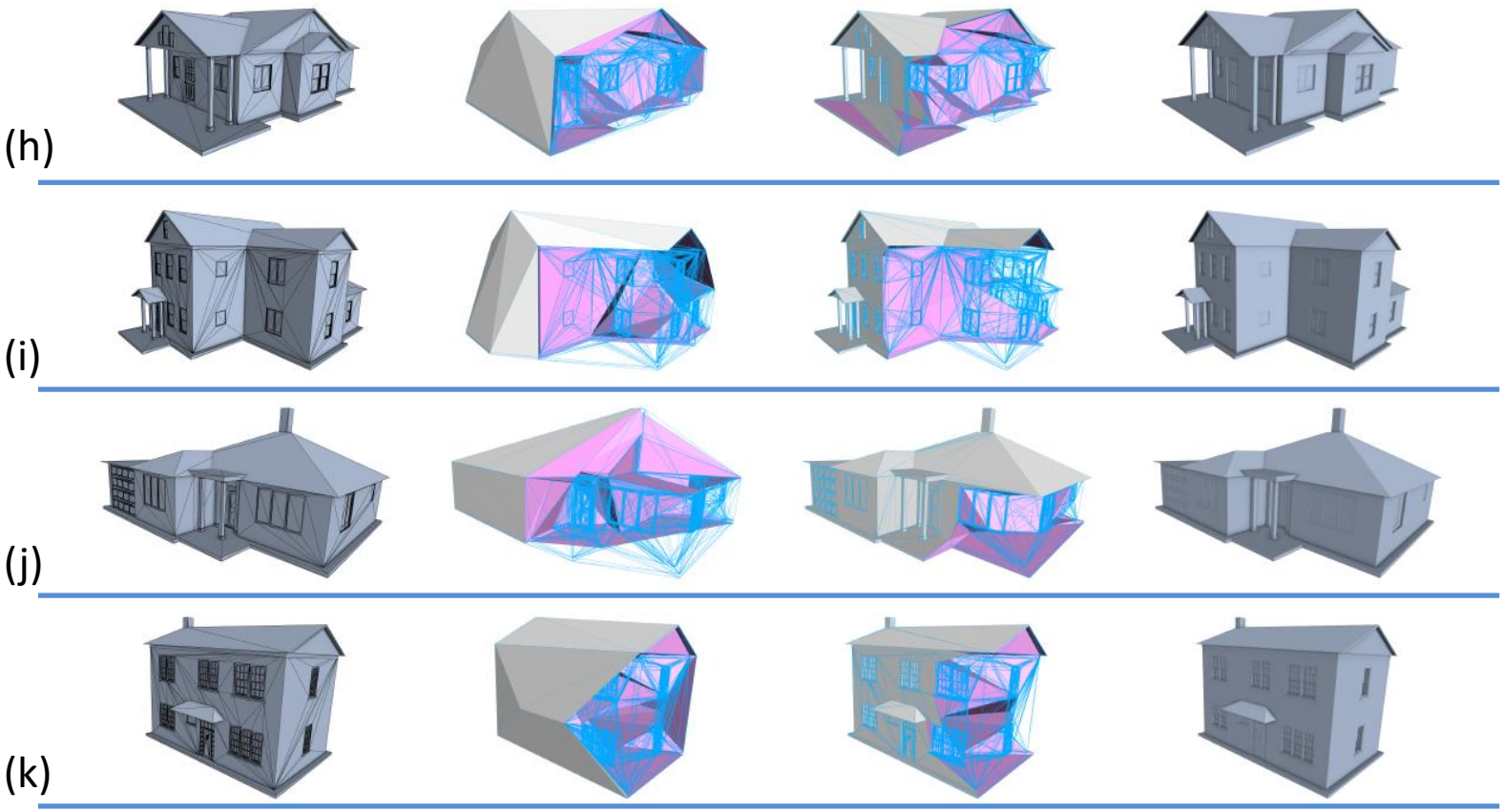
(f)



(g)



Results



Results

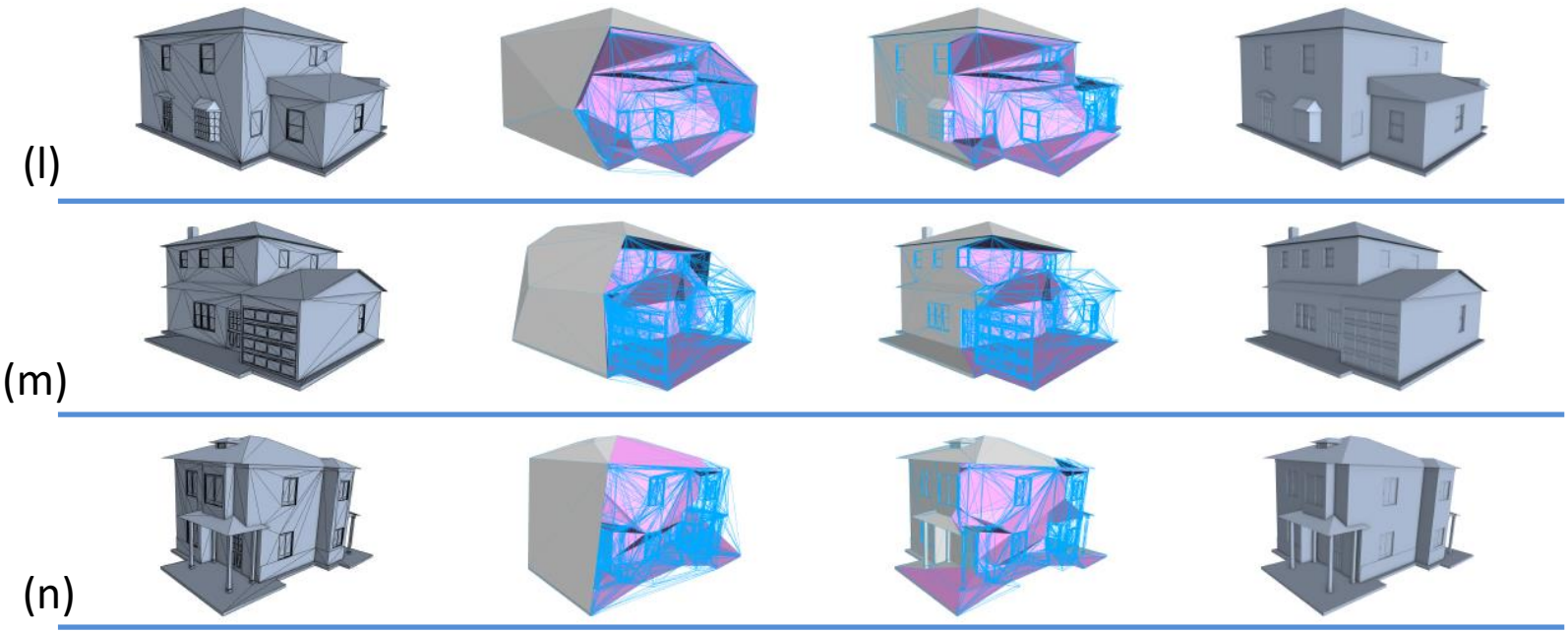


Figure 10: Stepwise results of our program. From left to right: input model, original CT, classified CT, outer surface. (a)~(n): task 1 x, task 2 x, task 4 x, task-1-x, task-2-x, task-7-x, task-8-x, task-11-x, task-12-x, task-16-x, task-17-x2, task-19-x, task-20-x, task-21-x.

Analysis

	CT	grouping	reuse	LoO (‰)
task_1_x	43530	12564	275	2.1
task_2_x	39983	11089	273	2.3
task_4_x	110550	61035	886	2.7
task-1-x	104543	56800	794	2.4
task-2-x	38459	9834	274	2.4
task-7-x	42774	12257	381	3.0
task-8-x	41911	11543	324	2.6
task-11-x	27293	7689	279	3.4
task-12-x	67098	22334	590	2.9
task-16-x	24241	9027	286	3.9
task-17-x2	50180	16863	345	2.3
task-19-x	33472	7989	314	3.1
task-20-x	35282	9494	320	3.0
task-21-x	39958	10221	292	2.4

Table 2: Effect of optimization. *CT*: number of tetrahedra in convex CT. *grouping*: number of tetrahedra that need to be classified. *reuse*: number of 2D boundaries constructed. *LoO*: level of optimization.

Analysis

	pre-process	CT	classification	OS extraction	in total
task_1_x	23.8	1.7	938.3	0.2	964.2
task_2_x	24.0	1.8	957.0	0.2	984.1
task_4_x	33.4	3.8	5820.4	0.2	5859.5
task-1-x	32.6	3.7	4338.7	0.2	4376.0
task-2-x	21.9	1.5	598.7	0.2	622.3
task-7-x	25.0	2.2	1838.0	0.2	1865.5
task-8-x	28.3	1.7	1219.8	0.2	1250.2
task-11-x	18.7	1.0	751.0	0.1	770.9
task-12-x	37.3	2.9	2759.1	0.4	2799.8
task-16-x	12.4	0.6	595.2	0.1	608.5
task-17-x2	30.7	2.3	1521.2	0.3	1554.8
task-19-x	19.1	1.2	967.0	0.2	987.6
task-20-x	19.5	1.4	1115.0	0.2	1136.2
task-21-x	23.0	1.1	914.8	0.2	939.2

Table 3: Execution times (s)

Analysis

	f	v	degeneracy	duplication	self-int	NM e	NM v	BD e
task_1_x_os	8166	4069	0	0	0	0	0	0
task_2_x_os	7260	3630	0	0	0	0	0	0
task_4_x_os	28998	14147	0	0	0	0	0	0
task-1-x_os	27276	13336	0	0	0	0	0	0
task-2-x_os	6018	2963	0	0	0	0	0	0
task-7-x_os	8904	4348	0	0	0	0	0	0
task-8-x_os	7358	3665	0	0	0	0	0	0
task-11-x_os	4970	2437	0	0	0	0	0	0
task-12-x_os	14790	7279	0	0	0	0	0	0
task-16-x_os	4642	2323	0	0	0	0	0	0
task-17-x2_os	11110	5550	0	0	0	0	0	18
task-19-x_os	6178	3025	0	0	0	0	0	0
task-20-x_os	7358	3591	0	0	0	0	0	8
task-21-x_os	5990	2961	0	0	0	0	0	0

Table 4. Validity check

Comparison

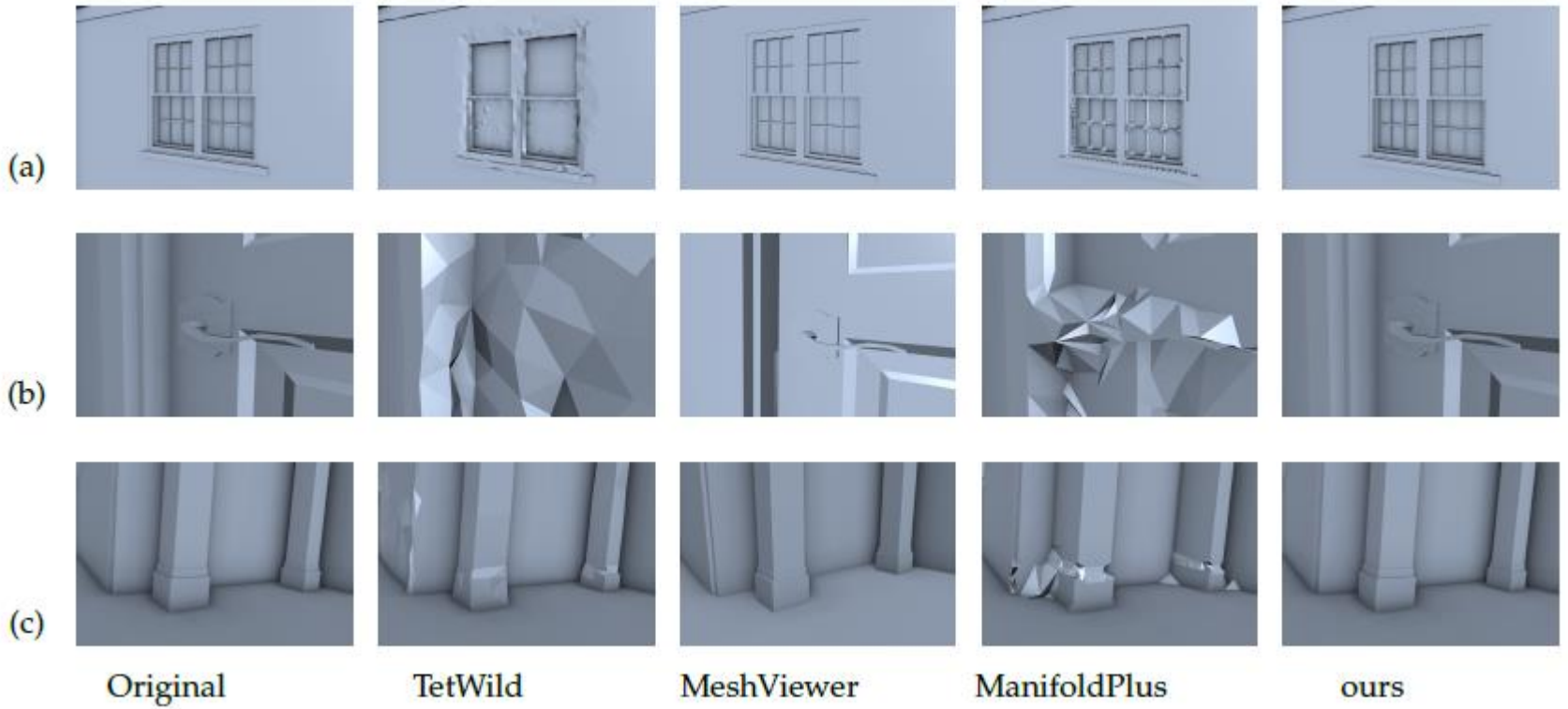


Figure 5.3: Close-ups of mesh details. (a) Window. (b) Handle. (c) Pillars.

Comparison

	NM e	NM v	BD e	time	deformation
original	767.9	26.1	3045.4	-	-
TetWild	726.5	252.4	0	1315.9	2.3e-5
MeshViewer	112.8	0.6	281.5	625.4	8.2e-3
ManifolfPlus	0	0	0	31.8	4.1e-6
ours	0	0	1.9	1765.6	1.6e-7

Table 5.5: Comparison with state-of-the-art methods. Statistics are averaged.

Discussion

	TetWild	MeshViewer	ManifoldPlus	Ours
Efficiency				
Validity				
Integrity				

- **Efficiency**. *ManifoldPlus* is the fastest among the four methods. Our method is quite time-consuming at this stage, especially that of classification.
- **Validity**. *TetWild* and *MeshViewer* are far from success. *ManifoldPlus* guarantees valid results for all testing models. Our method produces acceptable results with only a few boundary edges present in two models.
- **Integrity**. *TetWild* and *ManifoldPlus* can not preserve small features. Our method exactly reproduces details with minimum intrusion.

Outline

- Introduction
- Related works
- Methodology
- Results & discussion
- Conclusions & future works

Conclusions

*How can we accurately extract a **watertight** and **manifold** outer surface from an error-ridden 3D building model?*

By using a hybrid method i.e., both surface-oriented and volumetric

- Fully automatic
- Non-parametric
- Accurate
- Robust

Future works

- Complete exact arithmetic
- Non-manifoldness removal
- Multi-threading
- Test on more data

Thanks for your attention!

Questions?