MSc thesis in Geomatics

An automatic geometry repair framework for semantic 3D city models

Lisa Keurentjes
2024

LOD x.0
LOD x.1
LOD x.2
LOD x.3
LOD0
LOD0.0
LOD0.1
LOD0.2
LOD0.3
LOD1
LOD1.0
LOD1.1
LOD1.2
LOD1.3
LOD2
LOD2.0
LOD2.1
LOD2.2
LOD2.3
LOD3
LOD3.0
LOD3.1
LOD3.2
LOD3.3
Zuidplantsoen
Julianalaan
Michiel de Ruyterweg
Poortlandplein
Nassaulaan
Schoemakerstraat
Zuidplantsoen
Charlotte de Bourbonstraat
De Bouwpub
Greenwheels deelauto
TU Delft Faculteit Bouwkunde
The Berlage
Julianalaan
Mijnbouwstraat
Michiel de Ruyterweg
Explore Lab
Stichting Zeoliet Research Delft
De Vries van

Visibility analysis
Solar potential estimation
3D cadastre
Energy demand estimation
TUDelft
Infrastructure planning
Emergency response
Shadow estimation
Indoor navigation
Noise propagation
Utility management

Visibility analysis

Solar potential estimation

3D cadastre

Energy demand estimation

Emer

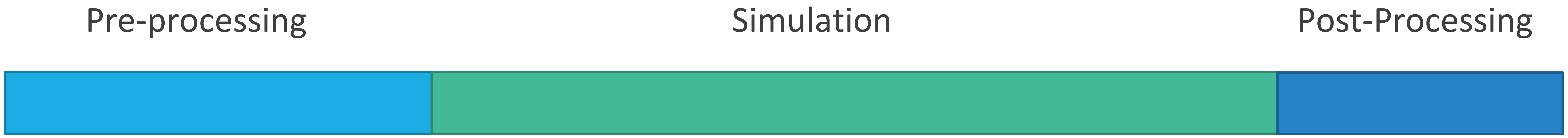
Shadow

Indoor navigation

Noise propagation

Utility management

Time spend

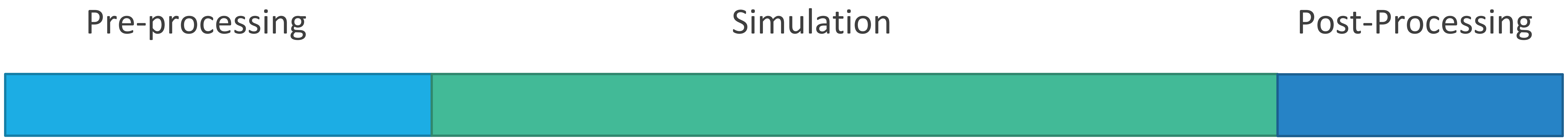


Pre-processing

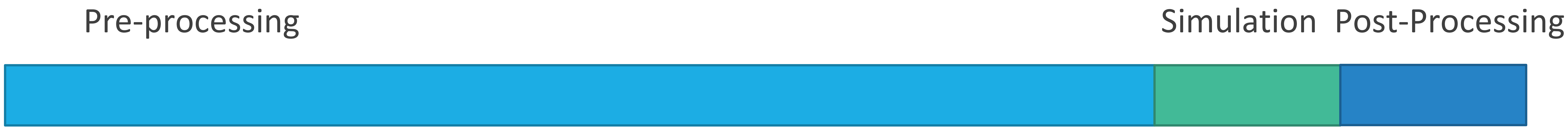
Simulation

Post-Processing

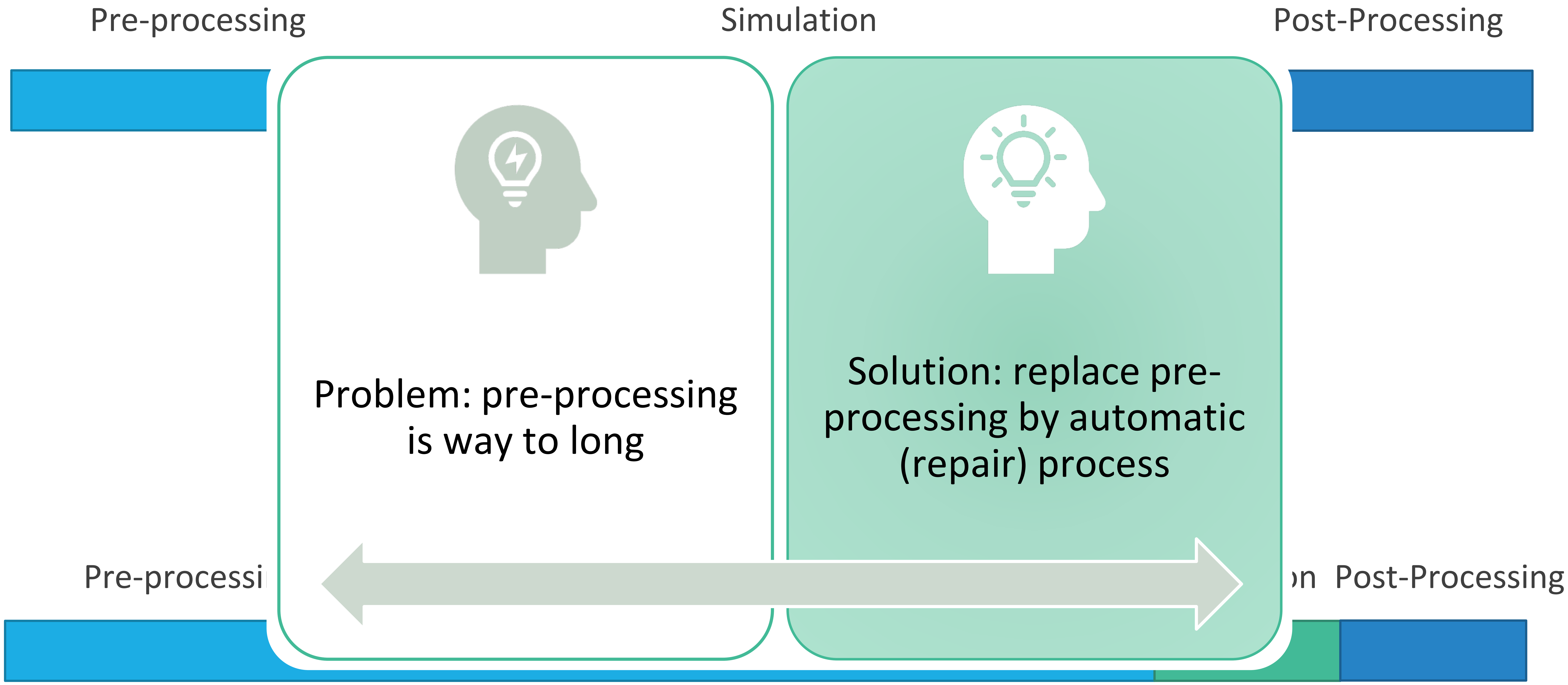
Time spend



Time spend realistic



Time spend

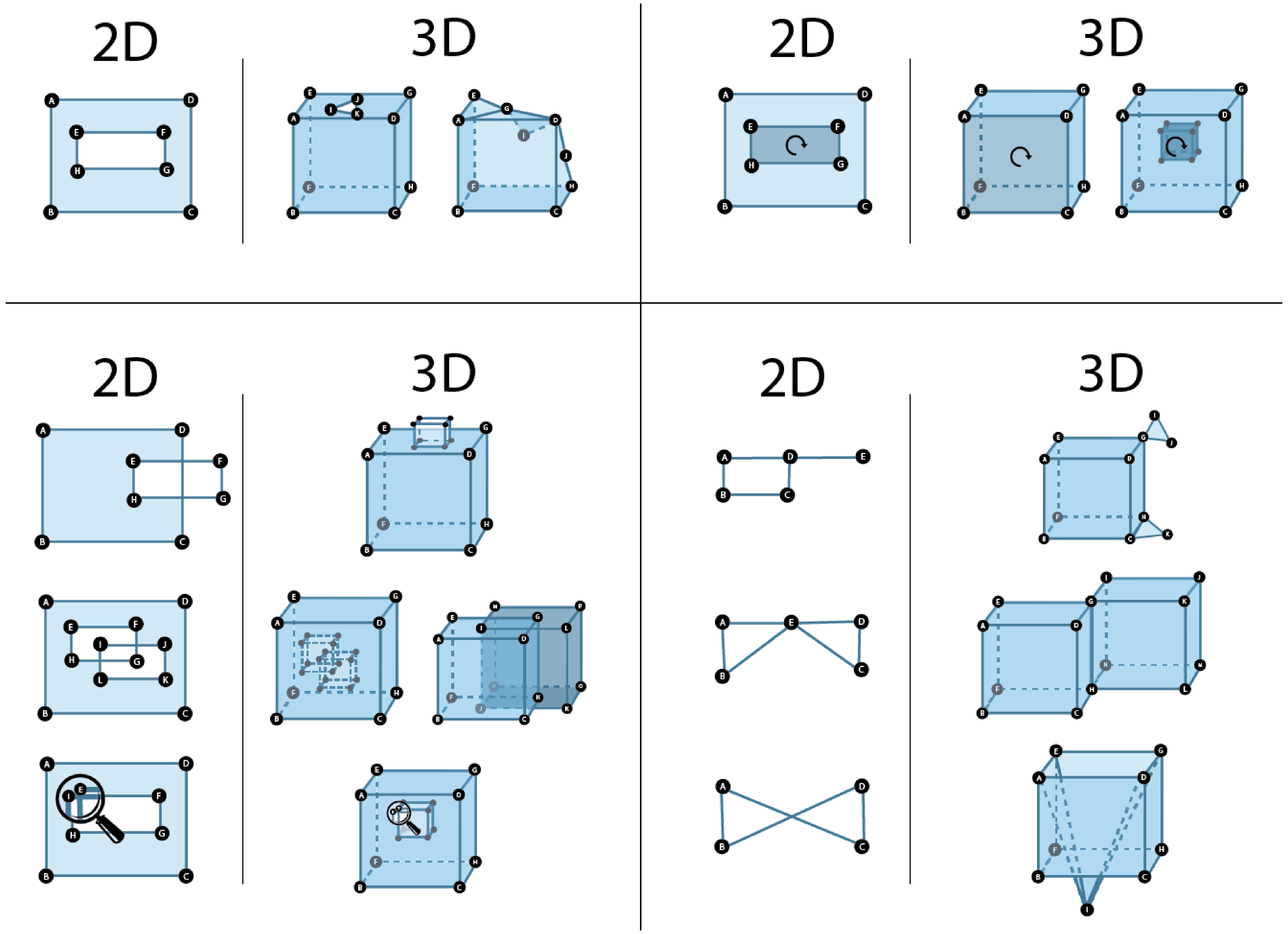


Problem: pre-processing is way to long

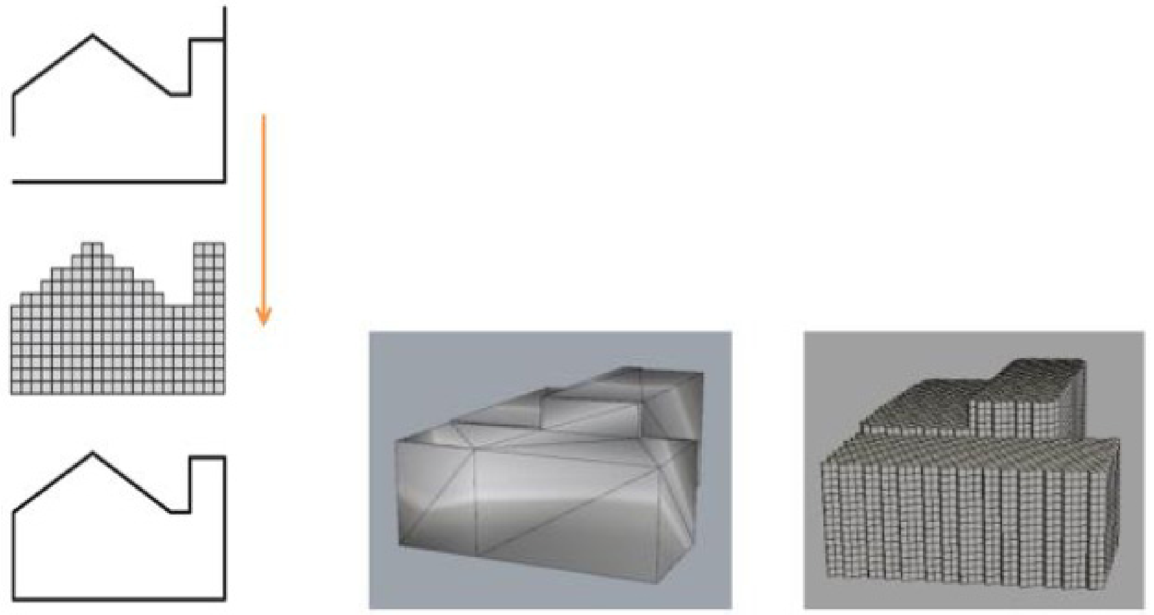
Solution: replace pre-processing by automatic (repair) process

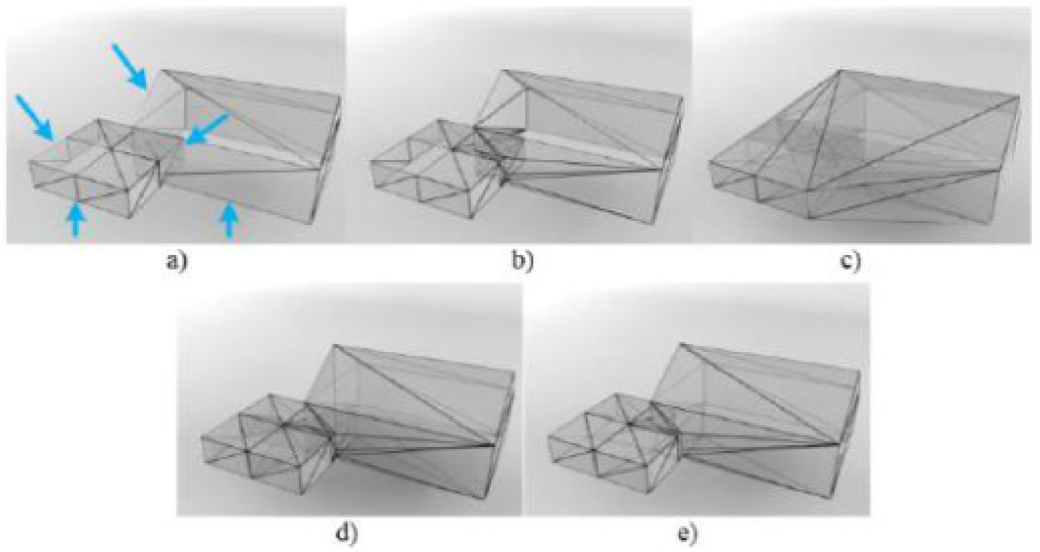
Existing repair methods

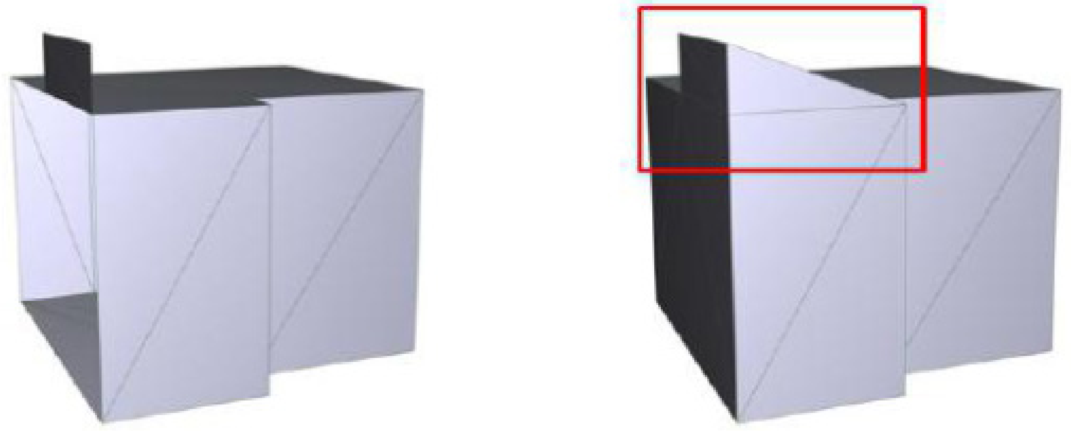
Local

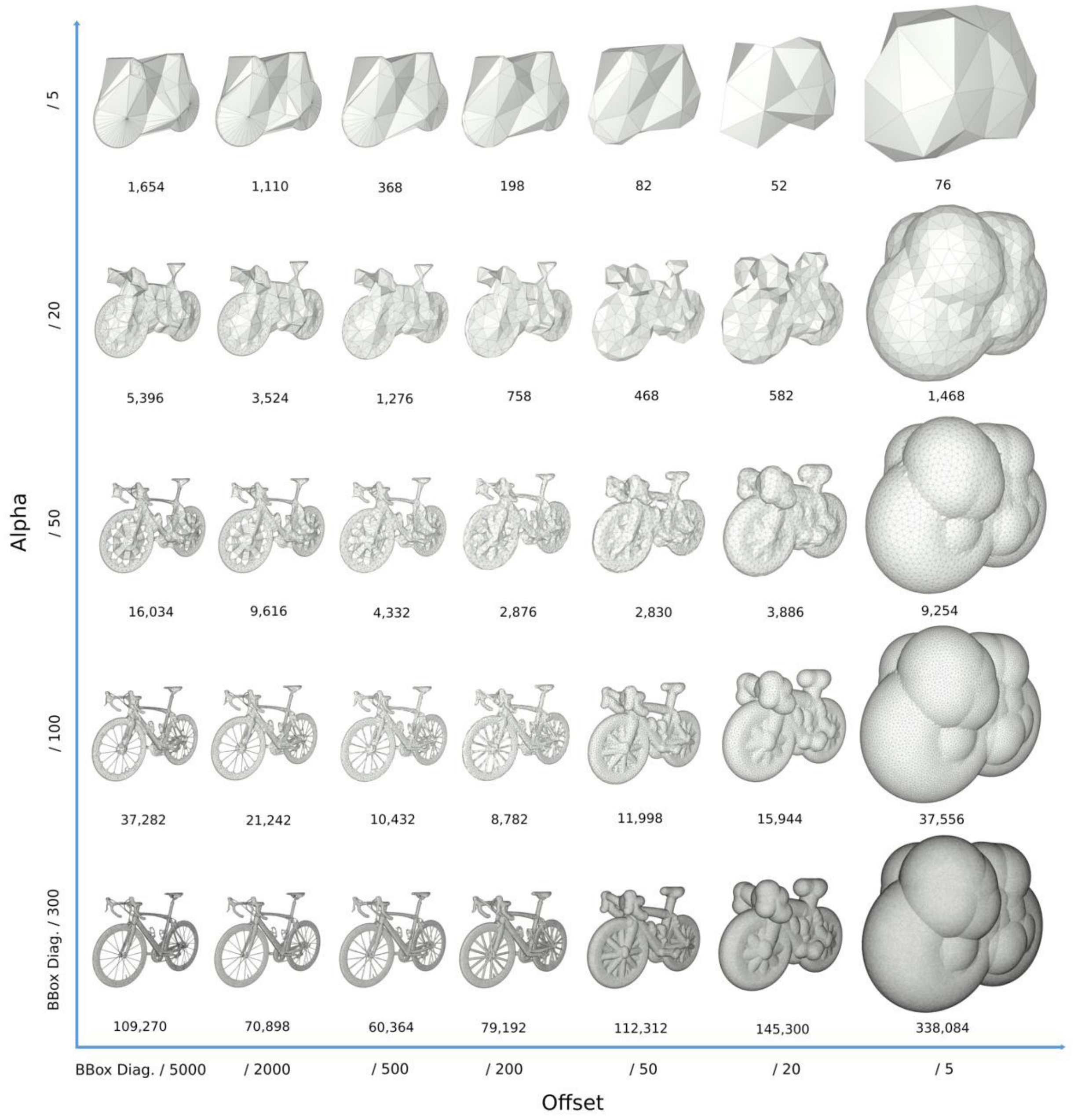


Global

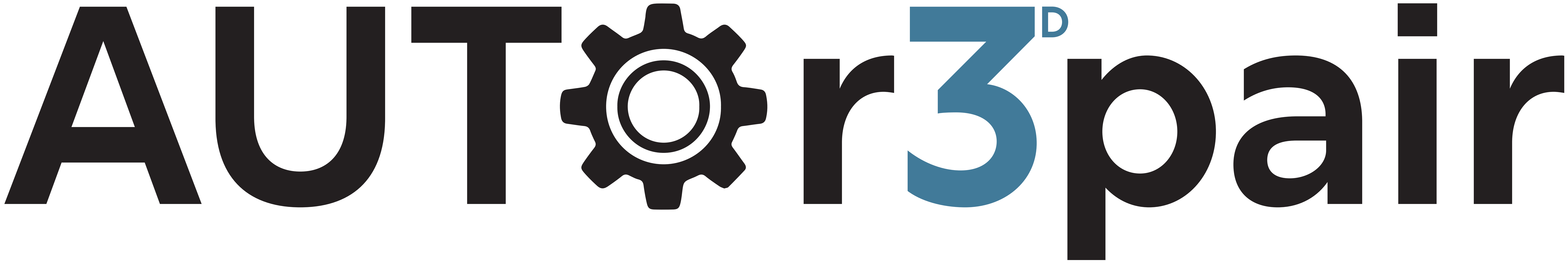


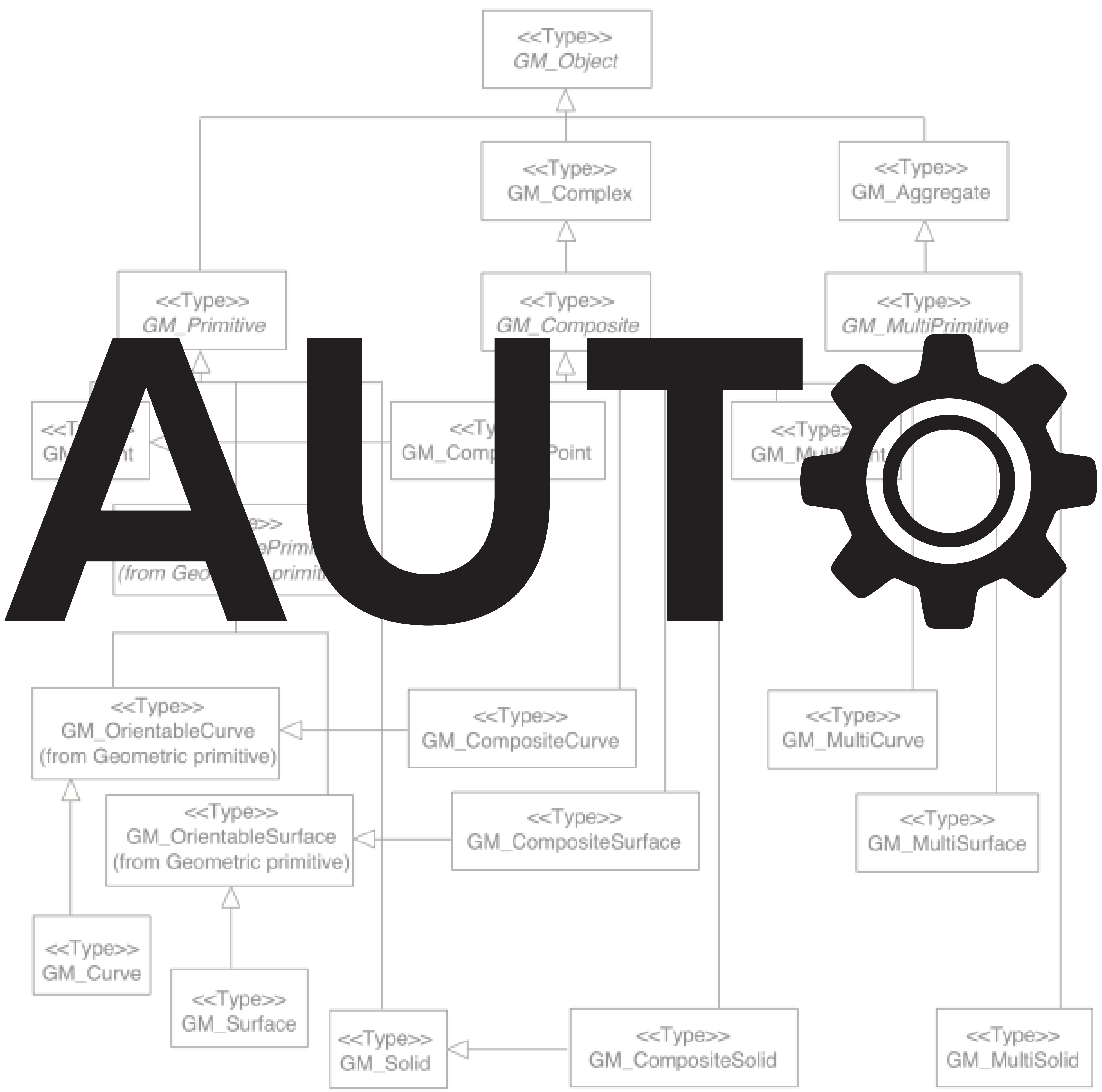


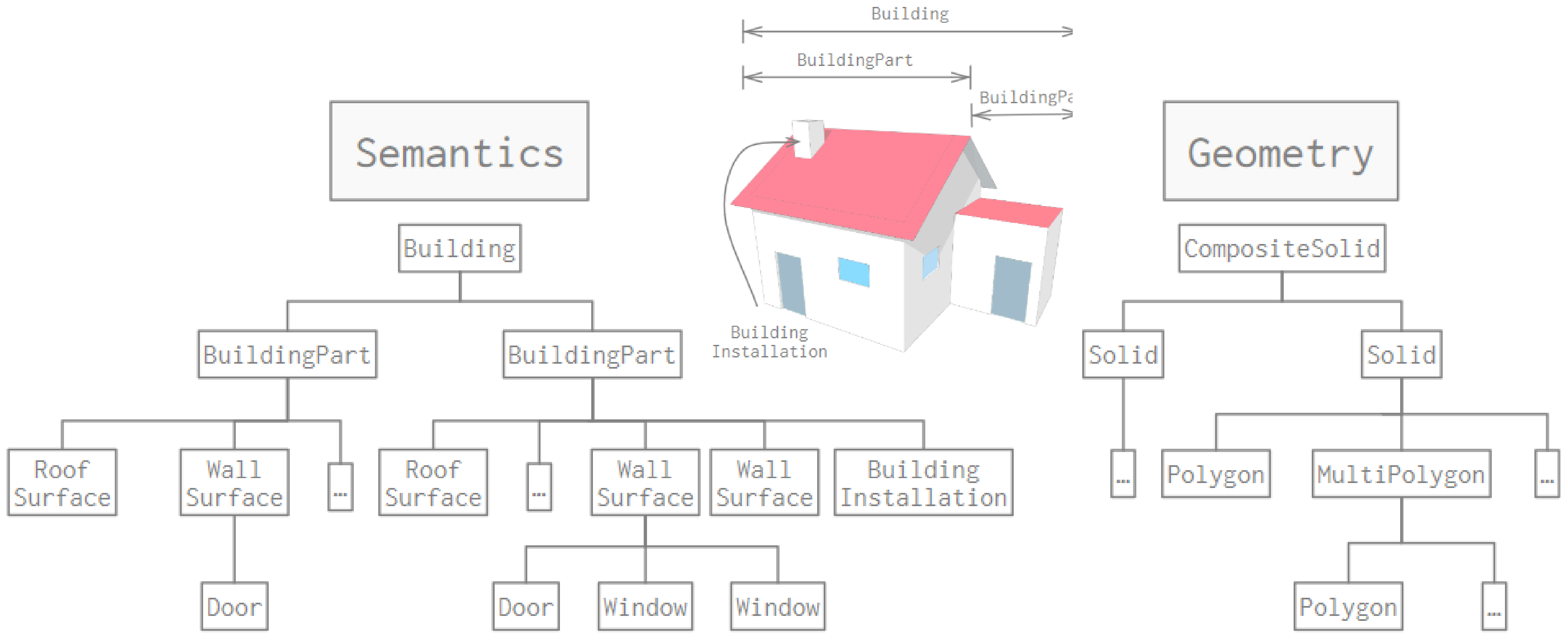




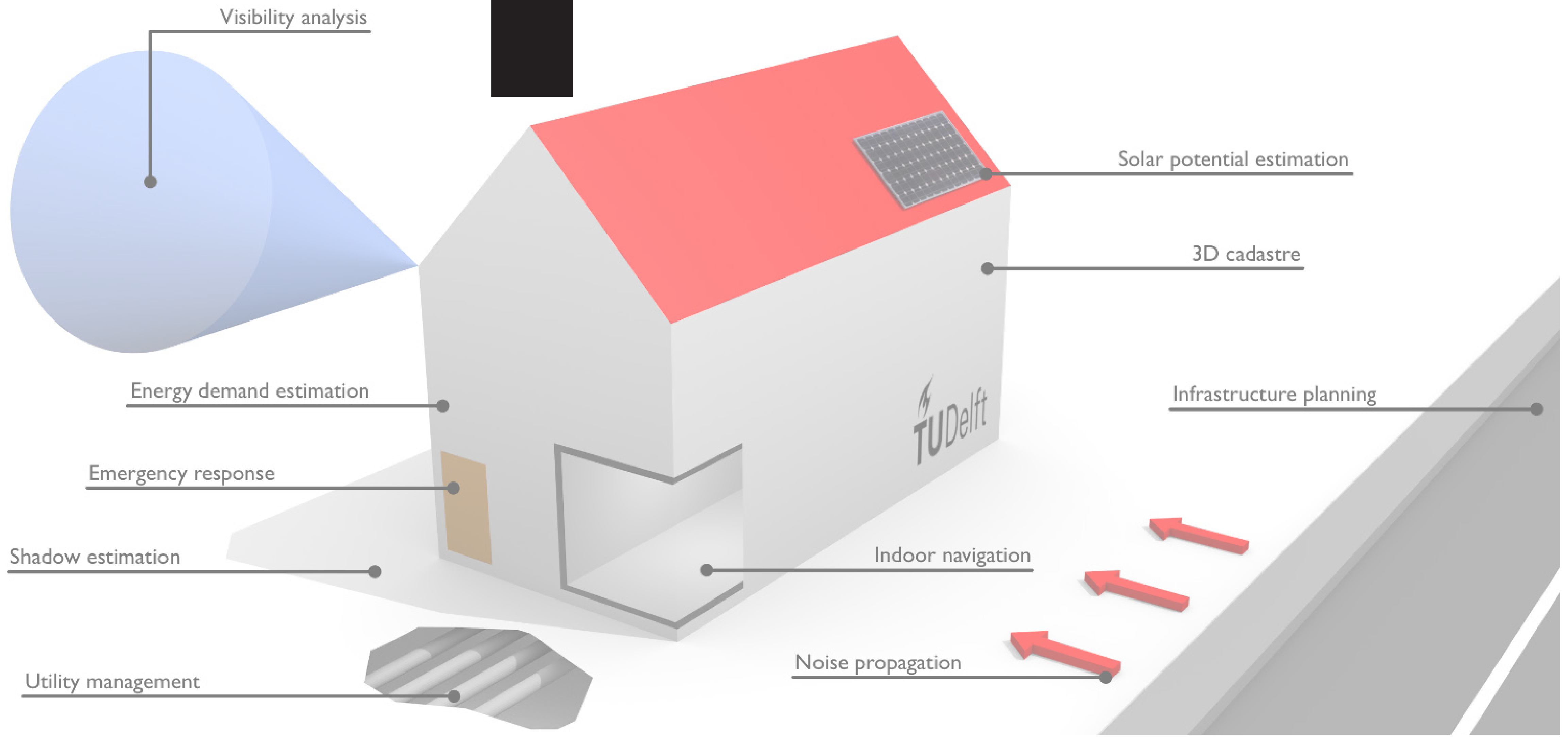
Develop a framework for the automatic repair and reconstruction of 3D city models to facilitate different use cases and implement a prototype.

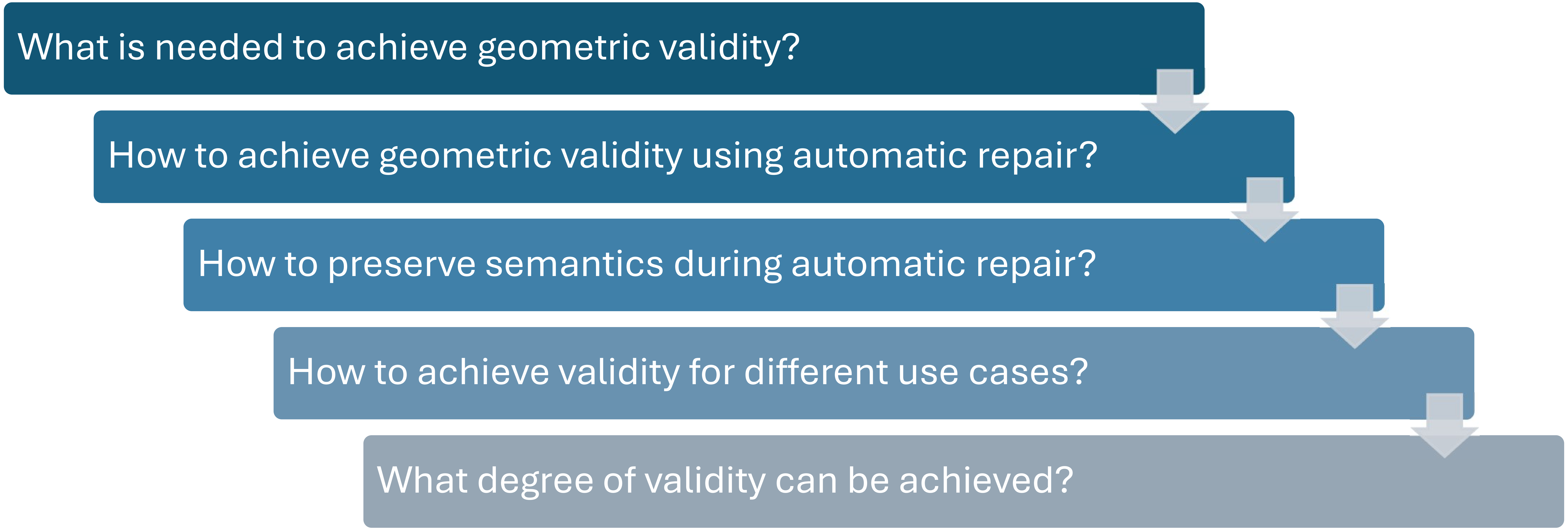

AUTOr3Dpair





AUTOr3Dpair





What is needed to achieve geometric validity?

How to achieve geometric validity using automatic repair?

How to preserve semantics during automatic repair?

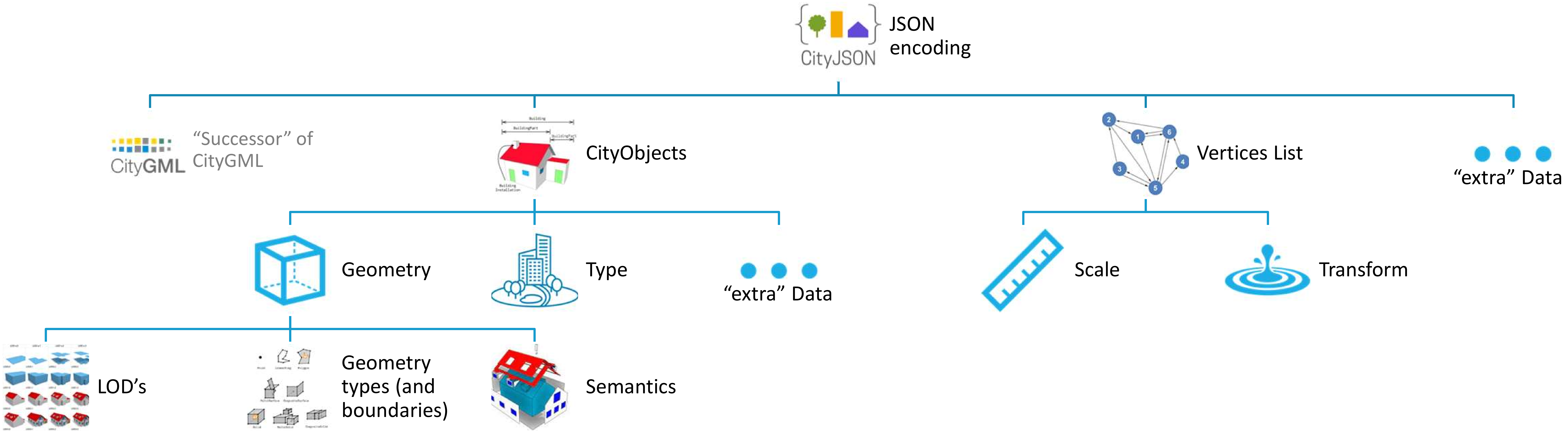
How to achieve validity for different use cases?

What degree of validity can be achieved?

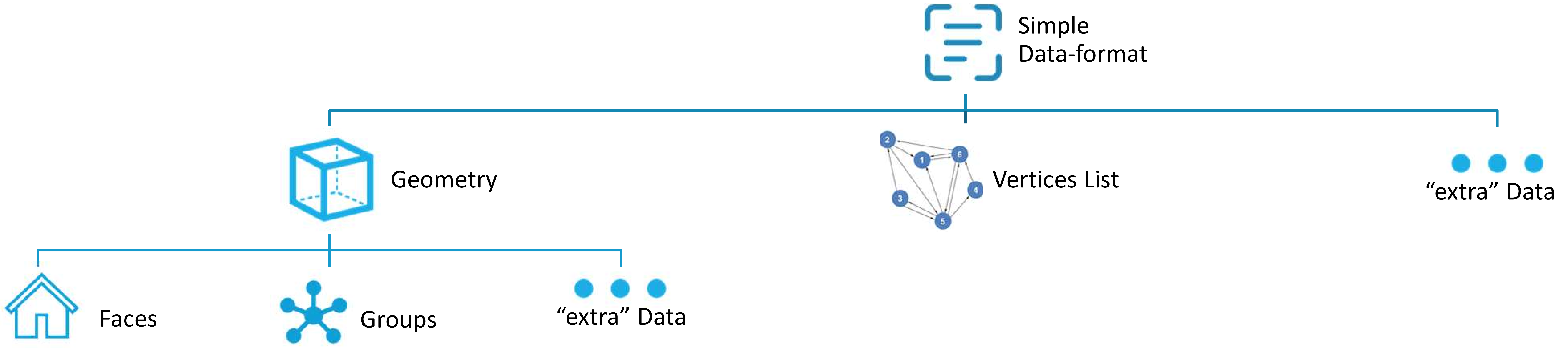
What is needed to achieve geometric validity?

Scope of the thesis

CityJSON

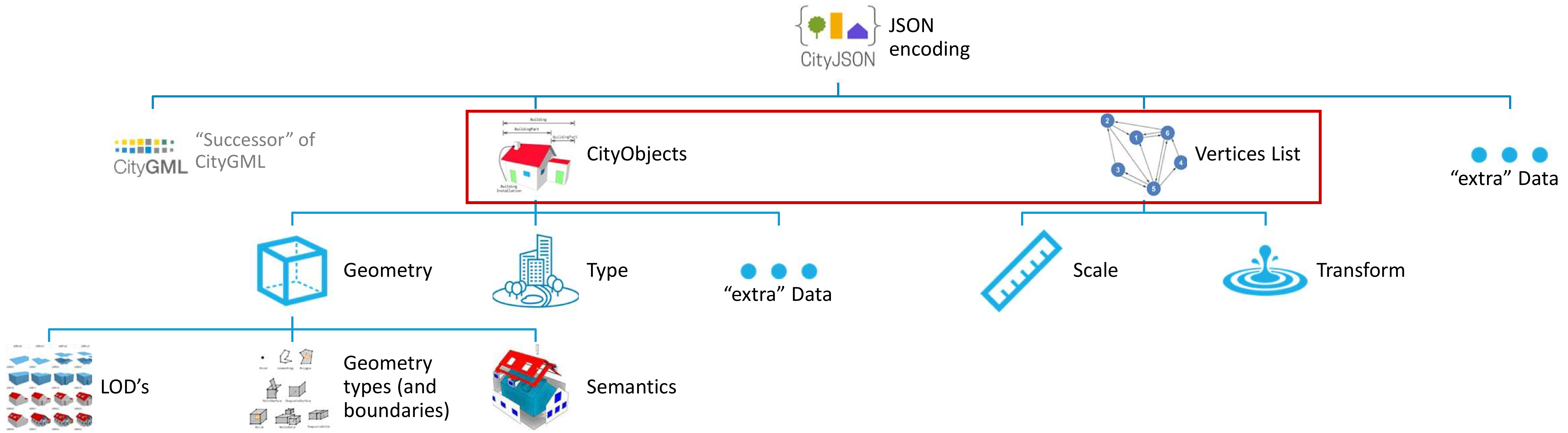


OBJ



Scope of the thesis

CityJSON



JSON encoding

“Successor” of CityGML

CityObjects

Vertices List

“extra” Data

Geometry

Type

“extra” Data

Scale

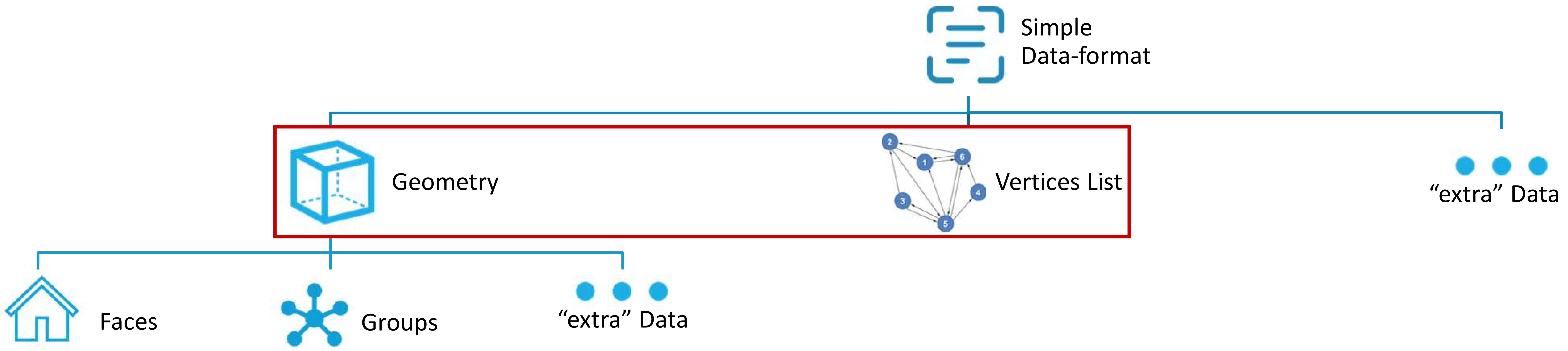
Transform

LOD’s

Geometry types (and boundaries)

Semantics

OBJ



Simple Data-format

Geometry

Vertices List

“extra” Data

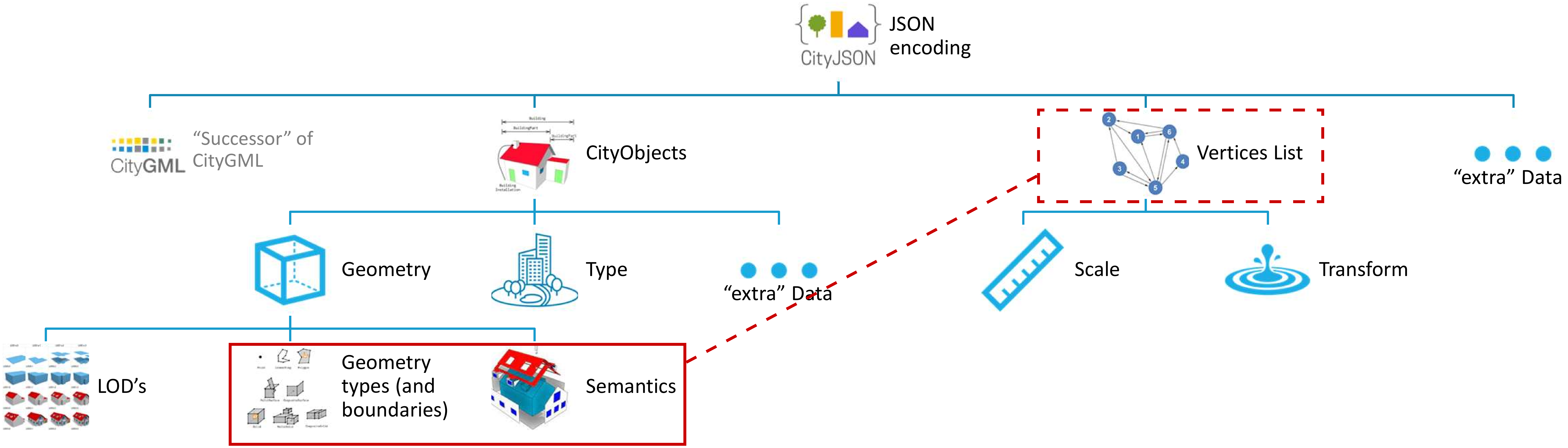
Faces

Groups

“extra” Data

Scope of the thesis

CityJSON



JSON encoding

“Successor” of CityGML

CityObjects

Vertices List

“extra” Data

Geometry

Type

“extra” Data

Scale

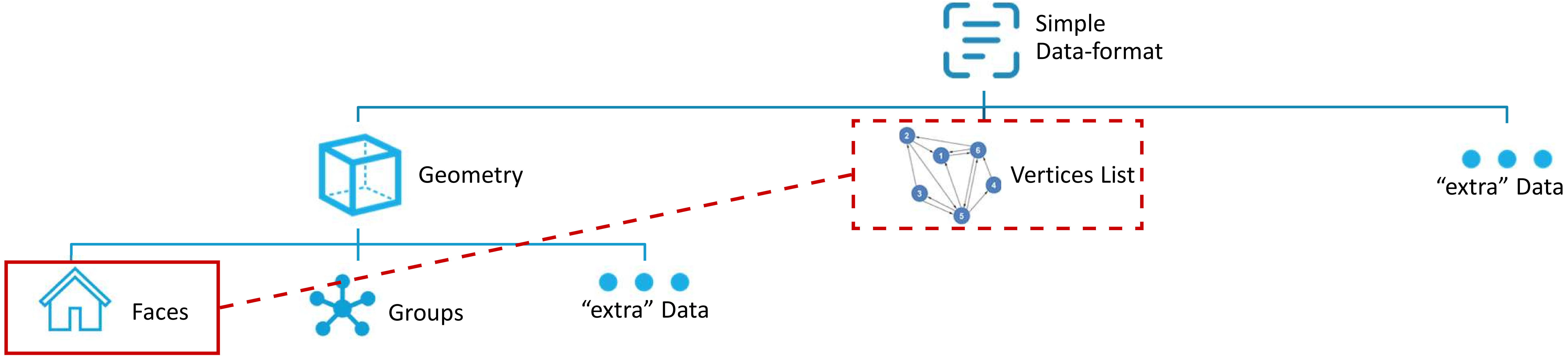
Transform

LOD’s

Geometry types (and boundaries)

Semantics

OBJ



Simple Data-format

Geometry

Vertices List

“extra” Data

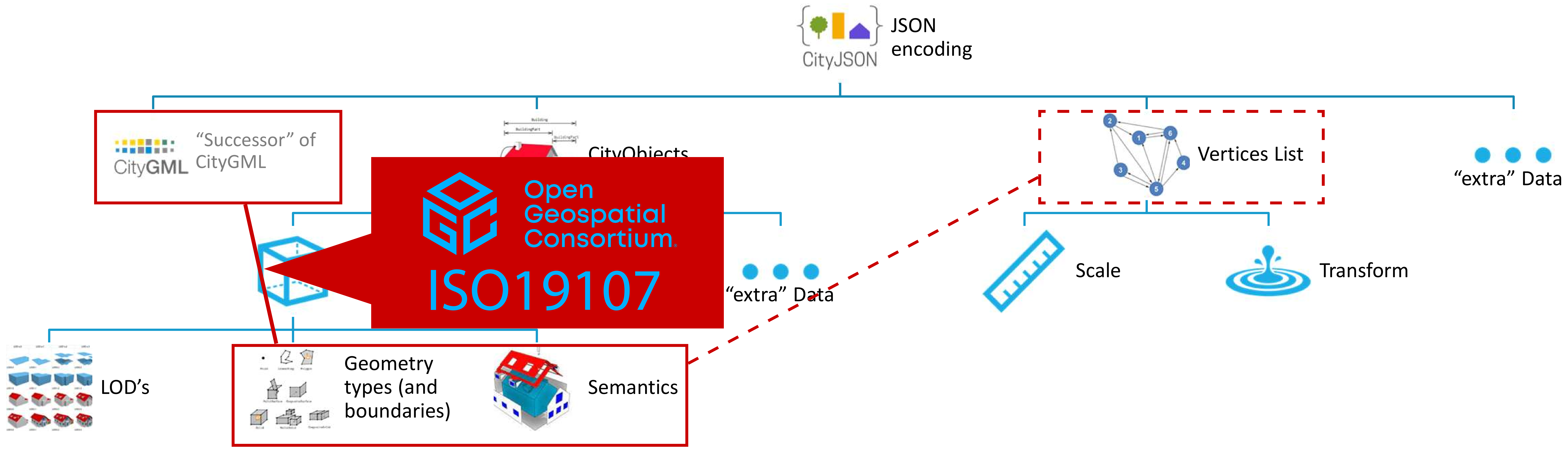
Faces

Groups

“extra” Data

Scope of the thesis

CityJSON



JSON encoding

"Successor" of CityGML

CityObjects

Open Geospatial Consortium

ISO19107

"extra" Data

Vertices List

"extra" Data

Scale

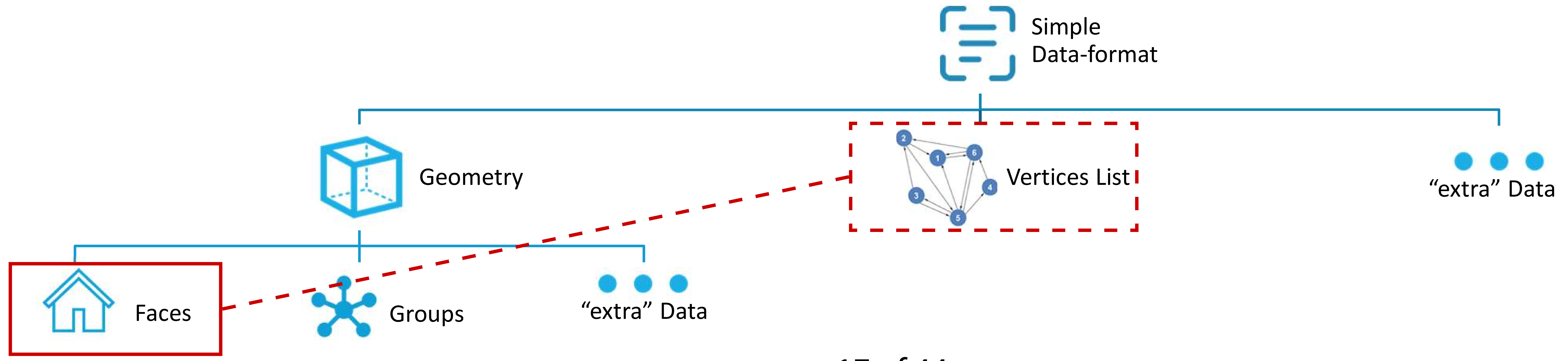
Transform

LOD's

Geometry types (and boundaries)

Semantics

OBJ



Simple Data-format

Geometry

Vertices List

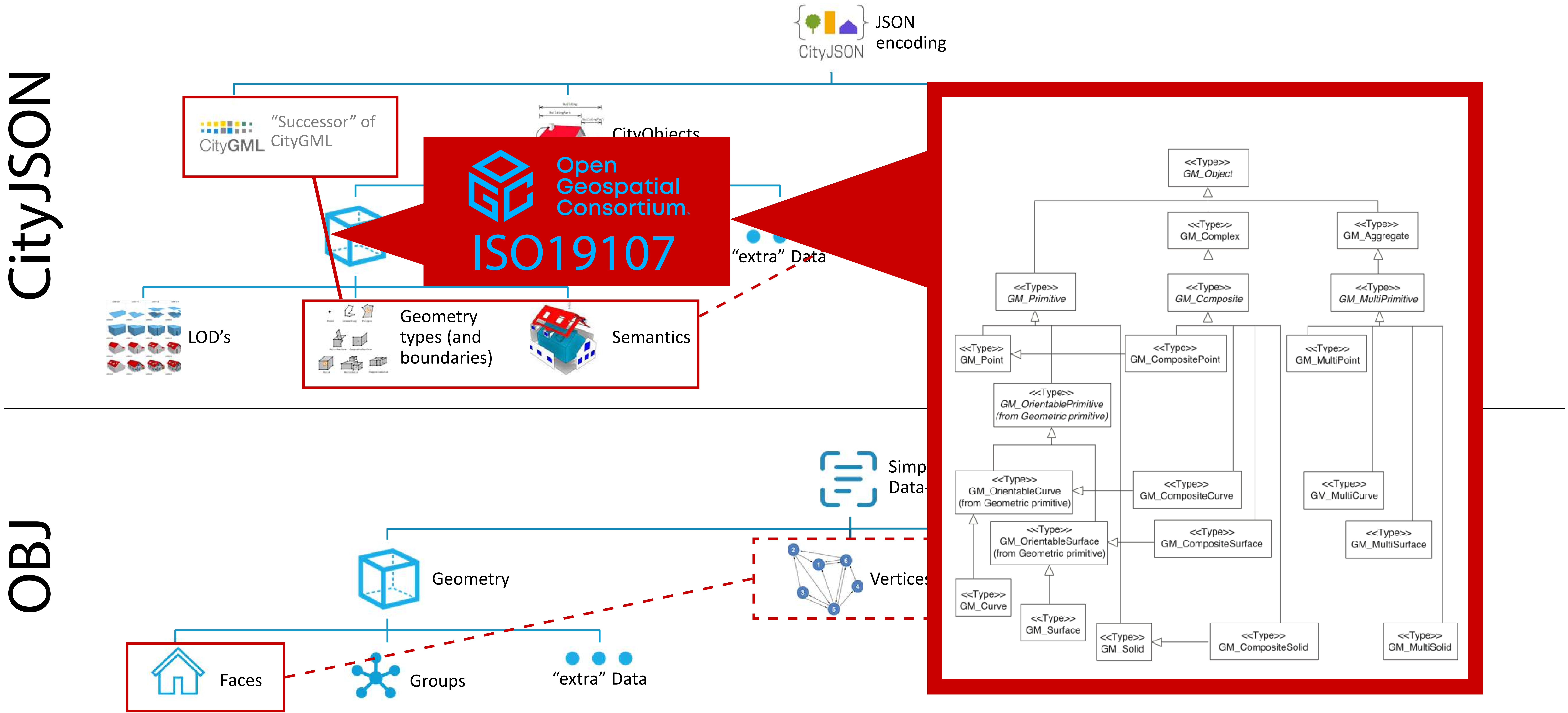
"extra" Data

Faces

Groups

"extra" Data

Scope of the thesis



CityJSON

JSON encoding

"Successor" of CityGML

CityObjects

Open Geospatial Consortium

ISO19107

"extra" Data

LOD's

Geometry types (and boundaries)

Semantics

OBJ

Simp Data-

Geometry

Vertices

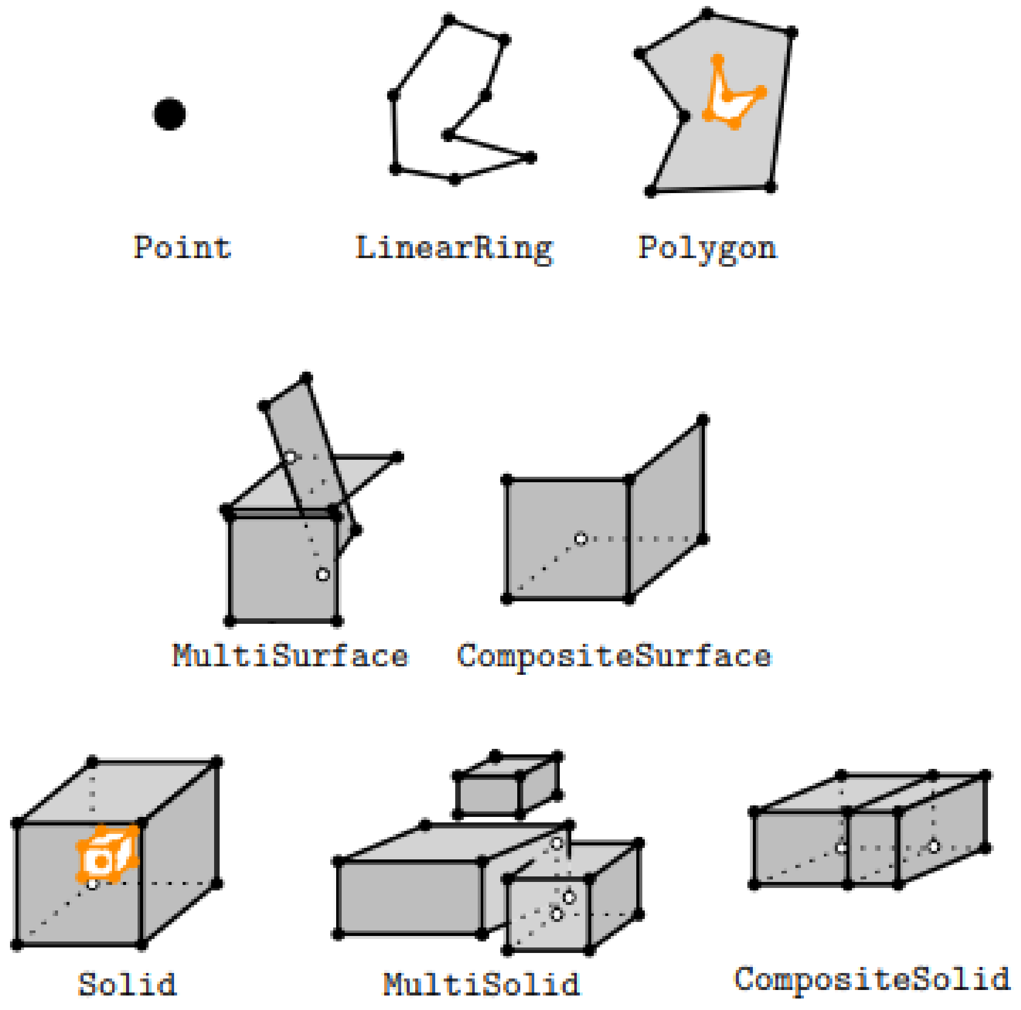
Faces

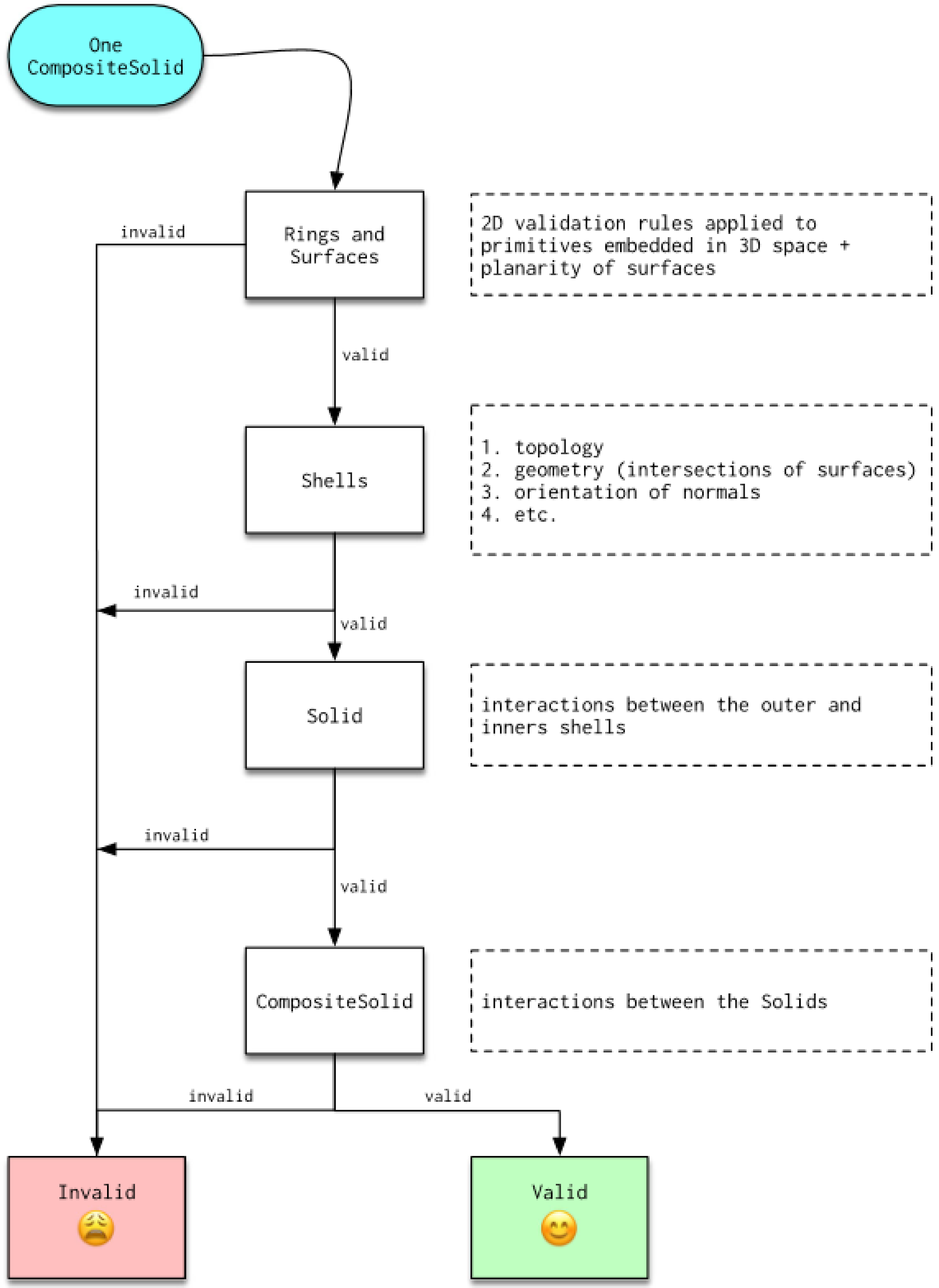
Groups

"extra" Data

CityJSON
CityGML
<<Type>> GM_Object
<<Type>> GM_Complex
<<Type>> GM_Aggregate
<<Type>> GM_Primitive
<<Type>> GM_Composite
<<Type>> GM_MultiPrimitive
<<Type>> GM_Point
<<Type>> GM_CompositePoint
<<Type>> GM_MultiPoint
<<Type>> GM_OrientablePrimitive (from Geometric primitive)
<<Type>> GM_OrientableCurve (from Geometric primitive)
<<Type>> GM_CompositeCurve
<<Type>> GM_MultiCurve
<<Type>> GM_OrientableSurface (from Geometric primitive)
<<Type>> GM_CompositeSurface
<<Type>> GM_MultiSurface
<<Type>> GM_Curve
<<Type>> GM_Surface
<<Type>> GM_Solid
<<Type>> GM_CompositeSolid
<<Type>> GM_MultiSolid
1
2
3
4
5
6

Geometric validity




One CompositeSolid

Rings and Surfaces

2D validation rules applied to primitives embedded in 3D space + planarity of surfaces

Shells

1. topology
2. geometry (intersections of surfaces)
3. orientation of normals
4. etc.

Solid

interactions between the outer and inners shells

CompositeSolid

interactions between the Solids

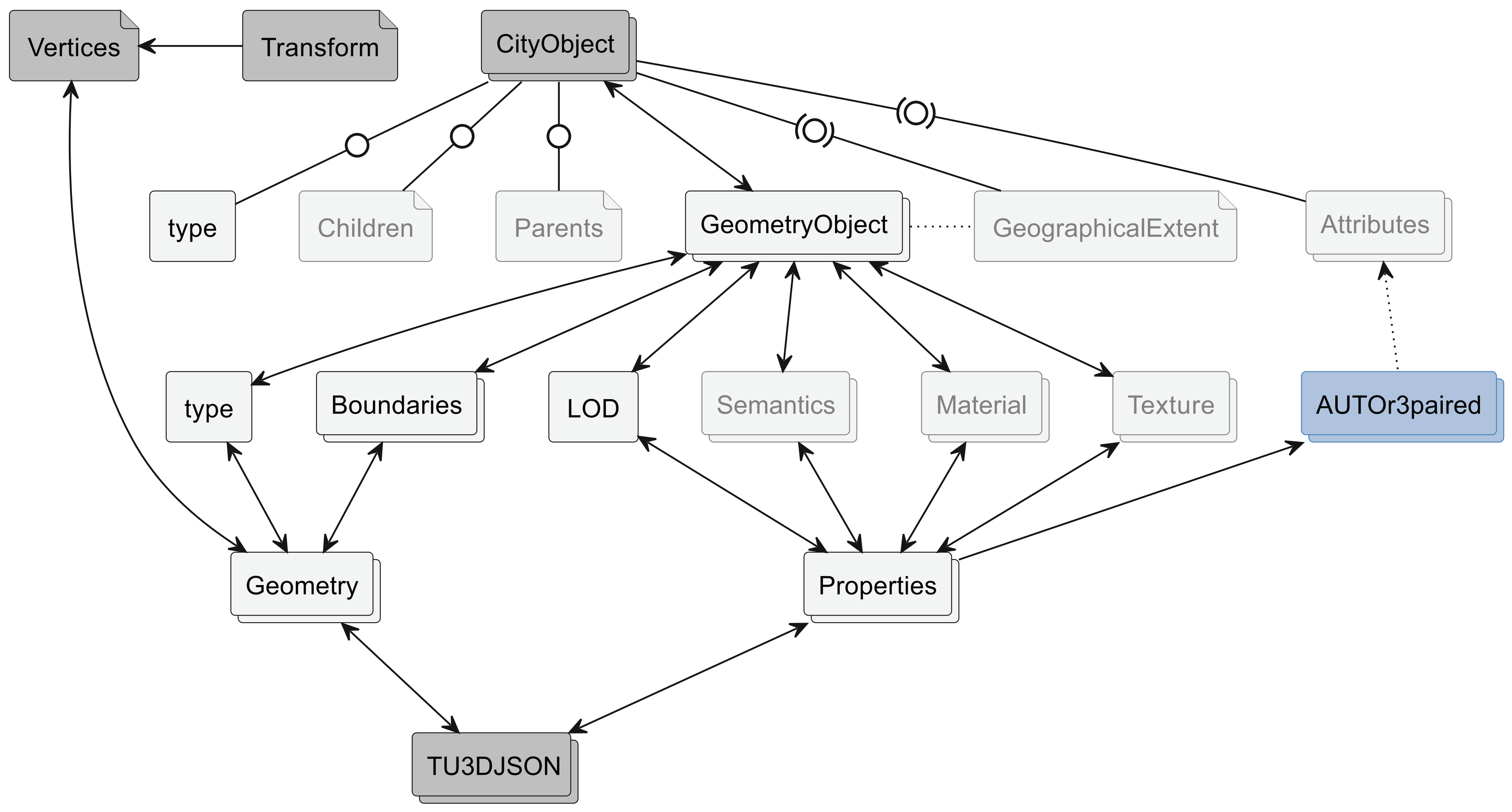
Invalid 😩

Valid 😊

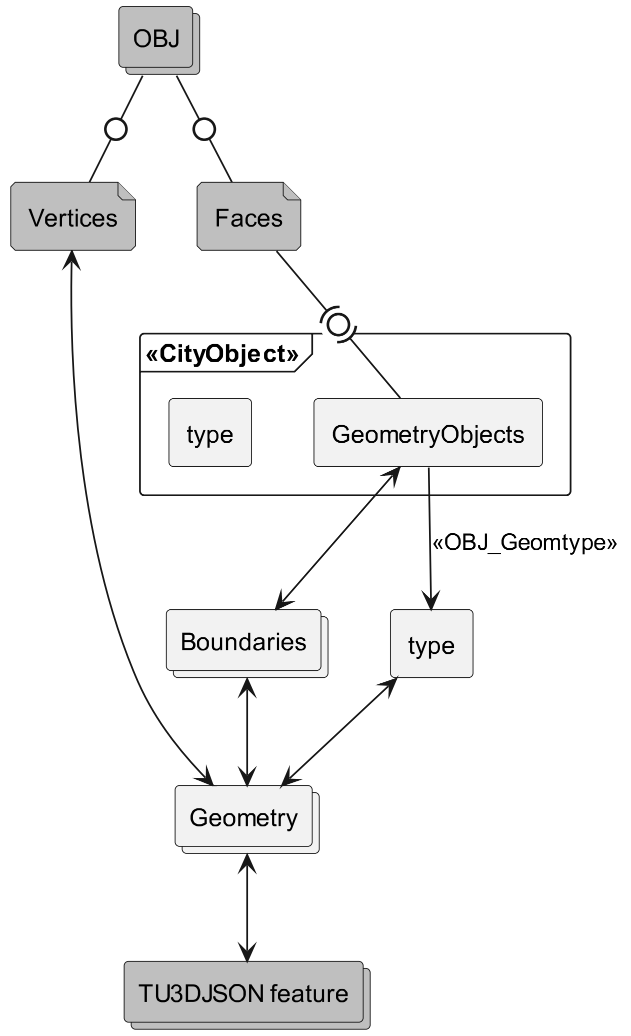
How to achieve geometric validity using automatic repair?

Convert to TU3DJSON

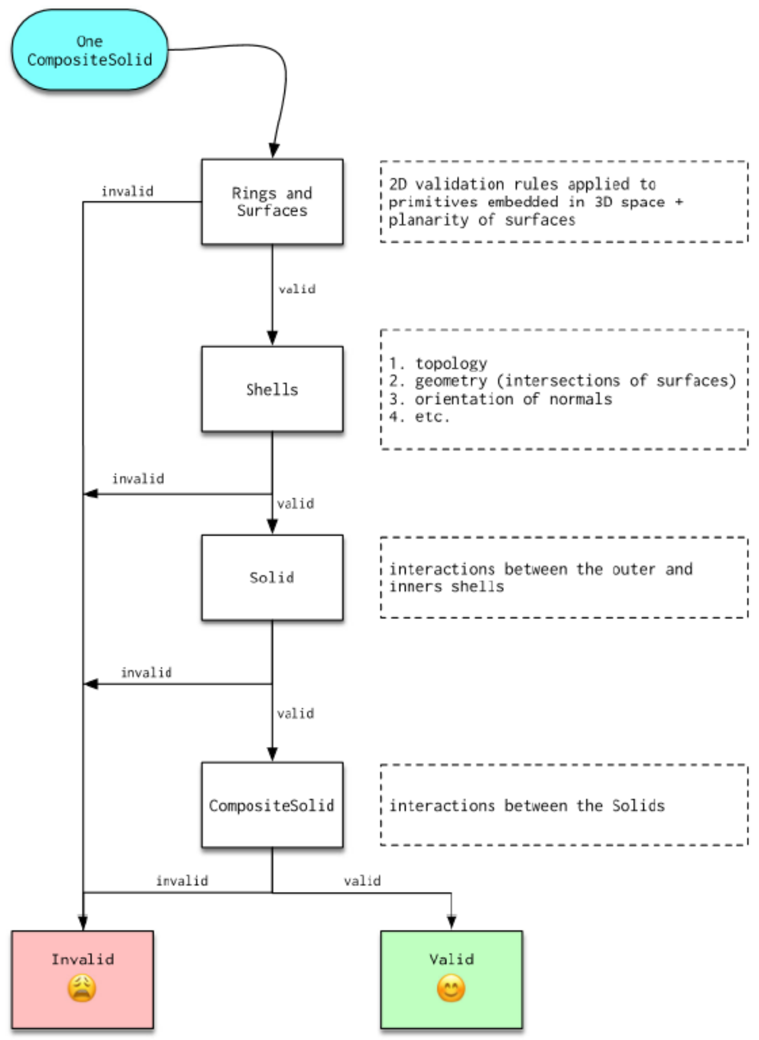
CityJSON



OBJ



Find defects



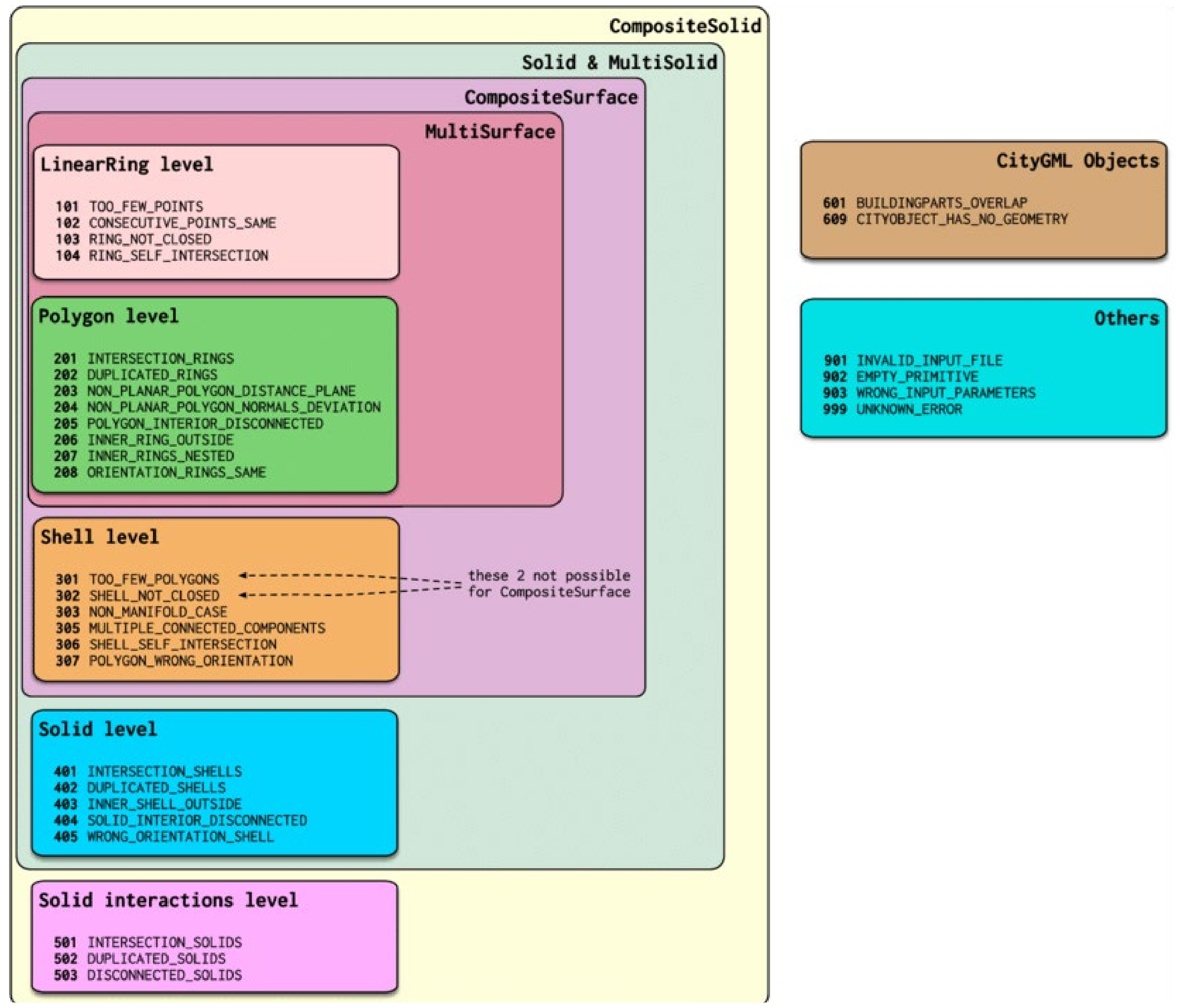
2D validation rules applied to primitives embedded in 3D space + planarity of surfaces

1. topology
2. geometry (intersections of surfaces)
3. orientation of normals
4. etc.

interactions between the outer and inners shells

interactions between the Solids

val3dity



CompositeSolid

Solid & MultiSolid

CompositeSurface

MultiSurface

LinearRing level

101 TOO_FEW_POINTS
102 CONSECUTIVE_POINTS_SAME
103 RING_NOT_CLOSED
104 RING_SELF_INTERSECTION

Polygon level

201 INTERSECTION_RINGS
202 DUPLICATED_RINGS
203 NON_PLANAR_POLYGON_DISTANCE_PLANE
204 NON_PLANAR_POLYGON_NORMALS_DEVIATION
205 POLYGON_INTERIOR_DISCONNECTED
206 INNER_RING_OUTSIDE
207 INNER_RINGS_NESTED
208 ORIENTATION_RINGS_SAME

Shell level

301 TOO_FEW_POLYGONS
302 SHELL_NOT_CLOSED
303 NON_MANIFOLD_CASE
305 MULTIPLE_CONNECTED_COMPONENTS
306 SHELL_SELF_INTERSECTION
307 POLYGON_WRONG_ORIENTATION

these 2 not possible for CompositeSurface

Solid level

401 INTERSECTION_SHELLS
402 DUPLICATED_SHELLS
403 INNER_SHELL_OUTSIDE
404 SOLID_INTERIOR_DISCONNECTED
405 WRONG_ORIENTATION_SHELL

Solid interactions level

501 INTERSECTION_SOLIDS
502 DUPLICATED_SOLIDS
503 DISCONNECTED_SOLIDS

CityGML Objects

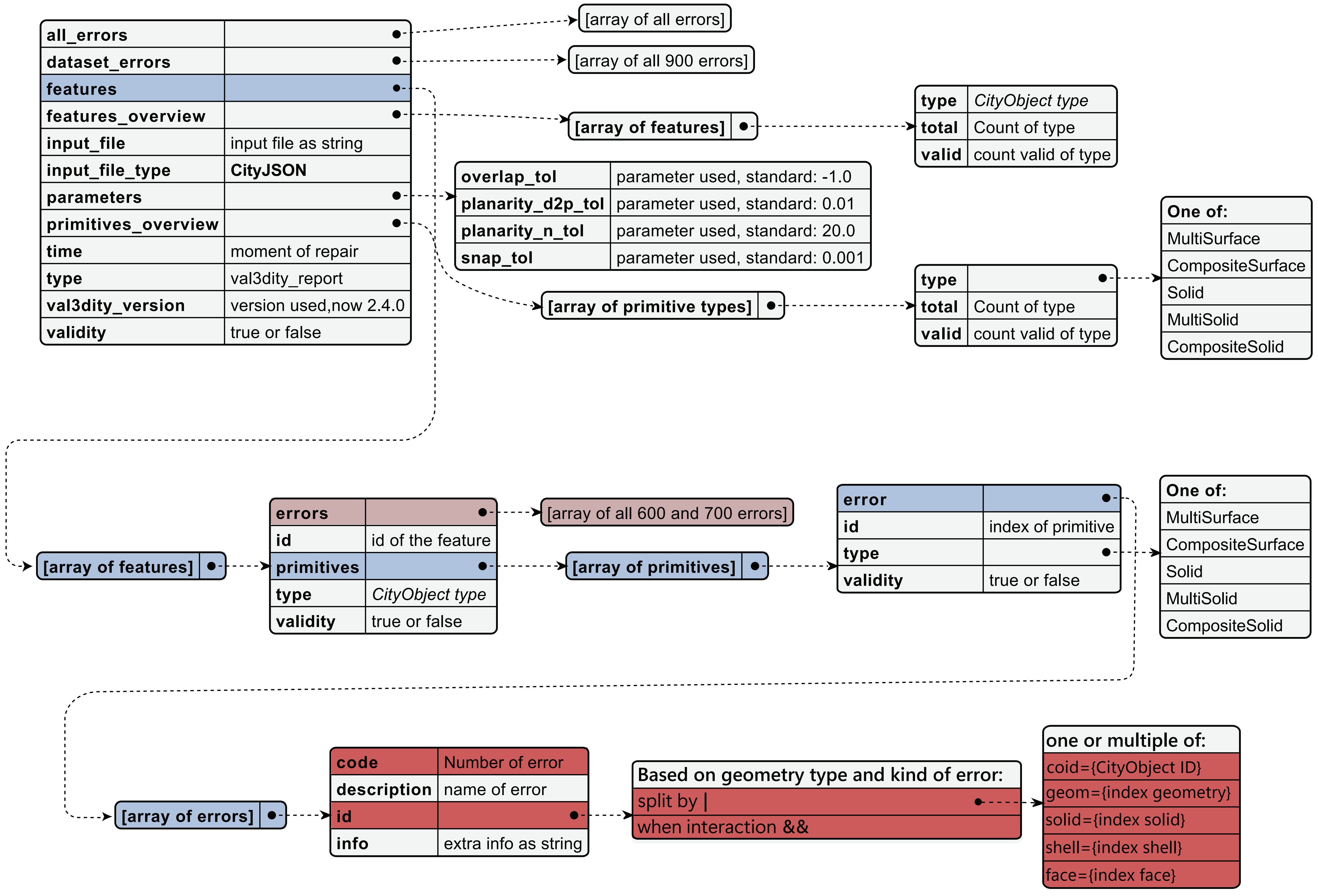
601 BUILDINGPARTS_OVERLAP
609 CITYOBJECT_HAS_NO_GEOMETRY

Others

901 INVALID_INPUT_FILE
902 EMPTY_PRIMITIVE
903 WRONG_INPUT_PARAMETERS
999 UNKNOWN_ERROR

Validate

Repair



all_errors	→ [array of all errors]
dataset_errors	→ [array of all 900 errors]
features	→ [array of features]
features_overview	→ [array of features]
input_file	input file as string
input_file_type	CityJSON
parameters	→
primitives_overview	→ [array of primitive types]
time	moment of repair
type	val3dity_report
val3dity_version	version used,now 2.4.0
validity	true or false

overlap_tol	parameter used, standard: -1.0
planarity_d2p_tol	parameter used, standard: 0.01
planarity_n_tol	parameter used, standard: 20.0
snap_tol	parameter used, standard: 0.001

[array of features]

type	CityObject type
total	Count of type
valid	count valid of type

[array of primitive types]

type	→
total	Count of type
valid	count valid of type

One of:
- MultiSurface
- CompositeSurface
- Solid
- MultiSolid
- CompositeSolid

[array of features]

errors	→ [array of all 600 and 700 errors]
id	id of the feature
primitives	→ [array of primitives]
type	CityObject type
validity	true or false

[array of primitives]

error	→
id	index of primitive
type	→
validity	true or false

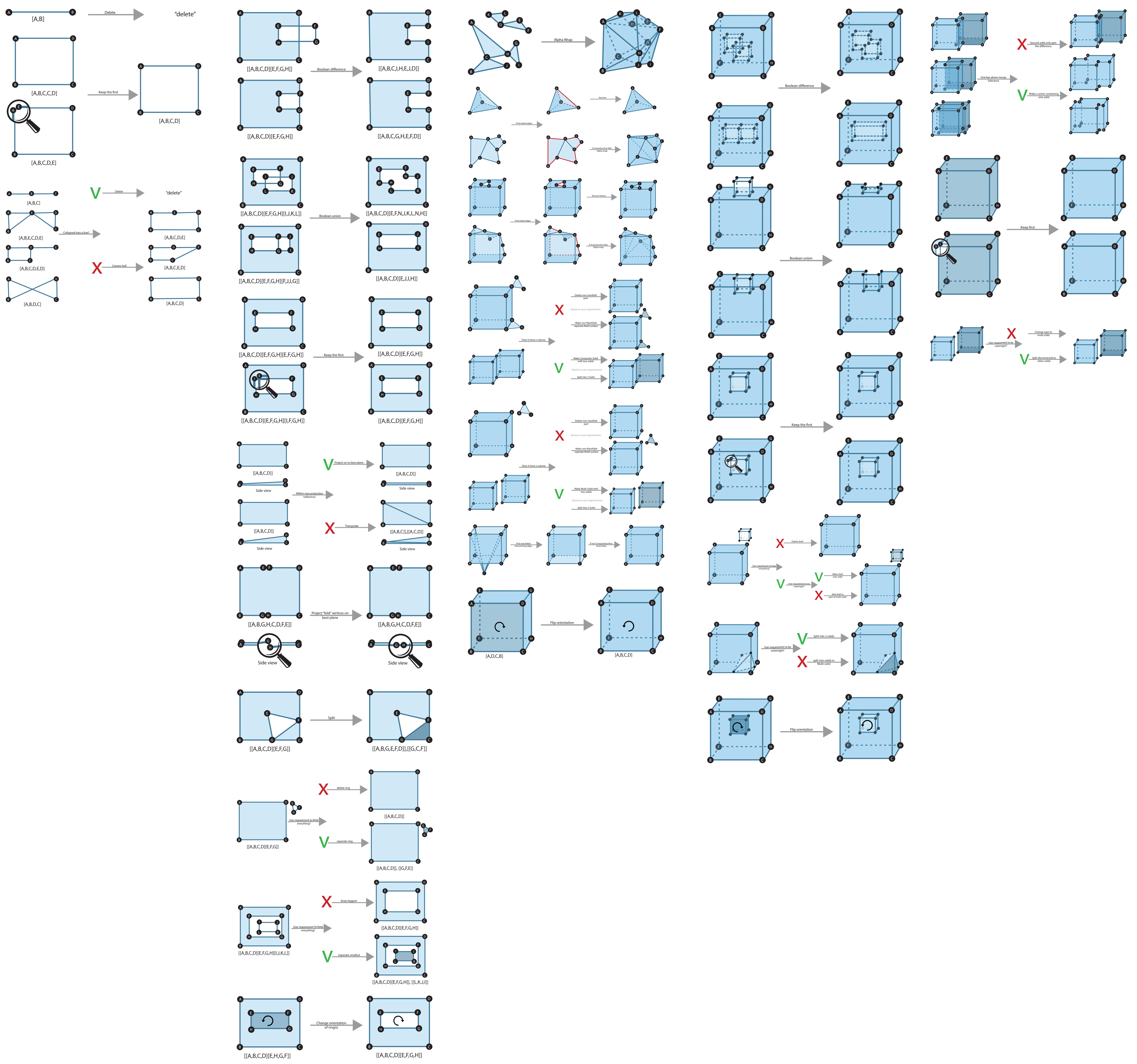
One of:
- MultiSurface
- CompositeSurface
- Solid
- MultiSolid
- CompositeSolid

[array of errors]

code	Number of error
description	name of error
id	→
info	extra info as string

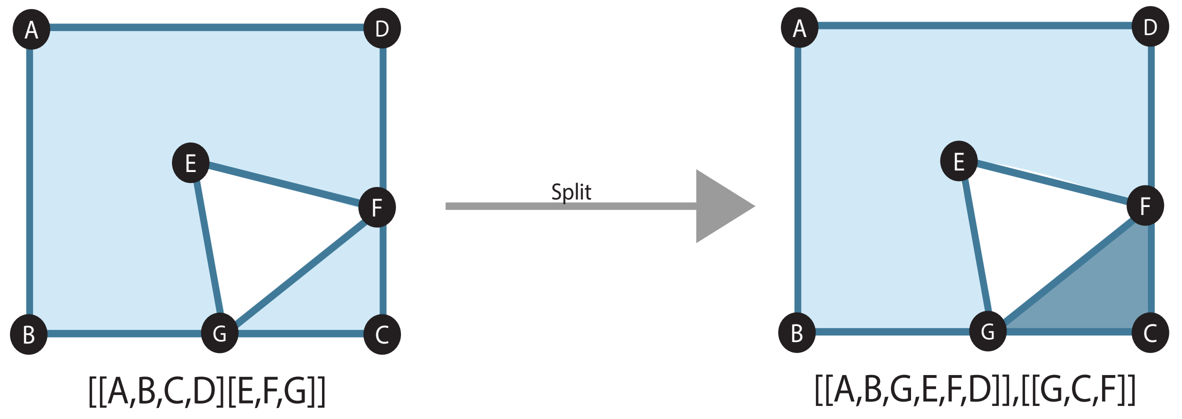
Based on geometry type and kind of error:
- split by |
- when interaction &&

one or multiple of:
- coid={CityObject ID}
- geom={index geometry}
- solid={index solid}
- shell={index shell}
- face={index face}

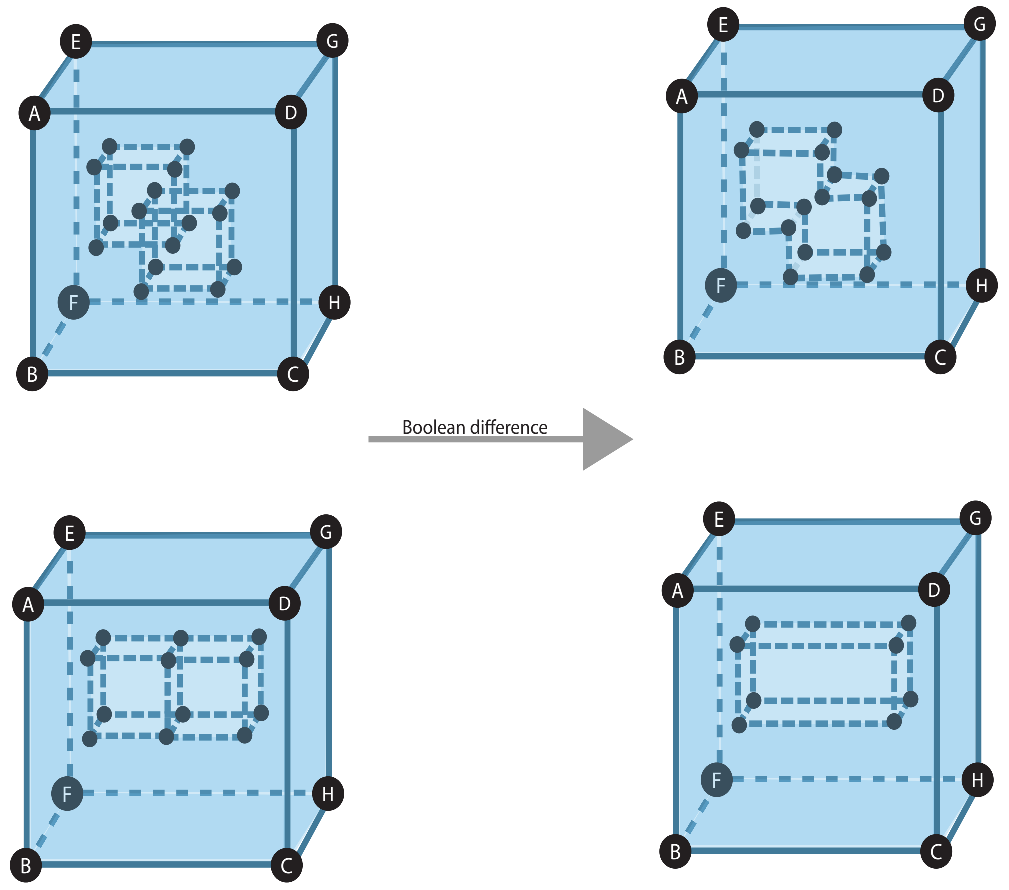


val3dity

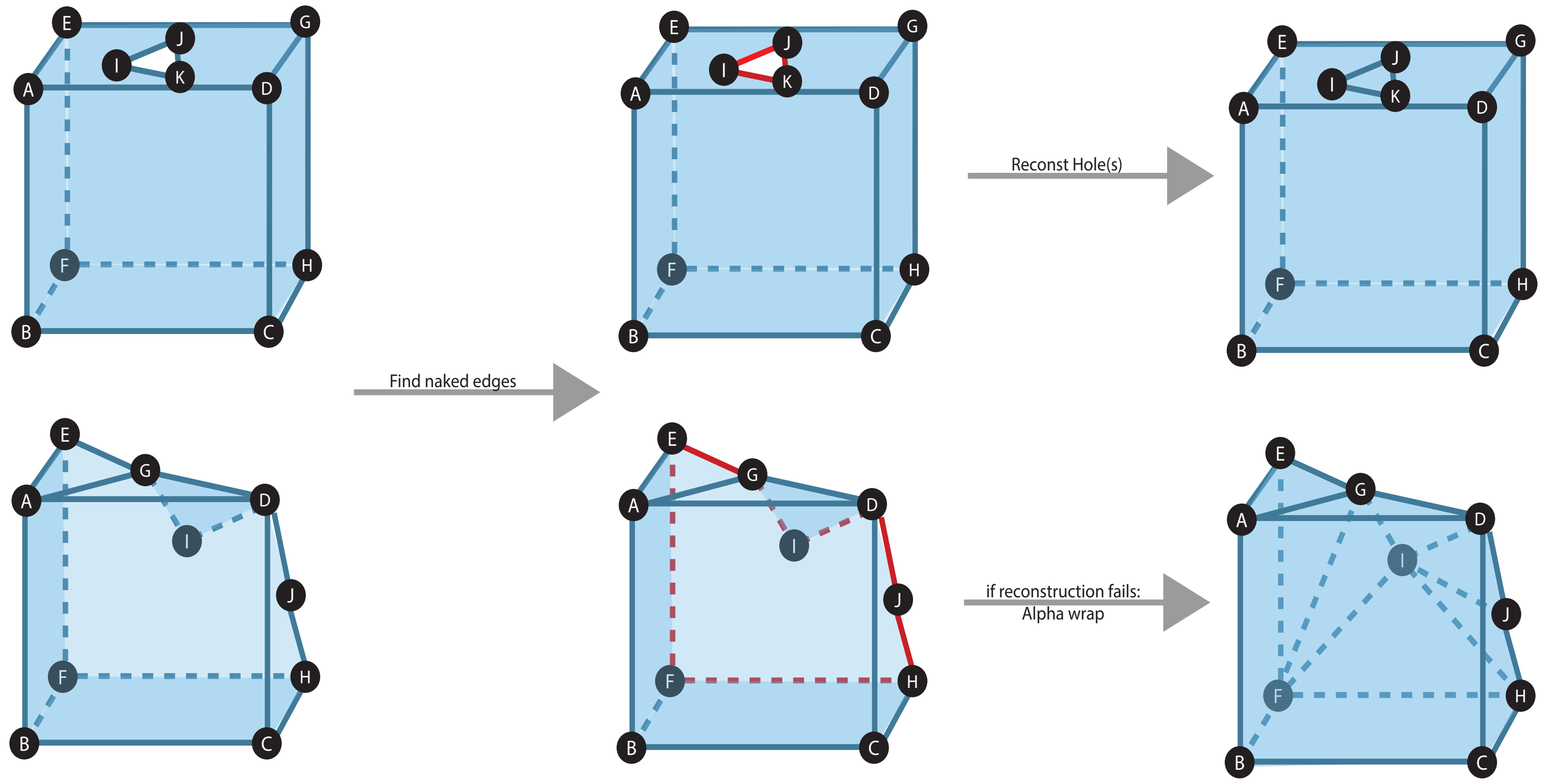
Example repairs



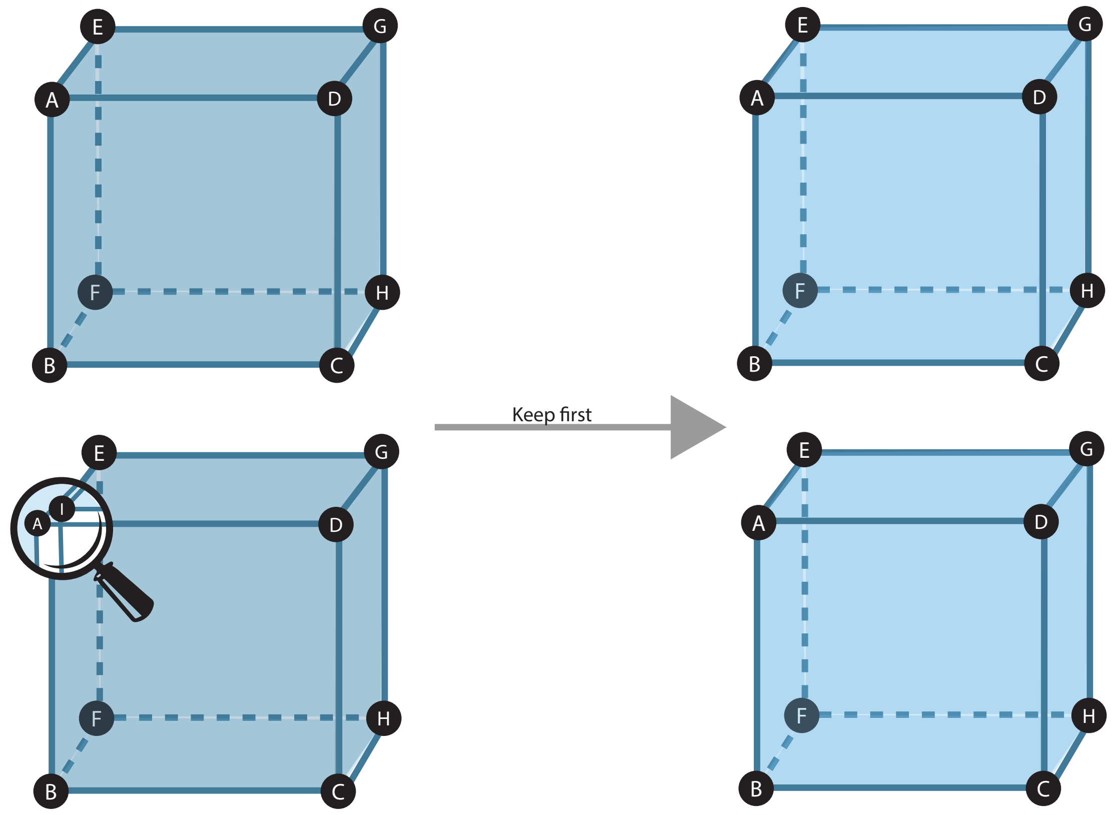
Polygon interior disconnected



Intersection shells

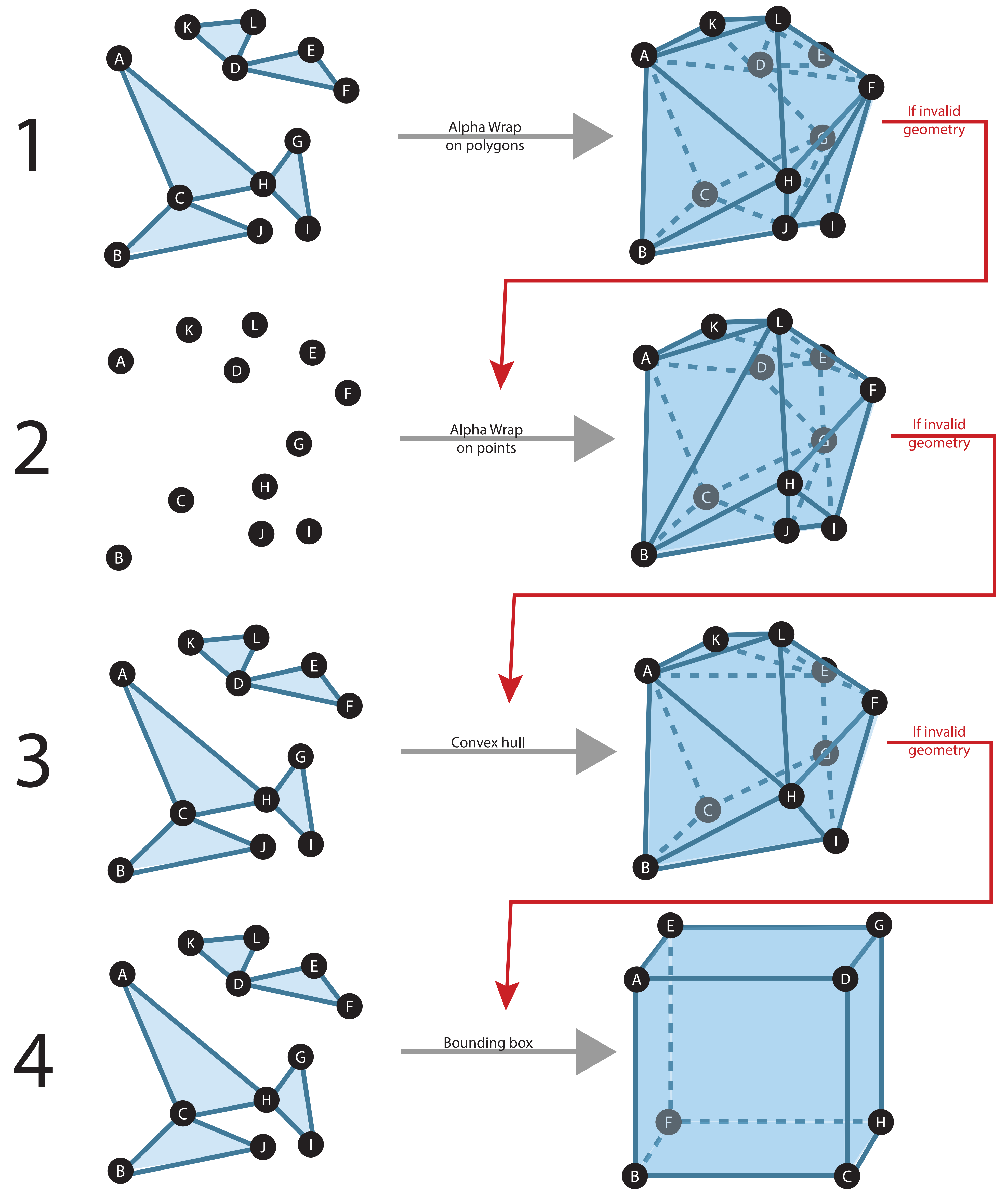


Shell not closed



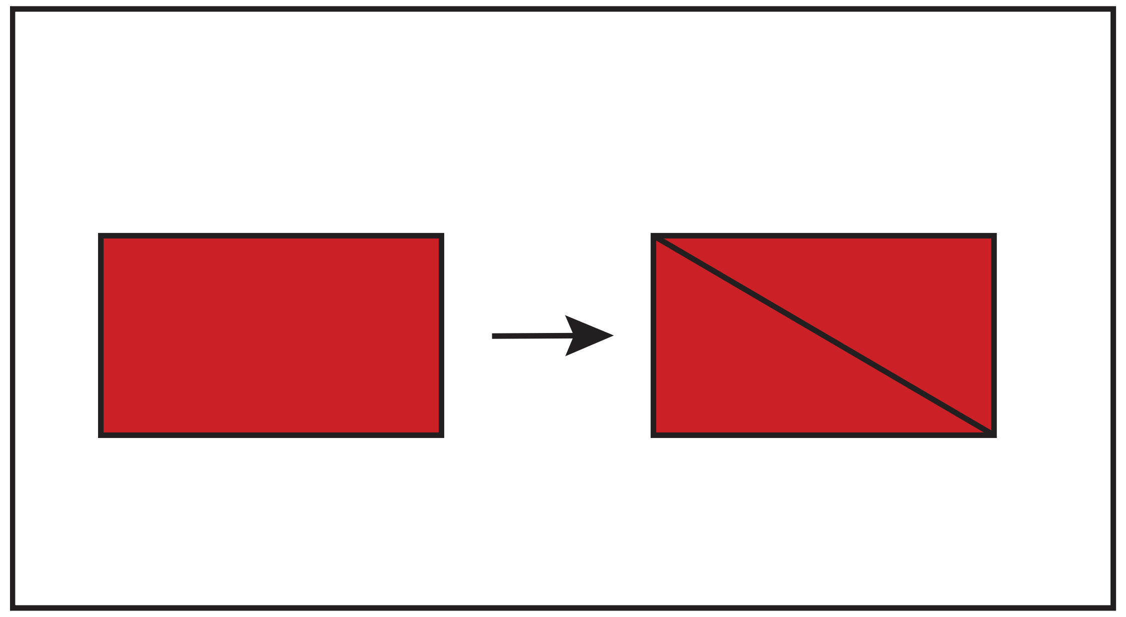
Duplicate solids

Global safety net

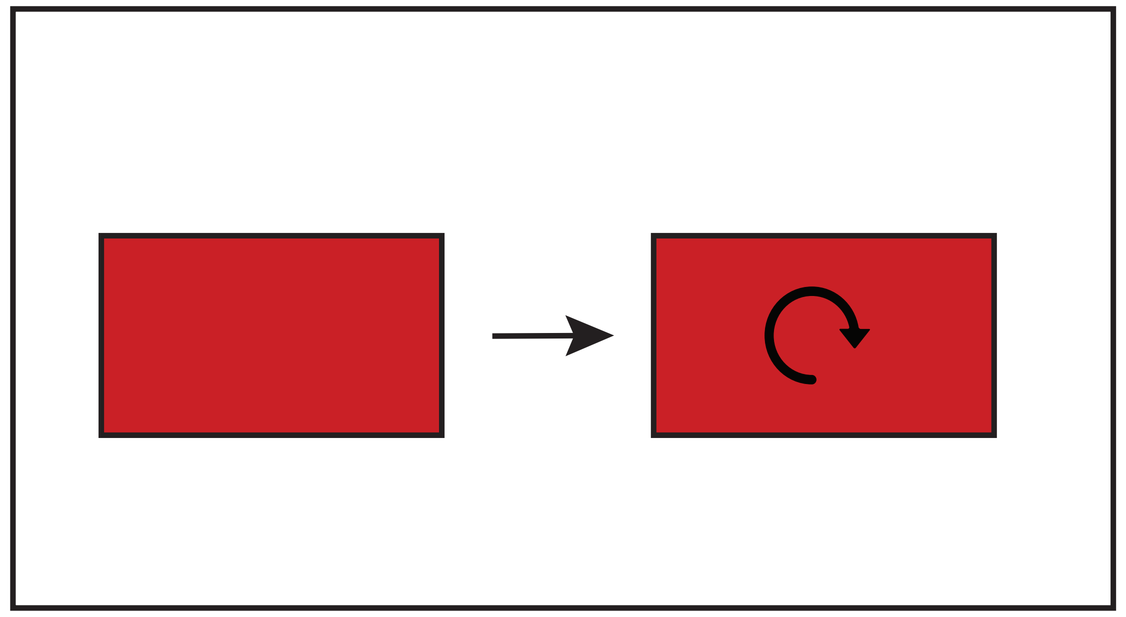


How to preserve semantics during automatic repair?

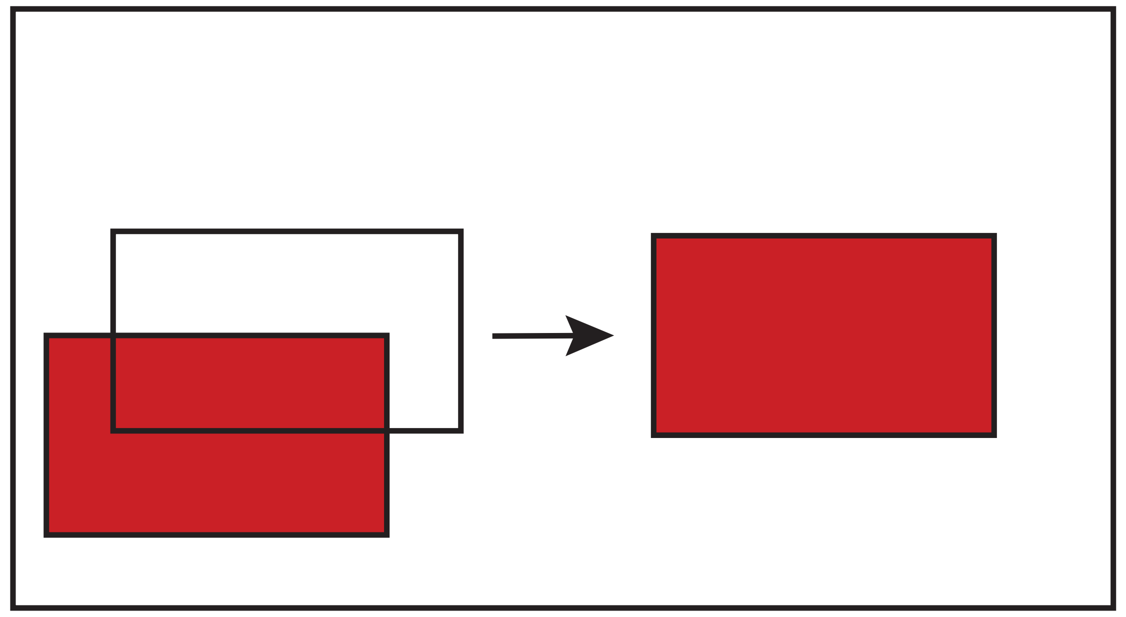
Preservation of semantics



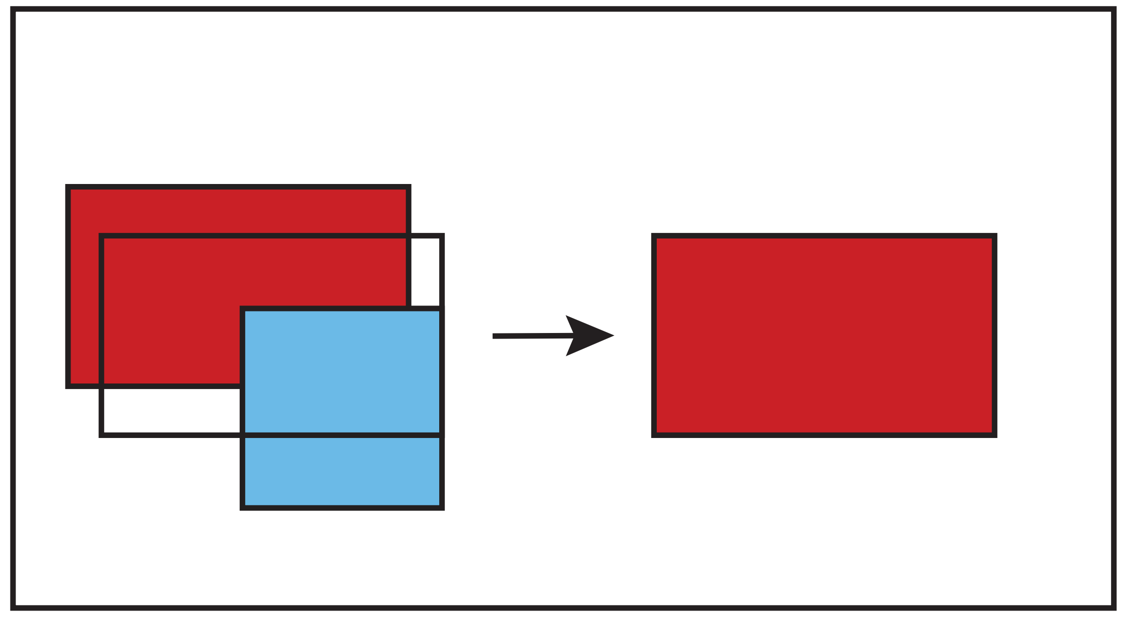
split



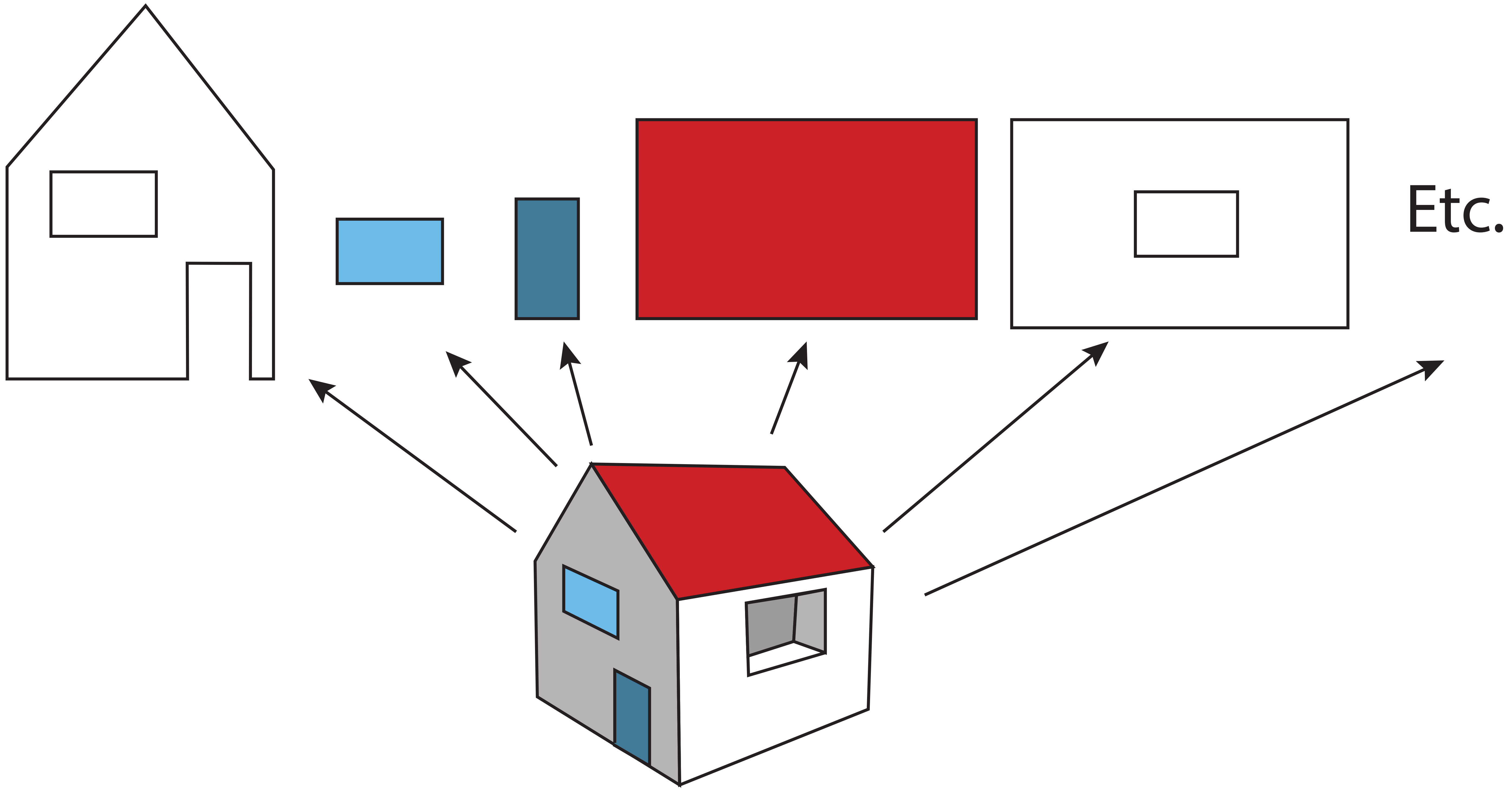
Flipped



Overlap

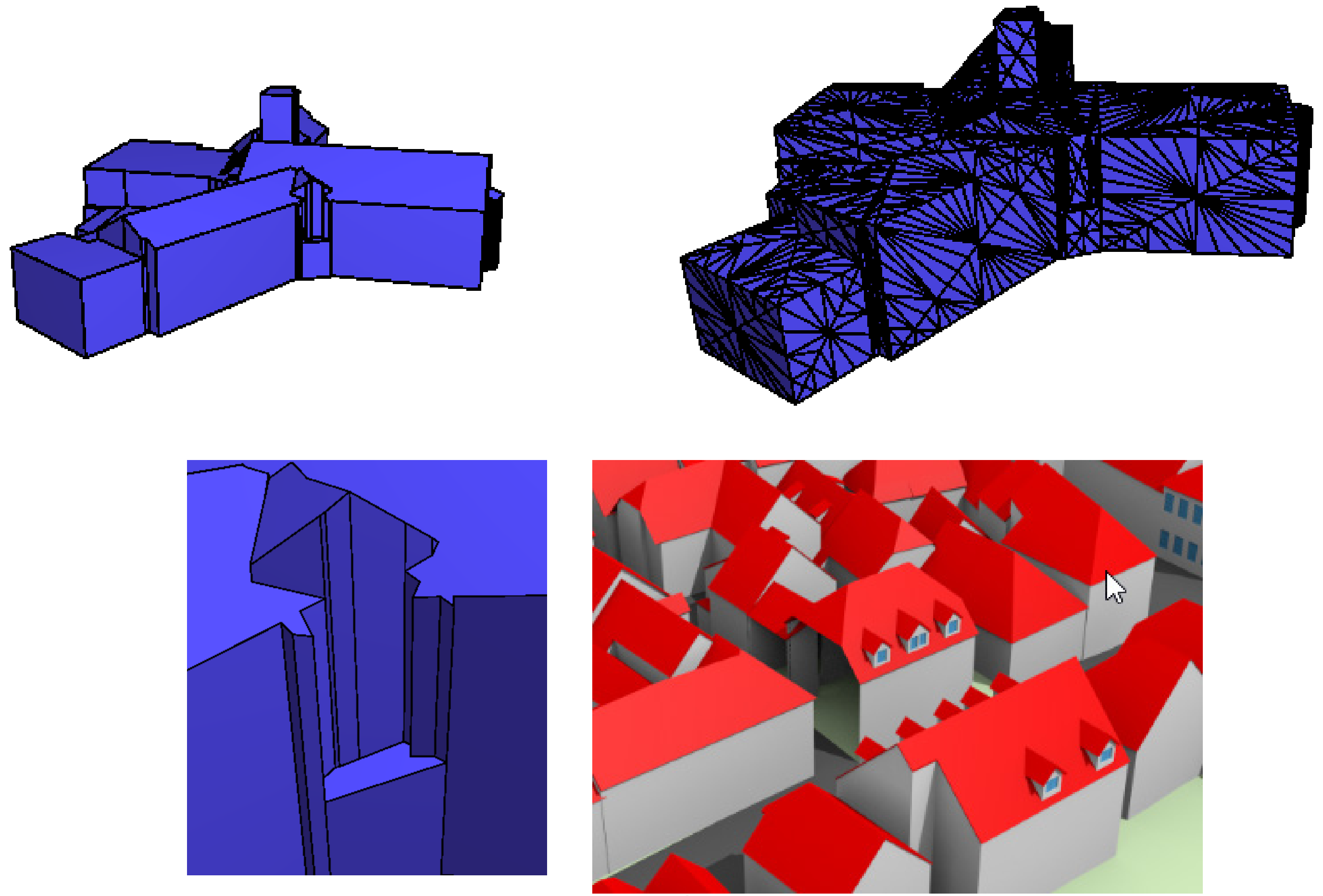


Merged

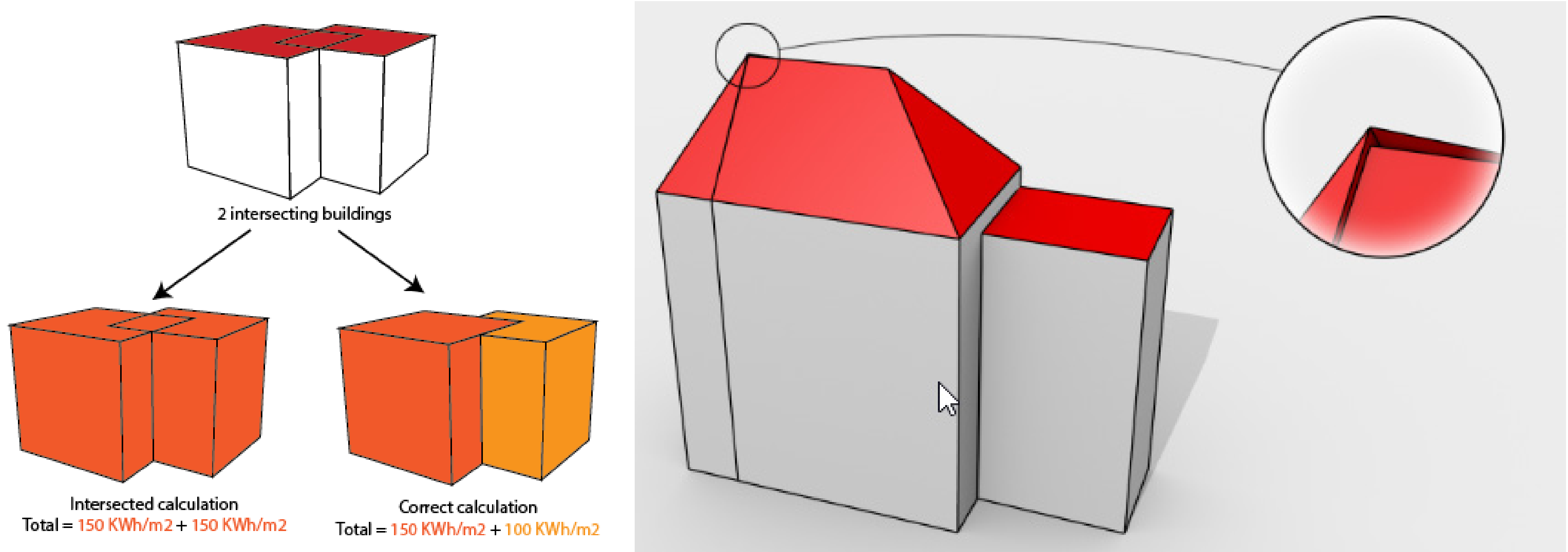


How to achieve validity for different use cases?

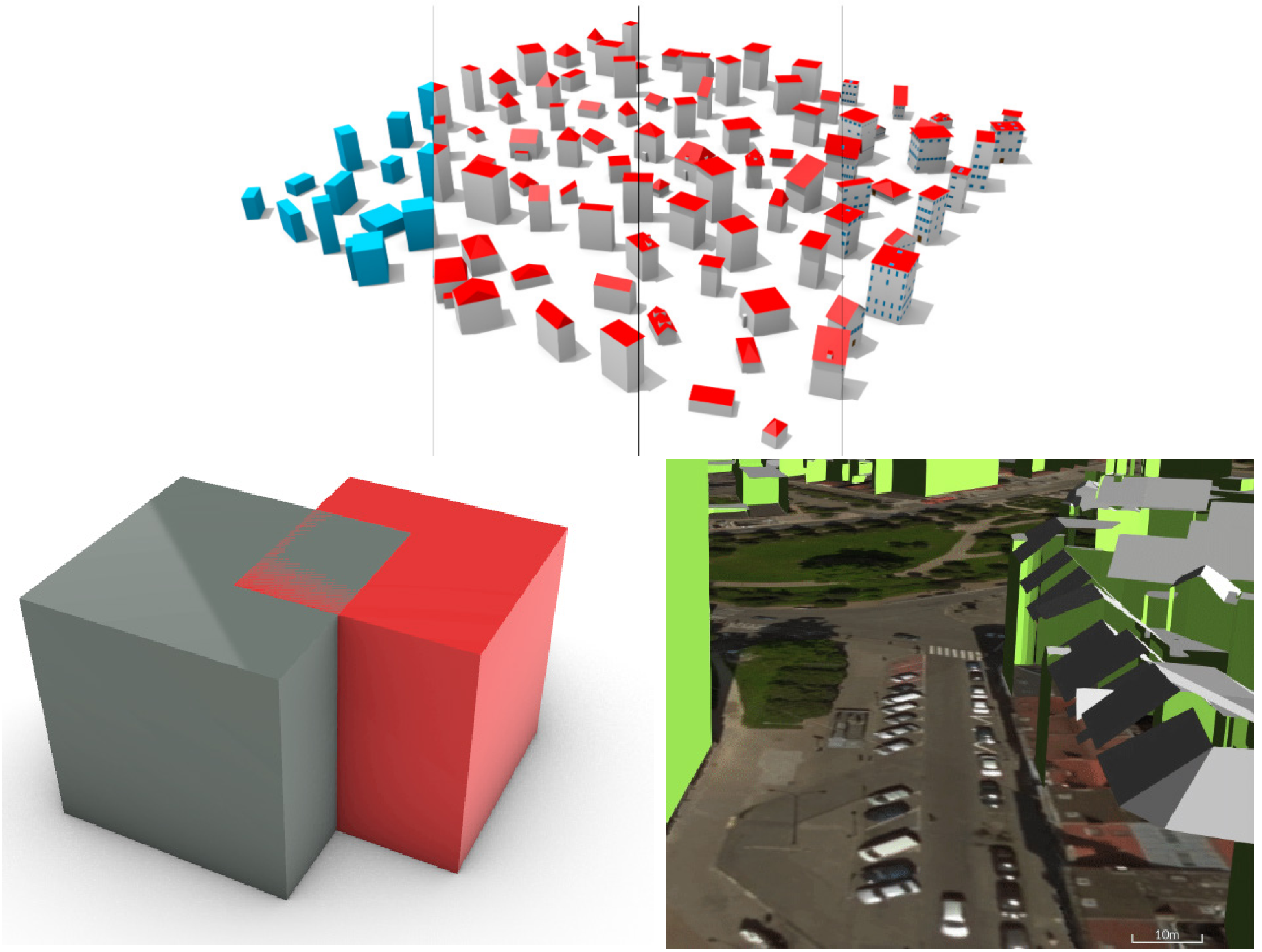
Scope of the use cases



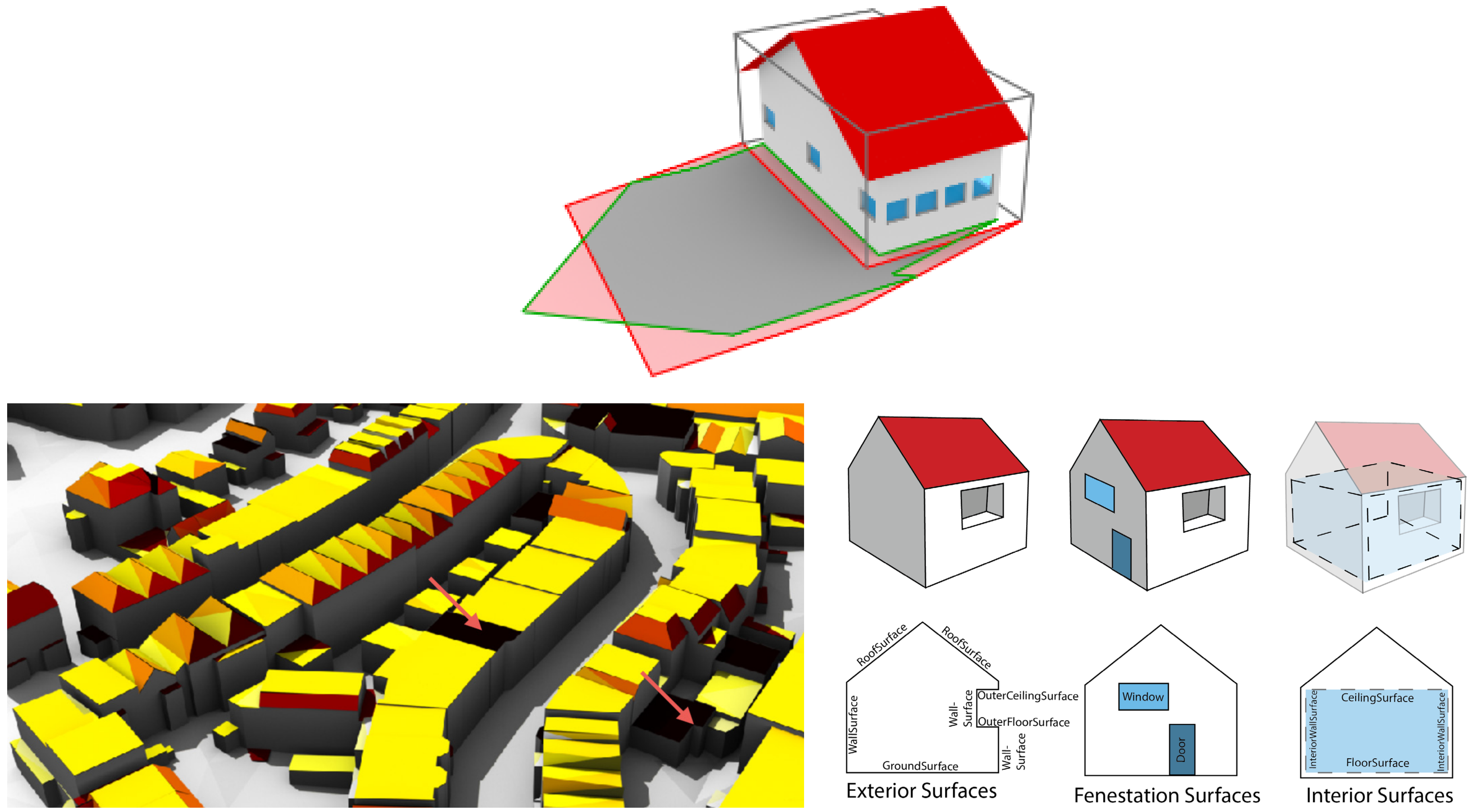
Computational fluid dynamics



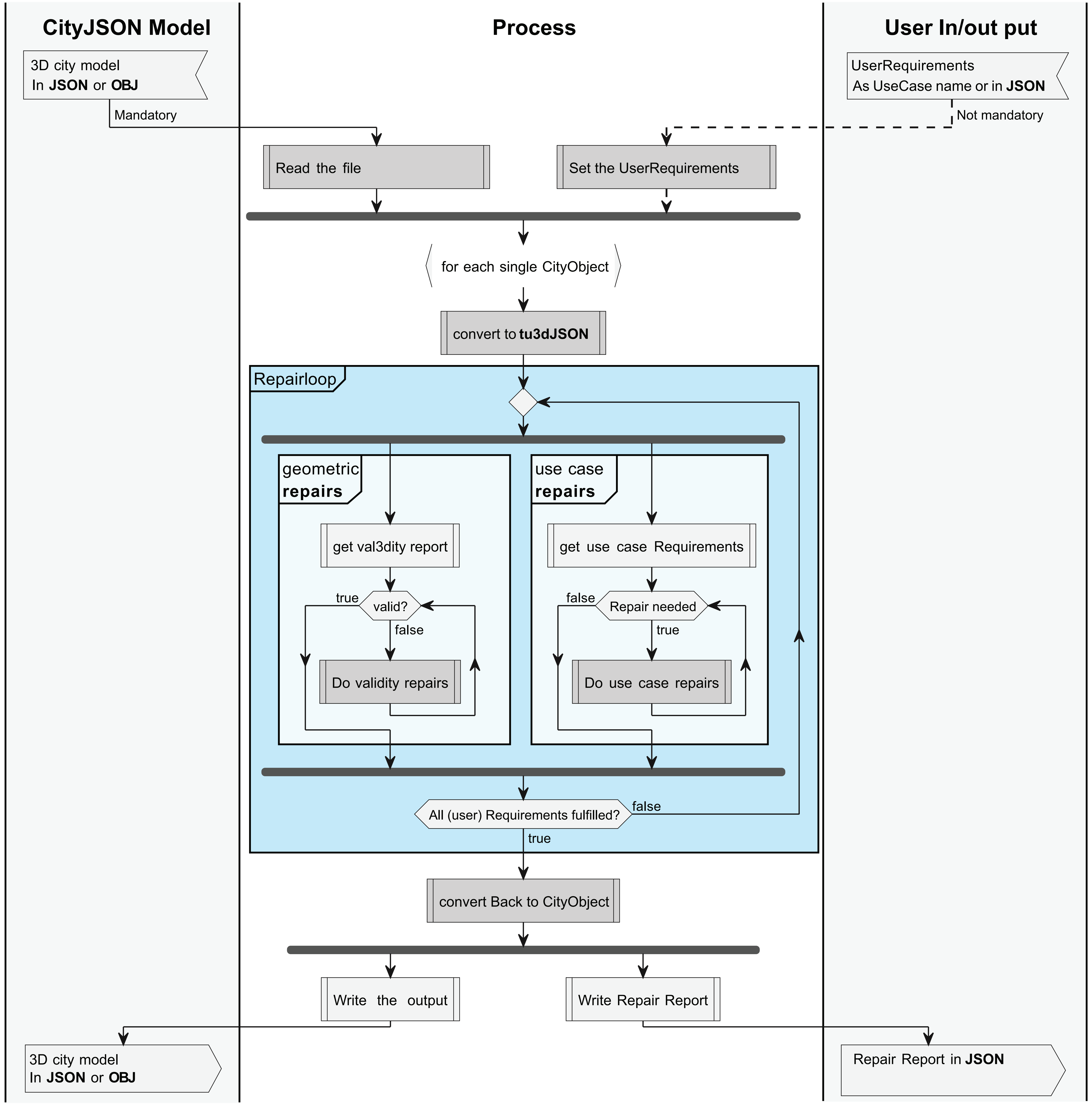
Energy demand



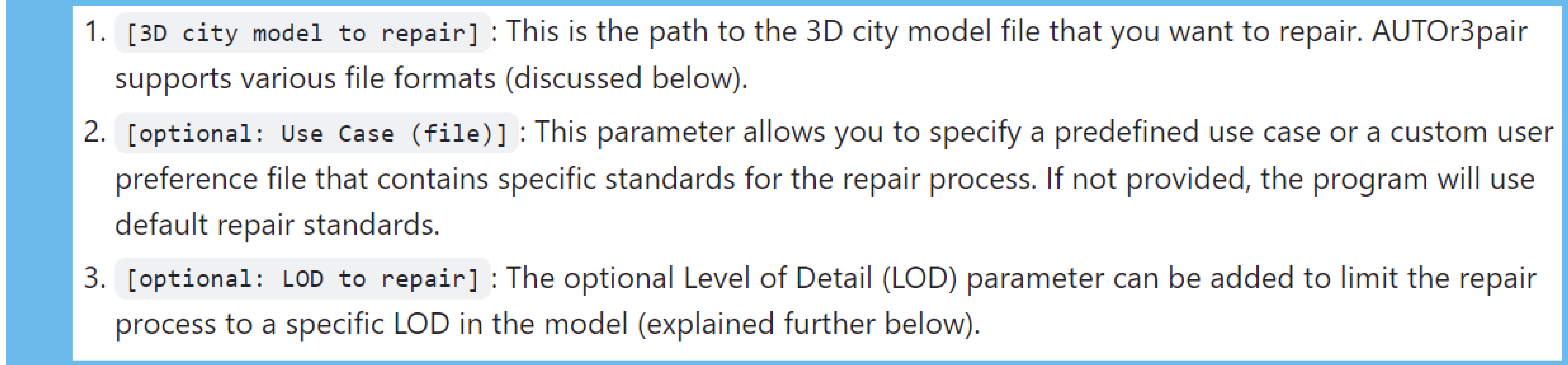
Visualization



Solar power estimation

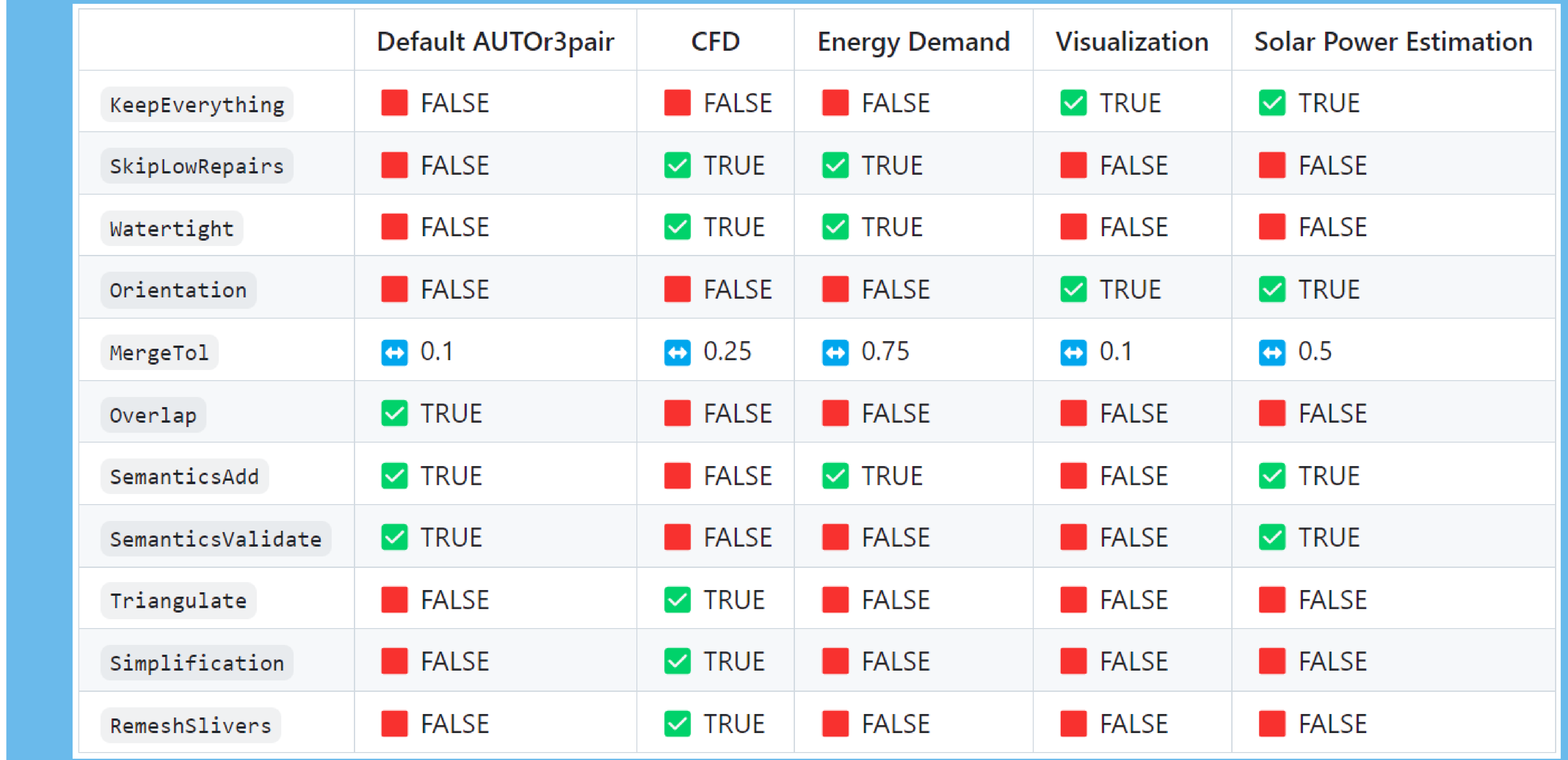


Input



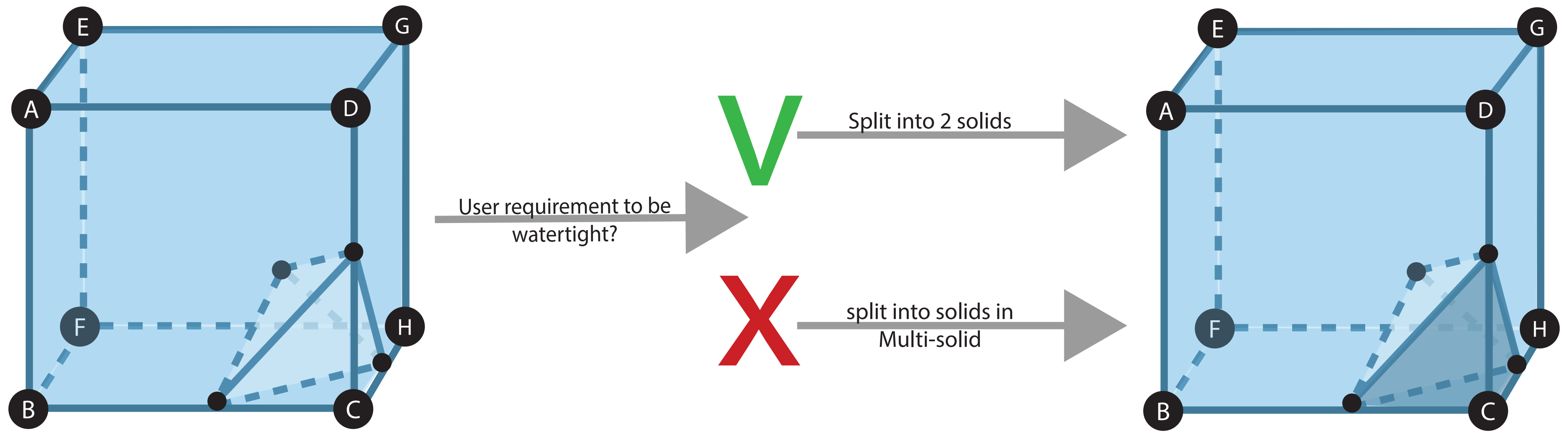
1. `[3D city model to repair]` : This is the path to the 3D city model file that you want to repair. AUTOr3pair supports various file formats (discussed below).
2. `[optional: Use Case (file)]` : This parameter allows you to specify a predefined use case or a custom user preference file that contains specific standards for the repair process. If not provided, the program will use default repair standards.
3. `[optional: LOD to repair]` : The optional Level of Detail (LOD) parameter can be added to limit the repair process to a specific LOD in the model (explained further below).

Use case



	Default AUTOr3pair	CFD	Energy Demand	Visualization	Solar Power Estimation
`KeepEverything`	🟥 FALSE	🟥 FALSE	🟥 FALSE	✅ TRUE	✅ TRUE
`SkipLowRepairs`	🟥 FALSE	✅ TRUE	✅ TRUE	🟥 FALSE	🟥 FALSE
`Watertight`	🟥 FALSE	✅ TRUE	✅ TRUE	🟥 FALSE	🟥 FALSE
`Orientation`	🟥 FALSE	🟥 FALSE	🟥 FALSE	✅ TRUE	✅ TRUE
`MergeTol`	↔ 0.1	↔ 0.25	↔ 0.75	↔ 0.1	↔ 0.5
`Overlap`	✅ TRUE	🟥 FALSE	🟥 FALSE	🟥 FALSE	🟥 FALSE
`SemanticsAdd`	✅ TRUE	🟥 FALSE	✅ TRUE	🟥 FALSE	✅ TRUE
`SemanticsValidate`	✅ TRUE	🟥 FALSE	🟥 FALSE	🟥 FALSE	✅ TRUE
`Triangulate`	🟥 FALSE	✅ TRUE	🟥 FALSE	🟥 FALSE	🟥 FALSE
`Simplification`	🟥 FALSE	✅ TRUE	🟥 FALSE	🟥 FALSE	🟥 FALSE
`RemeshSlivers`	🟥 FALSE	✅ TRUE	🟥 FALSE	🟥 FALSE	🟥 FALSE

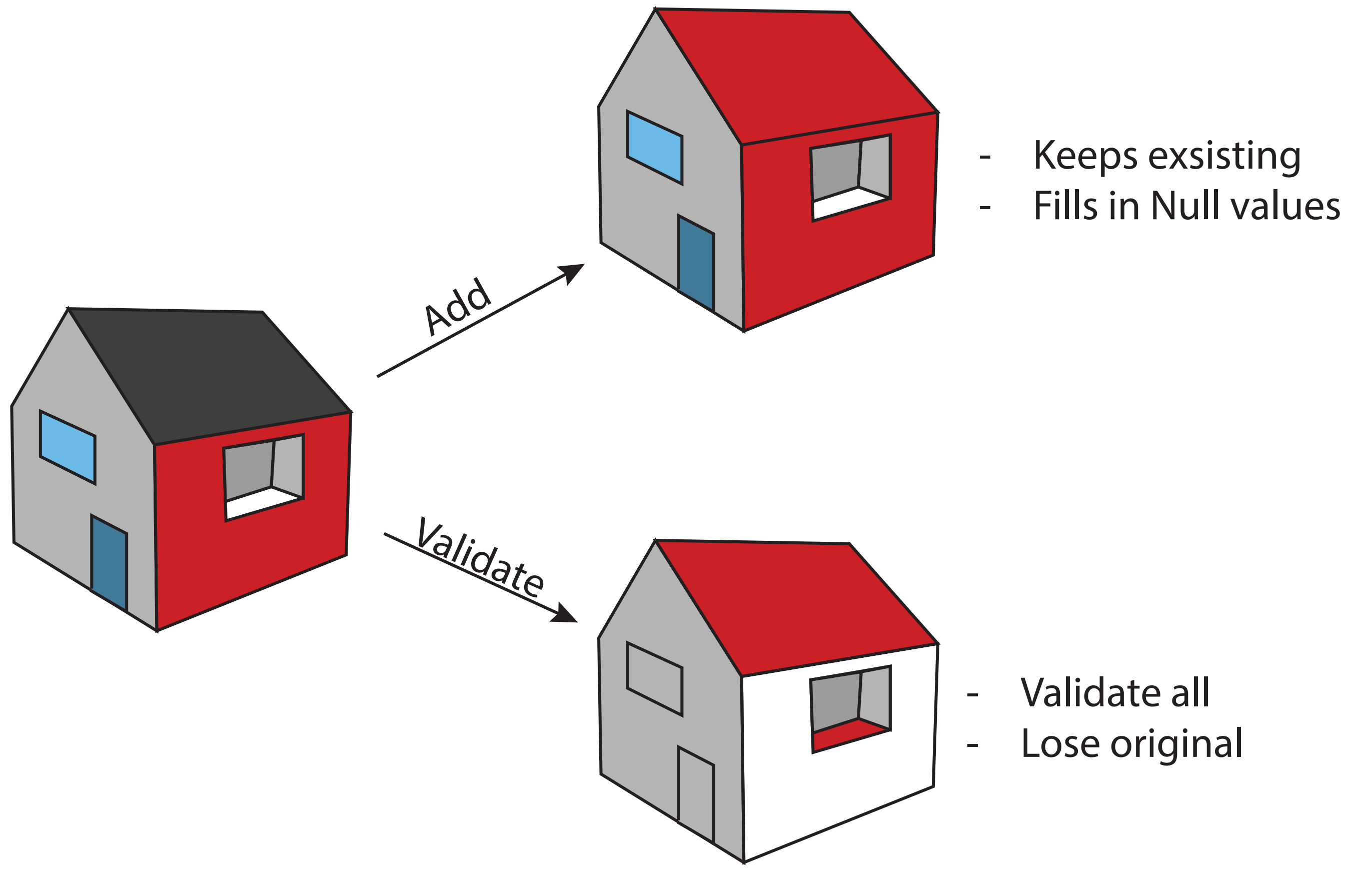
Examples use case repair


User requirement to be watertight?

V Split into 2 solids

X split into solids in Multi-solid

Solid interior disconnected

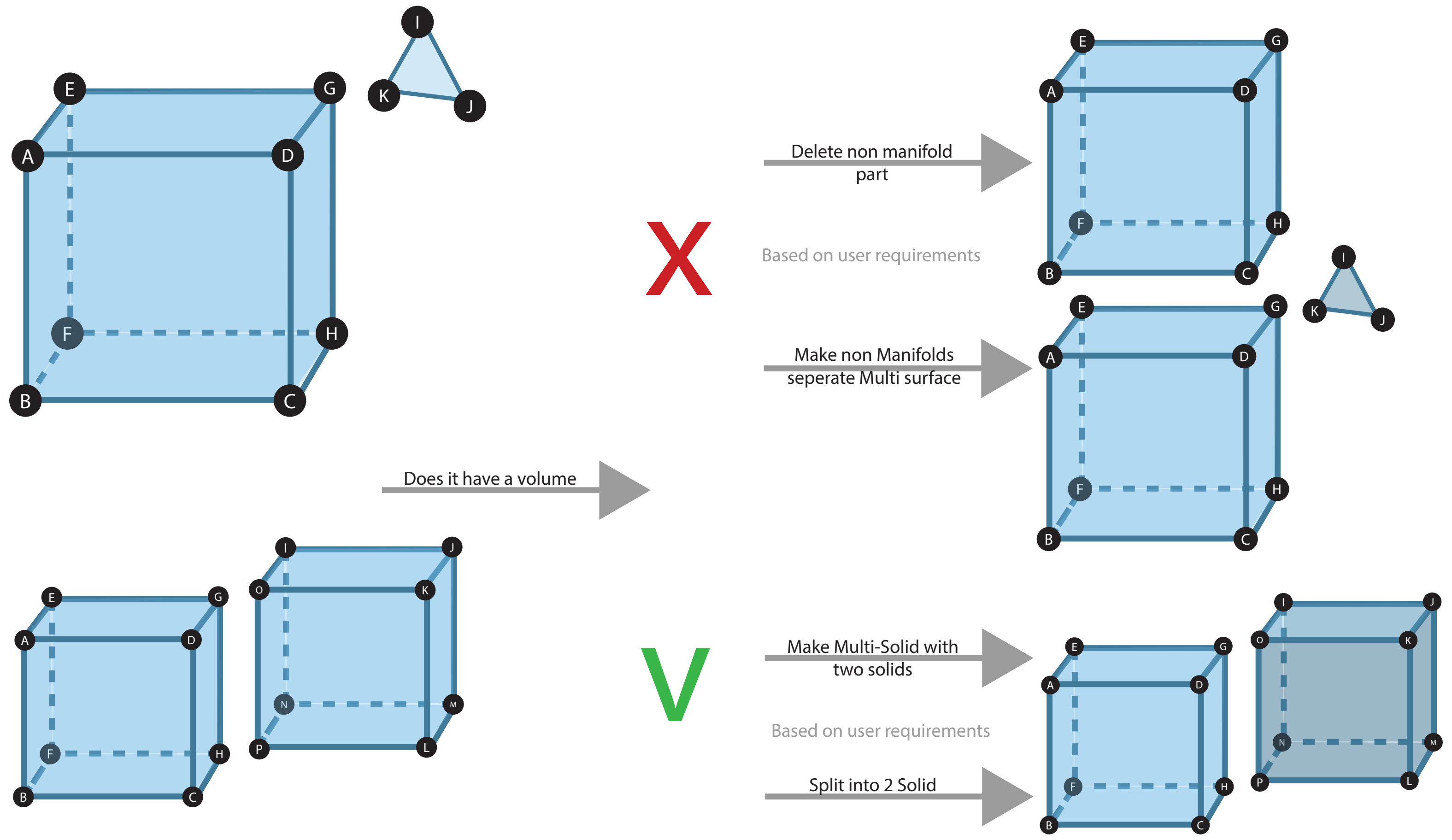

Add

- Keeps exsisting
- Fills in Null values

Validate

- Validate all
- Lose original

Semantic parameters


Does it have a volume

X

Delete non manifold part

Based on user requirements

Make non Manifolds seperate Multi surface

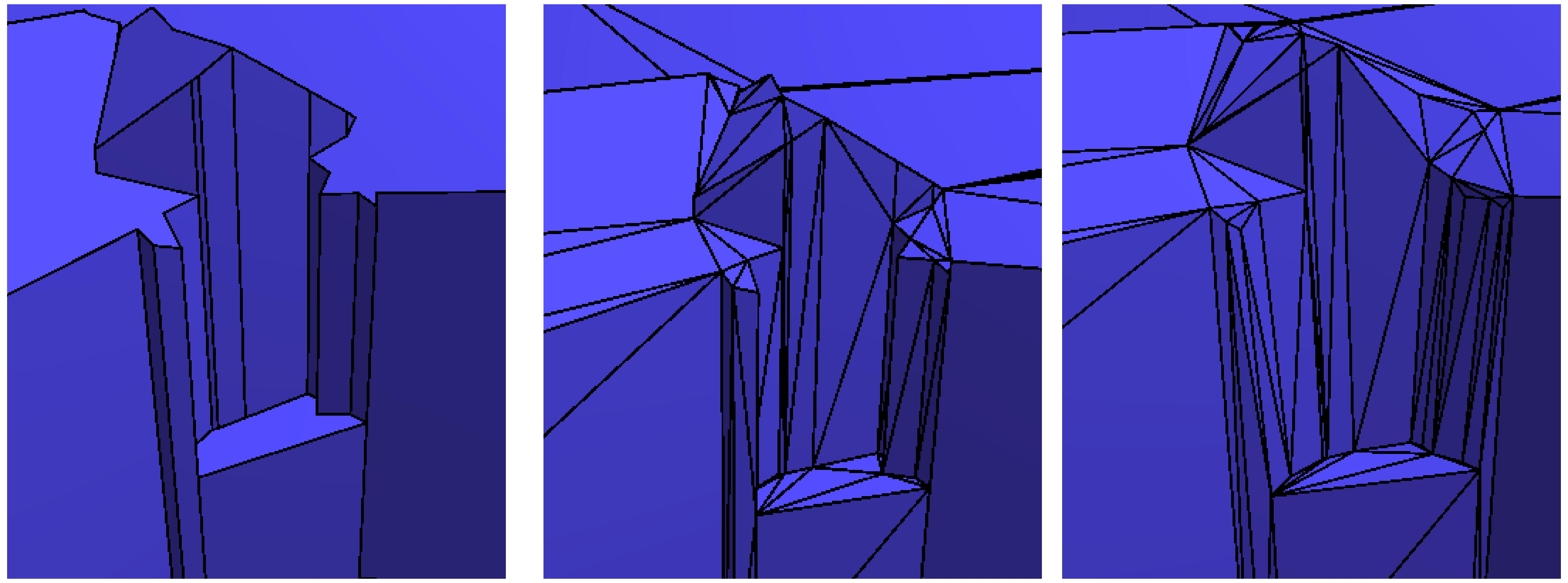
V

Make Multi-Solid with two solids

Based on user requirements

Split into 2 Solid

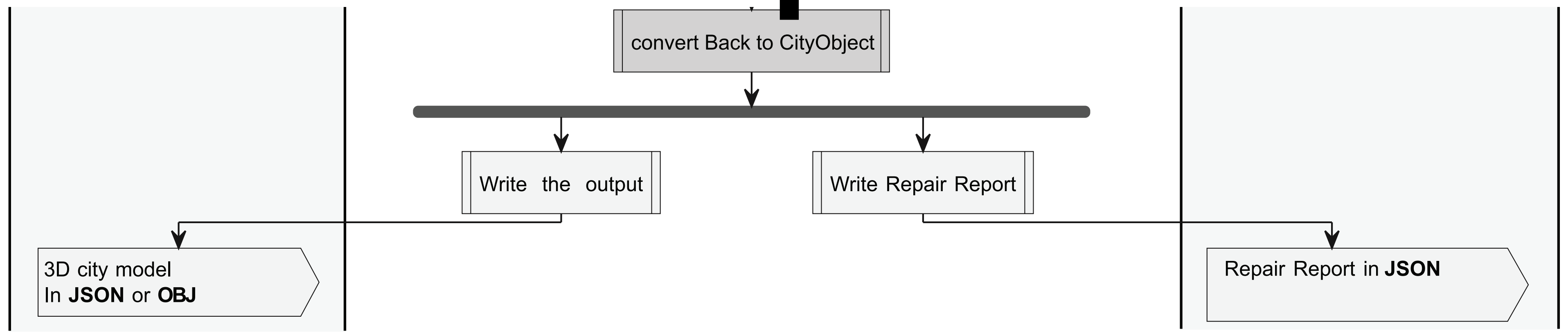
Multiple connected components



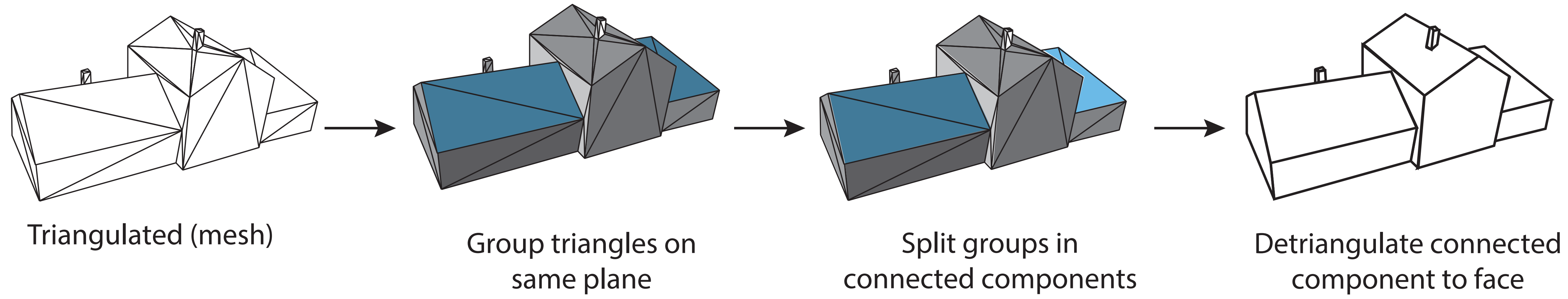
Simplification parameters

What degree of validity can be achieved?

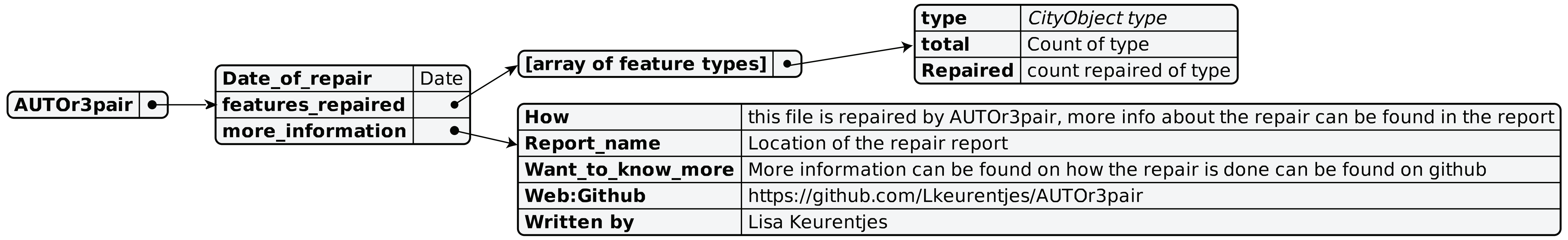
Output



Post processing - detriangulation


Triangulated (mesh)
Group triangles on same plane
Split groups in connected components
Detriangulate connected component to face

AUTOr3pair key



AUTOr3pair	●

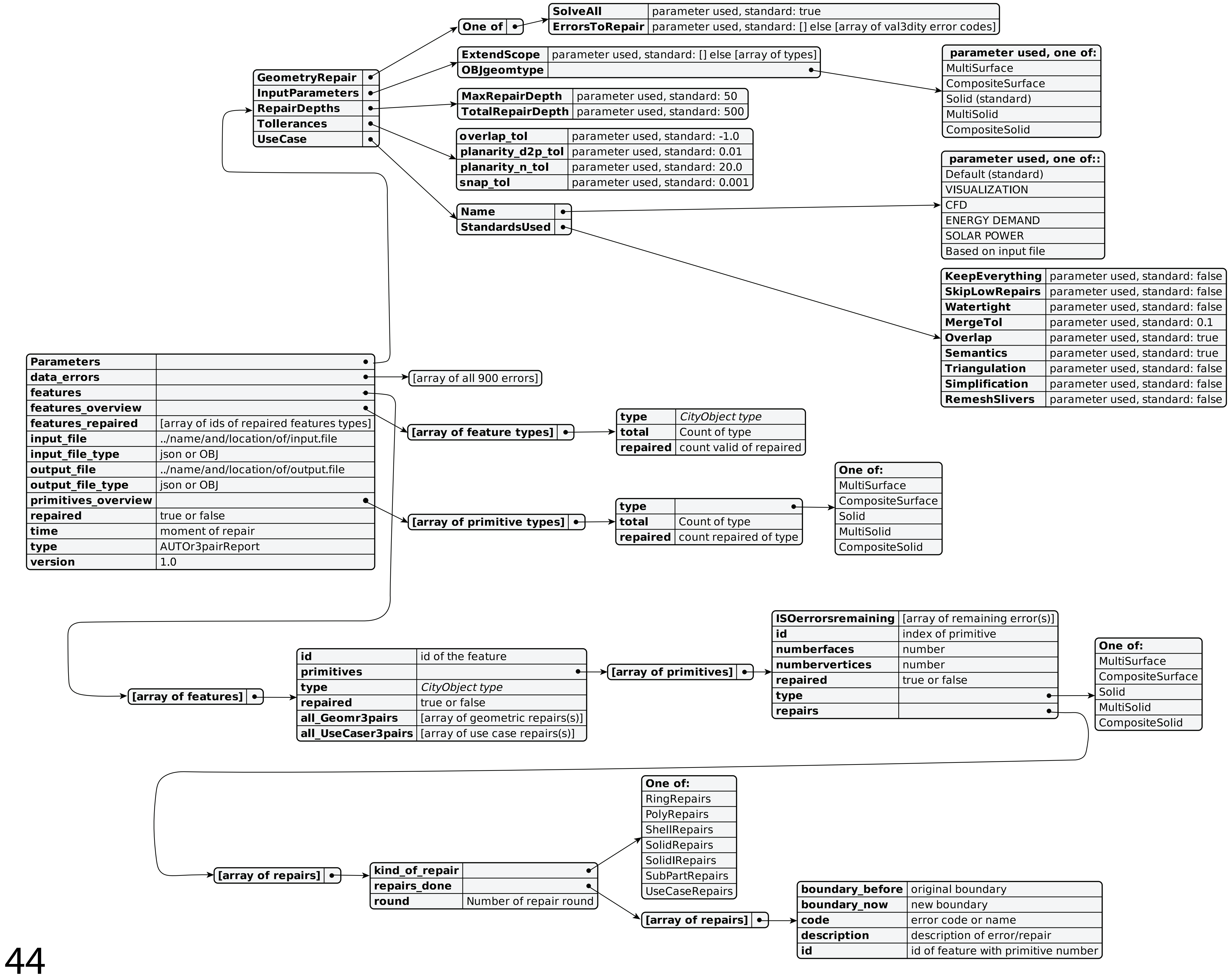
Date_of_repair	Date
features_repaired	●
more_information	●

[array of feature types]	●

type	*CityObject type*
total	Count of type
Repaired	count repaired of type

How	this file is repaired by AUTOr3pair, more info about the repair can be found in the report
Report_name	Location of the repair report
Want_to_know_more	More information can be found on how the repair is done can be found on github
Web:Github	https://github.com/Lkeurentjes/AUTOr3pair
Written by	Lisa Keurentjes

Repair report



Parameters	●
data_errors	●
features	●
features_overview	●
features_repaired	[array of ids of repaired features types]
input_file	../name/and/location/of/input.file
input_file_type	json or OBJ
output_file	../name/and/location/of/output.file
output_file_type	json or OBJ
primitives_overview	●
repaired	true or false
time	moment of repair
type	AUTOr3pairReport
version	1.0

GeometryRepair	●
InputParameters	●
RepairDepths	●
Tollerances	●
UseCase	●

One of	●

SolveAll	parameter used, standard: true
ErrorsToRepair	parameter used, standard: [] else [array of val3dity error codes]

ExtendScope	parameter used, standard: [] else [array of types]
OBJgeomtype	●

parameter used, one of:
MultiSurface
CompositeSurface
Solid (standard)
MultiSolid
CompositeSolid

MaxRepairDepth	parameter used, standard: 50
TotalRepairDepth	parameter used, standard: 500

overlap_tol	parameter used, standard: -1.0
planarity_d2p_tol	parameter used, standard: 0.01
planarity_n_tol	parameter used, standard: 20.0
snap_tol	parameter used, standard: 0.001

Name	●
StandardsUsed	●

parameter used, one of::
Default (standard)
VISUALIZATION
CFD
ENERGY DEMAND
SOLAR POWER
Based on input file

KeepEverything	parameter used, standard: false
SkipLowRepairs	parameter used, standard: false
Watertight	parameter used, standard: false
MergeTol	parameter used, standard: 0.1
Overlap	parameter used, standard: true
Semantics	parameter used, standard: true
Triangulation	parameter used, standard: false
Simplification	parameter used, standard: false
RemeshSlivers	parameter used, standard: false

[array of all 900 errors]

[array of feature types]	●

type	*CityObject type*
total	Count of type
repaired	count valid of repaired

[array of primitive types]	●

type	●
total	Count of type
repaired	count repaired of type

One of:
MultiSurface
CompositeSurface
Solid
MultiSolid
CompositeSolid

[array of features]	●

id	id of the feature
primitives	●
type	*CityObject type*
repaired	true or false
all_Geomr3pairs	[array of geometric repairs(s)]
all_UseCaser3pairs	[array of use case repairs(s)]

[array of primitives]	●

ISOerrorsremaining	[array of remaining error(s)]
id	index of primitive
numberfaces	number
numbervertices	number
repaired	true or false
type	●
repairs	●

One of:
MultiSurface
CompositeSurface
Solid
MultiSolid
CompositeSolid

[array of repairs]	●

kind_of_repair	●
repairs_done	●
round	Number of repair round

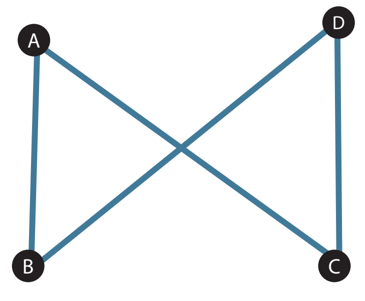
One of:
RingRepairs
PolyRepairs
ShellRepairs
SolidRepairs
SolidIRepairs
SubPartRepairs
UseCaseRepairs

[array of repairs]	●

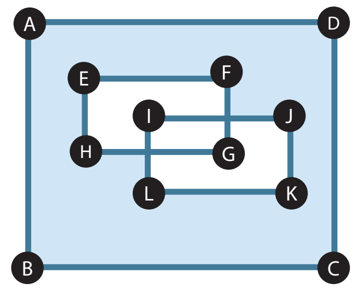
boundary_before	original boundary
boundary_now	new boundary
code	error code or name
description	description of error/repair
id	id of feature with primitive number

Demo

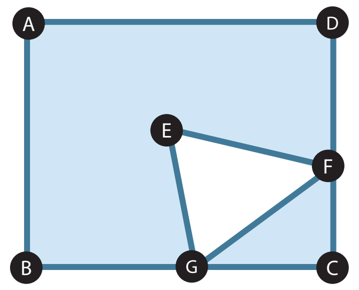
Unit tests



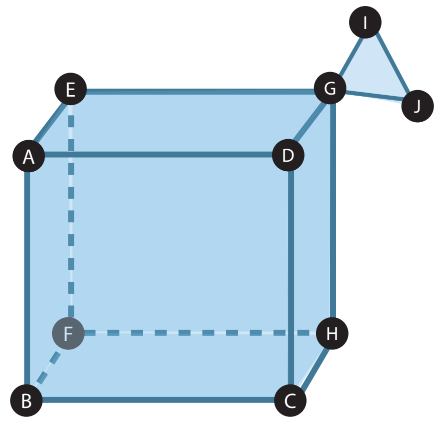
104



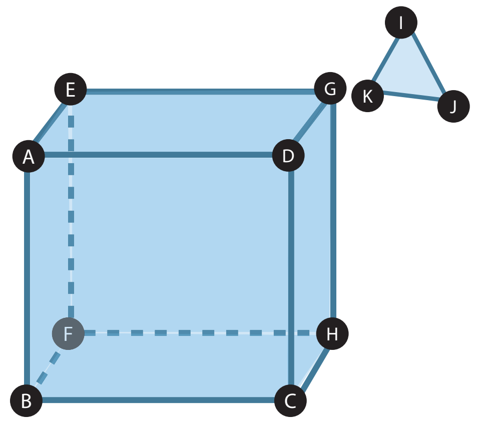
201



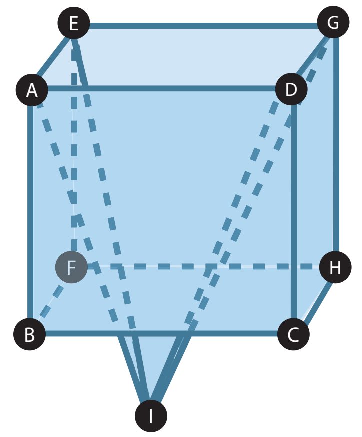
205



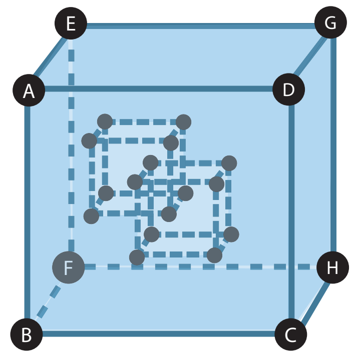
303



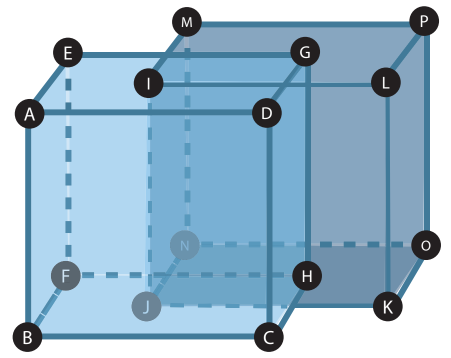
305



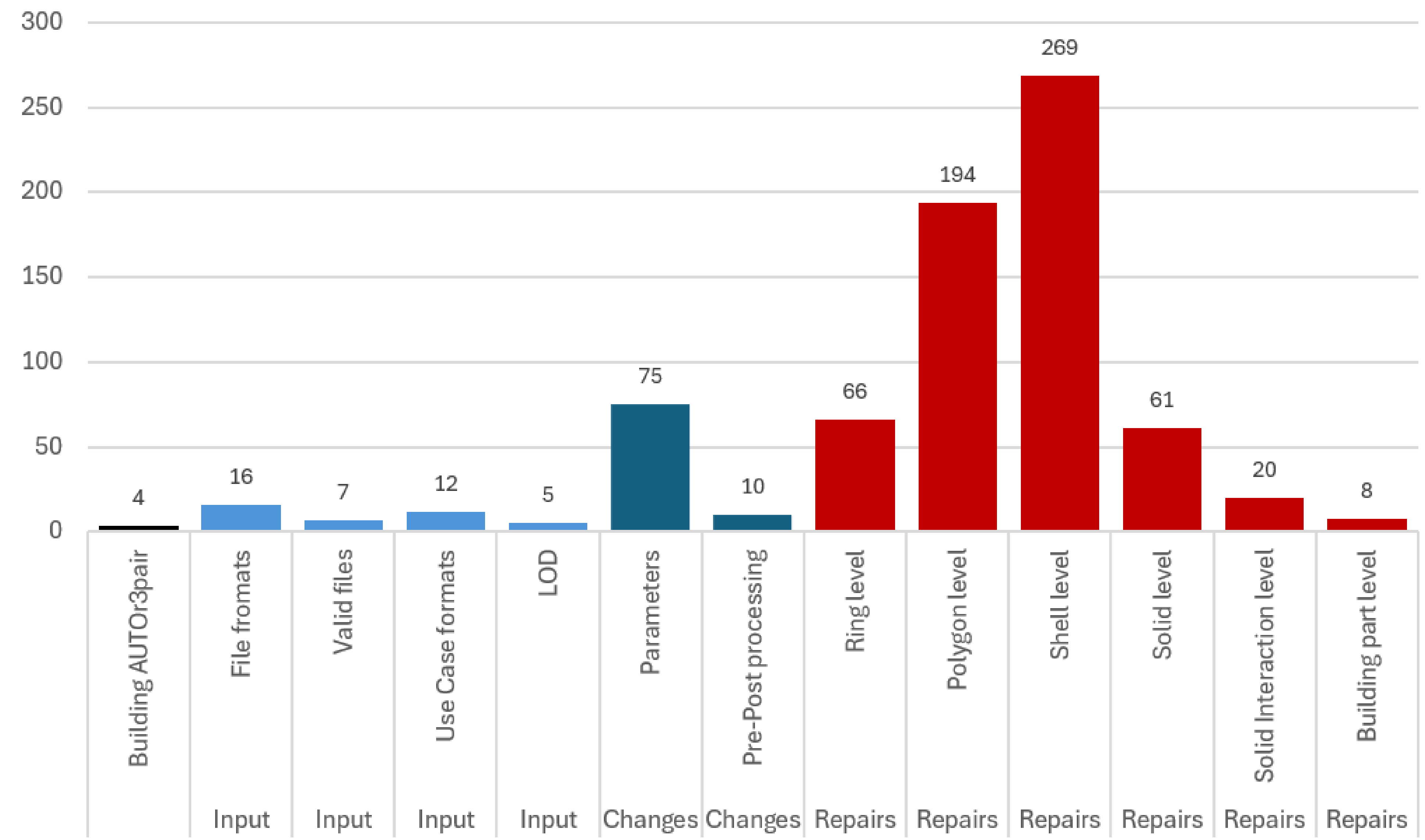
306



401



501



Result

- Validate that AUTOr3pair is working correctly

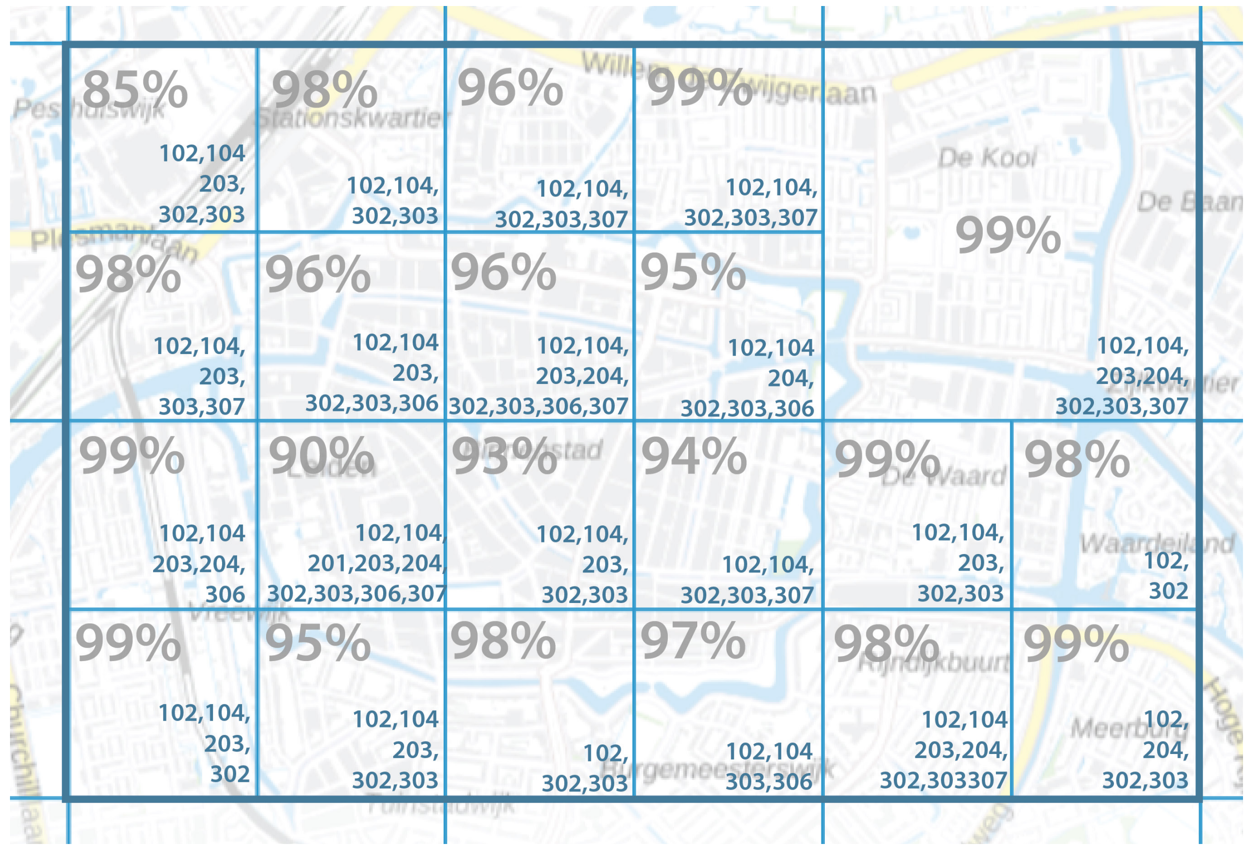
- All unit tests run automatically to verify compilation went smoothly and there are no bugs. Output is deleted after the tests

- For errors:
  1) Validate if error is present at start
  2) Repair (and check return code)
  3) Validate if error is not present
  4) Check result by evaluating boundaries

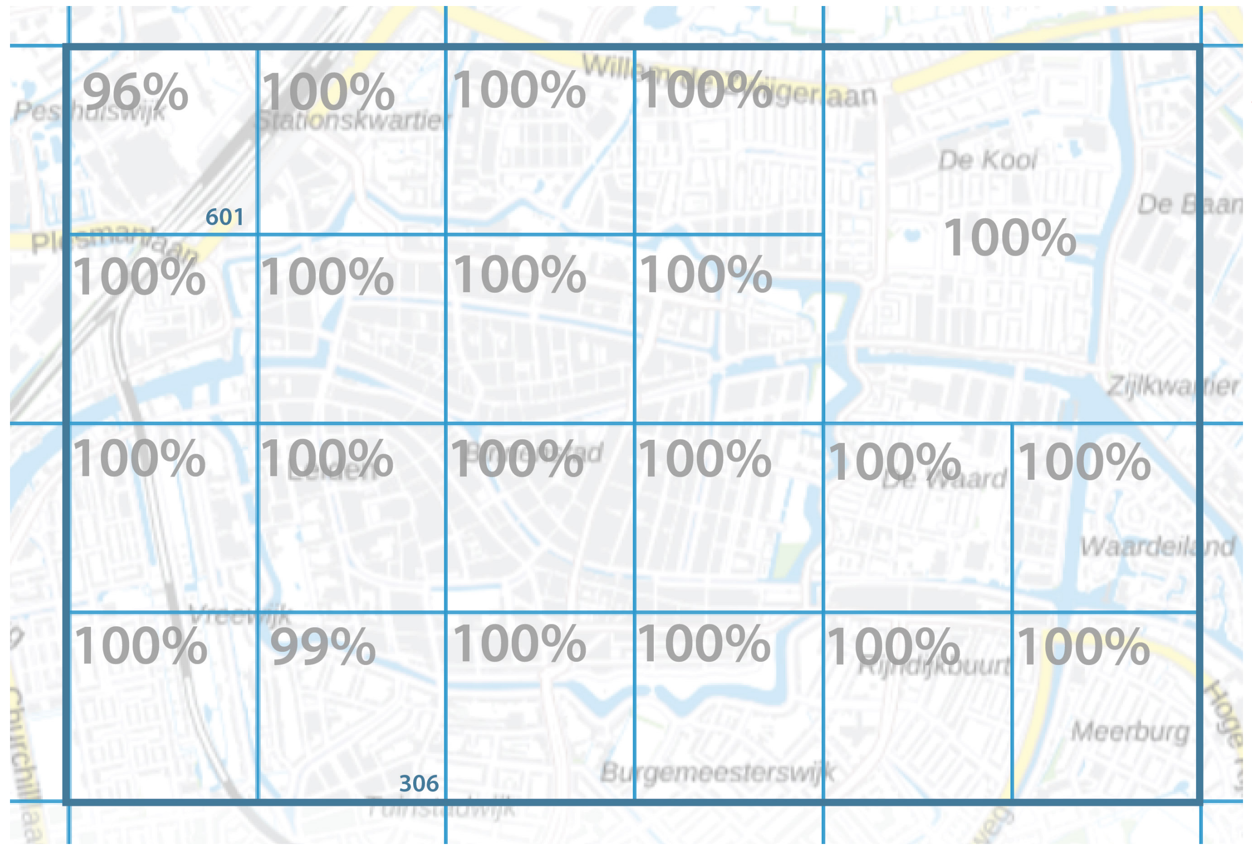
- Preserving of semantics is checked manually

BAG

input



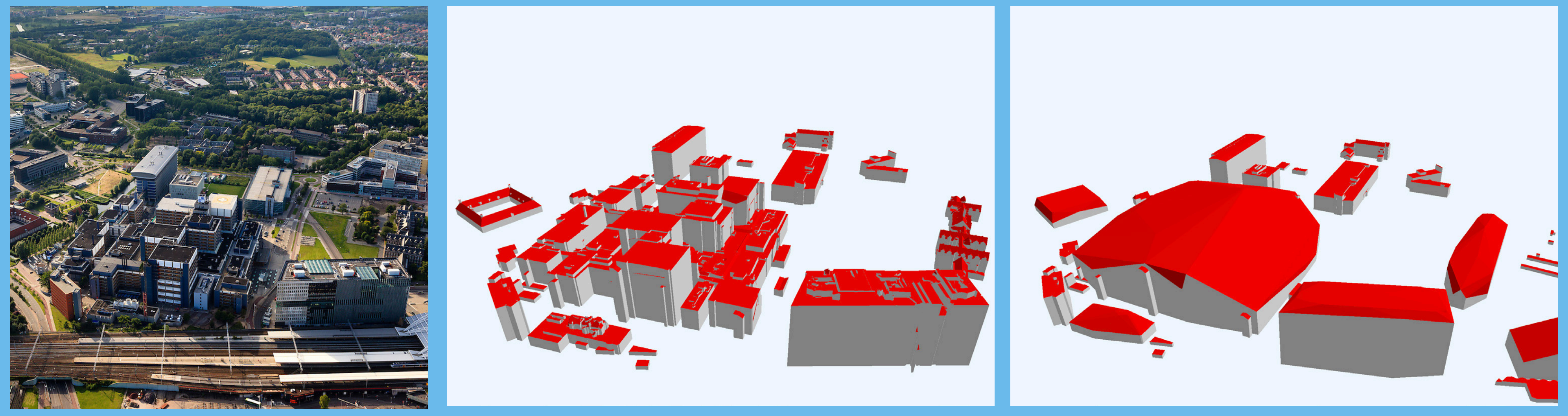
output



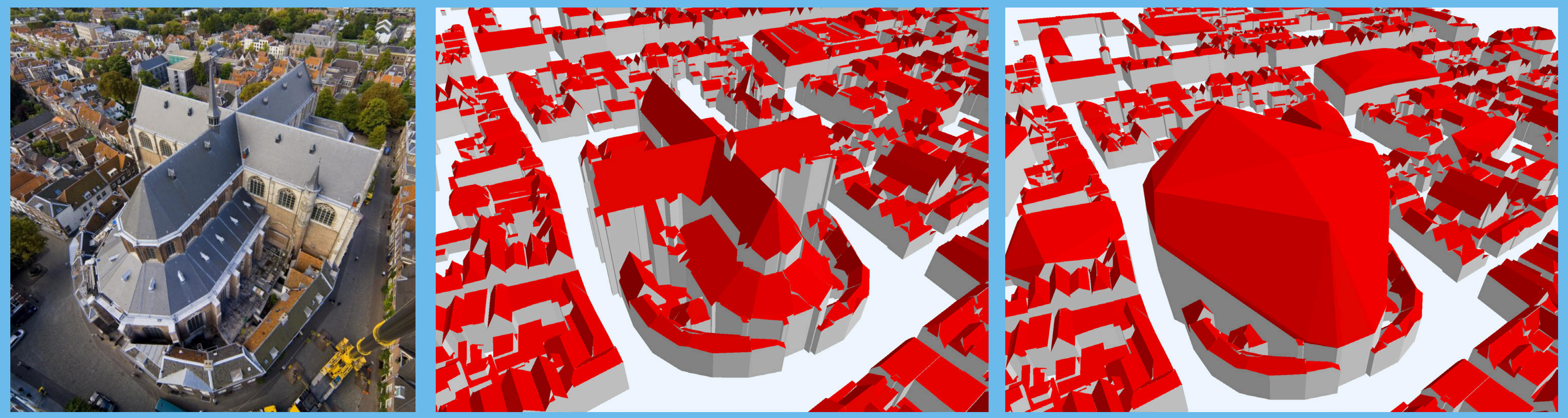
Result

- (Almost)100% valid
- Geomatric difference is small
- Global repairs needed in LOD 2.2

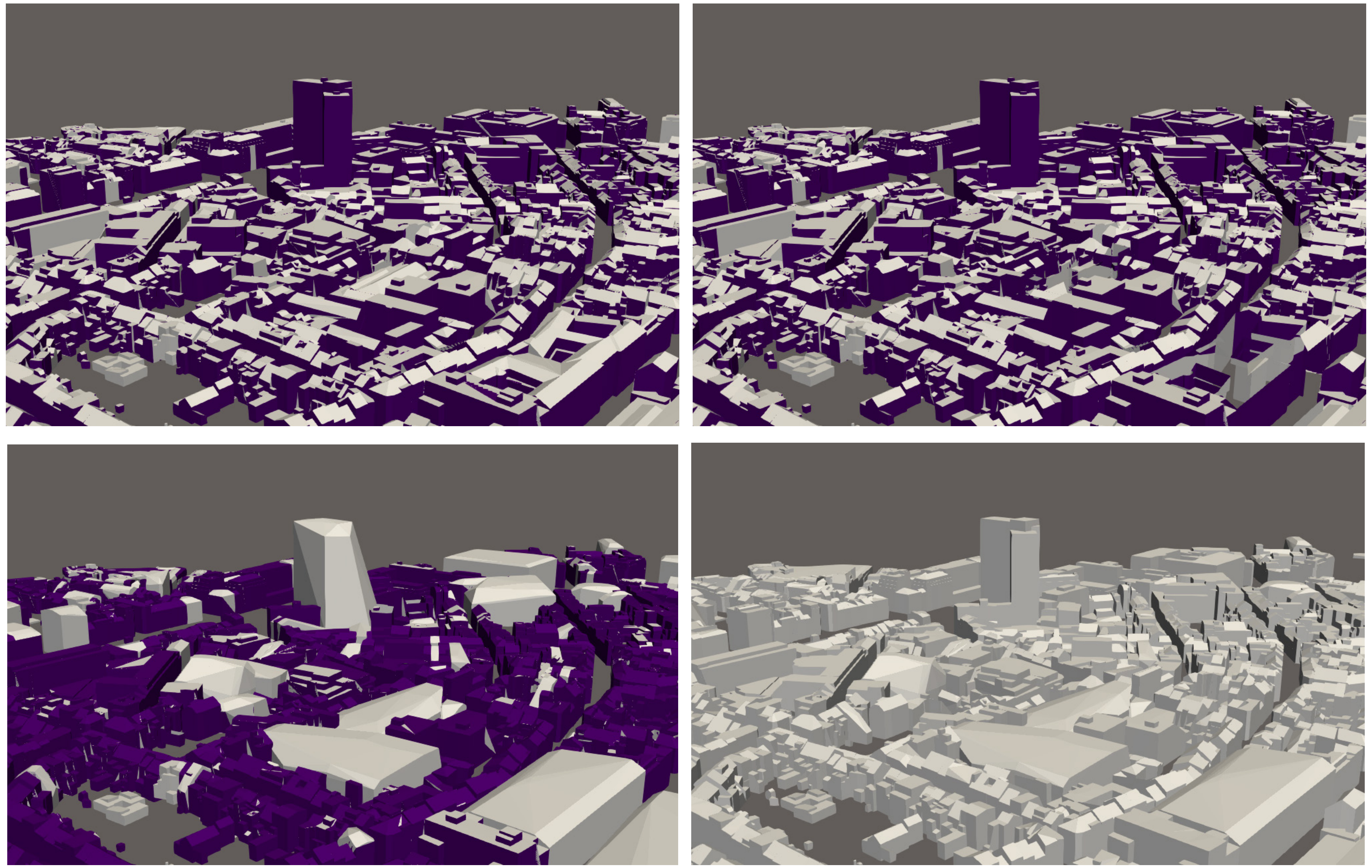
LUMC



Pieterskerk



Brussel



Input

Default

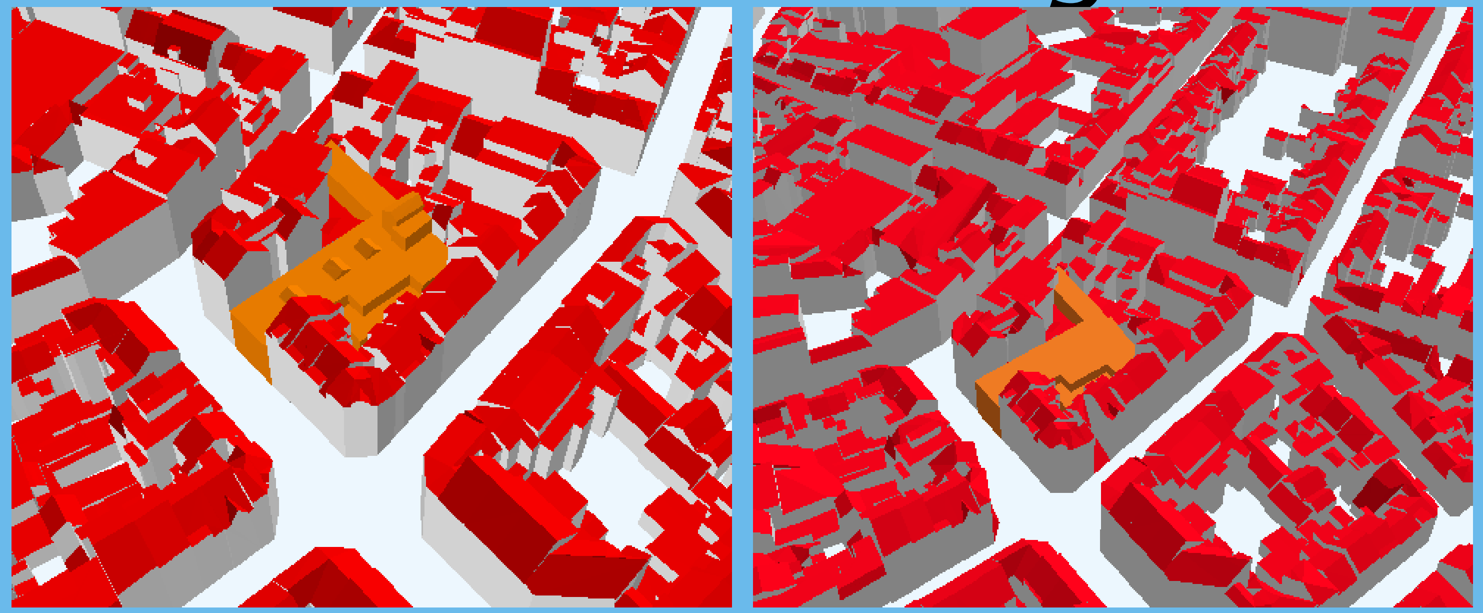
Orientation

Watertight

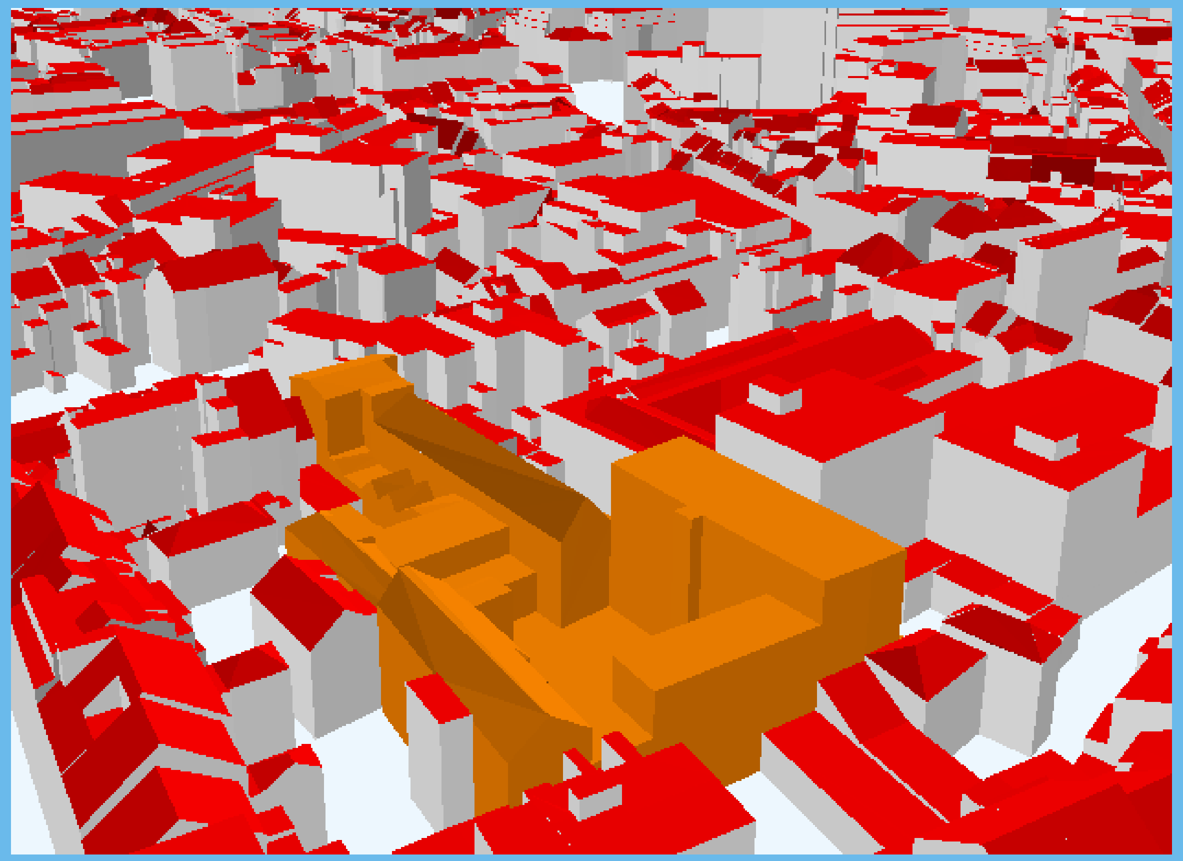
Result

- All 3 outputs are 99.9% valid

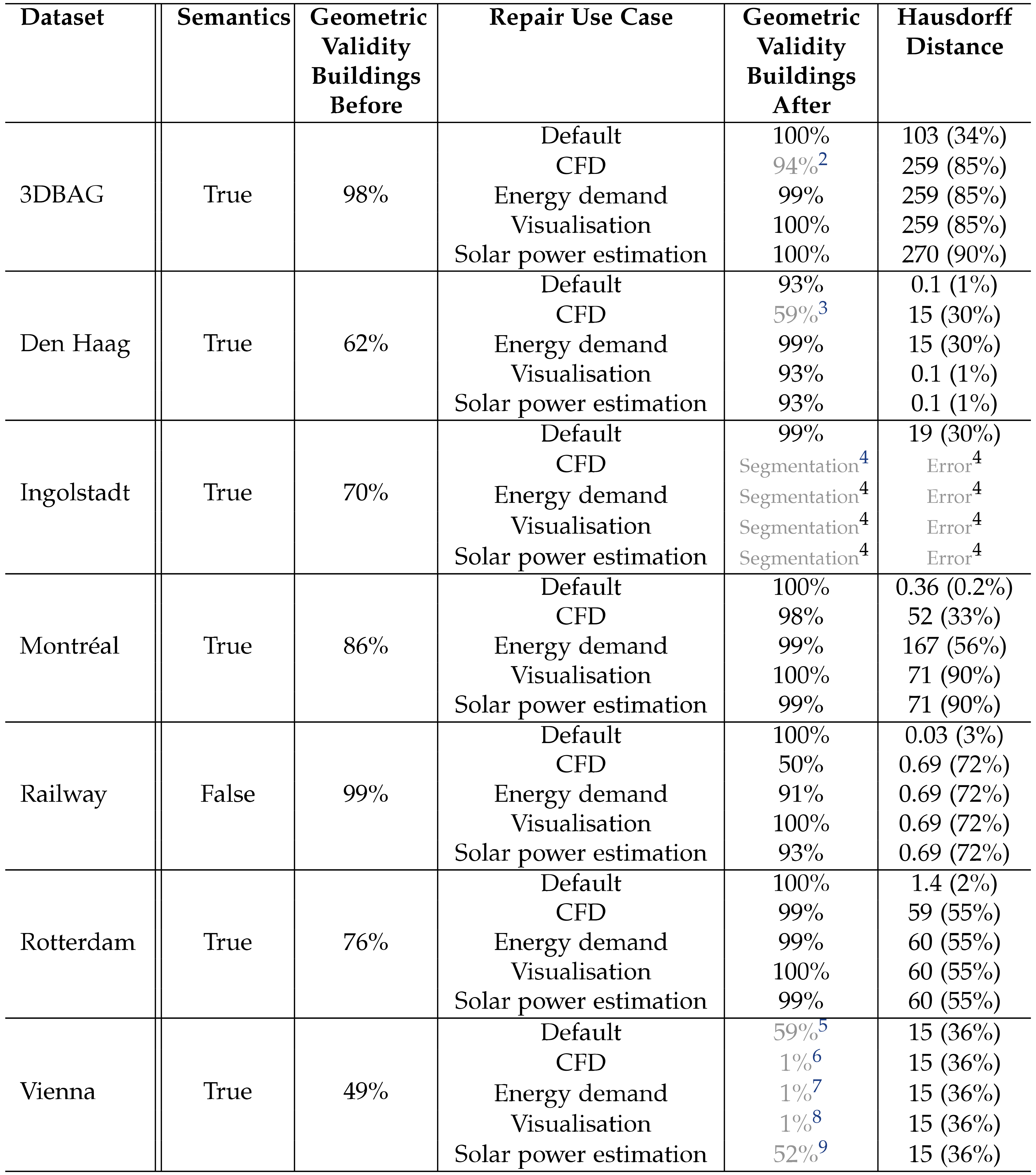
- Geomatric difference bigger for Orientation and watertight



- Global repair used for too complicated non-manifolds



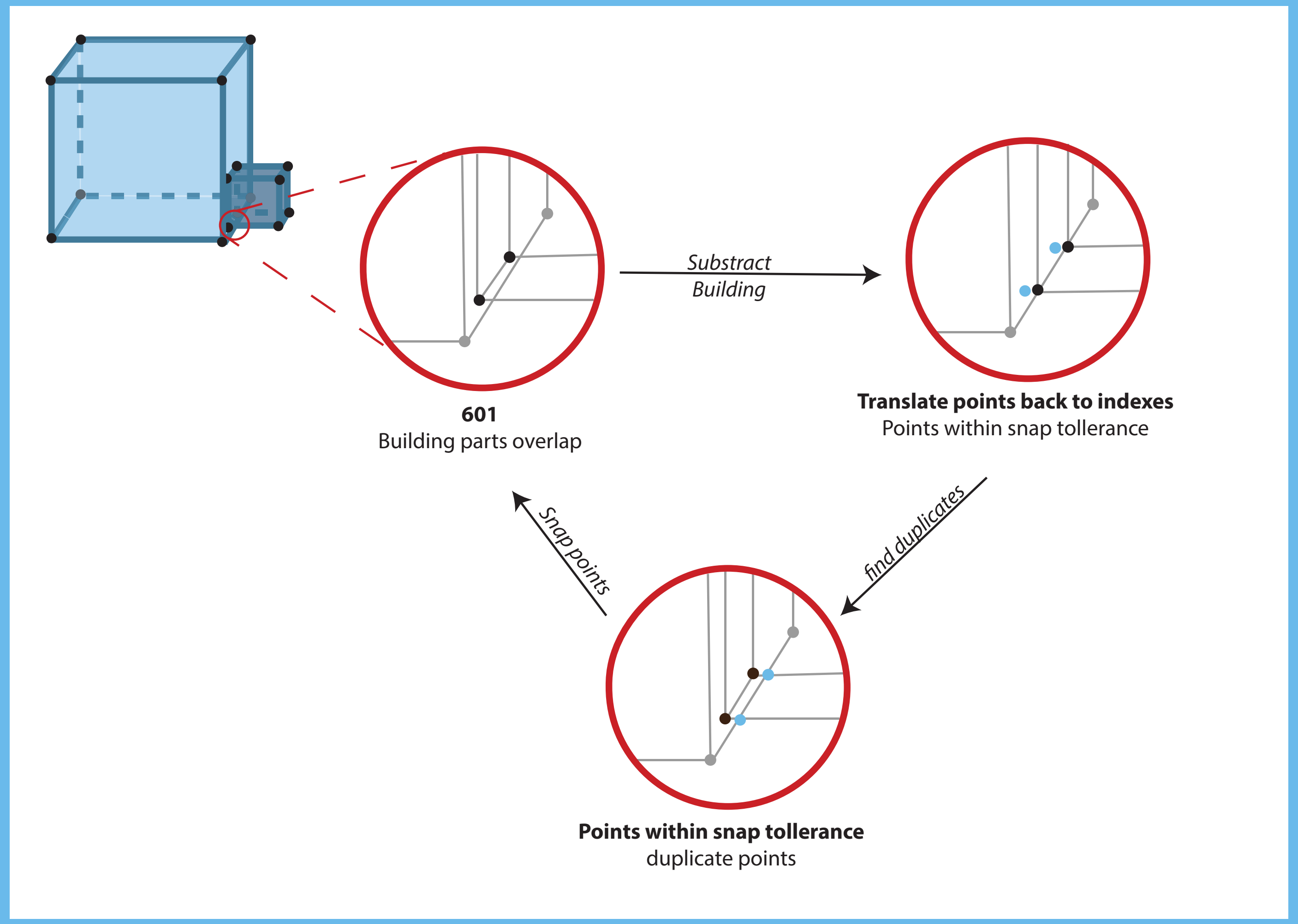
Use cases



Dataset	Semantics	Geometric Validity Buildings Before	Repair Use Case	Geometric Validity Buildings After	Hausdorff Distance
3DBAG	True	98%	Default	100%	103 (34%)
			CFD	94%[2]	259 (85%)
			Energy demand	99%	259 (85%)
			Visualisation	100%	259 (85%)
			Solar power estimation	100%	270 (90%)
Den Haag	True	62%	Default	93%	0.1 (1%)
			CFD	59%[3]	15 (30%)
			Energy demand	99%	15 (30%)
			Visualisation	93%	0.1 (1%)
			Solar power estimation	93%	0.1 (1%)
Ingolstadt	True	70%	Default	99%	19 (30%)
			CFD	Segmentation[4]	Error[4]
			Energy demand	Segmentation[4]	Error[4]
			Visualisation	Segmentation[4]	Error[4]
			Solar power estimation	Segmentation[4]	Error[4]
Montréal	True	86%	Default	100%	0.36 (0.2%)
			CFD	98%	52 (33%)
			Energy demand	99%	167 (56%)
			Visualisation	100%	71 (90%)
			Solar power estimation	99%	71 (90%)
Railway	False	99%	Default	100%	0.03 (3%)
			CFD	50%	0.69 (72%)
			Energy demand	91%	0.69 (72%)
			Visualisation	100%	0.69 (72%)
			Solar power estimation	93%	0.69 (72%)
Rotterdam	True	76%	Default	100%	1.4 (2%)
			CFD	99%	59 (55%)
			Energy demand	99%	60 (55%)
			Visualisation	100%	60 (55%)
			Solar power estimation	99%	60 (55%)
Vienna	True	49%	Default	59%[5]	15 (36%)
			CFD	1%[6]	15 (36%)
			Energy demand	1%[7]	15 (36%)
			Visualisation	1%[8]	15 (36%)
			Solar power estimation	52%[9]	15 (36%)

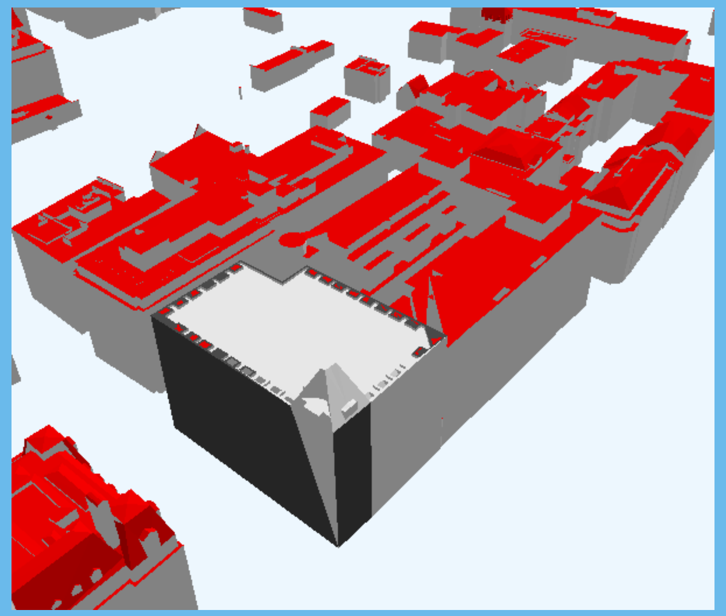
Results

- Validity improves significantly
- CGAL problems with meshes and Nef polyhedron --> falsely acused of overlap

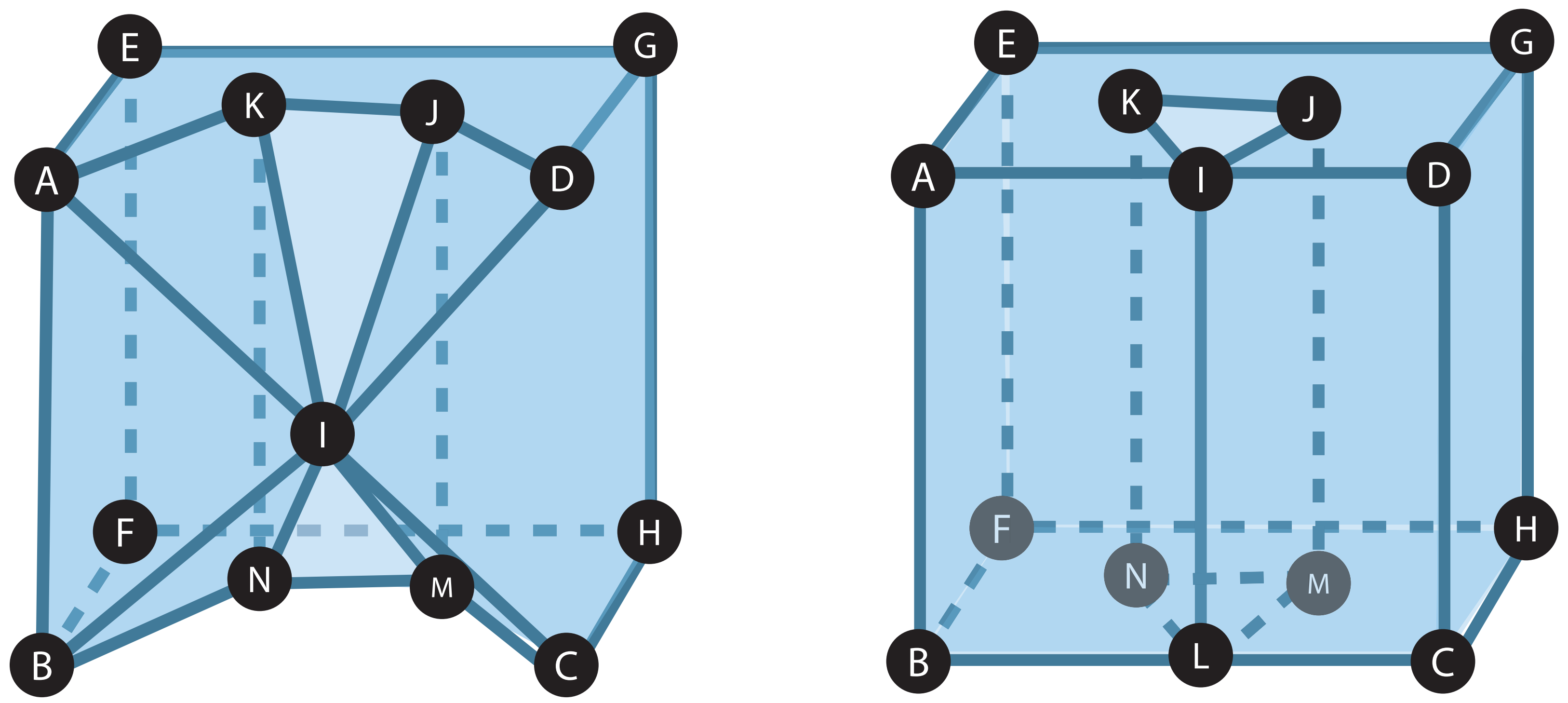


Substract Building
601
Building parts overlap
Translate points back to indexes
Points within snap tollerance
Snap points
find duplicates
Points within snap tollerance
duplicate points

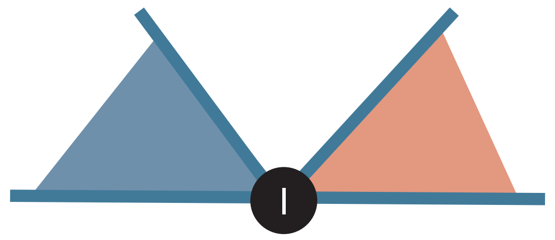
- CFD does give the “worst” results
- One case of wrongly preserved semantics



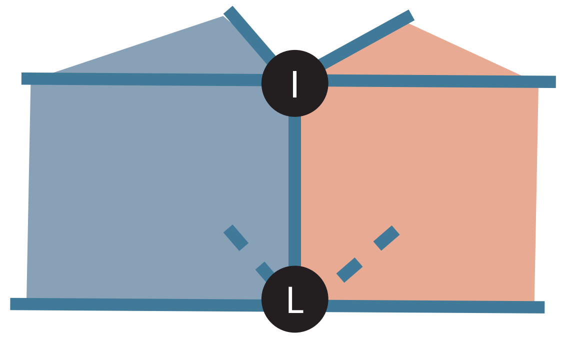
Global repair is needed



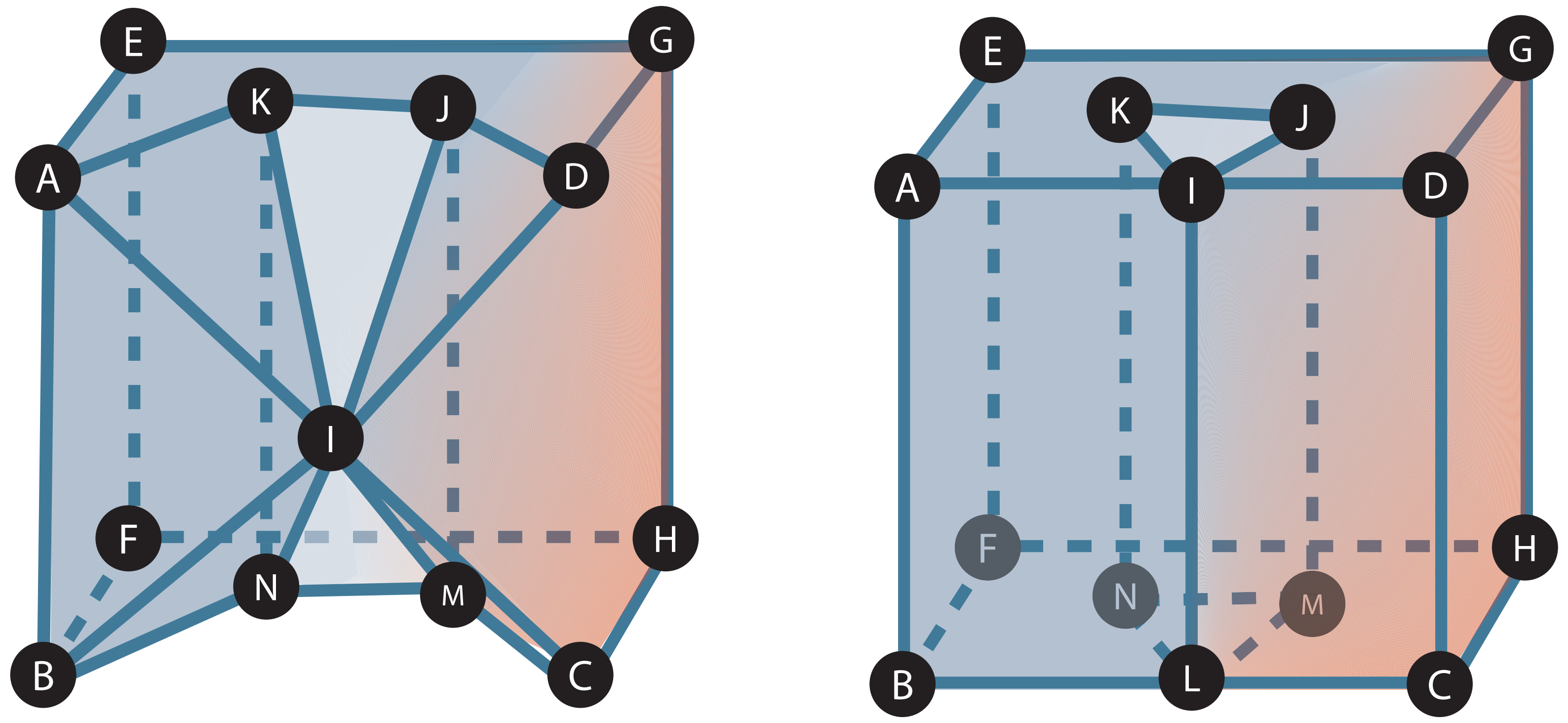
Normally would break down in parts



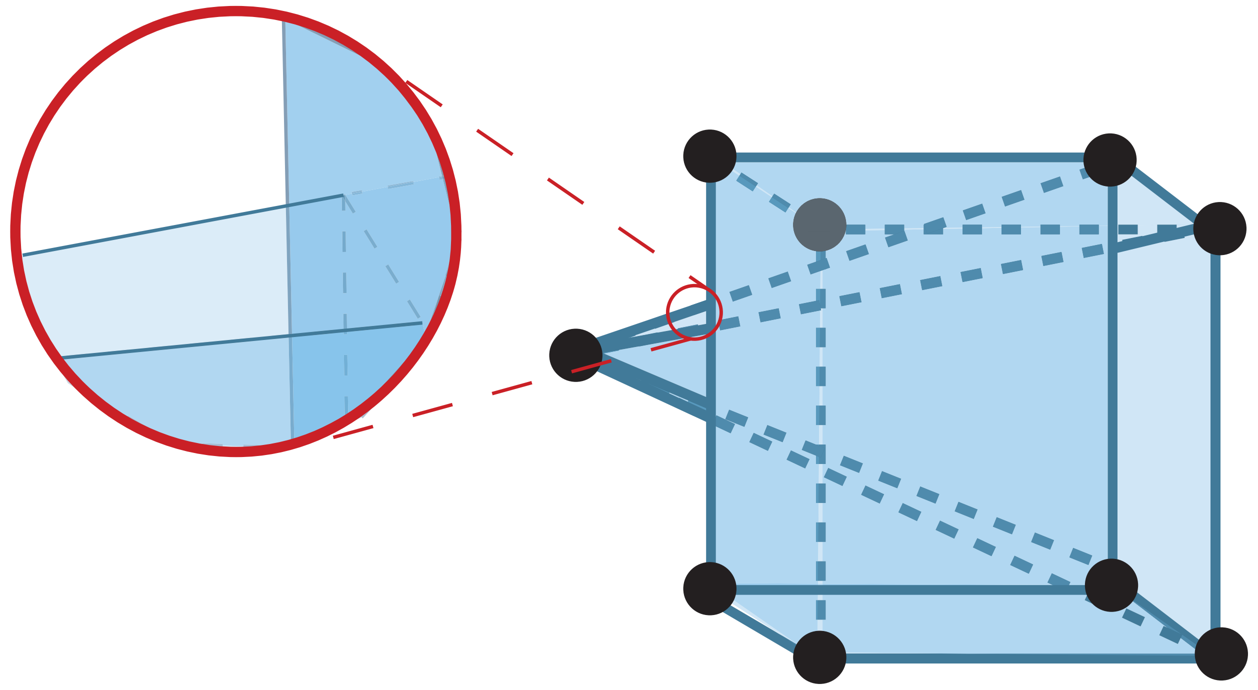
By cutting over point

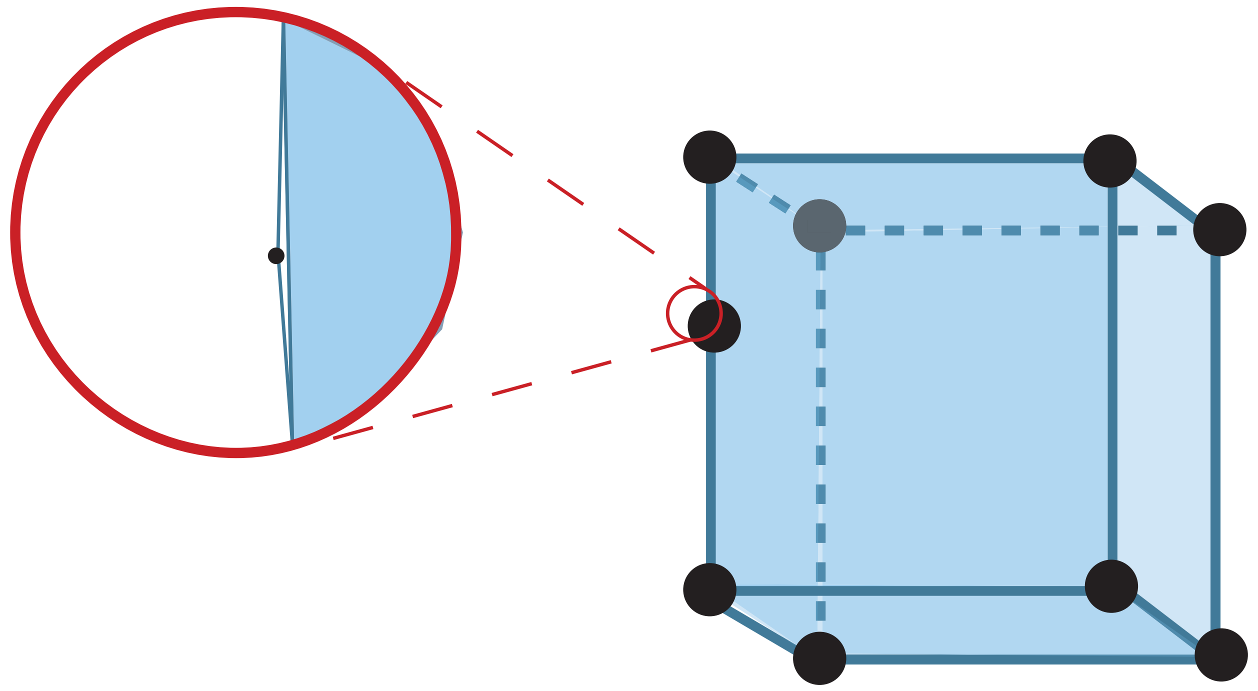


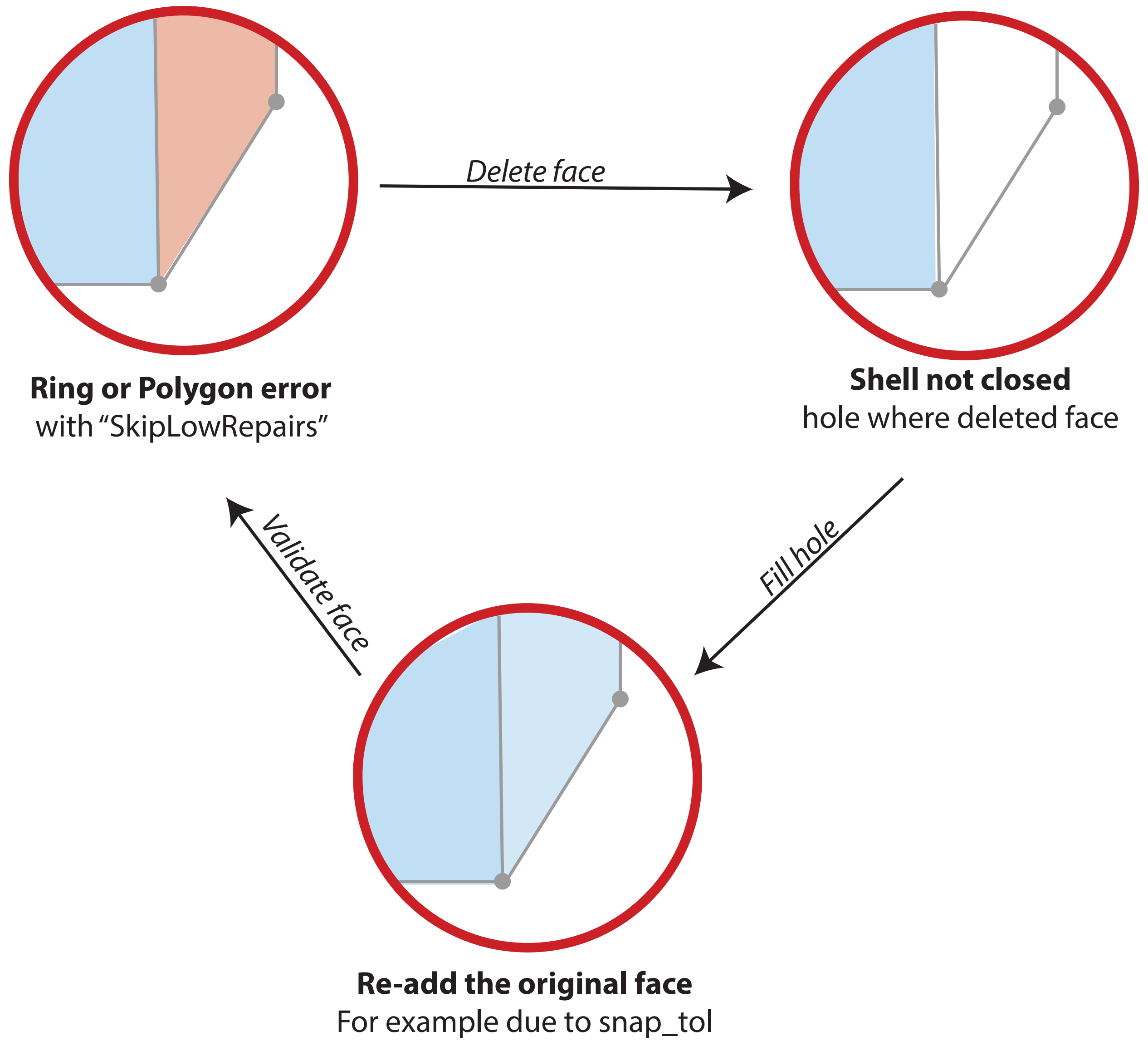
By cutting over Edge



But you cannot partition this shape by one cut







Delete face

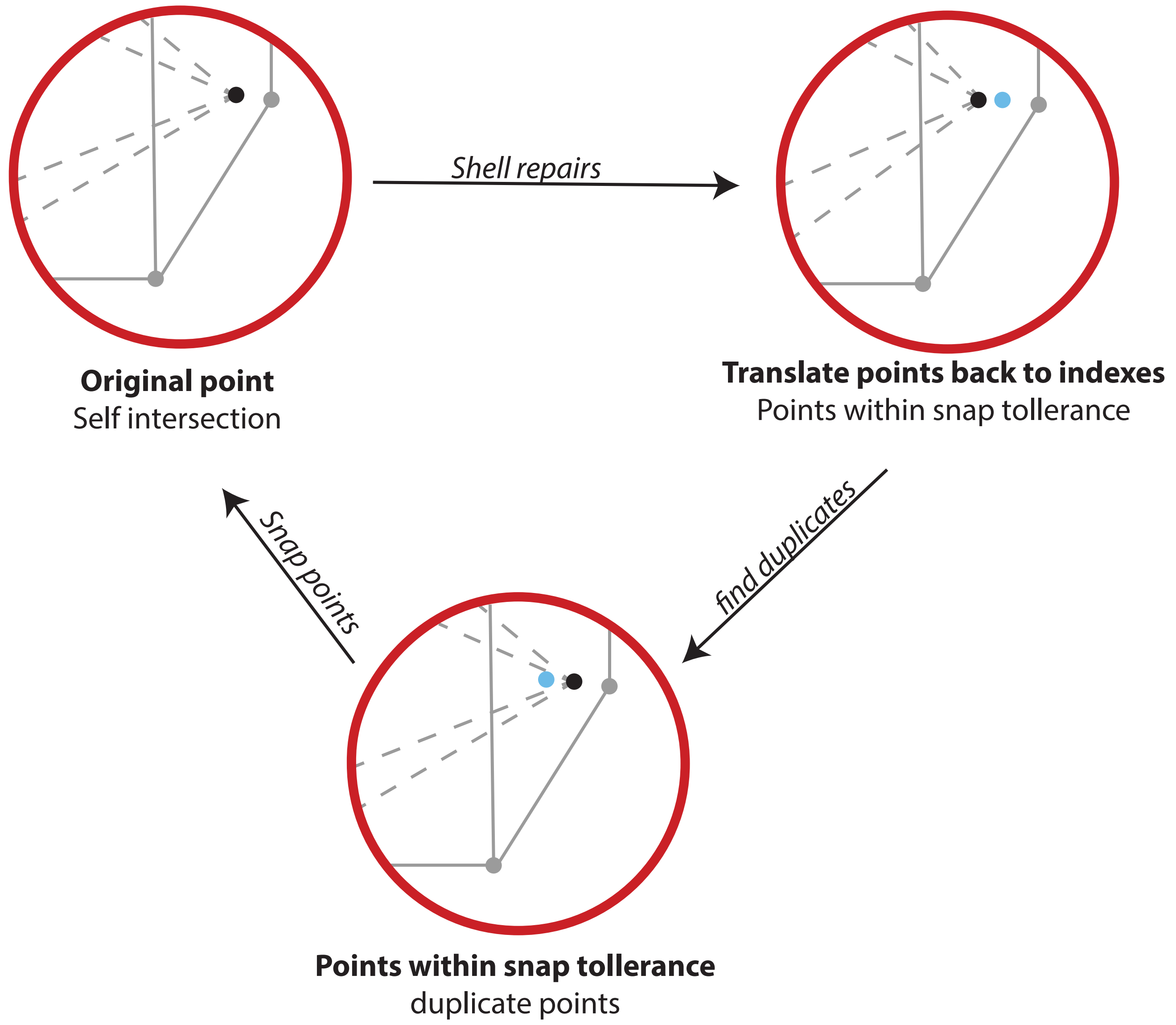
Ring or Polygon error
with “SkipLowRepairs”

Shell not closed
hole where deleted face

Fill hole

Re-add the original face
For example due to snap_tol

Validate face



Shell repairs

Original point
Self intersection

Translate points back to indexes
Points within snap tollerance

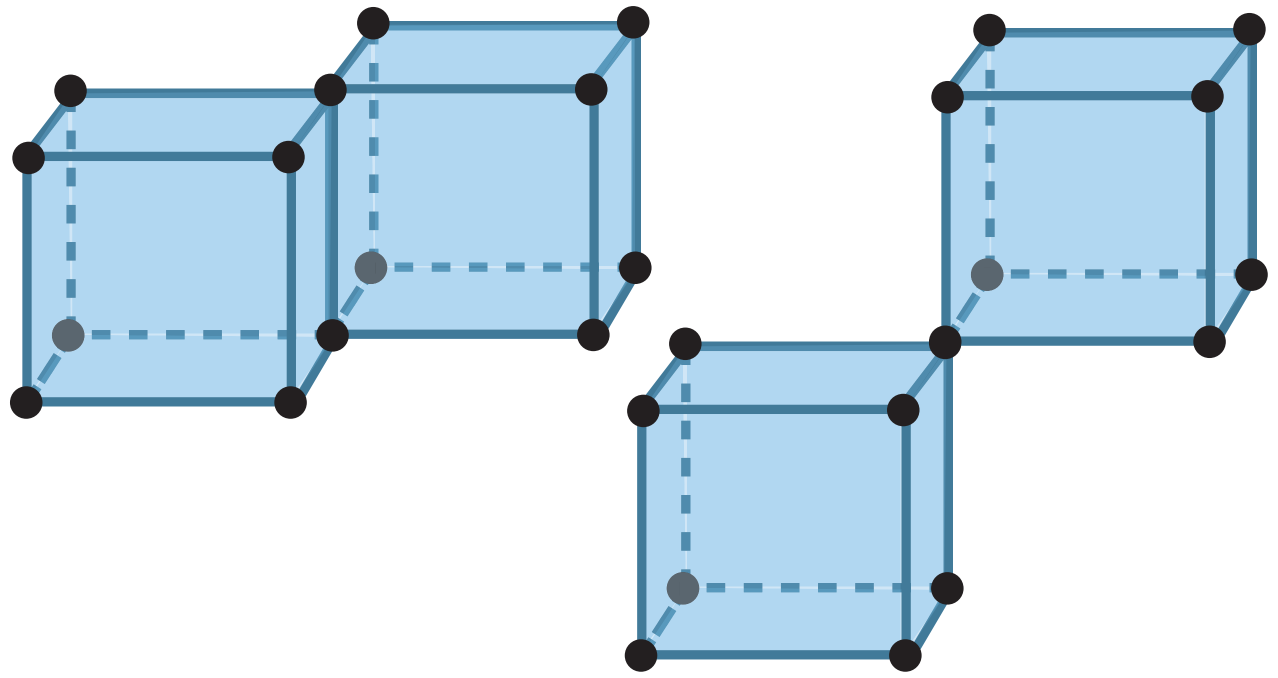
find duplicates

Points within snap tollerance
duplicate points

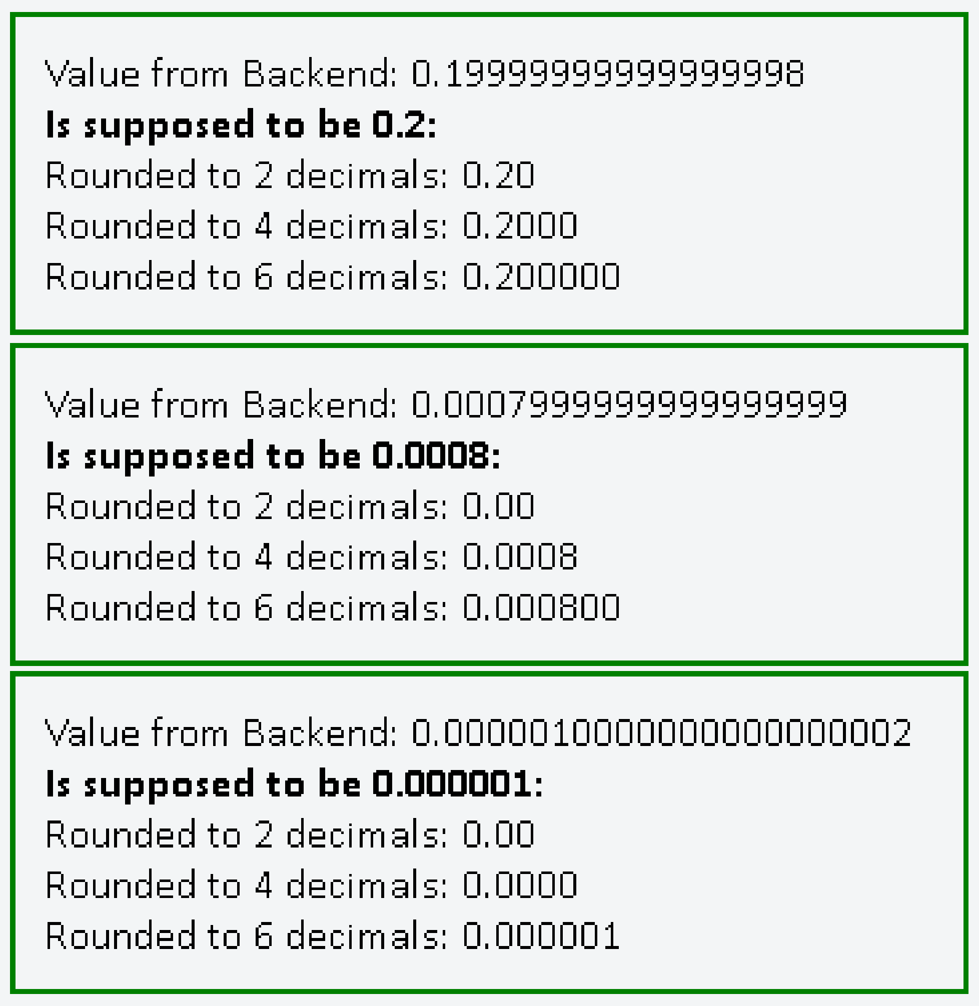
Snap points

Other issues

“non” repairs for OBJ



Floating point errors

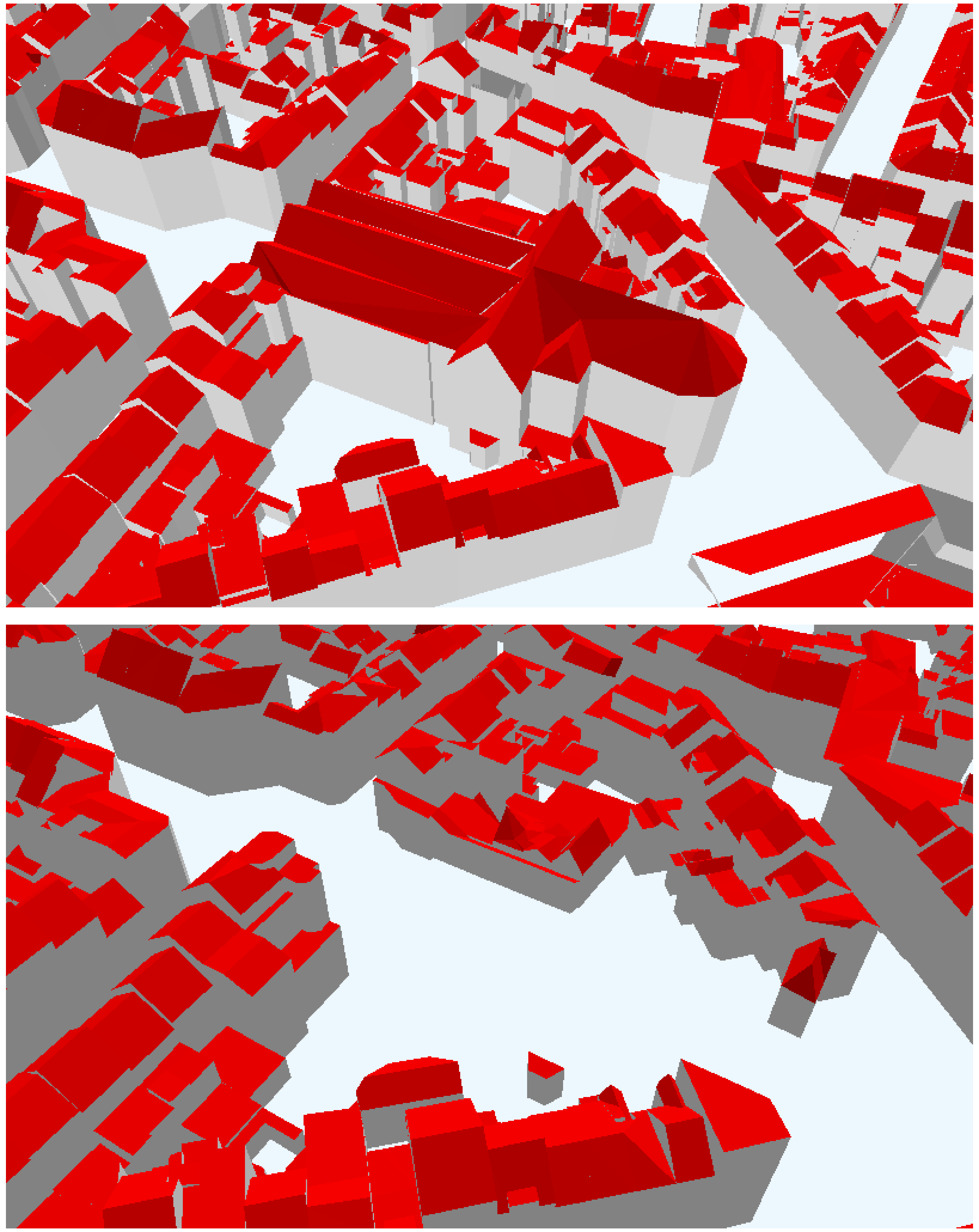


Value from Backend: 0.19999999999999998
Is supposed to be 0.2:
Rounded to 2 decimals: 0.20
Rounded to 4 decimals: 0.2000
Rounded to 6 decimals: 0.200000

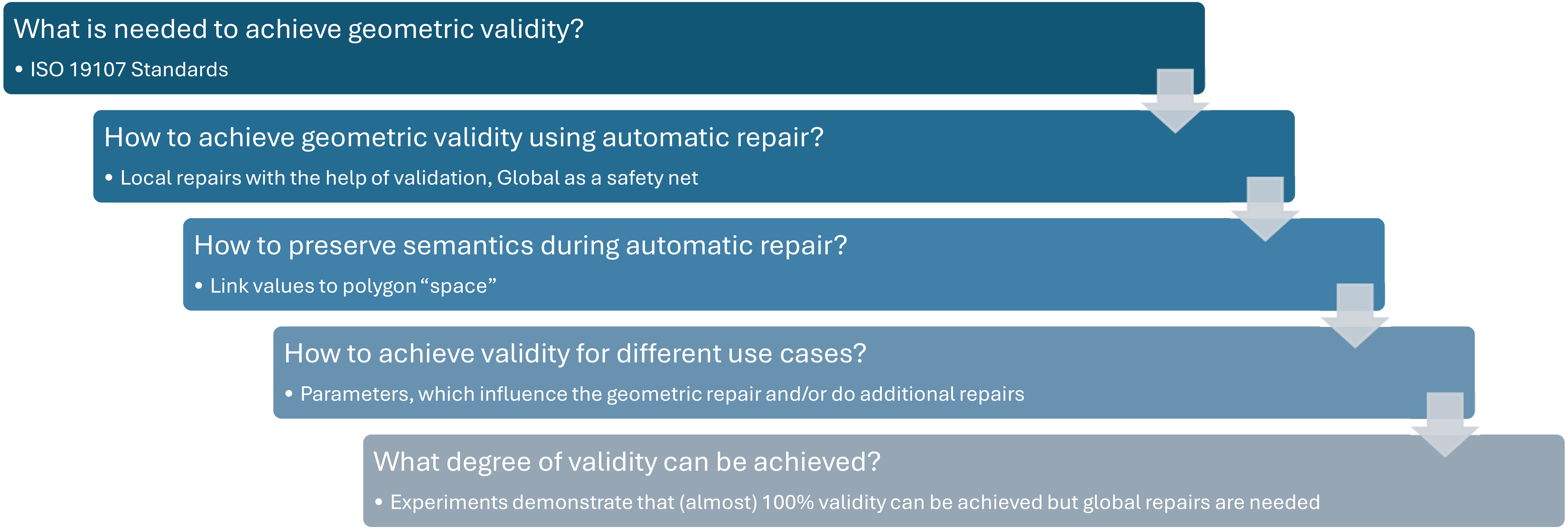
Value from Backend: 0.0007999999999999999
Is supposed to be 0.0008:
Rounded to 2 decimals: 0.00
Rounded to 4 decimals: 0.0008
Rounded to 6 decimals: 0.000800

Value from Backend: 0.0000010000000000000002
Is supposed to be 0.000001:
Rounded to 2 decimals: 0.00
Rounded to 4 decimals: 0.0000
Rounded to 6 decimals: 0.000001

Repair descisions



Conclusion



What is needed to achieve geometric validity?

- ISO 19107 Standards

How to achieve geometric validity using automatic repair?

- Local repairs with the help of validation, Global as a safety net

How to preserve semantics during automatic repair?

- Link values to polygon “space”

How to achieve validity for different use cases?

- Parameters, which influence the geometric repair and/or do additional repairs

What degree of validity can be achieved?

- Experiments demonstrate that (almost) 100% validity can be achieved but global repairs are needed

Recommendations for future work

More input file types & more use cases

Intergrating val3dity and AUTOr3pair into one tool

Automatic validation and repair for more semantic values

Validation for preserving of semantics

Research on keeping and extending textures

Intergrating automatic validation and repair for LODs

3D GIS application for preparing 3D City data

Develop a framework for the automatic repair and reconstruction of 3D city models to facilitate different use cases and implement a prototype.

Develop a framework for the automatic repair and reconstruction of 3D city models to facilitate different use cases and implement a prototype.