# DETECTION AND SEMI-AUTOMATIC CORRECTION OF GEOMETRIC INACCURACIES IN IFC FILES

G.N. Lilis[1], G.I. Giannakis[1] and D.V. Rovas[1,2]
[1]Department of Production Engineering and Management, Technical University of Crete, Chania, Greece
[2]Group of Energy Systems, Fraunhofer, Institute for Building Physics, Germany

## ABSTRACT

Accurate representation of geometry-related parameters is a prerequisite for the generation of building thermal simulation models. In case geometric data are extracted from IFC files, three types of geometrical errors are often encountered: clash errors where two building solids intersect, space definition errors where a building space volume is incorrectly defined leaving volume gaps between the space and neighboring building entities (walls, slabs,...) and surface orientation errors, where the normal vector of some boundary surfaces of building solids points to the wrong direction, generating a misconception on which area is inside or outside the solid. These errors are introduced either in the design process or during export from the BIM authoring tool. In this paper, error detection and correction algorithms are introduced to address each individual error type. The algorithms are tested on the geometrical data of the IFC files of two office buildings, where errors of the types mentioned above were detected and automatically corrected.

## INTRODUCTION

Energy performance simulation of buildings can be a valuable tool for estimation of energy-related parameters. In current state of practice a lot of effort is required to manually generate simulation model inputs, with significant part of this effort spent in geometry-related aspects. Building Information Models (BIM), such as the Industry Foundation Classes (IFC) (ISO-16739, 2013) provide an information-rich repository for Extraction of required data. Transformation of this information (e.g. by identification of second-level boundary information) (Bazjanac, 2010), and Loading to another data model (e.g. Simmodel) or direct generation of the input file (e.g. idf file for EnergyPlus) can help automate the model-generation process. Such Extraction, Transformation and Loading (ETL) layers have been the topic of intense research interest in recent years. The quality of the input IFC data is of paramount importance in establishing such ETL processes. Very often errors introduced in the design of the BIM model or during the extraction from the authoring tool to IFC, result in poor- quality models.
Unavoidably, poor quality of geometric input has a sizeable impact in the transformation process, most notably in the identification of the building's second-level space boundary topology. The second-level space boundary topology consists of a set of surface pairs which are defined at the boundaries between building constructions (walls, slabs,...) and internal building spaces. Thermal energy flows through these surface pairs from an internal building space to either: another internal building space or the building's environment (air or ground) (Figure 1). The accuracy of this topology can be guaranteed provided that any geometric conflict is removed using appropriate detection and correction processes. These detection and correction processes is an important prerequisite step of any second-level space-boundary-based generation process and is the subject of the present work.

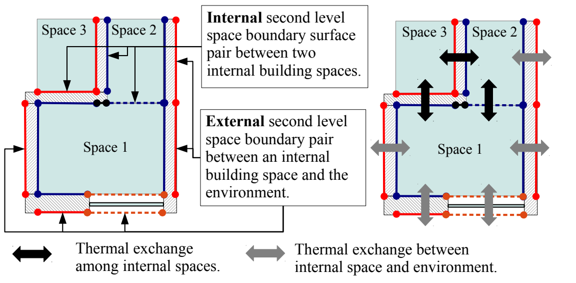
Space 3
Space 2
Space 1
**Internal** second level space boundary surface pair between two internal building spaces.
**External** second level space boundary pair between an internal building space and the environment.
Space 3
Space 2
Space 1
Thermal exchange among internal spaces.
Thermal exchange between internal space and environment.

*Figure 1: Example of second-level space boundary surface pairs*

Tools for detecting a wide variety of geometric conflicts been developed both for CAD models (Barequet and Kumar, 1997), (Barequet et al., 1998), and more recently for IFC BIM data models; commercial software for model checking include the Solibri Model Checker (SMC) (Khemlani, 2002) and the TEKLA Model Checker Suite (TEKLA, 2014). Algorithms which identify geometric clashes in geometric data of buildings have also been reported (van den Helm et al., 2010), (Klosowski et al., 1998). Finally, tools which detect and correct surface gaps in geometric multi-polygonal objects have been developed as well (Ju, 2004). Although these tools detect a wide variety of geometric inaccuracies, not all of these inaccuracies, affect the second level space boundary topology of a building. Additionally not all of these tools, provide mechanisms for automated correction (when possible).
The present work focuses on the geometric inaccuracies which affect the generation of the second-level space boundary topology and introduces detection and correction tools. These inaccuracies can be classified into three major categories: clash errors (I), space definition errors (II) and boundary surface errors (III), which can be distinguished further into surface orientation errors (IIIa) and incomplete shell errors (IIIb), as described in the error classification section. All of these error types can be detected in the geometric data of IFC files and some can be corrected (I,II,III and

IIIa), using detection and correction algorithms described in respective detection and correction sections. Finally, these detection and correction algorithms are applied on the IFC files of two office buildings and obtained results are presented in the examples section.

### Notation

To ease the comprehension of the introduced algorithms, certain mathematical notation is adopted, which consists of the following set of symbols:

- **Solid**: is denoted with a bold capital letter e.g. **S** and represents the subset of the three dimensional space, the solid occupies.
- **Surface set**: is denoted with a calligraphic capital letter e.g. $\mathcal{S}$. A surface set contains a finite number of polygonal surfaces e.g. $\mathcal{S} = \{S_1, S_2, ..., S_N\}$.
- **BSP-tree of surface set**: is denoted with T and the surface set letter e.g. $T_{\mathcal{S}}$. A BSP-tree is the binary space partitioning tree of an oriented space set $\mathcal{S}$ (Thibault and Naylor, 1987).
- **Boundary of a solid**: is denoted using $\partial$ in front of a bold capital letter which denotes the solid e.g. $\partial\mathbf{S}$. The boundary of a solid is a surface set which contains all the boundary surfaces of the solid.
- **Polygonal surface**: is denoted with a capital letter e.g. $S_i$.
- **Normal vector of a polygonal surfaces**: is denoted with $\hat{n}$ and the capital letter of the polygonal surface e.g. $\hat{n}_{S_i}$.

## GEOMETRIC REPRESENTATION

The developed geometric error detection and correction tools, use two geometric solid representations which are described in the following sections.

### Boundary representation (B-rep)

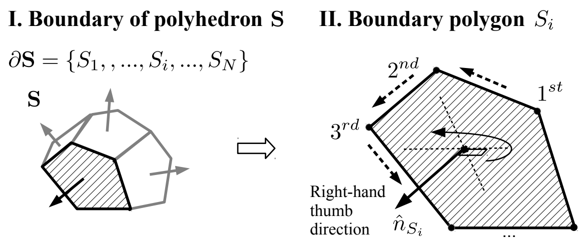
I. Boundary of polyhedron S
II. Boundary polygon $S_i$
$\partial\mathbf{S} = \{S_1, ..., S_i, ..., S_N\}$
S
$2^{nd}$
$1^{st}$
$3^{rd}$
Right-hand thumb direction
$\hat{n}_{S_i}$
...

*Figure 2: Example of a boundary representation (B-rep) of a building solid* **S***.*

The solid representation used in the detection and correction tools, is the boundary representation method, or B-rep (Jackson, 1995). A B-rep of a building solid **S** is defined as the set of the solid boundary polygon surfaces: $\partial\mathbf{S} = \{S_1, ..., S_N\}$, (part I of Figure 2). Each surface polygon in the B-rep $S_i$ is correctly oriented when its normal vector ($\hat{n}_{S_i}$) direction evaluated using the right-hand rule (part II of Figure 2) points outwards. If the boundary points are ordered following this right-hand rule (1st, 2nd, ...), then the boundary surface is oriented correctly (Part II of Figure 2).

### Binary Space Partitioning tree (BSP-tree)

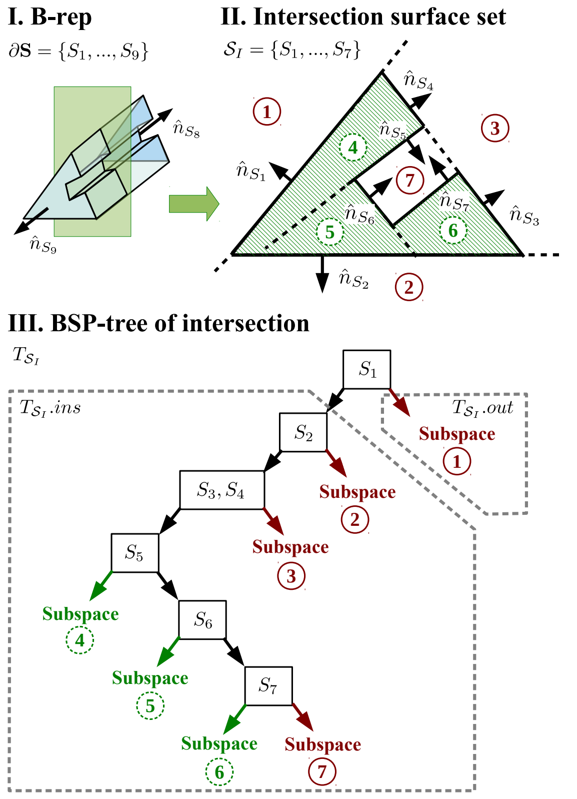

*Figure 3: Example of a Binary Space Partitioning tree (BSP-tree)* $T_{\mathcal{S}_I}$ *of a surface set* $\mathcal{S}_I = \{S_1, ..., S_7\}$ *which is a subset of the B-rep* $\partial\mathbf{S}$ *of a building solid* **S**.

The second solid representation used in the detection and correction tools is called Binary Space Partitioning tree representation or BSP-tree (Thibault and Naylor, 1987). BSP-tree is based on the information provided by the solid B-rep, given that all boundary surfaces are correctly oriented. A BSP-tree $T_{\mathcal{S}}$ of a set of surfaces $\mathcal{S}$, is a binary tree (Part III of Figure 3) which has: at its nodes (including the root) one or multiple **coplanar surface polygons** (node polygons $T_{\mathcal{S}}$.pol) with same direction normal vectors, in the left branches ($T_{\mathcal{S}}$.ins) a subtree containing at its nodes surface polygons which lie **outside** the half space pointed by the normal vector of the node polygon (inside the solid) and in the right branches ($T_{\mathcal{S}}$.out) a subtree containing at its nodes surface polygons which lie **inside** the half space pointed by the normal vector of the node polygon (outside the solid). In a broad sense, a BSP-tree partitions the three dimensional space into a finite number of sub-spaces, defined at its leaf nodes, depending on the orientation of the surface polygons at its nodes (sub-spaces 1,...,7 in Part III of Figure 3). Additionally, the node polygons may belong to a general surface set, e.g. the surface set $\mathcal{S}_I$ of the intersection displayed in Part II of Figure 3 which is a subset of a the solid B-rep ($\partial\mathbf{S}$), displayed in Part I of the same figure.

From this general perspective, a BSP-tree can be viewed as a filter where any polygonal surface, starting from the root and depending on its location with respect to the node polygons of the tree, can be filtered (partitioned) as it progresses down the tree into a finite number of surface parts. These parts end up in the sub-spaces of the tree leaf nodes. Such an operation is demonstrated in Figure 4 on a test surface $A$ using a BSP-tree $T_{\mathcal{S}_I}$ as a filter. The test surface $A$ is split into five segments $A_i, i = 1, \dots, 5$ two $(A_2, A_4)$ end up in the sub-spaces of the left leafs (inside the surface set $\mathcal{A}_I$) and the other three $(A_1, A_3, A_5)$ end up in the right leafs (outside the surface set $\mathcal{A}_I$).

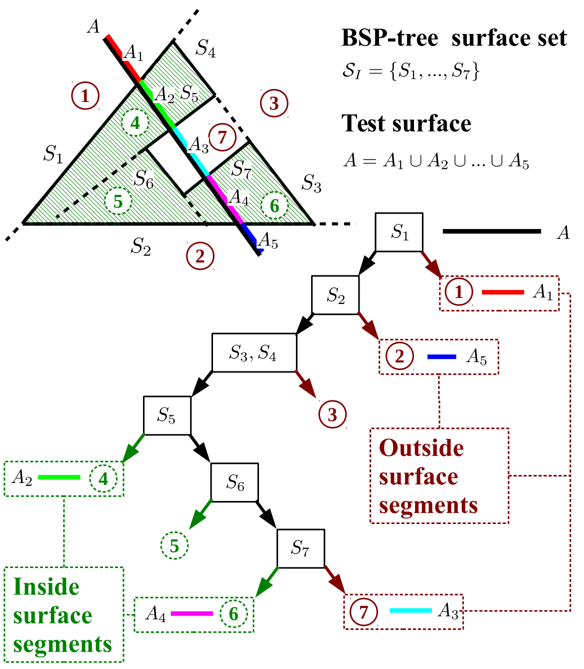

*Figure 4: Demonstration of a filtering capabilities of a BSP-tree $T_{\mathcal{S}_I}$ on a test surface $A$.*

This filtering ability of the BSP-tree is used by the filtering functions described in the next section.

## GEOMETRIC OPERATIONS

The detection and correction algorithms used to identify and correct the geometric errors contained in the IFC files, rely on three filtering functions: the **outside** $(F_{out})$, **inside** $(F_{ins})$ and **coplanar opposite direction** $(F_{cod})$ filtering functions. From a general point of view, these functions are applied on a surface set $\mathcal{A}$ using the BSP-tree representation $T_\mathcal{B}$ of another surface set $\mathcal{B}$, which acts as a filter.

These filtering functions return surface or parts of surfaces of $\mathcal{A}$ which: lie outside the surface set $\mathcal{B}$ (returned by $F_{out}$ see example I of Figure 5), lie inside the surface set $\mathcal{B}$ (returned by $F_{ins}$ see example II of Figure 5), or are coplanar with the surfaces of $\mathcal{B}$ and have opposite normal vectors, (returned by $F_{cod}$ see example III of Figure 5).

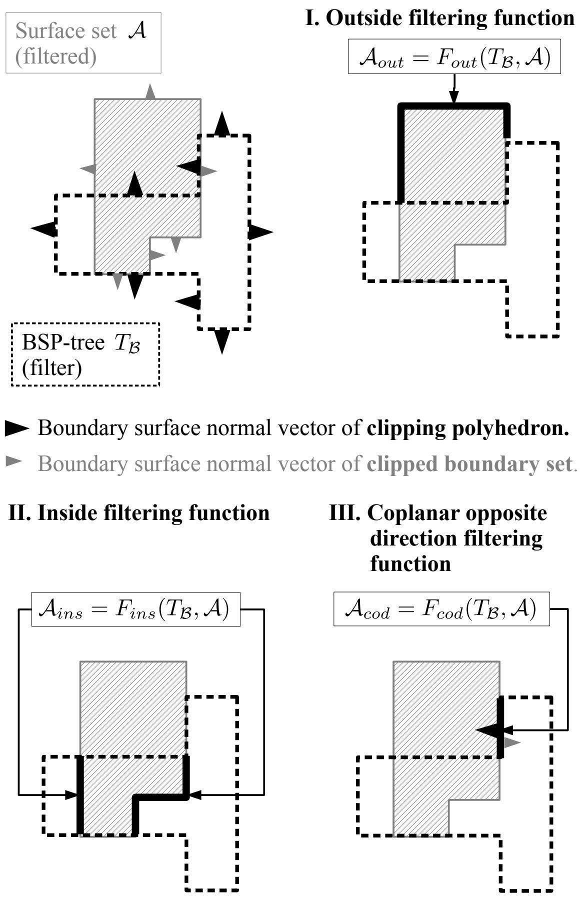
Surface set $\mathcal{A}$ (filtered)
I. Outside filtering function
$\mathcal{A}_{out} = F_{out}(T_\mathcal{B}, \mathcal{A})$
BSP-tree $T_\mathcal{B}$ (filter)
Boundary surface normal vector of clipping polyhedron.
Boundary surface normal vector of clipped boundary set.
II. Inside filtering function
III. Coplanar opposite direction filtering function
$\mathcal{A}_{ins} = F_{ins}(T_\mathcal{B}, \mathcal{A})$
$\mathcal{A}_{cod} = F_{cod}(T_\mathcal{B}, \mathcal{A})$

*Figure 5: **I. Outside filtering function:** $F_{out}$ returns parts of $\mathcal{A}$ which lie outside $\mathcal{B}$, **II. Inside filtering function:** $F_{ins}$ returns parts of $\mathcal{A}$ which lie inside $\mathcal{B}$, **III. Coplanar opposite direction filtering function:** $F_{cod}$ returns parts of $\mathcal{A}$ which are coplanar with boundary surfaces of $\mathcal{B}$ and have opposite normal vectors.*

## ERROR CLASSIFICATION

The geometric inaccuracies which appear in IFC files and affect the generation of a building's second level space boundary topology, can be classified into the following three categories:

### I. Clash errors

Clashes are non-empty and non-zero volume intersections among building entities. Clashes appear when the B-reps of building entities are not defined properly creating a nonzero volume space where the solid of one entity enters the solid of the other (van den Helm et al., 2010). An example of a clash between a wall and a slab is shown in Part I of Figure 6.

### II. Space definition errors

Space definition errors appear when the B-rep of an internal building space is not attached to all of the B-reps of its neighbor building entities. In such cases, the volume of the internal building space is not defined correctly, leaving small volume gaps of undefined space between the volume and neighbor building entities. An example of an incorrectly defined building space volume is displayed in Part II of Figure 6.

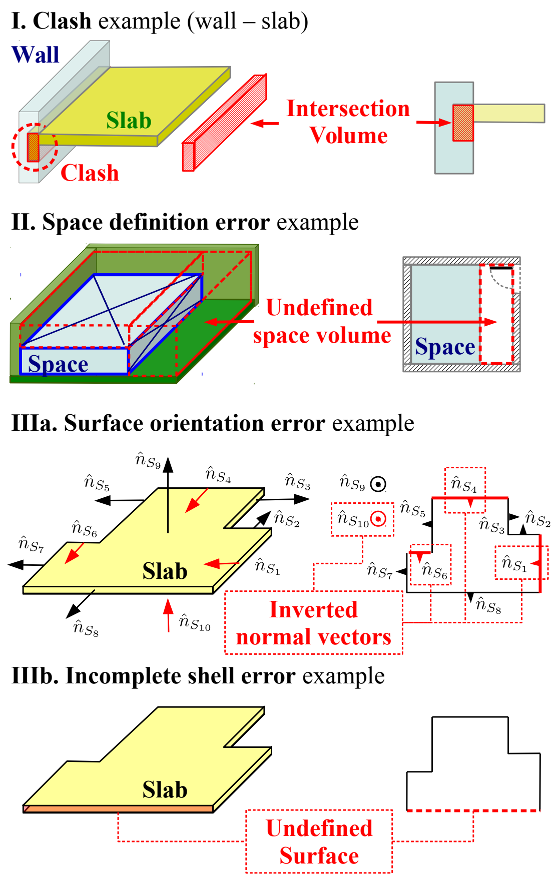
I. Clash example (wall – slab)
Wall
Slab
Clash
Intersection Volume
II. Space definition error example
Undefined space volume
Space
Space
IIIa. Surface orientation error example
Slab
Inverted normal vectors
IIIb. Incomplete shell error example
Slab
Undefined Surface

*Figure 6: **I. Clash error:** A building slab intersects with a building wall. **II. Space definition error:** An internal building space is incorrectly defined leaving an undefined space-volume gap. **IIIa. Space orientation error:** Some of the outward normal vectors of a building B-rep are inverted. **IIIb. Incomplete shell error:** One or more surfaces in the B-rep are not defined.*

### IIIa. Surface orientation errors

Surface orientation errors in IFC geometrical data occur due to bugs of IFC export. These errors appear when the order of the points of a surface polygon of a B-rep is reversed, causing its outward normal vector (calculated using the right hand rule), to point inwards instead of outwards. A surface orientation error creates a misconception on which of the two 3D half-spaces defined by the plane of the surface polygon, is inside or outside the building B-rep. An example of a B-rep with surface orientation errors is displayed in Part III of Figure 6.

### IIIb. Incomplete shell errors

An incomplete shell error type of appears when one or more surfaces of a B-rep of a building solid in an IFC file, is not defined or contain holes. Such case is illustrated in Part IV of Figure 6.

## DETECTION

### Clash error detection

Every pair of building solids (**A**,**B**), can be checked for clashes using the inside filtering function. More precisely is either $\mathcal{A}_{ins} = F_{ins}(T_{\mathbf{B}}, \partial\mathbf{A})$ or $\mathcal{B}_{ins} = F_{ins}(T_{\mathbf{A}}, \partial\mathbf{B})$ is nonempty then a clash is identified. In other words, if there is a part of the B-rep of one of the solids of the pair inside the other solid then these solids intersect, creating a clash error. A wall-slab clash detection example is displayed in part I of Figure 7.

### Space definition error detection

The building spaces which are incorrectly defined are identified by detecting boundary surfaces of space volumes which are not attached to any other building solid (wall, slab, other space, ...). These surfaces are named **space environment** surfaces, are gathered in a surface set $\mathcal{S}_e$ and are identified using the $F_{cod}$ filtering function. Initially, the common boundary surfaces of the B-rep of the space volume under consideration: $\partial\mathbf{S_p}$, with neighbor building solids $\mathbf{B}_j, j = 1, ..., M$ are obtained using $F_{cod}$ and gathered in a common boundary set $\mathcal{C}_b$:

$$\mathcal{C}_b = \bigcup_{j=1}^{M} F_{cod}(T_{\mathbf{B}_j}, \partial\mathbf{S_p}) \tag{1}$$

After the common boundary surfaces are gathered in the set $\mathcal{C}_b$, the space environment surfaces are obtained by subtracting from the B-rep surfaces of the space ($\mathcal{S}_p$) the common boundary surfaces:

$$\mathcal{S}_e = \partial\mathbf{S_p} - \mathcal{C}_b \tag{2}$$

Where the subtraction operation $\partial\mathbf{S_p} - \mathcal{C}_b$ between surface sets $\partial\mathbf{S_p}$ and $\mathcal{C}_b$ is defined as the collection of all possible pairwise surface polygon subtractions, $(S_i - C_j)$:

$$\partial\mathbf{S_p} - \mathcal{C}_b = \bigcup_{\substack{S_i \in \partial\mathbf{S_p} \\ C_j \in \mathcal{C}_b}} (S_i - C_j) \tag{3}$$

where $(S_i - C_j)$ is the polygon difference defined in (Vatti, 1992). In case $S_i, C_j$ are not coplanar then $S_i - C_j = \emptyset$.
At the end of this process if the set $\mathcal{S}_e$ is empty, then the space under consideration is correctly defined, otherwise the set $\mathcal{S}_e$ contains all the surfaces of the space Brep $\partial\mathbf{S_p}$ which are not attached to any other building solid (space-environment surfaces). A space definition error detection process is illustrated in part I of Figure 8.

### Surface orientation and incomplete shell error detection

The boundary surfaces ($S_i$ Figure 2), of a building solid B-rep are correctly oriented, creating a complete shell, when the summation of the products of their normal vectors ($\hat{n}_{S_i}$ Figure 2), weighted by their area $A_i$, returns the zero vector:

$$\sum_{i=1}^{N} A_i \hat{n}_{S_i} = \vec{0} \tag{4}$$

B-reps with incorrectly oriented surfaces (IIIa error) or B-reps defined by incomplete shells (IIIb error) can be detected using the vector sum of (4). If this sum, does not add to the zero vector, one of the two previous error types is identified. To determine the error type (IIIa or IIIb), the correction algorithm described in the surface orientation error correction section is applied on the B-rep. If the sum (4) of the corrected B-rep, adds to the zero vector, then IIIa error type is detected, which is corrected. Otherwise, IIIb error type is detected, which cannot be corrected. Correction of IIIb error type is discussed in (Ju, 2004).

## CORRECTION

### Clash error correction

Building solid pairs $\mathbf{A}, \mathbf{B}$ which intersect are corrected by keeping one solid intact and replacing the other with the difference of itself minus the intact solid. Among the two possible replacements $\mathbf{A} \leftarrow \mathbf{A} - \mathbf{B}$ and $\mathbf{B} \leftarrow \mathbf{B} - \mathbf{A}$ the selected one conforms to either a predefined priority rule or a cardinality-based selection criterion.

A priority rule can be applied when solids $\mathbf{A}$ and $\mathbf{B}$ are related to different building element types, e.g. wall and space types. In this case priority is given to one of the solids and as a result the other one is being replaced.

A cardinality-based selection criterion is based on the comparison of the cardinalities or the number of elements of two sets of surfaces. The first set is the set of surfaces of $\partial\mathbf{A}$ which are inside $\mathbf{B}$ ($\mathcal{S}_{AB}$) and the other set is the set of surfaces of $\partial\mathbf{B}$ which are inside $\mathbf{A}$ ($\mathcal{S}_{BA}$). These sets are obtained by applying the $F_{ins}$ filtering function: $\mathcal{S}_{AB} = F_{ins}(T_{\mathbf{B}}, \partial\mathbf{A})$ and $\mathcal{S}_{BA} = F_{ins}(T_{\mathbf{A}}, \partial\mathbf{B})$. After populating the sets $\mathcal{S}_{AB}$ and $\mathcal{S}_{BA}$ the cardinality based rule can be applied using the condition: if the cardinality of $\mathcal{S}_{AB}$ is bigger than that of $\mathcal{S}_{BA}$ then the replacement $\mathbf{A} \leftarrow \mathbf{A} - \mathbf{B}$ is chosen and vice versa. Cardinality-based rules are used when $\mathbf{A}$ and $\mathbf{B}$ refer to the same building entit, e.g. in a wall-wall clash error.

The new B-rep of the difference $\mathbf{A}-\mathbf{B}$ can be obtained by generating the following set of surfaces:

$$\partial(\mathbf{A}-\mathbf{B}) = \left\{ \begin{array}{l} F_{out}(T_{\mathbf{B}}, \partial\mathbf{A}), \\ F_{cod}(T_{\mathbf{B}}, \partial\mathbf{A}), \\ F_{ins}(T_{\mathbf{A}}, \partial\mathbf{B})^{-1} \end{array} \right\} \tag{5}$$

The superscript $-1$ in $F_{ins}(T_{\mathbf{A}}, \partial\mathbf{B})$ essentially reverses the orientation of the surface polygons returned by $F_{ins}$. If the replacement $\mathbf{B}-\mathbf{A}$ is selected its B-rep is obtained by (5) with the roles of $\mathbf{A}$ and $\mathbf{B}$ reversed.

The two replacement options of a wall-slab clash correction example are demonstrated in Parts II(a) and II(b) in Figure 7.

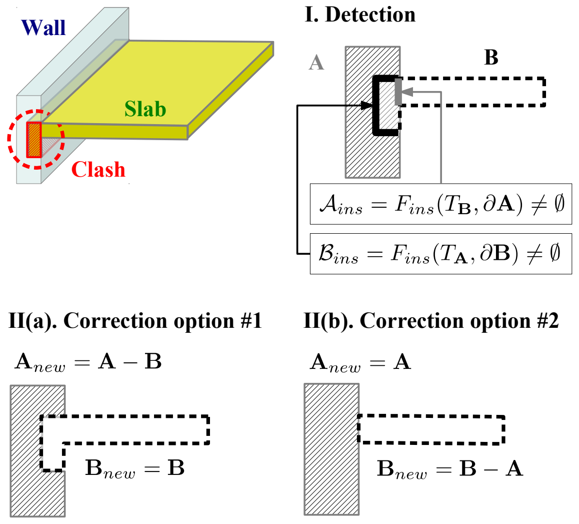
Wall
Slab
Clash
I. Detection
A
B
$\mathcal{A}_{ins} = F_{ins}(T_{\mathbf{B}}, \partial\mathbf{A}) \neq \emptyset$
$\mathcal{B}_{ins} = F_{ins}(T_{\mathbf{A}}, \partial\mathbf{B}) \neq \emptyset$
II(a). Correction option #1
$\mathbf{A}_{new} = \mathbf{A} - \mathbf{B}$
$\mathbf{B}_{new} = \mathbf{B}$
II(b). Correction option #2
$\mathbf{A}_{new} = \mathbf{A}$
$\mathbf{B}_{new} = \mathbf{B} - \mathbf{A}$

*Figure 7:* ***Clash error detection and correction. I. Detection:*** *The surface sets returned by $F_{ins}$ are nonempty.* ***II(b). Correction option #1:*** *$\mathbf{A}$ is replaced by $\mathbf{A} - \mathbf{B}$.* ***II(b). Correction option #2:*** *$\mathbf{B}$ is replaced by $\mathbf{B} - \mathbf{A}$.*

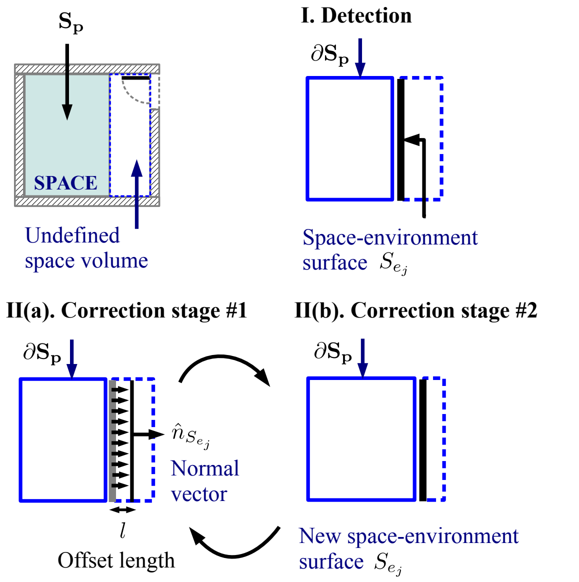

*Figure 8:* ***Space definition error detection and correction. I. Detection:*** *Space environment surfaces are extracted.* ***II(a). Correction stage #1:*** *The internal space is inflated towards the normal vector of every space-environment surface, for offset length l.* ***II(b). Correction stage #2:*** *New space-environement surfaces are extracted after the inflation.*

### Space definition error correction

Space volumes which are incorrectly defined, can be corrected by an iterative process which consists of two sequential stages: the **space-environment surface extraction** process (which is described in Section  and aims at determining the space environment surface of a given building space volume) and a **space inflation process** (which inflates a given space volume towards a direction vertical to the space environment surfaces obtained from the first stage). The process of inflation and new space-environment surface extraction is repeated until a no space-environment surfaces exist or a maximum number of iteration steps is reached. The two stage space definition error correction process is illustrated in parts II(a) and II(b) in Figure 8.

### Surface orientation error correction

If the boundary surface of a B-rep of a building solid are incorrectly oriented, the sum of (4) diverges from zero. This is due to the fact that some normal vectors point inwards instead of outwards. Generally a B-rep consisting of N boundary polygons can have $2^N$ possible surface orientations, out of which one is desired (all normal vectors point outwards). If the B-rep surfaces are not oriented properly a finite number of normal vector reversals are required in order to select the correct orientation combination out of the $2^N$ possible ones. Additionally, the case in which all normal vectors point inwards also give zero vector sum in (4) and should be avoided.

To address the previous issues, a recursive orientation correction algorithm have been developed which reduces the size of the solution space $2^N$ and excludes the inward normal direction solution option.

Initially no prior knowledge of any surface orientation is assumed and all the boundary surfaces are gathered in a "remaining" surface set $\mathcal{S}_{rem}$. As the algorithm progresses, surfaces with corrected orientation are collected in the "corrected" surface set $\mathcal{S}_{cor}$. The algorithm calls sequentially two functions: the convex partition set function $F_{cps}$ and then the connected surface set function $F_{css}$, as illustrated on a non-convex solid in part II of Figure 9. The index of $F_{cps}$ calls alternates form 1 to -1 with a value of 1 in the first $F_{cps}$ call.

$F_{cps}$ receives as input the corrected set $\mathcal{S}_{cor}$, a set of surfaces to be corrected $\mathcal{S}_c$ and a correction index which has a $\pm 1$ value. $F_{cps}$ isolates the surfaces of $\mathcal{S}_c$ which belong to the convex hull of $\mathcal{S}_c$ and sets their vector to point to a half space away from the convex hull if the correction index is 1 and towards the convex hull if the correction index is -1. After the normal vector of the convex hull surfaces of $\mathcal{S}_c$ is corrected, the convex hull surfaces are added to the corrected surface set $\mathcal{S}_{cor}$. The remaining surfaces (not corrected) which do not belong to the convex hull of $\mathcal{S}_c$ remain in $\mathcal{S}_c$.

$F_{css}$ receives as input the remaining surface set $\mathcal{S}_{rem}$ or the set to be corrected $\mathcal{S}_c$ and partitions it further into surface subsets which contain surfaces that are connected to each other by a common point or edge. The surfaces which belong to different subsets are not connected.

In the first step of the algorithm $F_{cps}$ is applied with input $\mathcal{S}_{cor} = \emptyset$, $\mathcal{S}_c = \partial \mathbf{S}$ and index value 1 and the surfaces which belong to the convex hull of the solid B-rep, are extracted and corrected. The remaining surfaces in $\mathcal{S}_{rem}$ are partitioned further into surface subsets to be corrected ($\mathcal{S}_c$'s) (part II(a) Figure 9). The process is repeated in the second step: the convex hull surfaces of the surface subset to be corrected ($\mathcal{S}_c$'s) are isolated and corrected using $F_{cps}$ with index -1 and the updated ($\mathcal{S}_c$'s) after the correction are partitioned further into new ($\mathcal{S}_c$'s) (part II(b) Figure 9). The algorithm terminates when all $\mathcal{S}_c$'s are empty and the corrected surface set contains all of the B-rep surfaces of the solid.

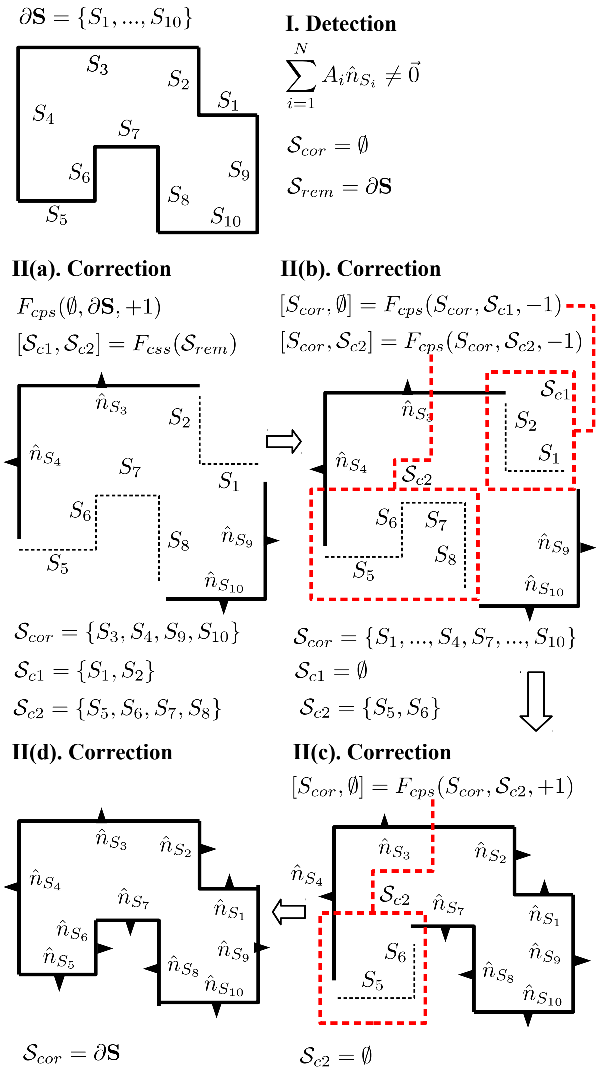
$\partial \mathbf{S} = \{S_1, ..., S_{10}\}$
I. Detection
$\sum_{i=1}^{N} A_i \hat{n}_{S_i} \neq \vec{0}$
$\mathcal{S}_{cor} = \emptyset$
$\mathcal{S}_{rem} = \partial \mathbf{S}$
$S_1$ $S_2$ $S_3$ $S_4$ $S_5$ $S_6$ $S_7$ $S_8$ $S_9$ $S_{10}$
II(a). Correction
$F_{cps}(\emptyset, \partial \mathbf{S}, +1)$
$[\mathcal{S}_{c1}, \mathcal{S}_{c2}] = F_{css}(\mathcal{S}_{rem})$
$\mathcal{S}_{cor} = \{S_3, S_4, S_9, S_{10}\}$
$\mathcal{S}_{c1} = \{S_1, S_2\}$
$\mathcal{S}_{c2} = \{S_5, S_6, S_7, S_8\}$
II(b). Correction
$[\mathcal{S}_{cor}, \emptyset] = F_{cps}(\mathcal{S}_{cor}, \mathcal{S}_{c1}, -1)$
$[\mathcal{S}_{cor}, \mathcal{S}_{c2}] = F_{cps}(\mathcal{S}_{cor}, \mathcal{S}_{c2}, -1)$
$\mathcal{S}_{cor} = \{S_1, ..., S_4, S_7, ..., S_{10}\}$
$\mathcal{S}_{c1} = \emptyset$
$\mathcal{S}_{c2} = \{S_5, S_6\}$
II(d). Correction
$\mathcal{S}_{cor} = \partial \mathbf{S}$
II(c). Correction
$[\mathcal{S}_{cor}, \emptyset] = F_{cps}(\mathcal{S}_{cor}, \mathcal{S}_{c2}, +1)$
$\mathcal{S}_{c2} = \emptyset$

*Figure 9: **Surface orientation errors detection and correction. I. Detection:** The vector sum of the surfaces does not add to zero. **II. Correction :** Convex hull normal corrections ($F_{cps}$) and connected surface set partitioning ($F_{css}$) are applied sequentially.*

## EXAMPLES

The proposed detection and correction algorithms were demonstrated on the geometrical data of the IFC files of two office buildings: one located in CARTIF research center in Spain and one in Technical University of Crete in Greece, illustrated in Figure 10. The

data were queried from the IFC files using the BIM server developed by TNO (Beetz et al., 2010).
Five wall-wall clash errors were detected in IFC file of CARTIF building (A,B,C,D and E Part I of Figure 11) and corrected using a cardinality based priority rule in the introduced clash correction algorithm. As it is displayed in part II of figure 11 the correction algorithm restored the wall volumes, in all of the five wall conflicts.
A space definition error was detected in TUC building (Part I, Figure 12) in a non-convex space volume in the first floor of the building. The associated space volume gap was eliminated, by the space definition error correction process as displayed in Part II of Figure 12.

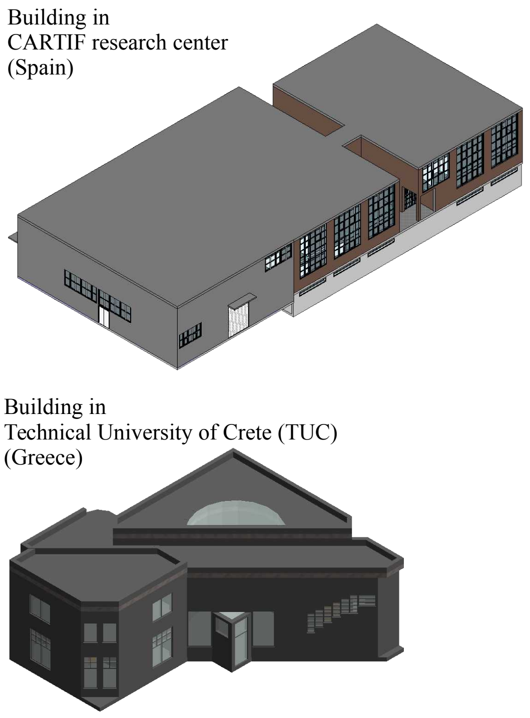

*Figure 10: Demonstration buildings.*

Finally, surface orientation errors were detected in all of the B-reps of the suspended ceilings of CARTIF building, as displayed in Part I of Figure 13. The orientation errors were identified at the surfaces along the small dimension of these B-reps (z-axis), were the respective normal vectors were pointing inwards. The orientation was reversed were necessary and the direction of the respective normal vectors was corrected as displayed in Part II of Figure 13.

## CONCLUSIONS

The accuracy of the geometrical data contained in IFC files is of paramount importance in any building simulation model generation process which is based on the definition of a buildings' second level space boundary topology. This accuracy can be ensured by identifying and removing, using the proposed tools, three types of geometrical errors which appear in IFC files due to either exporting program flaws, or design inaccuracies. These errors involve: **clash errors** where solid representations of architectural elements intersect, **space definition errors** where building space volumes are not attached to all of their surrounding building elements and **surface orientation errors** where the orientation of some of the surfaces of boundary representations of building solids is reversed.

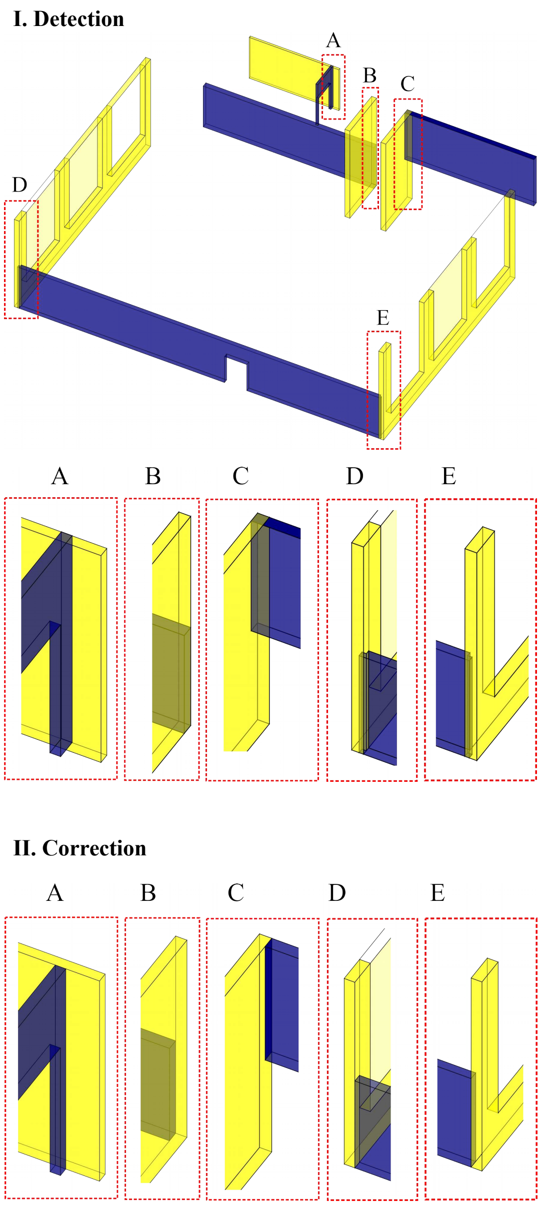

*Figure 11: Wall-wall class error example in CARTIF building: **detection (I)** and **correction (II)**.*

Although, **incomplete shell errors**, where boundary surfaces are missing from building B-reps, can be detected using the proposed algorithms, the correction of these error types, is a subject of future work. Such processes contribute significantly towards automating the process of building energy model creation using IFC data.

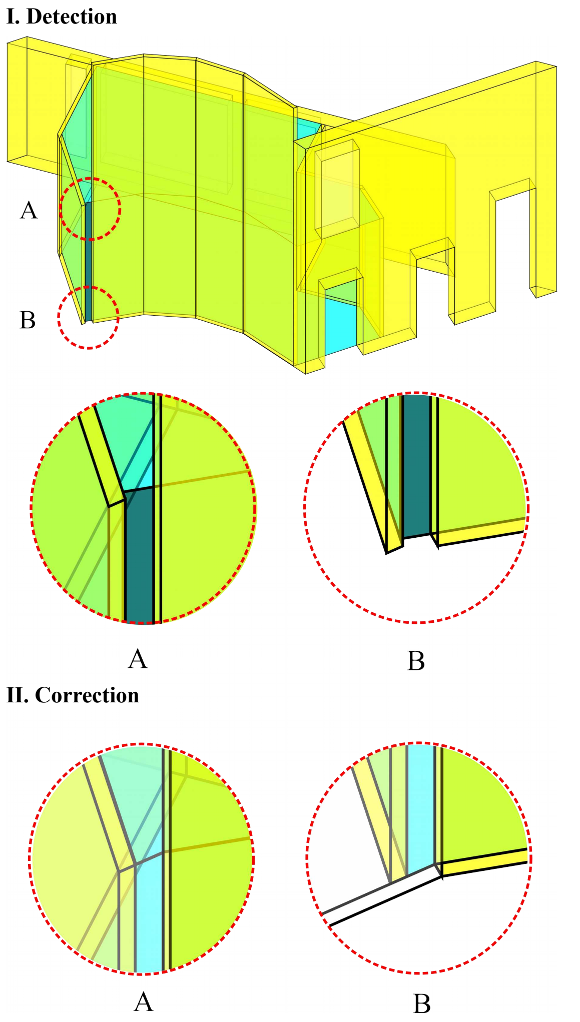

*Figure 12: Space definition error example in TUC building: **detection (I)** and **correction (II)**.*

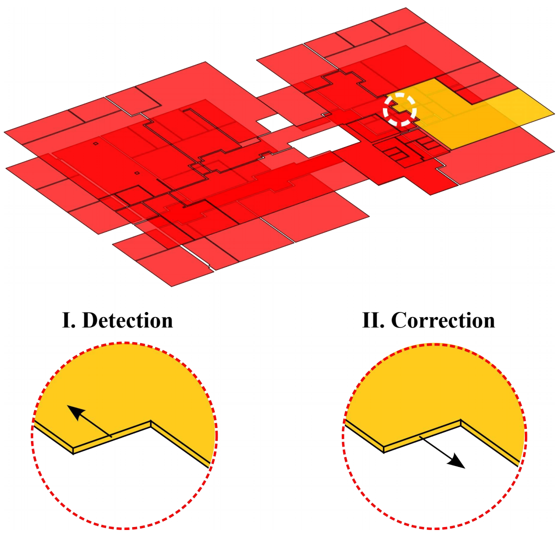

*Figure 13: Surface orientation error example in CARTIF building: **detection (I)** and **correction (II)**.*

## ACKNOWLEDGMENTS

This research has been partially funded by the European Commission: FP7-ICT-2011-6, ICT Systems for Energy Efficiency #288409 (BaaS) and Innovative design tools for refurbishing of buildings at district level H2020-EeB-05-2015 #680676 (Opteemal).

## REFERENCES

Barequet, G., Duncan, C. A., and Kumar, S. 1998. RSVP: A geometric toolkit for controlled repair of solid models. *IEEE Trans. on Visualization and Computer Graphics*, 4(2):162–177.

Barequet, G. and Kumar, S. 1997. Repairing CAD models. In *Visualization'97*, pages 363–370. IEEE.

Bazjanac, V. 2010. Space boundary requirements for modeling of building geometry for energy and other performance simulation. In *27th CIB W78 International Conference*, Cairo, Egypt.

Beetz, J., Van Berlo, L., de Laat, R., and Van den Helm, P. 2010. BIMserver. org - an open source IFC model server. In *27th CIB W78 International Conference*, Cairo, Egypt.

Jackson, D. 1995. Boundary representation modelling with local tolerances. In *3rd ACM symposium on Solid modeling and applications*, pages 247–254. ACM.

Ju, T. 2004. Robust repair of polygonal models. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 888–895. ACM.

Khemlani, L. 2002. Solibri model checker. *CADENCE-AUSTIN*, pages 32–34.

Klosowski, J. T., Held, M., Mitchell, J. S., Sowizral, H., and Zikan, K. 1998. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Trans. on Visualization and Computer Graphics*, 4(1):21–36.

TEKLA 2014. Model checker suite. ”`http://teklastructures.support.tekla.com/190/en/ext_model_checker_suite`”.

Thibault, W. and Naylor, B. 1987. Set operations on polyhedra using binary space partitioning trees. In *14th annual conference on Computer graphics and interactive techniques*, volume 21, pages 153–162. ACM.

ISO-16739 2013. ISO 16739: Industry foundation classes (IFC) for data sharing in the construction and facility management industries. *European Committee for Standardization (CEN), Brussels.*

van den Helm, P., Böhms, M., and van Berlo, L. 2010. IFC-based clash detection for the open-source bimserver. In *Int. Conference in Computing in civil and building engineering*, volume 30, pages 181–187, Nottingham, UK.

Vatti, B. 1992. A generic solution to polygon clipping. *Communications of the ACM*, 35(7):56–63.