International Journal of
Geo-Information

Article

Extrusion Approach Based on Non-Overlapping Footprints (EABNOF) for the Construction of Geometric Models and Topologies in 3D Cadasters

Yuan Ding [1,2,3], Nan Jiang [1,2,3], Zhaoyuan Yu [1,2,3], Binqing Ma [1,2,3], Ge Shi [1,2,3] and Changbin Wu [1,2,3,*]

1 Key Laboratory of Virtual Geographic Environment (Nanjing Normal University), Ministry of Education, Nanjing 210023, Jiangsu, China; 141301018@stu.njnu.edu.cn (Y.D.); njiang@njnu.edu.cn (N.J.); yuzhaoyuan@njnu.edu.cn (Z.Y.); 131302082@stu.njnu.edu.cn (B.M.); 161301020@stu.njnu.edu.cn (G.S.)
2 College of Geographic Sciences, Nanjing Normal University, Nanjing 210023, Jiangsu, China
3 Jiangsu Center for Collaborative Innovation in Geographic Information Resource Development and Application, Nanjing 210023, Jiangsu, China
* Correspondence: wuchangbin@njnu.edu.cn

Received: 25 May 2017; Accepted: 2 August 2017; Published: 3 August 2017

Abstract: Extrusion is widely used to construct models in 3D cadasters. However, the basic extrusion approach only supports relatively simple conditions, and a 3D cadastral data model that supports extruded 3D models that are associated with their corresponding footprints is not available. In this paper, we present a new extrusion approach based on non-overlapping footprints (EABNOF) that supports relatively complex 3D situations. In EABNOF, overlaps between overlapping footprints of the input data are removed, which also involves splitting extrusion intervals and handling the associated cadastral objects of footprints. The newly generated non-overlapping footprints are extruded to generate primitives. To construct geometric models and topologies for cadastral objects, three judgment criteria are proposed to identify and remove redundancies from these primitives, and then primitives of the same 3D spatial unit or topological feature are merged. Considering the feasibility of using EABNOF for current cadastral data, we design a data model that associates 3D cadastral data with the footprints of 2D cadasters. We examine two types of structures on Pozi Street to verify EABNOF: a building complex and property objects. The results demonstrate that EABNOF can construct geometric models and topologies in 3D cadasters. EABNOF is based on the footprints of 2D cadastral data, and thus is particularly suited to areas with 2D cadastral data to establish 3D cadasters with low costs.

Keywords: 3D cadaster; extrusion; geometric model; topology

1. Introduction

High-intensity land use necessitates the development of 3D space. In 3D cadasters, 3D parcels and other 3D cadastral objects constitute the basic spatial units that are used to manage 3D space. Modeling these objects involves a direct relationship with the implementation of 3D cadasters [1]. Numerous approaches are available to construct geometric models in 3D cadasters. Modeling from 3D surveying data (i.e., aerial LIDAR data) provides accurate and realistic 3D models, but surveying and algorithms for these data are difficult [2]. Moreover, this modeling approach disregards current 2D cadastral data and the costs are prohibitive in developing countries, such as China, in which most cadastral systems have been established over the past 10–20 years. Constructing geometric models in 3D cadasters should involve fully considering current 2D cadastral data. Extrusion is the simplest approach that can be used to automatically construct 3D models [3,4]. This approach involves using a set of footprints that represent projections of buildings and their corresponding height values. In turn, 3D models can be generated by extruding footprints. The extrusion approach is depicted in Figure 1. Extrusion is extensively used in many fields because of its simplicity. In CityGML, which is the international open standard for 3D city models, the well-known blocks model (lod1Solid) in LOD1 is generated by extrusion [5]. In 3D cadasters, constructing an exact model of a building or apartment is expensive; therefore, property spaces that consist of vertical and horizontal faces are often used to represent the 3D spatial extent. Footprints can be drawn from 2D cadastral data, so constructing geometric models via extrusion in 3D cadasters is feasible. Several studies on 3D cadastral modeling have involved the use of extrusion methods [6–10].

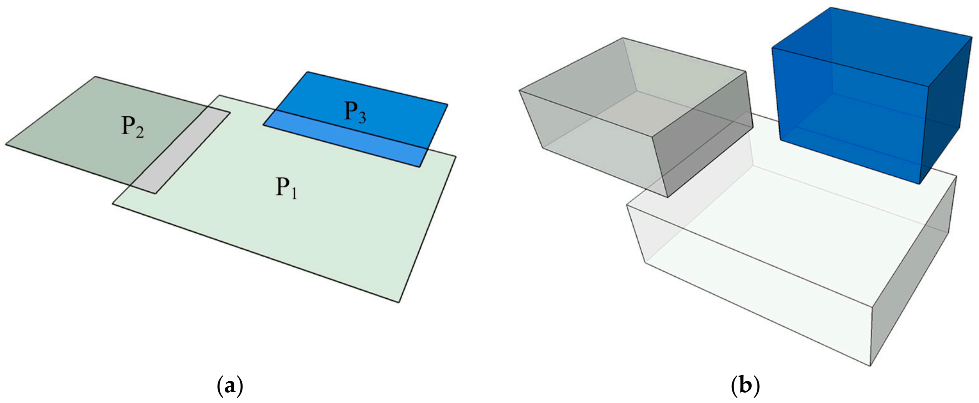


Figure 1. (**a**) Three overlapping 2D parcels in 3D space; and (**b**) 3D parcels generated by extruding 2D parcels in (**a**).

However, extrusion also presents limitations when applied in 3D cadasters. One of the greatest limitations is that if the projections of 3D objects have overlapping components, different footprints can also overlap. In the CityGML geometry model, the *MultiSurface* type provides support for overlapping footprints [5]. However, overlaps are not allowed for the same type of spatial unit in 2D cadastral maps because they may lead to ambiguities. For example, a parcel is not allowed to overlap any other parcel. Such a case is depicted in Figure 1, in which three 2D parcels (P_1, P_2 and P_3) overlap (Figure 1a). The three 2D parcels are used as footprints to construct models of 3D parcels (Figure 1b). If information on 3D parcels is not available, then misunderstandings that are related to topology errors between the 2D parcels (P_1 and P_2 or P_1 and P_3) may occur.

Topology is important for 3D cadastral data organization and spatial querying [11–13] and allows land administrators to determine the boundaries and extents of 3D spatial units. Many researchers have fully considered the topology of 3D cadasters [14–17]. The international standard ISO 19152 Land Administration Domain Model (LADM) [18] contains a "topology-based" spatial unit that is used when spatial units share boundary representations. This "topology-based" spatial unit reflects 3D topology. This unit is encoded in reference to its boundaries and with a common boundary (boundary point, boundary line or boundary face) between two adjacent spatial units and is stored only once [19]. Although many 3D topological models exist that consider 3D topology [20–22], only a few approaches implement 3D topology and extrusion-based 3D models [3,23]. In addition, these approaches only support simple 3D models.

This paper describes (1) an extrusion approach based on non-overlapping footprints (EABNOF) for constructing geometric models and topologies in 3D cadasters and (2) a 3D cadastral data model for applying our EABNOF in practice. EABNOF mainly consists of two components: one is removing the overlaps between footprints (footprint points, edges and faces), in which extrusion intervals and cadastral objects are considered; the other is constructing geometric models and topologies, in which redundancies are removed and the primitives of a single topological feature or 3D spatial unit are merged together.

The remainder of this paper is organized as follows. Section 2 presents a review of previous studies that are related to our EABNOF, which mainly include extrusion approaches for constructing 3D models in Geographic Information Systems (GIS) and 3D cadasters. Section 3 describes a 3D cadastral data model for our EABNOF. In Section 4, we provide a detailed description of our EABNOF. Section 5 describes several implementation details for EABNOF. In Section 6, we illustrate the feasibility of our EABNOF through two case studies of Pozi Street, which is the most famous landmark with commercial and residential space in Taizhou, Jiangsu, China.

2. Related Work

Considering the importance of topology in GISs, the ISO 19107 Spatial Schema [24], which is the most important international standard for describing and manipulating the spatial characteristics of geographic features, contains topology packages for geometric features. In these packages, n-dimensional topological primitives are constructed from $(n - 1)$-dimensional primitives. Several topological primitives can be aggregated into a topological complex. Each topological primitive has a geometric realization. These packages can support 3D topology. Additionally, many other frameworks support 3D topology, such as the formal data structure (FDS) [21], the Tetrahedral Network (TEN) [22], the Simplified Spatial Model (SSM) [4], the Urban Data Model (UDM) [20] and others.

As previously stated, footprint extrusion is a simple approach to construct 3D models. The commonly used GIS platform ArcGIS provides an extrusion function through ArcScene [25]. The 3D spatial engine of Oracle 11g can also extrude 2D footprints to 3D solid geometries [26]. Additionally, the 3D modeling software 3D Studio Max has an extrusion function to generate 3D models. When using these software products, the topological relationships between 3D geometries are not represented [3]. Google SketchUp supports the topological relationships between 3D geometries but does not support 3D solids [27,28]. To construct a geometric model for a relatively complex object, such as a building with different structures on each floor, these software products require the use of several footprints of different heights. However, these footprints cannot be accurately represented in a 2D plane because they overlap one another.

In CityGML, a building is represented through the blocks model (lod1Solid) in LOD1. One of the simplest methods of generating this structure involves extruding the footprint (lod0FootPrint) of a building in LOD0. The blocks model and footprint of a building are unified in CityGML and thus can be simultaneously represented. However, building footprints in LOD0 (Figure 2a) often describe buildings with different height values. In LOD1 (Figure 2b), these buildings should be modeled by extruding several footprints that are different from those in LOD0. Therefore, topological connections cannot be created between LOD0 footprints and the LOD1 blocks model in CityGML. Figure 2a,b show the LOD0 footprint and LOD1 model.

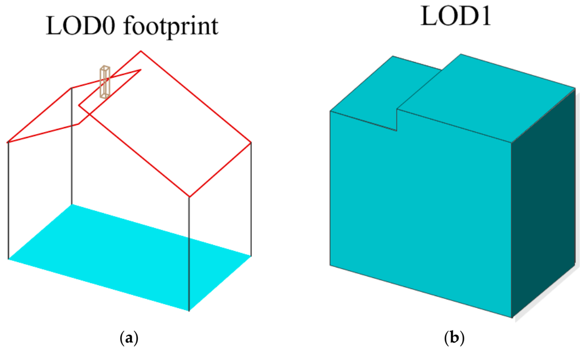


Figure 2. LOD0 footprint (**a**) and LOD1 blocks model (**b**) in CityGML (images: KIT Karlsruhe [5]).

Gröger and Plümer [23] described an approach to obtain 3D models of urban GISs based on 2D maps. This approach extends 2D and 2.5D models to a 2.8D model by allowing for overhangs and vertical walls. The 2.8D model is an extension of a digital elevation model (DEM) [29] and allows each (x, y) location to have more than one height value. To guarantee consistency, the faces of an object cannot overlap. The topology and consistency of the 2.8D model is maintained by using a nested map. Thus, we can construct most objects in a 3D GIS. However, a vertical wall between two connected buildings cannot be represented (Figure 3).

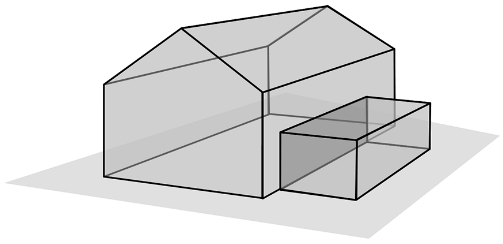


Figure 3. Vertical common wall between two connected buildings.

Ledoux and Meijers [3] proposed an extrusion approach for 3D city models that considers the topological consistency between different objects. This approach uses 2D topologically consistent data as the input and outputs 3D models with consistent topology. This approach can manage a polygon with holes and islands and therefore can represent the inner components of 3D objects, as shown in Figure 3. Arroyo, Ledoux and Stoter [30] presented an extrusion algorithm for a higher-dimensional model based on a generalized map. In this algorithm, a cell complex is used to decompose a topological space into cells. An *n*D cell complex is constructed by using ($n-1$) cells and extrusion intervals, and each ($n-1$)-cell is allowed to have multiple extrusion intervals. These authors designed an algorithm to propagate extrusion intervals from higher-dimensional cells to lower-dimensional cells and split the intersecting extrusion intervals. The topological relationships between higher-dimensional cells are represented by these lower-dimensional cells. Although the above two approaches support topological consistency between different 3D objects, they do not consider that a 3D model can be generated by extruding several overlapping footprints.

Boolean set operations are widely used in 3D modeling. Constructive solid geometry (CSG) [31], which was proposed by Requicha and Voelcker, constructs a complex object by using a tree of Boolean set operations. This approach has many practical applications in computer-aided designs and game engines, such as AutoCAD and the Unreal engine. Binary Space Partitioning (BSP) Trees [32] constructs a polyhedron by recursively partitioning subspaces, which can be used to speed up the intersection tests in CSG. However, this method does not support topology. Gursoz, Choi and Prinz [33] developed an algorithm for non-manifold boundary models that can perform Boolean operations between objects of different dimensionality and solids. The basic idea of this algorithm is to systematically handle singular intersections in an ordered manner from vertex to edge and face elements. Additional modeling approaches that involve Boolean set operations can be found in [34–37]. Few studies have examined Boolean set operations between extruded based models because these Boolean set operations can be converted to Boolean set operations between footprints.

Many studies split footprints into several components before extrusion to improve the accuracy of reconstructed 3D models. Kada and McKinley [38] automatically reconstructed 3D building models from LIDAR data and existing ground plans. These authors use a method called cell decomposition, which uses line segments of buildings' outlines to decompose footprints into pieces and provide each piece a roof shape through parameter estimation based on LIDAR points. Vallet et al. [39] presented a framework to improve building-footprint databases that consisted of 2D polygons. These authors split each polygon into several simple polygons through a DEM and merged the split polygons to minimize their number. These splitting and merging operations were based on a Mumford and Shah-like energy function to characterize the quality of the segmentation. Commandeur [40] proposed an approach to model 3D objects that involves extruding decomposed footprints. This method includes decomposing a single footprint of a building via generalized line equations. In the above three studies, each footprint is used to construct a single volume, and these studies do not consider footprints that are extruded along several extrusion intervals to generate more than one volume. Additionally, models that are constructed by several overlapping footprints were not considered in these studies. Moreover, these footprints are approximately split and thus can be used for accurate 3D cadastral management.

Extrusion is a common approach in 3D cadasters to model 3D spatial units. The property space can be easily modeled for buildings in which each floor has the same structure, as has been noted in many other papers [10,28,41]. For a relatively complex building with floors that have different structures, the footprints of different floors can be extruded [7]. However, this approach often results in overlaps between footprints. Boolean set operations can be used as a complementary approach to extrusion when modeling irregular 3D cadastral objects [42]; however, model data that are constructed through this approach cannot be associated with 2D cadastral data.

3. 3D Cadastral Data Model for EABNOF

This section introduces a 3D cadastral model that we designed for EABNOF. This model is oriented towards the application of EABNOF in a 3D cadastral system. The model consists of three interconnected components: geometry, entity and topology components. In this model, both 2D and 3D spatial units are supported and 3D topology is supported. Considering the versatility of this model, we use classes in ISO 19107 to define the geometry of our classes. This model is shown in Figure 4 through a Unified Modeling Language (UML) class diagram.

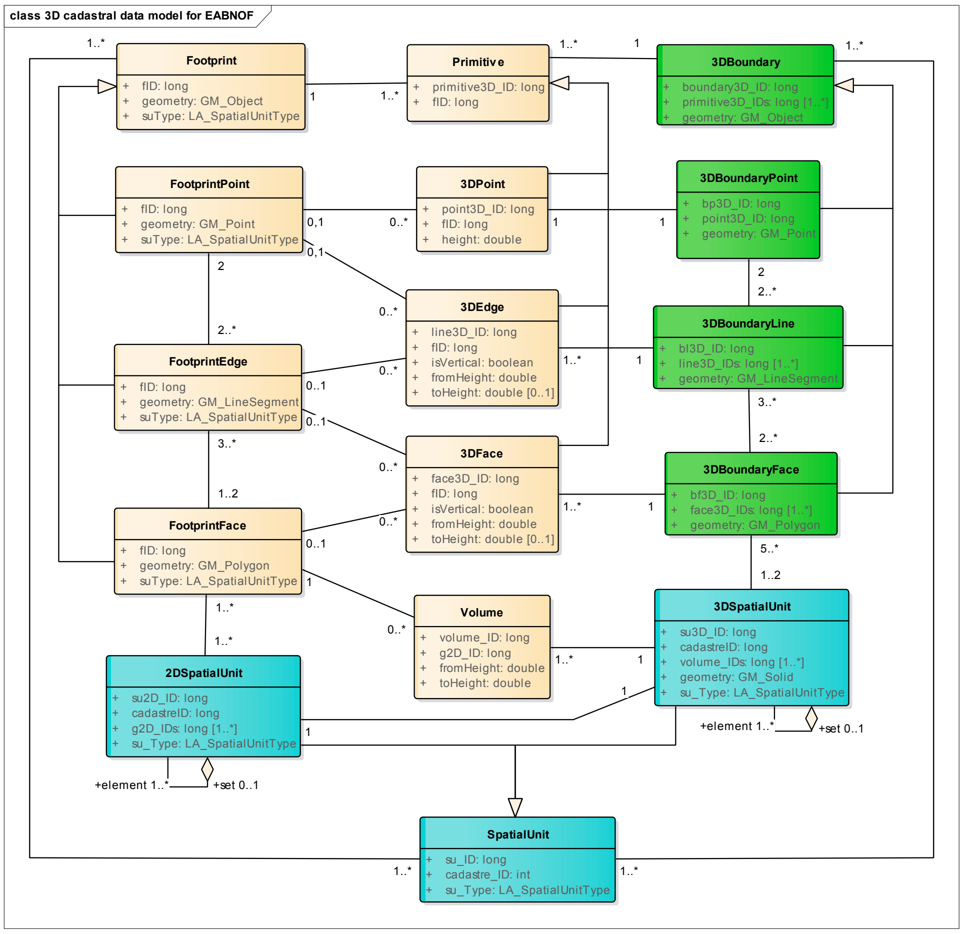


Figure 4. 3D cadastral data model for EABNOF.

3.1. Geometry

The geometry component provides geometric objects that are used to construct 2D/3D spatial units and 3D topological features. This component contains the classes *FootprintPoint*, *FootprintEdge*, *FootprintFace*, *3DPoint*, *3DEdge*, *3DFace* and *Volume*. *FootprintPoint*, *FootprintEdge* and *FootprintFace* are footprints of different dimensions that are embedded in a 2D plane, and *3DPoint*, *3DEdge*, *3DFace* and *Volume* are primitives of different dimensions that are embedded in 3D space. The geometries are only recorded in the three footprint classes (*FootprintPoint*, *FootprintEdge*, *FootprintFace*), and the four primitives (*3DPoint*, *3DEdge*, *3DFace* and *Volume*) record the references to footprint classes and extrusion intervals.

1. ***Footprint***: This class is the base class of all the footprint classes (*FootprintPoint*, *FootprintEdge* and *FootprintFace*).
2. ***FootprintPoint***: A footprint point is a 0D geometry that is embedded in a 2D plane. This class has X, Y coordinates, and its geometry type is ISO 19107 GM_Point in 2D.
3. ***FootprintEdge***: A footprint edge is a 1D geometry that is embedded in a 2D plane. This class is a straight-line segment that is bounded by two footprint points, and its geometry type is ISO 19107 GM_LineSegment in 2D.
4. ***FootprintFace***: A footprint face is a 2D geometry that is embedded in a 2D plane. This class is a polygon that is bounded by three or more footprint edges. A footprint face has only one outer boundary and zero or more inner boundaries. Thus, this class can have holes but no interior points of self-intersection. This class' geometry type is ISO 19107 GM_Polygon in 2D.
5. ***Primitive***: This class is the base class of all the primitive classes (3D point, 3DEdge, 3DFace and Volume).
6. ***3DPoint***: A 3D point is a type of 0D primitive that is embedded in 3D space. This class has X, Y, and Z coordinates and is obtained by adding a Z value to the footprint point.
7. ***3DEdge***: A 3D edge is a type of 1D primitive that is embedded in 3D space. This class is a straight line segment and its boundary is a set of two associated 3D points. Two types of 3D edges exist: a horizontal 3D edge and vertical 3D edge. A horizontal 3D edge is generated by adding the same Z value to the two nodes of a footprint edge, and a vertical 3D edge is constructed via extruding a footprint point along an extrusion interval.
8. ***3DFace***: A 3D face is a type of 2D primitive that is embedded in 3D space. This class is a polygon and its boundary is a set of three or more 3D edges. This class has only one outer boundary and zero or more inner boundaries and thus can include holes. Two types of 3D faces exist: a horizontal 3D face and vertical 3D face. A horizontal 3D face is generated by adding the same Z value to the vertices of a footprint face, and a vertical 3D face is constructed by extruding a footprint edge along an extrusion interval.
9. ***Volume***: A 3D volume is a type of 3D primitive that is embedded in 3D space. This class is a prism and its boundary is a set of five or more 3D faces. This class has only one outer boundary (shell) and no inner boundary (shell).

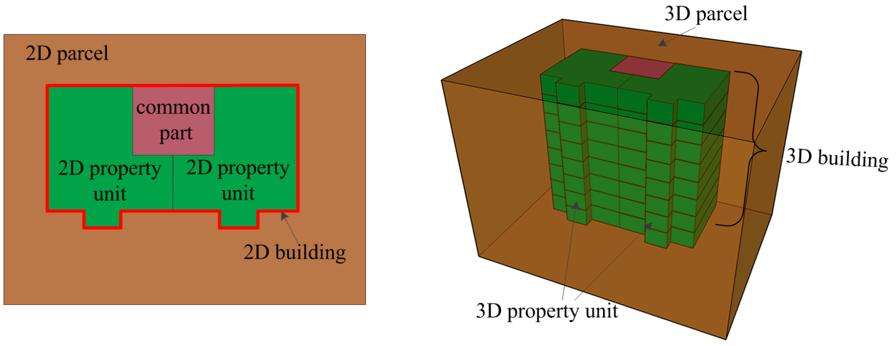
If a 3D edge or 3D face is vertical, its field *isVertical* should be set to true and the fields *fromHeight* and *toHeight* should be assigned the minimum and maximum values of the corresponding extrusion interval. If a 3D edge or 3D face is horizontal, its field *isVertical* should be set to false, the field *fromHeight* is assigned a Z value and the field *toHeight* should be empty. The Z value is obtained from the boundary value of the extrusion interval.

3.2. Entity

The entity component contains the classes *2DSpatialUnit* and *3DSpatialUnit* and records and manages the spatial units of 2D and 3D cadasters. These two classes are constructed from the classes *FootprintFace* and *Volume,* respectively.

1. ***SpatialUnit:*** This class is the base class of the classes *2DSpatialUnit* and *3DSpatialUnit*.
2. ***2DSpatialUnit***: A 2D spatial unit can be a 2D parcel, building, property unit, etc. (Figure 5a). All types of 2D spatial units share the same class *2DSpatialUnit*. The geometry is not recorded in *2DSpatialUnit.* Its geometry is represented by the combination of footprint faces.
3. ***3DSpatialUnit***: A 3D spatial unit can be a 3D parcel, 3D property unit, etc. (Figure 5b). All types of 2D spatial units share the same class *2DSpatialUnit*. Its geometry type is ISO 19107 GM_Solid. Its geometry is constructed from a set of volumes.

A cadastral object (i.e., a parcel) may represented by both 2D and 3D spatial units. Thus, *2DSpatialUnit* and *3DSpatialUnit* are associated through the field cadastreID. Figure 5 shows a case with a parcel and a building. The building consists of property units and common components. In Figure 5a, the parcel, building and property units are represented by 2D spatial units (2D parcel, 2D building and 2D property unit). In Figure 5b, the parcel, building and property units are represented by 3D spatial units (3D parcel, 3D building and 3D property unit).



(**a**) (**b**)

Figure 5. Representations of 2D spatial units (**a**) and 3D spatial units (**b**) for the same cadastral objects.

3.3. Topology

The topology component records and manages the topological features of 3D spatial units and includes the classes *3DBoundaryPoint*, *3DBoundaryLine* and *3DBoundaryface*.

1. ***3DBoundary:*** This class is the base classes of all the 3D boundary classes (*3DBoundaryPoint*, *3DBoundaryLine* and *3DBoundaryFace*).
2. ***3DBoundaryPoint***: A 3D boundary point represents a 0D boundary of a 3D spatial unit or a 0D common boundary between 3D spatial units. The geometry of *3DBoundaryPoint* is constructed from a 3D point, and its type is ISO 19107 GM_point in 3D.
3. ***3DBoundaryLine***: A 3D boundary line represents a 1D boundary of a 3D spatial unit or a 1D common boundary between 3D spatial units. The geometry of *3DBoundaryLine* is constructed from one or more 3D edges, and its type is ISO 19107 GM_LineSegment in 3D.
4. ***3DBoundaryFace***: A 3D boundary face represents a 2D boundary of a 3D spatial unit or a 2D common boundary between 3D spatial units. The geometry of *3DBoundaryFace* is constructed from one or more 3D faces, and its type is ISO 19107 GM_Polygon in 3D.

3.4. Limitations of the Data Model

In the real world, many cadastral/property objects are not limited to simple volumes, such as block models in LOD1 (CityGML); therefore, we often use an approximation to represent such objects. Whether the approximation correctly represents the topological relationship depends on how we approximately represent the cadastral/property objects. The approach of approximately representing different cadastral/property objects is another research subject and will not be covered in this study.

Although we cannot ensure that all the approximations do not break the topological relationships between nearby or adjacent cadastral/property objects in the 3D cadastral data model, we can avoid situations that obviously break the topological relationships. In 3D cadasters, many 3D situations can be approximately represented without breaking the topological relationships. Although no jurisdiction legally restricts all cadastral/property objects, some restrictions exist for these objects in practice, such as height limitations for buildings. With improvements in laws for 3D cadasters, more detailed restrictions for the spatial extents of cadastral/property objects will likely be implemented.

4. EABNOF

EABNOF consists of two components, which are presented in the following sections: (1) the generation of non-overlapping footprints (footprint points, edges and faces) by removing overlaps between footprints of the same dimension and propagating intervals from higher-dimensional footprints to corresponding lower-dimensional footprints; and (2) the extrusion of all footprints to generate primitives (3D points, edges, faces and volumes) , the removal of redundant primitives and the merging of valid primitives to form integrated 3D geometric models and topological features.

4.1. Boolean Set Operations in EABNOF

In our EABNOF, the difference (\) and intersection (∩) Boolean set operations can be performed between two footprints with the same dimension. Figure 6 shows examples of the difference and intersection between two footprint points, edges and faces. In these examples, the difference and intersection operations only compute the interiors of two footprints. Thus, the result of the difference or intersection between two nD (n = 0, 1, 2) footprints can only be empty or one or more nD footprints.

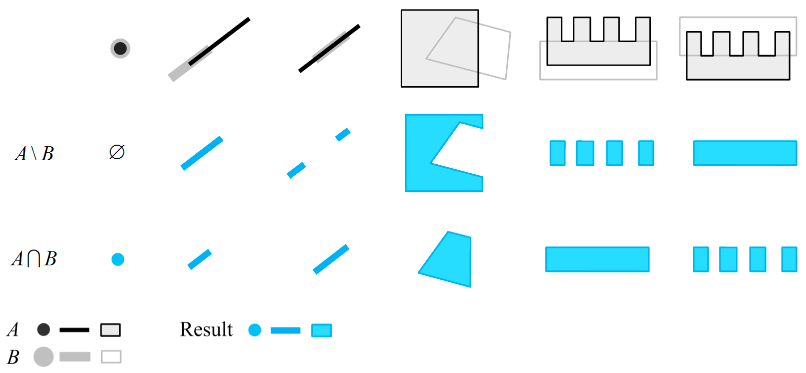


Figure 6. Boolean set operations for footprint points, edges and faces.

For footprints with different dimensions, the numbers of footprints in the result becomes different. The difference or intersection between two footprint points are one footprint point or empty. The difference between two footprint edges is one or two footprint edges or empty, and the intersection between two footprint edges is one footprint edge or empty. The difference or intersection between two footprint faces could be one or more footprint faces or empty.

4.2. Generating Non-Overlapping Footprints

The input of our EABNOF is a set of footprint faces, and each footprint face has its corresponding extrusion intervals and other attributes. We define several data structures to explicitly explain our EABNOF as follows:

```
Structure 1: IntervalBoundary
    Value: double; //the value of the interval boundary
    CadastreIDs: set<long>; // indicates cadasteral objects corresponding to this interval boundary
```

Structure 2: Interval

```
MinVal: double; //the maximum value of the extrusion interval
MaxVal: double //the minimum value of the extrusion interval
CadastreIDs: set<long>; // indicates cadaster objects corresponding to this interval
```

Structure 3: Footprint

```
Geometry: GM_Object; //the geometry of the footprint
IntervalSet: set< Interval >; // the interval set of this footprint
IntervalBoundarySet: set< IntervalBoundary>; // the interval boundary set of the footprint
```

The structure Footprint (Structure 1) is not consistent with the class Footprint in the data model because it is only used for computation. The structure Footprint is a common description for footprint points, edges and faces. This factor has different types of geometries for different types of footprints. The main function of the structure Footprint is to associate each interval and boundary value of a footprint with one or more cadastral objects by using cadaster IDs. Thus, the structures Interval and IntervalBoundary were designed, and the sets of the two structures can be encapsulated in the structure Footprint.

The structure Interval (Structure 2) contains a member CadastreIDs that indicates a primitive that is generated by extruding a footprint along an extrusion interval, which may belong to one or more cadastral objects. For example, a vertical 3D face that is generated by extruding a footprint edge along an extrusion interval is the common face of two 3D spatial units for two cadastral objects.

An extrusion interval is bounded by a top and bottom boundary (called interval boundaries), whose values are the maximum and minimum values of the extrusion interval. The values of the interval boundaries are used as Z values, which are added to the footprints to generate horizontal primitives. The structure IntervalBoundary (Structure 3) was designed to encapsulate the value and attributes of an interval boundary. The member CadastreIDs of IntervalBoundary has the same meaning as that of Interval.

4.2.1. Splitting the Extrusion Intervals of Two Footprints

When generating non-overlapping footprints, the intersection of two overlapping footprints forms a new footprint, which should produce the extrusion intervals of both overlapping footprints. Overlapping intervals may exist when the extrusion intervals of the two footprints are put into a set, and these intervals can result in overlapping primitives. These overlapping intervals should be split to obtain non-overlapping intervals.

Arroyo, Ledoux and Stoter [30] propagated and split intervals with an algorithm called PropagateRanges, which inserts each extrusion interval into another non-overlapping interval set. This algorithm considers extrusion intervals that correspond to 3D shapes of the same 3D object (i.e., a 3D building) and thus cannot handle extrusion intervals that correspond to 3D shapes of two or more 3D objects. In our EABNOF, the extrusion intervals of a footprint may correspond to the primitives of more than one cadastral object, so the cadastral objects that correspond to each extrusion interval should be considered when splitting intervals. To satisfy this requirement, we designed an algorithm whose pseudo-code is shown in SplitIntervals (Algorithm 1). This algorithm makes four main extensions and modifications to PropagateRanges, which are as follows:

(1) SplitIntervals associates the intervals of a footprint with their corresponding cadastral objects through CadastreIDs, while PropagateRanges only handles intervals that correspond to 3D shapes of the same 3D object.

(2) SplitIntervals repeatedly splits an interval (iv) until the intersecting interval set IV''_{Inter} is empty and the maximum and minimum values ($r_{\min}$ and $r_{\max}$) of iv are modified in the splitting

operation, while PropagateRanges does not modify the maximum or minimum values and the splitting operation is not repeated, so PropagateRanges cannot handle the condition of an interval that intersects with several intervals. This problem for PropagateRanges may be a mistake in the writing.

(3) SplitIntervals considers a condition (Condition 1 in lines 6–10) in which both the maximum and minimum values ($r_{\min}$ and $r_{\max}$) of an interval (iv) are in the same interval (iv''); this condition is also considered in PropagateRanges.

(4) Empty intervals that are used for extrusion are retained in PropagateRanges, while SplitIntervals removes these empty intervals because our EABNOF records interval boundaries instead of empty intervals.

Algorithm 1: SplitIntervals

```
    Input: two interval sets $IV$ and $IV'$
    Output: output interval set $IV''$ in which intervals are non-overlapping
1   $IV'' \leftarrow IV'$
2   foreach interval $iv$ in $IV$ do
3     Find intervals $IV''_{Inter} \subseteq IV''$ whose interiors intersect with the interior of $iv$ and $IV''_{Inter}$
      is kept sorted in ascending sort
4     $r_{\min} \leftarrow iv.$ MinVal $r_{\max} \leftarrow iv.$ MaxVal
5     repeat
6     if both $r_{\min}$ and $r_{\max}$ are in a same interval $iv'' \in IV''$ then //Condition 1
7         Put interval {$iv''$. MinVal, $r_{\min}$, $iv''$.CadastreIDs} in $IV''$
8         Put interval {$r_{\min}$, $r_{\max}$, $iv.$ CadastreIDs$\cup iv''$. CadastreIDs} in $IV''$
9         Put interval {$r_{\max}$, $iv''$. MaxVal, $iv''$. CadastreIDs } in $IV''$
10        Break
11      if $r_{\min}$ is in an interval $iv'' \in IV''$ then //Condition 2
10        Put interval {$iv''$. MinVal, $r_{\min}$, $iv''$.CadastreIDs} in $IV''$
11        Put interval {$r_{\min}$, $iv''$. MaxVal, $iv.$ CadastreIDs$\cup iv''$. CadastreIDs} in $IV''$
12        $r_{\min} \leftarrow iv''$. MaxVal
13        Remove $iv''$ from $IV''_{Inter}$
14      if $r_{\min}$ is outside all intervals in $IV''$ then //Condition 3
15        $r''_{\min} \leftarrow$ the minimum value of the first interval in $IV''_{Inter}$ interval
16        Put interval {$r_{\min}$, $r''_{\min}$, $iv.$ CadastreIDs} in $IV''$
17        $r_{\min} \leftarrow r''_{\min}$
18      if $r_{\max}$ is in an interval $iv'' \in IV''$ then //Condition 4
19        Put interval {$iv''$. MinVal, $r_{\max}$, $iv.$ CadastreIDs $\cup iv''$. CadastreIDs} in $IV''$
20        Put interval {$r_{\max}$, $iv''$. MaxVal, $iv''$. CadastreIDs} in $IV''$
21        $r_{\max} \leftarrow iv''$. MinVal
22        Remove $iv''$ from $IV''_{Inter}$
23      if $r_{\max}$ is outside all intervals in $IV''$ then //Condition 5
24        $r''_{\max} \leftarrow$ the maximum value of the last interval in $IV''_{Inter}$ interval
25        Put interval {$r_{\max}$, $r''_{\max}$, $iv.$ CadastreIDs} in $IV''$
26    $r_{\max} \leftarrow r''_{\max}$
27      if $IV''_{Inter} = \varnothing$ then
28        Put interval {$r_{\min}$, $r_{\max}$, $iv$.CadastreIDs} in $IV''$
29    until $IV''_{Inter} = \varnothing$
30  Remove each empty interval whose MinVal and MaxVal are equal from $IV''$
```

Figure 7a–e show Conditions 1–5 for splitting intervals in SplitIntervals. The visited interval (iv), the interval (iv'') that overlaps the visited interval, and the new split intervals are marked in red, blue and purple, respectively. The cadastral IDs that correspond to these intervals are shown as C_1, C_2 or $C_1 \cup C_2$, and the modifications of the maximum and minimum values are also marked.

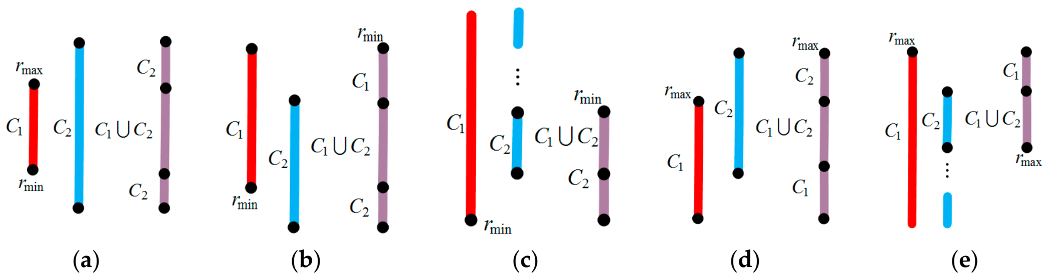


Figure 7. Five conditions of splitting intervals: (**a**) the maximum and minimum values of an interval are within another interval; (**b**) the minimum value of an interval is within another interval; (**c**) the minimum value of an interval is outside all the intervals of another interval set; (**d**) the maximum value of an interval is within another interval; and (**e**) the maximum value of an interval is outside all the intervals of another interval set.

If we assume that the two sets IV and IV' in SplitIntervals have m and n intervals, $m + n$ intervals has at most $2(m + n)$ interval boundaries, so the output interval set IV'' has at most $2(m + n) - 1$ intervals. Thus, each interval iv in IV overlaps with at most $2(m + n) - 1$ intervals in IV'', so at most m$(2(m + n) - 1)$ split operations are performed. To obtain a sorted set IV''_{Inter} and keep it sorted, we can sort set IV'' first and place new split intervals in the proper positions according to their corresponding original intervals. The set IV'' can be sorted in O(mlogm) by quicksort. Thus, the computational complexity of SplitIntervals is $O(m^2 + mn)$. In practice, we can always choose the interval set with less intervals as set IV with m intervals, which can ensure that mn is always greater than m^2. Thus, the computational complexity of SplitIntervals is simplified to O(mn).

4.2.2. Removing Overlaps between Two Footprints

The basic idea of removing overlaps between two footprints is to use the combination of Boolean set operations that was defined in Section 4.2.1. Figure 8 shows how overlaps among footprint points, edges and faces are removed by Boolean set differences and intersections. The overlaps between two overlapping footprint edges or faces A and B can be removed by splitting them into three components $A \backslash B$, $B \backslash A$ and $A \cap B$. Removing overlaps between two overlapping footprint points is much easier because only $A \cap B$ is required.

Considering the intervals and corresponding cadastral objects, removing overlaps between footprints could become much more complicated. The procedure of removing overlaps between two footprints (footprint points, edges and faces) is described in *RemoveOverlaps* (Algorithm 2), which involves operations for geometries, extrusion intervals and cadaster IDs. This algorithm uses *FootprintsDifference* (Algorithm 3) and *FootprintsIntersection* (Algorithm 4) to compute the difference and intersection between two footprints. In these two algorithms, *ExtractIntervalBoundaries* (Algorithm 5) is used to compute interval boundary sets of the interval sets. Moreover, *SplitIntervals* (Algorithm 1) is used to split the intervals for the overlapping components of two footprint edges and points in *FootprintsIntersection*. No overlapping intervals exist for the overlapping components of two footprints faces because extruding these components along overlapping intervals results in overlapping volumes, which cannot exist in 3D cadasters. The detailed implementations of the Boolean set difference ($\backslash$) and intersection ($\cap$) between geometries are not described because Boolean set operations between geometries can be easily implemented by many APIs for GISs or computational geometry, such as the ArcGIS Engine and CGAL.

Algorithm 2: RemoveOverlaps

```
   Input: two overlapping footprints $f$ and $f'$ of the same dimension
   Output: set $F$ of output non-overlapping footprints
1  if footprints $f$ and $f'$ are footprint faces or edges then
2      Put the footprints of FootprintsDifference($f$, $f'$) in $F$
3      Put the footprints of FootprintsDifference($f'$, $f$) in $F$
4  Put the footprints of FootprintsIntersection ($f$, $f'$) in $F$
```

Algorithm 3: FootprintsDifference

```
   Input: base footprint $f$
          comparison footprint $f'$
   Output: set $F_{diff}$ of output non-overlapping footprints including parts of $f$ but not $f'$
1  Compute the geometry set $G_{diff}$ ← ($f$. Geometry) ($f'$. Geometry)
2  IntervalBoundarySet $IB$←ExtractIntervalBoundaries($f$. IntervalSet)
3  foreach geometry $g$ in $G_{\text{diff}}$ do
4      Create a new footprint $f''$
5      $f''$. Geometry ←$g$
6      $f''$. IntervalSet ←$f$. IntervalSet
7      $f''$. IntervalBoundarySet ←$IB$
8  Put $f''$ in $F_{\text{diff}}$
```

Algorithm 4: FootprintsIntersection

```
    Input: two footprints $f$ and $f'$
    Output: set $F_{\text{inter}}$ of output non-overlapping footprints including parts of both $f$ and $f'$
1   Compute the geometry set $G_{\text{inter}}$ ← ($f$. Geometry)∩($f'$. Geometry)
2   Put the intervals of $IV$ and $IV'$ in a new IntervalSet $IV''$
3   IntervalBoundarySet $IB$← ExtractIntervalBoundaries ($IV''$)
4   if footprints $f$ and $f'$ are footprint points or edges then
5       IntervalSet $IV$← SplitIntervals($IB$)
6   else
7       IntervalSet $IV$←($f$. IntervalSet)∪($f'$. IntervalSet)
8   foreach geometry $g$ in $G_{\text{inter}}$ do
9       Create a new footprint $f''$
10      $f''$.Geometry ←$g$
11      $f''$. IntervalSet ←$IV$
12      $f''$. IntervalBoundarySet ←$IB$
13      Put $f''$ in $F_{\text{inter}}$
```

Algorithm 5: ExtractIntervalBoundaries

```
    Input: Interval set $IV$ in which intervals may overlap each other
    Output: Interval boundary set $IB$ in which each interval boundary has a different value
1   foreach $iv$ in $IV$ do
2       Create two new interval boundaries $ib$ and $ib'$
3       $ib$. Value ←$iv$. Range. maximum $ib'$. Value ←$iv$. Range. minimum
4       $ib$. CadastreIDs ←$iv$. CadastreIDs
5       Put $ib$ and $ib'$ in $IB$
6   Sort the interval boundary set $IB$
7   foreach pair of IntervalBoundaries $ib$ and $ib'$ whose values are equal do
8       Create a new interval boundary $ib''$
9       $ib''$.Value ←$ib$. Value
10      $ib''$. CadastreIDs ←($ib$. TopCadastreIDs) ∪($ib'$. TopCadastreIDs)
11      Remove $ib$ and $ib'$ from $IB$
12      Put $ib''$ in $IB$
```

In *ExtractIntervalBoundaries* (Algorithm 5), all the $m + n$ intervals (m and n are the number of intervals for each footprint) of the two input footprints have $2(m + n)$ interval boundaries. The interval boundary set *IB* (line 6) can be sorted in $O((m+n)\log(m+n))$ by quicksort, and searching interval boundaries with the same values (line 7) in the sorting set only requires visiting each interval boundary once, so the time complexity of *ExtractIntervalBoundaries* is $O((m+n)\log(m+n))$.

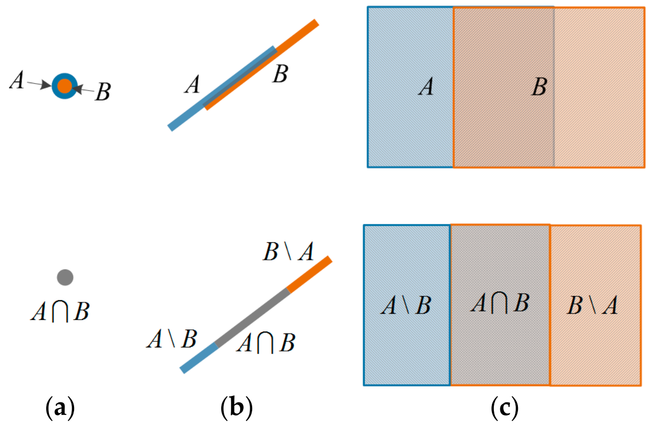


Figure 8. Removal of overlaps between footprint points (**a**), footprint edges (**b**) and footprint faces (**c**) by the combination of Boolean set operations.

In *FootprintsDifference* (Algorithm 3) and *FootprintsIntersection* (Algorithm 4), Boolean set operations between the geometries of two footprint faces are more time-consuming than those between the geometries of two footprint edges or points. The methods TopologicalOperator.Difference and ITopologicalOperator.Intersect in ArcEngine are used to implement the difference (\) and intersection (∩) between two geometries of two footprint faces. The implementation algorithms of the two methods are not published, but we can use the Weiler–Atherton clipping algorithm [43] as a reference, which was published in 1977 and is one of the most famous polygon-clipping algorithms. The time complexity of the Weiler–Atherton clipping algorithm is $O(pq)$, where p and q are the numbers of vertices of the two input polygons. If we assume that *FootprintsDifference* and *FootprintsIntersection* generate k and l new footprints, their time complexities are $O(pq + k + (m+n)\log(m+n))$ and $O(pq + l + (m+n)\log(m+n) + mn)$, respectively, where $O((m+n)\log(m+n))$ and $O(mn)$ are the time complexities of *ExtractIntervalBoundaries* (Algorithm 5) and *SplitIntervals* (Algorithm 1).

Summing the time complexities of the three steps (lines 2, 3 and 4) of *RemoveOverlaps* (Algorithm 2) produces the time complexity $O(pq + t + (m+n)\log(m+n) + mn)$, where p and q are the numbers of vertices of the two geometries of the two input footprints, t is the total number of newly generated geometries, and m and n are the numbers of intervals of the two input footprints. When the number of input footprints is more than two, *RemoveOverlaps* should be executed for each overlapping footprint, and the total execution time depends on the number of newly generated footprints in each execution.

4.2.3. Propagating Extrusion Intervals

The input footprint faces and their extrusion intervals (and other attributes) can only be used to construct 3D faces and volumes. To generate 3D edges and points, the extrusion intervals of footprint faces (Figure 9a) should be propagated to footprint edges (Figure 9b) and then to footprint points (Figure 9c). The propagation algorithm PropagateRanges by Arroyo, Ledoux and Stoter [30] both propagates intervals and splits extrusion intervals, while our propagation operation only propagates extrusion intervals because extrusion intervals are split by SplitIntervals (Algorithm 1) in our EABNOF.

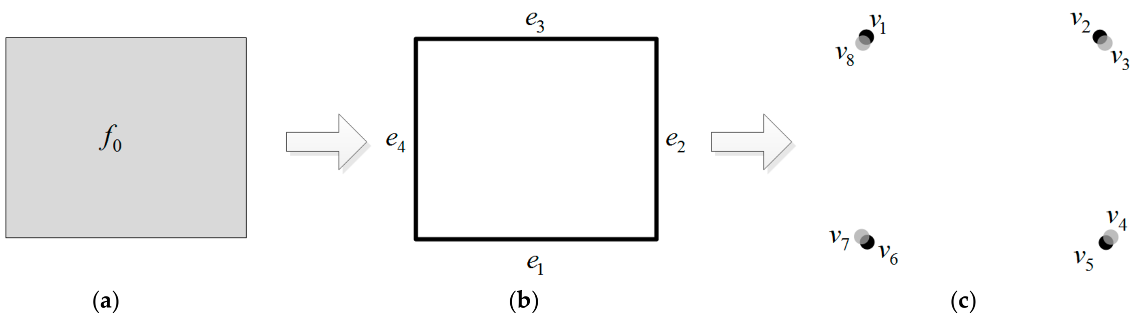


Figure 9. Propagation of extrusion intervals from footprint faces (**a**) to footprint edges (**b**) and then to footprint points (**c**).

Three rules are defined in our propagation operation:

(**Rule** 1) Extrusion intervals are propagated only from *n*D ($n = 1, 2$) footprints to ($n - 1$)D footprints, which are ($n - 1$)D boundaries of the *n*D footprints.

(**Rule** 2) Extrusion intervals of higher-dimensional footprints can be propagated to their corresponding lower-dimensional footprints only when overlaps between the higher-dimensional footprints are removed.

(**Rule** 3) For several adjacent footprint edges (i.e., e_1 and e_2 in Figure 9) whose boundaries intersect at a point, each footprint edge should propagate their intervals to a different footprint point at the intersection (v_5 and v_4).

Rule 3 does not generate only one footprint point at the intersection because each footprint edge may propagate different intervals to its corresponding footprint point, and these different intervals should be split.

4.2.4. Process for Generating All Non-Overlapping Footprints

Overlaps should be removed and intervals should be propagated to obtain all non-overlapping footprints (footprint faces, footprint edges and footprint points). Figure 10 provides an example of generating all non-overlapping footprints. This example has three input footprint faces (f_1, f_2 and f_3), in which two footprint faces (f_1 and f_2) overlap the interiors of each other and two footprint faces (f_2 and f_3) touch each other. The process of generating all non-overlapping footprints consists of five steps:

Step 1: Remove overlaps between each overlapping footprint face by using RemoveOverlaps (Algorithm 1). In the example, four non-overlapping footprint faces (f_3, f_4, f_5 and f_6) are generated by removing the overlaps between f_1 and f_2.

Step 2: Extract footprint edges from the footprint faces and propagate interval sets of the footprint faces to their corresponding footprint edges. In the example, eighteen footprint edges ($e_1 - e_{18}$) are generated, in which eight footprint edges (e_3, e_5, e_7, e_8, e_{10}, e_{11}, e_{13} and e_{15}) overlap one other footprint edge.

Step 3: Remove overlaps between each overlapping footprint edge by using RemoveOverlaps. In the example, six non-overlapping footprint edges (e_{19}, e_{20}, e_{21}, e_{22}, e_{23} and e_{24}) are generated by removing overlaps.

Step 4: Extract footprint points from the footprint edges and propagate interval sets of the footprint edges to their corresponding footprint points. In the example, thirty-two footprint points (v_1-v_{32}) are generated, each of which overlaps one or more other footprint points.

Step 5: Remove overlaps between each overlapping footprint point by using RemoveOverlaps. In the example, thirteen non-overlapping footprint edges (v_{33}-v_{45}) are generated by removing overlaps.

After these steps, the non-overlapping footprint faces, edges and points that are generated in Steps 1, 3 and 5 are output.

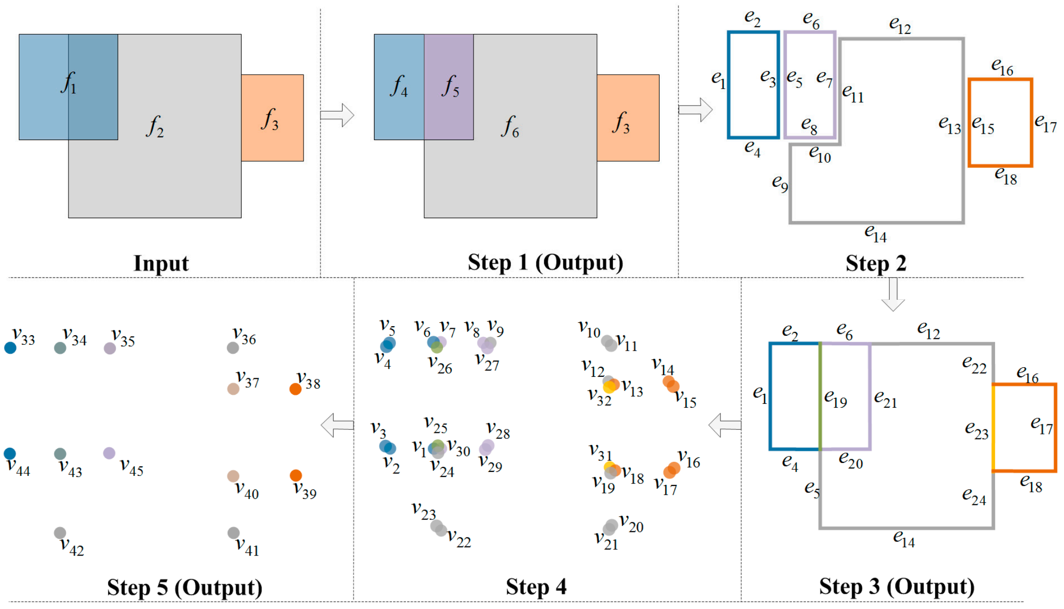


Figure 10. Process of generating all non-overlapping footprints.

4.3. Constructing Geometric Models and Topology without Redundancies

With these non-overlapping footprints, we can easily generate primitives (3D points, edges, faces and volumes) by extruding along their corresponding extrusion intervals and adding interval boundaries as Z values. Figure 11a shows a footprint face (f_0) and its corresponding footprint edges (e_1, e_2, e_3 and e_4) and points (v_1, v_2, v_3 and v_4). In Figure 11b, vertical primitives (i.e., v_3^{r1}, v_3^{r2}, v_3^{r3}, e_3^{r1}, e_3^{r2}, e_3^{r3}, f_0^{r1}, f_0^{r2} and f_0^{r3}) are generated by extruding footprints (v_3, e_3 and f_0) along intervals r_1 = (a, b), r_2 = (b, c) and r_3 = (d, e), and horizontal primitives (i.e., v_3^a, v_3^b, v_3^c, v_3^d, v_3^e, e_2^a, e_2^b, e_2^c, e_2^d, e_2^e, f_0^a, f_0^b, f_0^c, f_0^d and f_0^e) are generated by adding the interval boundaries a, b, c, d and e to the footprints (v_3, e_3 and f_0) as Z values. In these primitives, volumes (f_0^{r1}, f_0^{r2} and f_0^{r3}) are used to construct geometric models for 3D spatial units, and other primitives (3D points, edges and faces) are used to construct topological features (3D boundaries). However, not all these 3D points, edges and faces can be used to construct topological features for spatial units because redundancies (invalid values) exist.

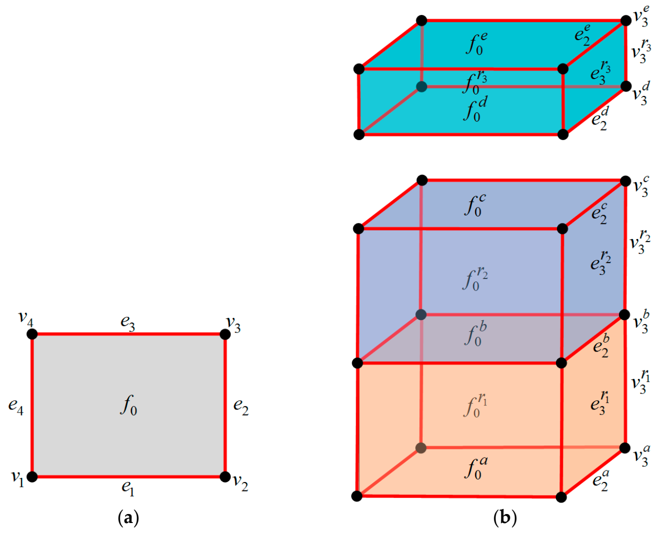


Figure 11. (**a**) Footprint face and its corresponding footprint edges and points; and (**b**) primitives (3D points, edges and faces) that were generated by extruding these footprints.

4.3.1. Identifying Redundant Primitives

Generally, a redundant primitive is a lower-dimensional primitive within a higher-dimensional topological feature (or 3D spatial unit), such as a 3D point within a 3D boundary line. An example in Figure 12 explains such redundancies. Figure 12a shows two non-overlapping footprint faces (f_1 and f_2), nine non-overlapping footprint edges (*ab*, *be*, *ec*, *cd*, *da*, *ef*, *fg*, *gh* and *hc*) and eight non-overlapping footprint points (*a*, *b*, *c*, *d*, *e*, *f*, *g* and *h*), in which the footprints f_1, *ab*, *be*, *cd*, *da*, *a*, *b* and *d* have the interval (0, 6) and interval boundaries 0 and 6; the footprints f_2, *ef*, *fg*, *gh*, *hc*, *f*, *g* and *h* have the interval (0, 4) and interval boundaries 0 and 4; and the footprints *ec*, *e* and *c* have the intervals (0, 4) and (4, 6) and interval boundaries 0, 4 and 6. All these intervals and interval boundaries are associated with the same cadastral object.

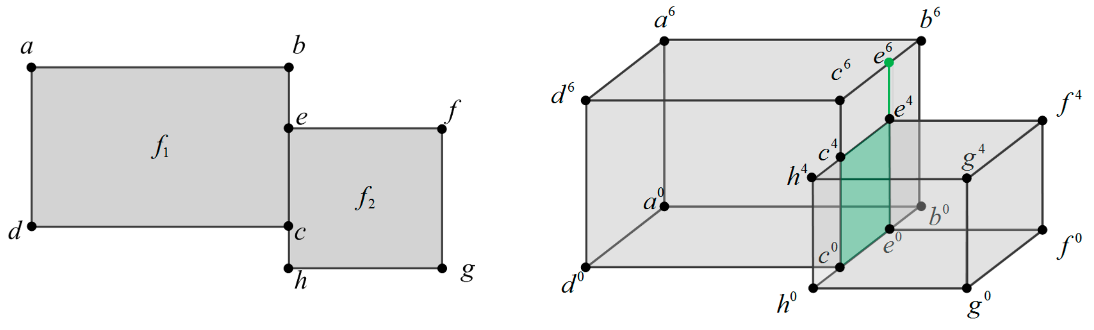


Figure 12. Non-overlapping footprints (**a**) and primitives (**b**) that were generated by extruding footprints in (**a**), which contain redundancies.

Figure 12b shows the primitives that were generated by the footprints in Figure 12a, in which the 3D point e^6, 3D edge e^4 e^6 and 3D face $c^0c^4e^4e^0$ (marked in green) are redundancies. These redundancies occurred because e^6 is within the 3D boundary line b^6 c^6, e^4 e^6 is within $c^4c^6b^6b^0e^0e^4$ and $c^0c^4e^4e^0$ is within the 3D spatial unit that consists of $a^0b^0c^0d^0a^6b^6c^6d^6$ and $e^0f^0g^0h^0e^4f^4g^4h^4$. These redundancies should be removed from primitives. We provide three judgment criteria to accurately identify redundancies.

(**Criterion** 1) A 3D point δ_0 is the common point of only two valid 3D edges. If the two 3D edges are collinear and associated with the same cadastral object(s), then δ_0 is a redundancy.

(**Criterion** 2) A 3D edge δ_1 is the common edge of only two valid 3D faces. If the two 3D faces are coplanar and associated with the same cadastral object(s), then δ_1 is a redundancy.

(**Criterion** 3) A 3D face δ_2 is the common face of two valid volumes. If the two volumes are associated with the same cadastral object(s), then δ_2 is a redundancy.

For Criteria 1 and 2, "only two" and "valid" designations are used as two key points for identifying redundancies.

An example can help clarify the "only two" designation. Figure 13a shows a non-manifold spatial unit *A* that consists of volumes (V_1 and V_2). The 3D edge e_1 is the common edge of four 3D faces: f_1, f_2, f_3 and f_4, in which f_1 and f_4 are coplanar and f_2 and f_3 are coplanar. All four 3D faces are associated with the cadastral object *A*. According to *Criterion* 3, the four 3D faces are not redundancies. Thus, e_1 touches coplanar valid 3D faces that are associated with the same cadastral object; however, this edge is not a redundancy because the number of valid 3D faces does not fit the "only two" description.

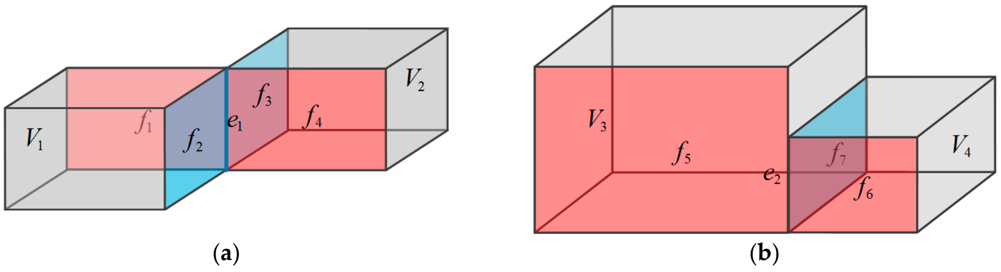


Figure 13. Two examples of identifying redundancies based on the proposed conditions: (**a**) a non-manifold spatial unit; and (**b**) a spatial unit consisting of two adjacent volumes.

Another example shows the importance of the "which are not redundancies" designation. Figure 13b shows a spatial unit B that consists of two volumes (V_3 and V_4). The 3D edge e_2 is the common edge of the three 3D faces f_5, f_6 and f_7, which does not fit the "only two" designation. However, e_2 is a redundancy because f_7 is a redundancy as the common face of two valid volumes V_3 and V_4 and is associated to the same cadastral object of B, and e_2 touches only two valid 3D faces f_5 and f_6, which are coplanar and associated with the same cadastral object B.

When applying these three criteria, we should adhere to the following order: *Criterion* 3, *Criterion* 2 and *Criterion* 1. Before applying *Criterion* 2, we must identify any redundant (invalid) 3D faces, which should be performed according to *Criterion* 3. Before applying *Criterion* 1, we must identify redundant (invalid) 3D edges, which should be performed based on *Criterion* 2.

4.3.2. Construction of Geometric Models and Topological Features

In our EABNOF, geometric models and topological features of 3D spatial units are constructed from primitives. A single geometric model (or topological feature) may be constructed from one or more primitives. The corresponding primitives of a geometric model (or topological feature) that is constructed from more than one primitive must be merged into a single object.

Two nD (n = 1, 2, 3) primitives must always have an (n – 1D common boundary (common point, edge or face) to be merged, which is redundant because the common boundary is within a 3D boundary that is generated by merging the primitives. We can construct geometric models and topological features by merging primitives and removing redundancies based on this property, which are described in the following three steps:

Step 1: Identify all the redundancies from the set P of primitives by using Criteria 1–3 and put them into a redundancy set RD. Remove the redundancies of set RD from P.

Step 2: For each redundancy rd in the set RD, merge its corresponding two primitives (p_1 and p_2) and place the merged result in P. Remove the two primitives p_1 and p_2 from set P. Remove the redundancy rd from set RD.

Step 3: Repeat Step 2 until set RD is empty.

Figure 14 shows the geometric model and topological features that were constructed from the primitives in Figure 12b, in which the 3D edges $b^6\ e^6$ and $e^6\ c^6$, 3D faces $c^4c^6e^6e^4$ and $e^0e^6b^6b^0$, and volumes $a^0b^0c^0d^0a^6b^6c^6d^6$ and $e^0f^0g^0h^0e^4f^4g^4h^4$ are merged and redundancies (e^6, $e^4\ e^6$ and $c^0c^4e^4e^0$) are removed.

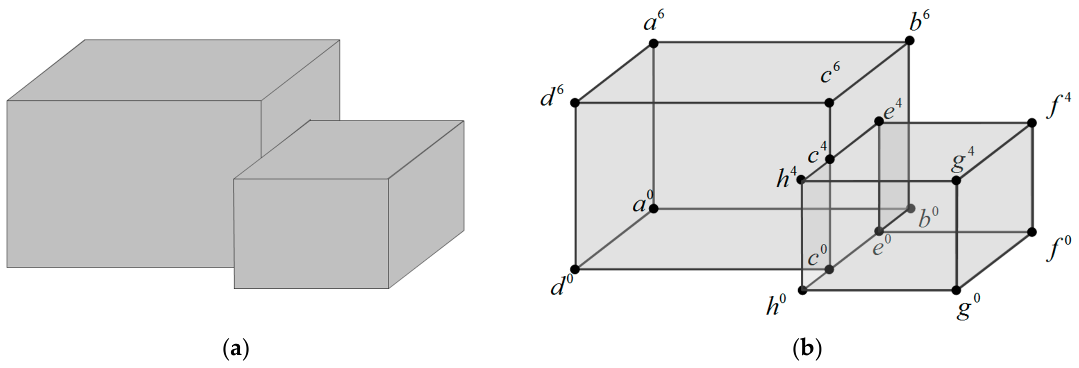


Figure 14. Geometric model (**a**) and topological features (**b**) that were constructed by merging primitives and removing redundancies in Figure 12b.

5. Implementation

This section introduces some important implementation details for our approach. We implemented our EABNOF and 3D cadastral model by using ArcGIS (Version 10.2), which is available through Microsoft Visual Studio 2008 and the ESRI Personal GeoDatabase. To handle the initial input data, we wrote a processing tool in Python (Version 2.7) based on ArcPy (Version 10.2), which provides access to geoprocessing tools.

5.1. Constructing Polygons for Footprint Faces

Our initial input data were floor plans of buildings, and most of the data were in the *DWG* format and created in AutoCAD. In these data, footprint faces were only represented by lines or polylines (Figure 15a), so we wrote a DWG conversion tool in Python based on ArcPy to construct polygons from these lines or polylines. The output (Figure 15b) of the tool was in *shapefile* format, and the tool could be added to ArcToolBox.

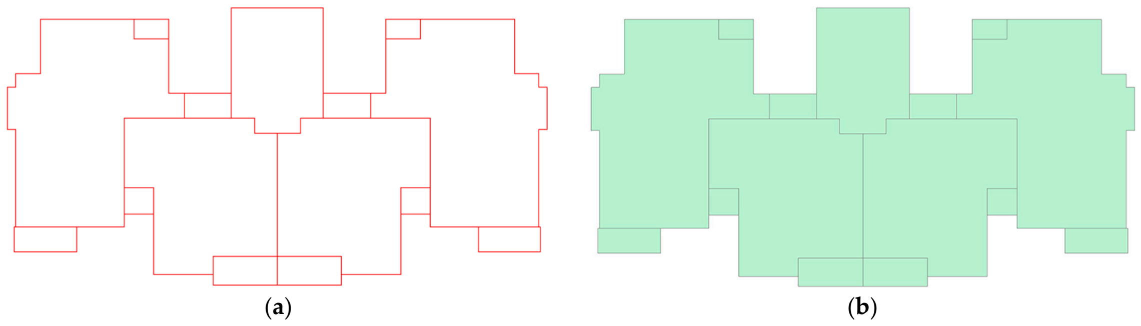


Figure 15. (**a**) Footprint faces represented by lines or polylines in the original data; and (**b**) polygons that were converted from lines or polylines by the conversion tool.

5.2. Adjusting the Positions of Footprint Faces

In most cases, each floor plan of a building was drawn in a different position that did not overlap other floor plans, so the footprint faces of different floors for a building were not in the same position (Figure 16a). We used the spatial adjustment tool in ArcMap to adjust the footprint faces of different floors to the same position. The footprint faces of a floor could be adjust to the position of the footprint faces of another floor by creating four displacement links between the footprint faces of the two floors. Figure 16b shows the adjusted footprint faces in Figure 16a.

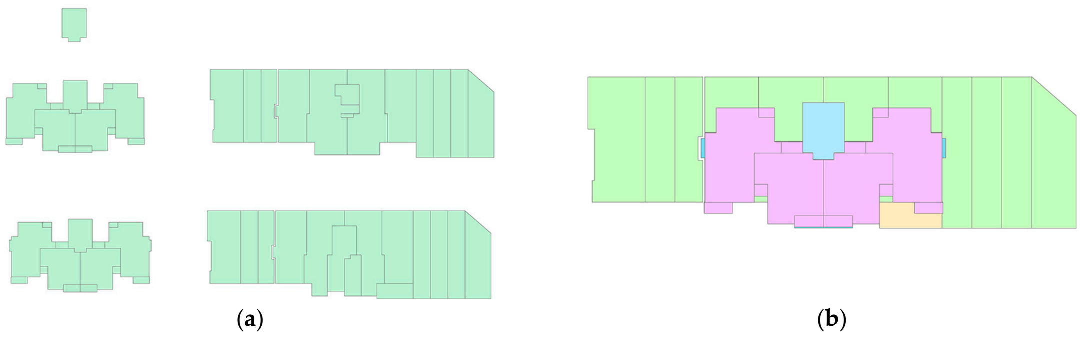


Figure 16. (**a**) Footprint faces of different floors in different positions; and (**b**) footprint faces of different floors adjusted to the same position.

6. Case Study

This section presents two 3D cadastral case studies of Pozi Street (Taizhou, Jiangsu, China) based on our EABNOF.

6.1. Current Cadaster of Pozi Street

Pozi Street is the most famous landmark in Taizhou, Jiangsu, China, and is a commercial pedestrian street with commercial and residential space. Figure 17a,b show an overall image and an overhead view of Pozi Street, respectively, with the building area in Figure 17b denoted with a red border. The first floor contains many brand-name stores, such as TESIRO, SAMSUNG, and an Apple Store. The second and third floors include many gourmet restaurants and other stores, and apartments are located above the third floor. Additionally, the underground level beneath the first floor includes over 400 small stores that sell a variety of goods.

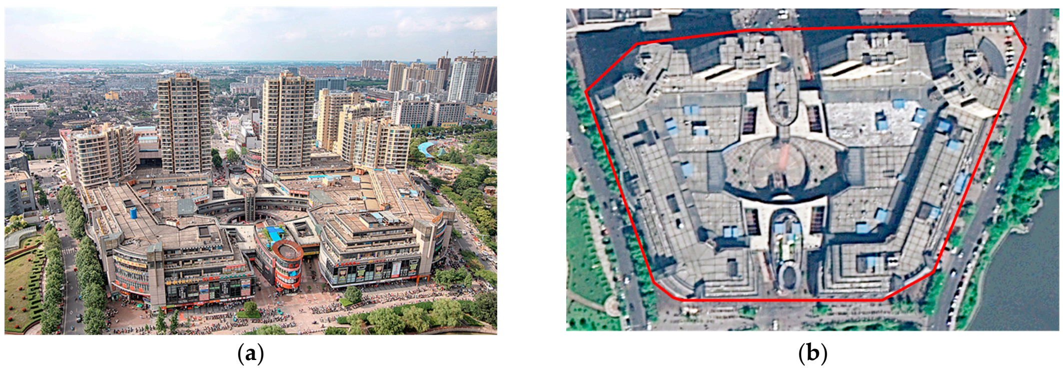


Figure 17. Overall image (**a**) and overhead view (**b**) of Pozi Street.

Many mixed land uses (residential and commercial areas) have been applied along Pozi Street to fully exploit the land space. An example of a mixed-use building is shown in Figure 18a. In this case, department stores and apartments of different heights are located within the same building.

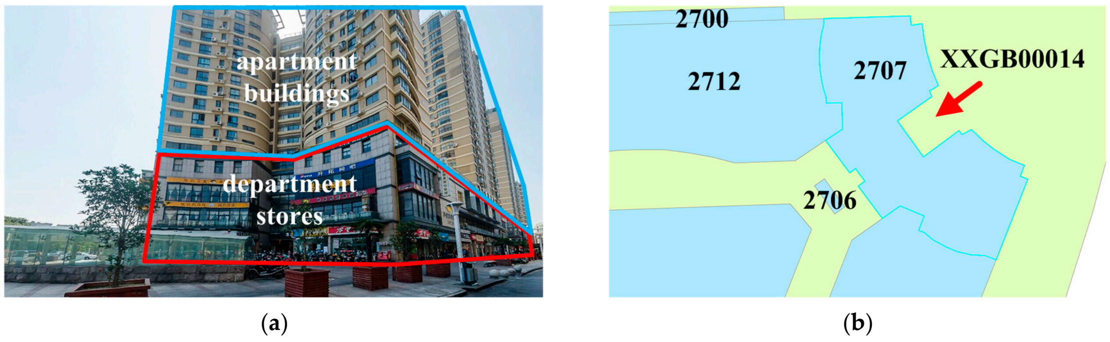


Figure 18. A mixed-use building (**a**) on Pozi Street represented on a 2D cadastral map (**b**).

In the current 2D cadaster, the property space of a mixed-use building was stored and managed as a 2D polygon. Figure 18b shows the mixed-use building (Figure 18a) number 2707 on a 2D cadastral map, and the red arrow denotes the observation position of the building. In the 2D cadastral map, both department stores and apartment buildings are represented by 2D polygons, resulting in the ambiguous definition of a mixed-use building. Moreover, the property rights of different owners could not be clearly represented. Therefore, a 3D cadaster is required for Pozi Street.

6.2. Case 1: Building Complex

Figure 19a shows 22 footprint faces of a building complex on Pozi Street. The rooftop, residential (Floor 2) and commercial (Floor 1) areas are located on different floors. These footprint faces could be used to build models for the 3D property units (spatial units) of the building complex. However, the three footprint faces overlapped in the 2D plane. We assigned different colors to the footprint faces of different layers in Figure 19b to clearly show these overlaps. These overlaps created ambiguities in the 2D cadastral map.

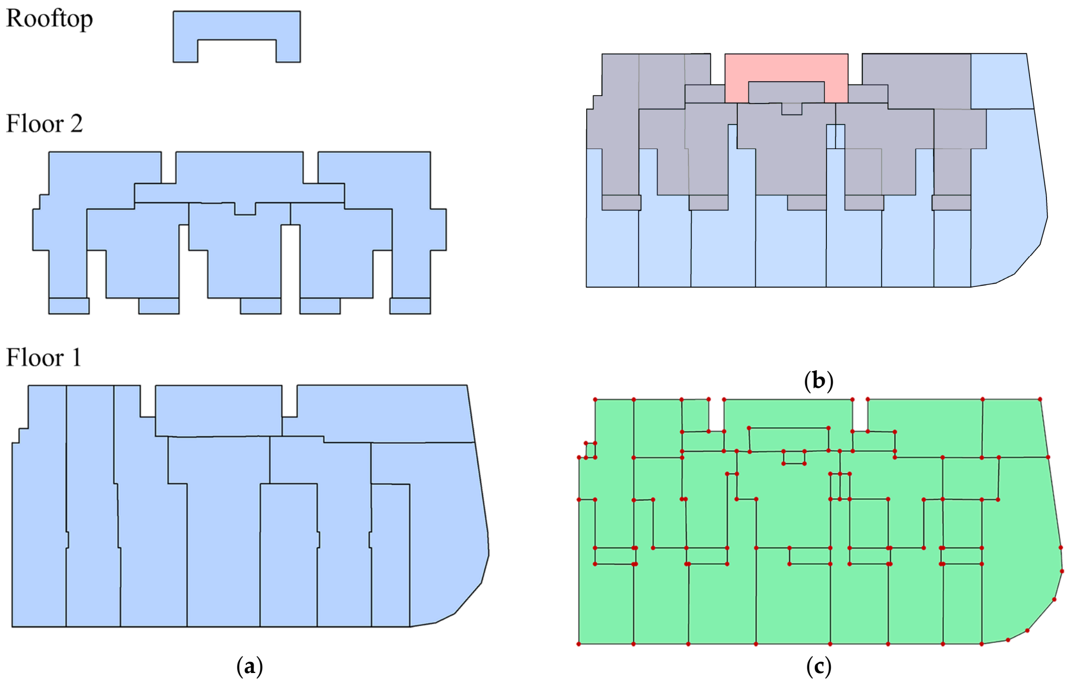


Figure 19. Building complex on Pozi Street: (**a**) rooftop, residential (Floor 2) and commercial footprint faces (Floor 1); (**b**) overlaps between footprint faces; and (**c**) non-overlapping footprint faces, edges and points that were generated by EABNOF.

We used EABNOF to remove overlaps and generate non-overlapping footprint faces, edges and points (Figure 19c) with intervals. We obtained 33 footprint faces, 137 footprint edges and 105 footprint points. This combination of footprint faces could represent 2D spatial units of the building complex. We extruded footprints, removed redundancies and merged primitives to obtain 3D geometric models and topological features for the 3D spatial units of the building complex, which are shown in Figure 20a,b.

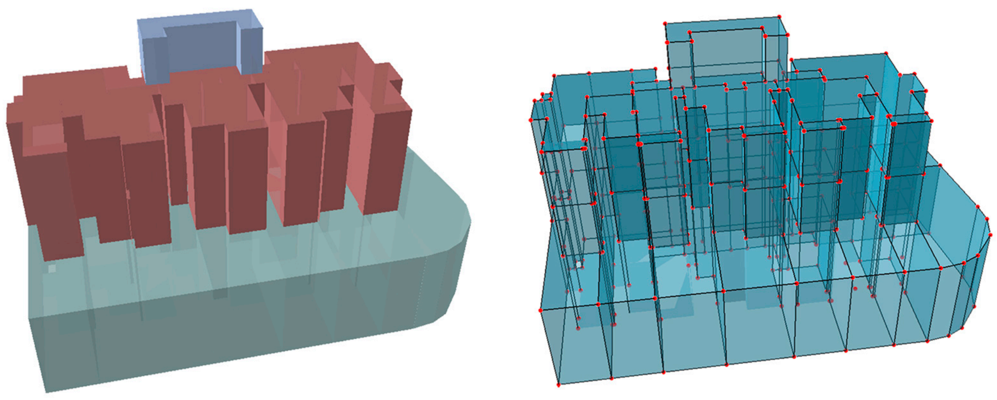

(**a**) (**b**)

Figure 20. 3D geometric model (**a**) and topological features (**b**) of the property units of the building complex.

6.3. Case 2: All the Property Objects of Pozi Street

Figure 21a shows the contours of all the property objects (buildings) on Pozi Street. We used our DWG conversion tool to construct polygons and obtained the polygons of footprint faces for all the property objects on different floors (Figure 21b). Many overlaps occurred between these footprint faces.

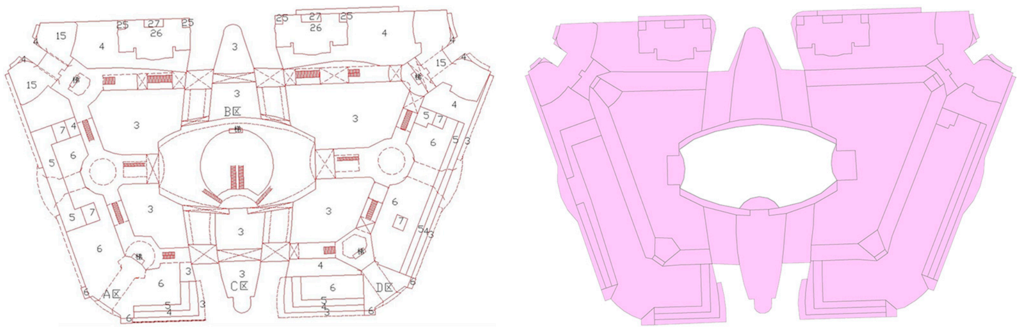

(**a**) (**b**)

Figure 21. All the property objects of Pozi Street: (**a**) contours of all the property objects and (**b**) footprints shown as different floors.

We constructed 3D geometric models (Figure 22a) and topological features (Figure 22b) for the 3D property objects by applying EABNOF to this case. EABNOF generated 75 footprint faces, 675 footprint edges and 459 footprint points.

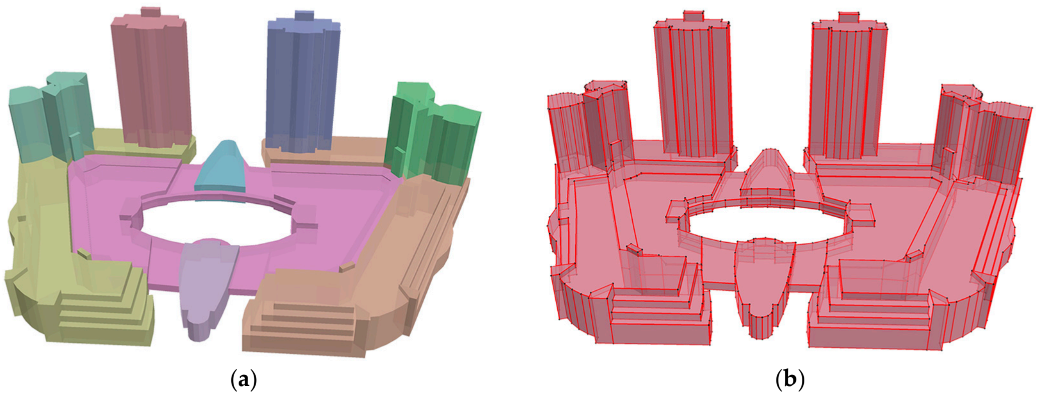


Figure 22. 3D geometric models (**a**) and topologies (**b**) of all the property objects on Pozi Street.

7. Conclusions

Recently, 3D model-building methods have received considerable attention because of the urgent need for 3D cadasters. The widely used extrusion approach is simple and practical but is associated with many obvious drawbacks. This paper presented a new extrusion approach (EABNOF) that is suitable for constructing geometric models and topologies in 3D cadasters. This approach constructs geometric models and topologies of 3D cadasters from the footprints of 2D cadastral data and supports significantly more 3D conditions than previous methods. Overlaps between input footprints can be removed through this approach, which also involves splitting the extrusion intervals of footprints and associating extrusion intervals to their corresponding cadastral objects. We used three judgment criteria to ensure no redundancies in primitives that are constructed via extruding non-overlapping footprints. EABNOF should provide some new ideas for handling footprints that are used for extrusion and building 3D topologies based on 2D data.

In the 3D cadastral data model that was designed for EABNOF, 3D cadastral data are associated with the footprints of a 2D cadaster, and the primitives of the 3D cadaster represent their geometries through footprints and height values. Therefore, 3D cadastral geometric models and topologies can be maintained by footprints in 2D. Thanks to these advantages, 3D cadastral data models can be used as a transitional scheme in countries that have accumulated a considerable amount of 2D cadastral data.

In future work, we plan to apply EABNOF to 4D cadaster conditions (3D cadaster + time), which will require us to modify the main operations of EABNOF. The time dimension can also be represented as an interval, so we can perform operations on time intervals that are similar to those that are applied for height values. Moreover, we plan to use EABNOF to construct models of LOD0 and LOD1 and identify the topological connections between them. We also plan to study approaches to approximate cadastral/property objects and the influences of the approximations on topological relationships.

Acknowledgments: This work was supported by the National Natural Science Foundation of China under Grant Nos. 41471318 and 41501431.

Author Contributions: All six authors contributed to the work in this paper. Yuan Ding designed the main operations of the EABNOF (Section 4); Nan Jiang and Changbin Wu designed the 3D cadastral data model and reviewed the literature (Sections 2 and 3); Binqing Ma implemented the EABNOF (Section 5); Zhaoyuan Yu improved the EABNOF and applied it to 3D cadasters (Section 4.3 and 6); and Ge Shi contributed to the introduction and conclusions (Sections 1 and 6). All the authors worked collaboratively in writing this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Aien, A.; Rajabifard, A.; Kalantari, M.; Williamson, I. Aspects of 3D Cadastre: A Case Study in Victoria. In Proceedings of the FIG Working Week 2011, Marrakech, Morocco, 18–22 May 2011.

2. Stoter, J.; Salzmann, M. Towards a 3D cadastre: Where do cadastral needs and technical possibilities meet? *Comput. Environ. Urban Syst.* **2003**, *27*, 395–410. [CrossRef]
3. Ledoux, H.; Meijers, M. Topologically consistent 3D city models obtained by extrusion. *Int. J. Geogr. Inf. Sci.* **2011**, *25*, 557–574. [CrossRef]
4. Zlatanova, S. 3D GIS for Urban Development. Ph.D. Thesis, ITC, Delft, The Netherlands, 2000.
5. Gröger, G.; Kolbe, T.; Nagel, C.; Häfele, K. OGC City Geography Markup Language (CityGML) Encoding Standard, Version 2.0. Available online: http://www.opengeospatial.org/standards/citygml (accessed on 25 May 2017).
6. Chiang, H. Data Modelling and Application of 3D Cadastral in Taiwan. In Proceedings of the 3rd International Workshop on 3D Cadastres: Developments and Practices, Shenzhen, China, 25–26 October 2012; pp. 137–157.
7. García, J.M.O.; Soriano, L.I.V.; Martín-Varés, A.V. 3D Modeling and Representation of the Spanish Cadastral Cartography. In Proceedings of the 2nd International Workshop on 3D Cadastres, Delft, The Netherlands, 16–18 November 2011; pp. 209–222.
8. Stoter, J.E.; Ploeger, H.D. Property in 3D—Registration of multiple use of space: Current practice in Holland and the need for a 3D cadastre. *Comput. Environ. Urban Syst.* **2003**, *27*, 553–570. [CrossRef]
9. Ying, S.; Guo, R.; Li, L.; He, B. Application of 3D GIS to 3D cadastre in urban environment. In Proceedings of the 3rd International Workshop on 3D Cadastres: Developments and Practices, Shenzhen, China, 25–26 October 2012; pp. 253–272.
10. Ying, S.; Li, L.; Guo, R. Building 3D Cadastral System Based on 2D Survey Plans with SketchUp. *Geo-Spat. Inf. Sci.* **2011**, *14*, 129–136. [CrossRef]
11. Benhamu, M.; Doytsher, Y. Toward a spatial 3D cadastre in Israel. *Comput. Environ. Urban Syst.* **2003**, *27*, 359–374. [CrossRef]
12. Guo, R.; Li, L.; Ying, S.; Luo, P.; He, B.; Jiang, R. Developing a 3D cadastre for the administration of urban land use: A case study of Shenzhen, China. *Comput. Environ. Urban Syst.* **2013**, *40*, 46–55. [CrossRef]
13. Stoter, J.E. 3D Cadastre. Ph.D. Thesis, ITC, Delft, The Netherlands, 2004.
14. Thompson, R.J.; Van Oosterom, P. Validity of Mixed 2D and 3D Cadastral Parcels in the Land Administration Domain Model. In Proceedings of the 3rd International Workshop on 3D Cadastres: Developments and Practices, Shenzhen, China, 25–26 October 2012.
15. Yu, C.; Li, L.; Ying, S.; He, B.; Zhao, Z.; Wan, Y. Designing a Title Certificate for the Chinese 3D Cadastre. In Proceedings of the 3rd International Workshop on 3D Cadastres: Developments and Practices, Shenzhen, China, 25–26 October 2012; pp. 1–21.
16. Peres, N.; Benhamu, M. 3D Cadastre GIS—Geometry, Topology and Other Technical Considerations. In Proceedings of the FIG Working Week 2009, Eilat, Israel, 3–8 May 2009.
17. Billen, R.; Zlatanova, S. 3D spatial relationships model: A useful concept for 3D cadastre? *Comput. Environ. Urban Syst.* **2003**, *27*, 411–425. [CrossRef]
18. Van Oosterom, P.; Lemmen, C.; Uitermark, H. ISO 19152: 2012, land administration domain model published by ISO. In Proceedings of the FIG Working Week 2013, Abuja, Nigeria, 6–10 May 2013.
19. Lemmen, C.; Van Oosterom, P.; Thompson, R.; Hespanha, J.P.; Uitermark, H. The Modelling of Spatial Units (Parcels) in the Land Administration Domain Model (LADM). In Proceedings of the XXIV FIG International Congress 2010, Sydney, Australia, 11–16 April 2010.
20. Coors, V. 3D-GIS in networking environments. *Comput. Environ. Urban Syst.* **2003**, *27*, 345–357. [CrossRef]
21. Molenaar, M. A formal data structure for three-dimensional vector maps. In Proceedings of the EGIS' 90, Amsterdam, The Netherlands, 10–13 April 1990; pp. 770–781.
22. Pilouk, M. Integrated modelling for 3D GIS. Ph.D. Thesis, ITC, Delft, The Netherlands, 1996.
23. Gröger, G.; Plümer, L. How to Get 3-D for the Price of 2-D—Topology and Consistency of 3-D Urban GIS. *Geoinformatica* **2005**, *9*, 139–158. [CrossRef]
24. Herring, J.R. The OpenGIS abstract specification, Topic 1: Feature geometry (ISO 19107 Spatial schema), Version 5. In *OGC Document*; Open Geospatial Consortium (OGC): Wayland, MA, USA, 2001.
25. Craig McCabe, E.M.C.T. Creating and Texturing Multipatch Features. Available online: http://www.esri.com/news/arcuser/0111/3dcity.html (accessed on 13 July 2017).
26. Oracle Database Online Documentation 11g Release 2. Available online: http://docs.oracle.com/cd/E11882_01/index.htm (accessed on 10 June 2017).

27. Kazar, B.M.; Kothuri, R.; Van Oosterom, P.; Ravada, S. On Valid and Invalid Three-Dimensional Geometries. In *Advances in 3D Geoinformation Systems*; van Oosterom, P., Zlatanova, S., Penninga, F., Fendel, E.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 19–46.
28. Ying, S.; Guo, R.; Li, L.; Van Oosterom, P.; Ledoux, H.; Stoter, J. Design and Development of a 3D Cadastral System Prototype based on the LADM and 3D Topology. In Proceedings of the 2nd International Workshop on 3D Cadastres, Delft, The Netherlands, 16–18 November 2011.
29. Okabe, A.; Boots, B.; Sugihara, K.; Chiu, S.N. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
30. Arroyo Ohori, K.; Ledoux, H.; Stoter, J. A dimension-independent extrusion algorithm using generalised maps. *Int. J. Geogr. Inf. Sci.* **2015**, *29*, 1166–1186. [CrossRef]
31. Requicha, A.A.; Voelcker, H.B. *Constructive Solid Geometry*; Technical Memorandum 25, Production Automation Project; University of Rochester: Rochester, NY, USA, 1977.
32. Thibault, W.C.; Naylor, B.F. Set Operations on Polyhedra Using Binary Space Partitioning Trees. In Proceedings of the ACM SIGGRAPH Computer Graphics, Anaheim, CA, USA, 27–31 July 1987; pp. 153–162.
33. Gursoz, E.L.; Choi, Y.; Prinz, F.B. Boolean set operations on non-manifold boundary representation objects. *Comput.-Aided Des.* **1991**, *23*, 33–39. [CrossRef]
34. Requicha, A.A.; Voelcker, H.B. Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algorithms. *Proc. IEEE* **1985**, *73*, 30–44. [CrossRef]
35. Masuda, H. Topological operators and Boolean operations for complex-based nonmanifold geometric models. *Comput.-Aided Des.* **1993**, *25*, 119–129. [CrossRef]
36. Hachenberger, P.; Kettner, L.; Mehlhorn, K. Boolean operations on 3D selective Nef complexes: Data structure, algorithms, optimized implementation and experiments. *Comput. Geom.* **2007**, *38*, 64–99. [CrossRef]
37. Naylor, B.; Amanatides, J.; Thibault, W. Merging BSP trees yields polyhedral set operations. *ACM Siggr. Comput. Graph.* **1990**, *24*, 115–124. [CrossRef]
38. Kada, M.; McKinley, L. 3D building reconstruction from LiDAR based on a cell decomposition approach. In Proceedings of the International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Paris, France, 3–4 September 2009; pp. 47–52.
39. Vallet, B.; Pierrot-Deseilligny, M.; Boldo, D.; Brédif, M. Building footprint database improvement for 3D reconstruction: A split and merge approach and its evaluation. *ISPRS J. Photogramm. Remote Sens.* **2011**, *66*, 732–742. [CrossRef]
40. Commandeur, T. Footprint Decomposition Combined with Point Cloud Segmentation for Producing Valid 3D Models. Master's Thesis, ITC, Delft, The Netherlands, 2012.
41. Pouliot, J.; Roy, T.; Fouquet-Asselin, G.; Desgroseilliers, J. 3D Cadastre in the province of Quebec: A First experiment for the construction of a volumetric representation. In *Advances in 3D Geo-Information Sciences*; Kolbe, T., König, G., Nagel, C., Eds.; Springer: Berlin, Germany, 2011; pp. 149–162.
42. Guo, R.; Luo, F.; Zhao, Z.; He, B.; Li, L.; Luo, P.; Ying, S. The Applications and Practices of 3D Cadastre in Shenzhen. In Proceedings of the 4th International Workshop on 3D Cadastres, Dubai, UAE, 9–11 November 2014; pp. 299–312.
43. Weiler, K.; Atherton, P. Hidden surface removal using polygon area sorting. In Proceedings of the ACM SIGGRAPH Computer Graphics, San Jose, CA, USA, 20–22 July 1977; pp. 214–222.