Modelling and analysing 3D buildings with a primal/dual data structure[*]

Pawel Boguslawski and Christopher Gold and Hugo Ledoux

December 17, 2010

While CityGML permits us to represent 3D city models, its use for applications where spatial analysis and/or real-time modifications are required is limited since at this moment the possibility to store topological relationships between the elements is rather limited and often not exploited. We present in this paper a new topological data structure, the *dual half-edge* (DHE), which permits us to represent the topology of 3D buildings (including their interiors) and of the surrounding terrain. It is based on the idea of simultaneously storing a graph in 3D space and its dual graph, and to link the two. We propose Euler-type operators for incrementally constructing 3D models (for adding individual edges, faces and volumes to the model while updating the dual structure simultaneously), and we also propose navigation operators to move from a given point to all the connected planes or polyhedra for example. The DHE also permits us to store attributes to any element. We have implemented the DHE and have tested it with different CityGML models. Our technique allows us to handle important query-types, for example finding the nearest exterior exit to a given room, as in disaster management planning. As the structure is locally modifiable the model may be adapted whenever a particular pathway is no longer available. The proposed DHE structure adds significant analytic value to the increasingly popular CityGML model.

1 Introduction

With the newly adopted Open Geospatial Consortium (OGC) standard CityGML it is possible to represent the different aspects of three-dimensional (3D) city models (OGC, 2008; Kolbe, 2008). This representation can be multi-scale since five levels of details (LODs) for the same city can be stored: from LOD0 where for example the terrain and the transportation network are stored, to LOD4 where buildings have detailed roofs structures, windows, rooms and even pieces of furniture. Also, extensions to the base model are possible for specific applications: there exist extensions for the modelling of noise, of floods (Schulte and Coors, 2008) and of sub-surface infrastructures. However, as explained Section 2, while CityGML provides great flexibility for the storage and the representation of 3D buildings, the spatial data model used currently —the Geography Markup Language GML (OGC, 2007)—has limited capabilities for storing explicitly topology, and offers no tools/algorithms to build the topological data model. In most cases, only the geometry of the 3D models are stored. While the connectivity information between the elements of the model can always be extracted on-the-fly, for many applications it is highly desirable to have them explicitly stored in a topological data structure (Ellul and Haklay, 2006; Ellul, 2007; van Oosterom et al., 2002; Zlatanova et al., 2004). Examples of such applications are when spatial analysis functions and/or dynamism is involved: updating the model in real-time, routing for emergency situations, etc.

In this paper we propose a new topological data structure and a set of construction and manipulation operators that permit us to construct, represent and *analyse* the buildings (and their interiors) of 3D city models. Our structure, called the *dual half-edge* (DHE), allows us to store explicitly: (i) the geometry of the elements of the model (the different buildings, the rooms, the terrain, etc.); (ii) the topological relationships between the elements of different dimensionality (so we can navigate from a given point to all the connected faces, from polyhedron to polyhedron, etc.); and (iii) the attributes for any elements of the model (points, line segments, faces and polyhedra). We achieve this by storing simultaneously both the *primal* and the *dual* subdivisions of a 3D city model (we assume here that the model divides the 3D space into different polyhedra and that

[*]Premilinary version of a paper accepted for publication in ISPRS Journal of Photogrammetry and Remote Sensing.

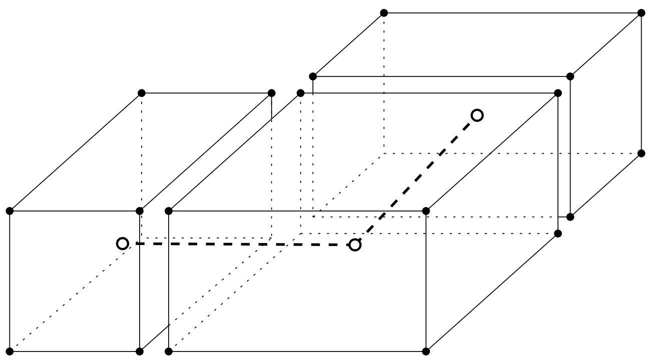

Figure 1: The DHE models 3D subdivisions by representing the boundary of each polyhedron separately with a graph (black lines), and two adjacent polyhedra are linked together by the dual graph (dashed lines). Both graphs are inter-connected.

we have one "universe" polyhedron). Duality, which is described in Section 3, is a concept that implies that two subdivisions are inter-connected: they represent the same thing but from a different point-of-view. Fig. 1 shows the general idea behind the DHE, which is that every polyhedron is represented independently with an edge-based structure (a *b-rep* model), and adjacent polyhedra are linked together by their dual edge. We thus use the primal subdivision for the geometry of the model, and the dual for the connectivity.

The DHE builds upon our previous work on a structure called the *augmented quad-edge* (AQE) (Gold et al., 2005; Ledoux and Gold, 2007). It permitted us to simultaneously store the Delaunay tetrahedralization and its dual the 3D Voronoi diagram, but the storage of arbitrary polyhedra and the incremental construction of models was not possible. As explained in Section 3, the DHE circumvents these problems by defining atomic elements and atomic construction operators (based on Euler operators) that can maintain a partially valid model (i.e. dangling edges and dangling faces are allowed). In Section 2 we compare the DHE with other 3D topological structures used in GIS and CAD. It should also be noticed that this paper builds on early work related to primal/dual data structures (Boguslawski and Gold, 2010) and contains several significant improvements including the automatic construction of CityGML models, the handling of holes and cavities, and the automatic construction of the dual graph for navigation.

The fact that both the primal and the dual subdivisions of a 3D model are explicitly stored and inter-connected has many advantages in practice: see for instance Gold (1991) for a discussion in 2D and Ledoux and Gold (2007) in 3D. We demonstrate in Section 4 how the DHE and its construction and navigation operators are beneficial for the modelling and the analysis of 3D buildings, stored for instance in CityGML. An important benefit is that the dual is always constructed and preserved automatically, which permits us not only to navigate between rooms but also to perform more complex operations, like routing in buildings, without having to reconstruct the dual graph each time the model is modified, and to identify all the parts of a building that are accessible from a given starting location (for security purposes). Perhaps more importantly, the DHE is locally modifiable (i.e. without a global reconstruction of the structure) so that real-time modelling of buildings is possible; one can think of applications related to disaster management when a certain part of a building is inaccessible and the model must be updated and, consequently, the navigation network. Finally, we believe our new data structure to be flexible and capable of handling common real-world situations such holes in faces/polyhedra and non-manifold situations (e.g. when 2 polyhedra are only adjacent by one line segment or a vertex).

2 Related work

The related data structures to store 3D models are many since they have been developed in different disciplines for different purposes (e.g. CAD, solid modelling, GIS, and meshing). We describe in this section only the most relevant in the context of modelling several 3D buildings and their interior.

2.1 GIS data models

In the context of GIS and of city modelling, 3D objects (e.g. houses, bridges, towers) are often represented with "b-rep" models (boundary representation) using primitive elements such as nodes, edges and faces. Many applications represent individual 3D objects, and topological relationships are not explicitly stored. The rationale behind that choice is that these can always be extracted on-the-fly. However, in practice, doing so in 3D is still a problem, especially if the dataset is not consistent (Gröger and Plümer, 2010). Perhaps the best known topological data model is the 3D *formal data structure* (FDS) of Molenaar (1992), which defines four primitives (nodes, edges, faces and solid) linked together by certain topological relationships; geometric constraints are also defined to ensure that for instance polyhedra do not overlap. Several have simplified this space-consuming model: see for instance Zlatanova (2000) and Coors (2003). The ISO-19107 abstract specifications ISO (TC211) also uses the four primitives to represent the topology of 3D objects: a solid is formed by an unordered set of faces, and faces have links to the 2 solids they bound. Other topological relationships, more complex to calculate—co-boundaries for instance—are optional. By comparison, the DHE uses only 2 primitives (nodes and edges) and it explicitly stores more relationships. While GML3 implements the ISO-19107 specifications, at this moment CityGML does not support the topological primitives, only faces shared by 2 solids can be re-used with `xlink` (Kolbe, 2008).

One should be careful in comparing these data models to the data structure we present in this paper. As explained in Frank (1992), a data model is more abstract than a data structure, and the implementation of a data model is made with a data structure. Another major difference is that these models do not in general offer algorithms or methods to compute the topological relationships: it is assumed that they can be performed. The GIS data models already mentioned can be realised with the DHE because it provides construction and navigation operators, as Sections 3 and 4 show.

2.2 Indoor space navigation

To perform indoor navigation and routing, Lee (2007) and Lee and Kwan (2005) also use the Poincaré duality. They define a 3D topological data model to represent buildings, and based on it they extract the dual graph (which is used for routing). However, unlike the DHE, the primal and dual graphs are not "linked", which means that each modification to the primal implies a reconstruction of the dual graph. Also, their data model for the primal subdivision has severe restrictions: faces must be either horizontal or vertical (they extract the dual graph based on a project of the rooms to a 2D plane). The DHE does not have such restrictions (although the examples shown in Section 4 contain only vertical and horizontal planes), and the graphs are inter-connected (if one is modified, the dual is automatically updated by the same operator). Observe also that Becker et al. (2009) have recently extended the model of Lee (2007) and Lee and Kwan (2005), but their dual graph is also not connected to the primal.

2.3 CAD

Several structures to represent 2-manifolds have been proposed. Unlike GIS where faces are often explicitly stored these consider the b-rep as a graph (nodes and edges) embedded in 3D space (faces are implicit). Early CAD data structures were based on the *winged-edge* (Baumgart, 1975), and different variations such as the the DCEL (Muller and Preparata, 1978) and the *half-edge* (Mäntylä, 1988) have been used. The latter also detailed the requirements for a set of topological operators (called Euler operators) that could modify the boundary of a single 2-manifold (also called a *shell*) without breaking its topology. These are used extensively in CAD to construct and modify models. A shell (or 2-manifold) is a 2D boundary of a 3D object—this must remain unbroken during modification by Euler Operators. If the boundary is 'awkward', e.g. with protruding edges or faces, then this is a non-manifold model (see Fig. 2).

2.4 Dual representation

The quad-edge data structure (QE) of Guibas and Stolfi (1985) was also developed to represent 2-manifolds, but it goes further than the CAD structures mentioned since it simultaneously stores both the primal and dual subdivisions. Each quad-edge represents one geometrical edge of a subdivision and its dual edge. Tse and Gold (2004) showed that sets of Euler operators could be formed in a particularly easy fashion by using the quad-edge. The quad-edge data structure for a connected graph on a 2-manifold was of particular interest because it combined the primal and dual graphs, and showed that this made navigation within the graph particularly easy, whether or not the dual was of specific interest in the application. Applying this to Euler operators indicated the potential value of an explicit dual.

2.5 Data Structures for non-manifold models

The data structures so far are appropriate for 2-manifolds but are insufficient for complex models—such as building interiors—where more than two faces may meet at an edge. The *radial-edge* data structure of Weiler (1988) was introduced to handle non-manifold models, where the junction of more than two faces at an edge, for

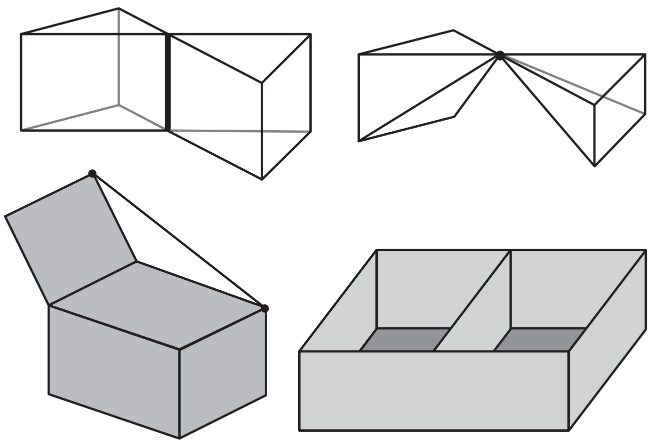

Figure 2: Examples of non-manifold models (Lee, 1999).

example, precluded the use of manifold models. This complex but well-used structure is applied in many non-manifold CAD systems. Lienhardt (1994) defined the *G-Map* (Generalized Map) for cellular models—which are generalizations of b-reps since each k-cell is recursively decomposed into cells of lower dimensionality, and the topological relationships between adjacent k-cells are kept. For example, in two dimensions an atomic cell tuple might be a vertex "seen from" an edge "seen from" a face. All combinations of tuples are stored, which results in a space-consuming data structure. Although G-Maps store only one subdivision, it offers efficient functions to extract all the elements of the dual subdivision. The *coupling-entity* data structure is perhaps closest to our approach (Yamaguchi and Kimura, 1995). There is no skeletal data structure (with multiple edges meeting at an edge), as is the case in the Radial-Edge. The atomic element is the 'feather': a combination of face, edge and vertex. Each of these requires two types of pointers: a mate pointer to its partner (a fan, blade or wedge, around a particular face-use or paired face-use) and a cyclic pointer (around a loop cycle, a radial cycle or a disk cycle). (A face-use is a copy of a face to be used by the polyhedron on one side of that face.) They point out that due to dependencies these six pointers may be reduced to three: they choose to use mate pointers exclusively. However it appears to us that it does not handle the case of two cells joined only at a vertex. Like the DHE there are pointers to adjacent shells, but there is no representation of the dual.

To our knowledge, one of the few structure permitting the simultaneous representation of both the primal and dual subdivisions of a cell complex is the *facet-edge* structure of Dobkin and Laszlo (1989). It is based on face-edge pairs: part of an edge together with part of its associated face. For every face-edge pair there are four facet-edges, corresponding to the four ways of describing 'clockwise' directions within the face and around the edge. The facet-edge structure comes with a set of operations to modify cells and to navigate within both subdivisions. Its generality makes manipulation of a single cell too complex, and hence the authors suggest storing extra information for each edge and using the quad-edge operators. Also, to our knowledge, it has never been fully implemented.

The augmented quad-edge (AQE), on which the work in this paper is based, uses the quad-edge to individually represent each polyhedron (Ledoux and Gold, 2007). With this structure, it is possible to navigate within a single cell with the quad-edge operators, but it was initially impossible to navigate to adjacent cells. That problem was solved by using the dual edge to link two adjacent polyhedra. Thus the ability to navigate the primal and dual graphs of a single edge using the quad-edge approach is preserved, and the 3D dual edge forms part of a complete dual cell complex with exactly the same structure. As mentioned in the Introduction, Ledoux and Gold (2007) showed that this structure provides navigation for 3D Voronoi/Delaunay structures, however the construction operators were complex and arbitrary 3D models and non-manifold cases were not supported.

3 The dual half-edge (DHE)

The constraints of this paper prevent the presentation of all the details of the implementation of this structure: the authors are preparing a companion paper for publication within the Computer Aided Design literature. Here we can only give a summary.

In our research we focus on cell complex construction and manipulation methods. Poincaré duality is an important issue there—we use the dual graph to take all the primal cells (which are individual b-reps) and connect them into one complex. Our models can be considered as complexes of 2-manifold cells topologically

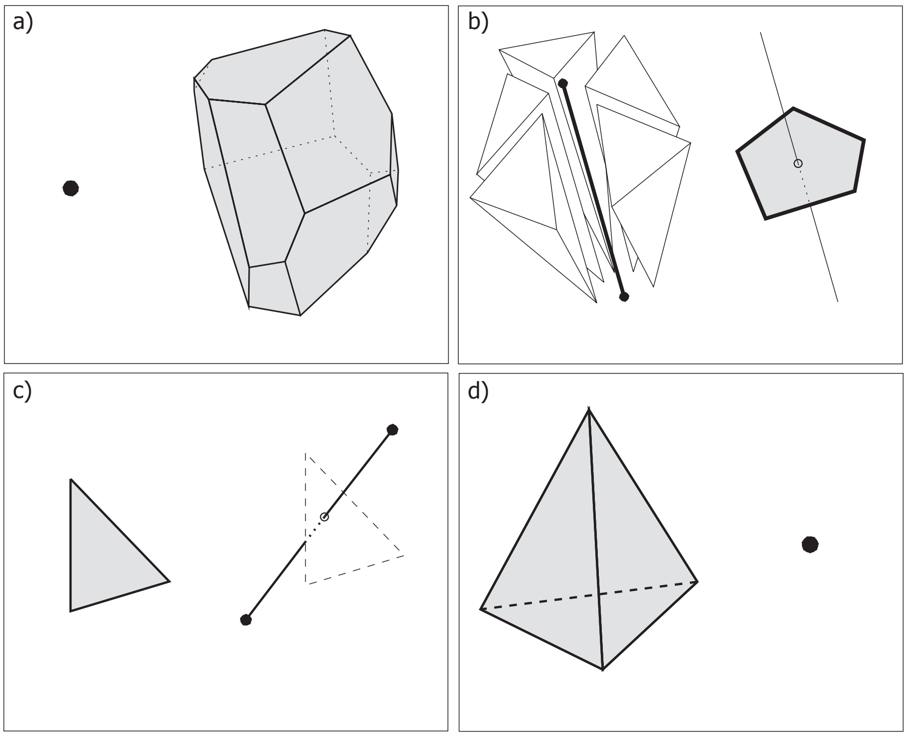

Figure 3: 3D Poincaré duality: a) A primal vertex and a dual cell; b) A primal edge and a dual face; c) A primal face and a dual edge; d) A primal cell and a dual vertex. (Ledoux and Gold, 2007)

connected by a dual structure (non-manifold models can be 'simulated' with our models). The only entities used in a construction process are edges and vertices. Poincaré duality states that for a space of dimension d and an element of dimension $k <= d$ a dual element exists of dimension $d - k$. Thus in 3D a vertex has a dual cell (volume), a face has a dual penetrating edge, etc. (see Fig. 3). Therefore a cell can be represented as a single dual vertex—there is no need to create a special class of objects to represent a cell: properties of a cell (i.e. id, volume) can be assigned to its dual vertex. Adjacent cells of a complex are connected by a shared face, which is represented by a dual edge. This edge links two dual vertices representing the adjacent cells. In our model each face is represented by a bundle of dual edges—each dual edge is permanently linked with an associated edge from a primal face. Because edges in a bundle are bounded by the same vertices, and they form a loop, so navigation around the bundle is possible. (We consider the bundle as an edge in terms of the Poincaré duality.) Properties of a connection between two adjacent cells (i.e. distance, wall finishing) can be assigned to one of the edges in the bundle and considered as properties of this bundle, or references to these properties can be assigned to all edges in the bundle.

3.1 Details of the data structure / Atomic elements

The DHE was inspired by the half-edge of Mäntylä (1988) and the AQE of Ledoux and Gold (2007). It uses the half-edges to represent each polyhedron, but like the AQE adds the dual to each element. Thus we have pairs of half-edges (see Fig. 4), one in primal space (he) and one in dual space ($he.D$), which are permanently linked together. Each half-edge has pointers: to a vertex ($he.V$); to the paired half-edge that forms the opposite side of the edge ($he.S$); to the next half edge around the associated vertex ($he.N_V$); and to the next half edge around its own face ($he.N_F$)—so the primal part contains a loop pointer around the face of a single cell and the dual part contains a pointer around the face in dual space—which is equivalent to a radial loop around an edge in the primal space. This simplifies the overall structure considerably.

It is interesting to note that the original half-edge structure consists of twinned pairs of half-edges in 2D, and the same would be true for a model of the dual. The quad-edge combines these two into an element containing four half-edges, with improved navigation features. The DHE splits these four elements in a different way: one primal and one dual in each element. This preserves the 2-manifold navigation properties of the QE and the easy navigation between cells of the AQE—while allowing flexible 'twinning' of half-edges in the construction process. Thus each half edge may be linked to a single other half edge, forming part of a simple shell, as with b-rep models. To access adjacent shells one goes to the dual of one of these edges and moves around the dual

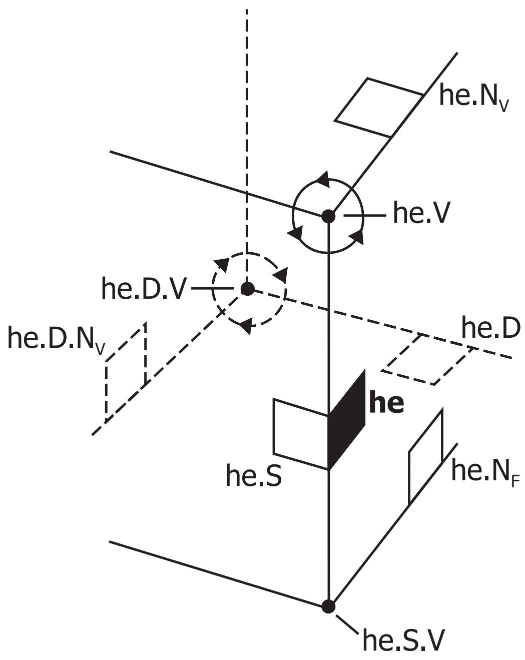

Figure 4: DHE pointer based data structure; primal graph (solid lines) is connected permanently with the dual graph (dashed lines); he - original half-edge; S, N_V, N_F, D, V - pointers.

face until the adjacent primal shell is found. There are no shells without an adjacent shell: the boundaries of the model are enclosed by an exterior shell; see Section 3.2 below.

A complete primal ('geometric' or 'building') model involves vertices, edges, faces and volumes. Primal vertices and edges are equivalent to volumes and faces in the dual model. Primal faces and volumes are equivalent to edges and vertices in the dual model. Thus all elements may be represented by a graph structure consisting of vertices and edges—half in primal space and half in dual space. Observe that each element has a dual role: attributes may be assigned to its primal and/or dual meaning.

In terms of storage, whether in a graph structure or a data base, there are only two atomic element types: vertices and edges. Both may have attribute information associated with their primal or dual roles, but only edges have topological pointer information: to the second half of the edge, to the dual edge, to the next edge around a vertex, to the next edge around a face, and to its associated vertex.

3.2 External cell

While this model is complete in the interior of a model, there could be problems at the exterior boundary. Each room has a simple quad-edge type shell. The dual edge for each face connects the interior 'room' node with that of an adjacent room. If there is no adjacent room the dual graph structure—itself a collection of simple shells—will be incomplete, and some navigation operations could fail by "falling off the edge of the world". To prevent this, and to make all the required Euler operators viable, a simple rule of the DHE is that there is always a single exterior shell—the 'exterior' or the 'rest of the world'. Thus an exterior face of a room will have a dual edge connecting the 'room' vertex with a single universal 'exterior' vertex: all exterior faces will have dual edges connecting to this vertex. This property is particularly useful when joining initially-separate buildings (cell complexes).

Without the external cell it is not possible to create dangling edges, and thus the incremental construction of models, as explained in the next section, would not be possible. Connection of two half-edges in primal space (geometry) does not cause a problem, but in the dual 'not-paired' halves prohibit navigation within the model. However since the external cell is kept along with the internal cells, every half-edge has its counterpart in the adjacent cell: in the external or internal cell.

3.3 Construction of models with Euler operators

The construction of a 3D model has several stages. First the cells of a complex are created, and then all adjacent cells are connected. However, not all cells are the same: arbitrary polyhedra can have different shapes,

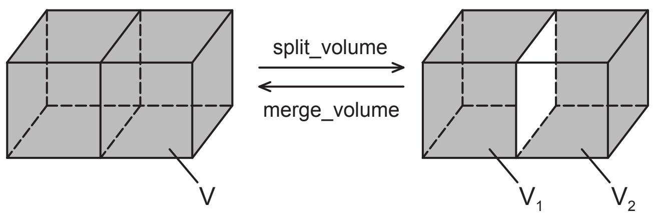

Figure 5: `split(merge)_volume` - an example of the extended Euler operator introduced by Masuda (1993): `split_volume` splits volume V in two volumes V_1 and V_2; `merge_volume` is a reverse operator.

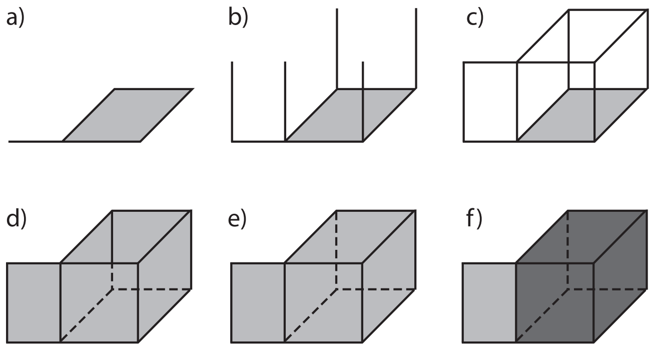

Figure 6: Lifting a face and a wired edge. (Masuda, 1993)

a different number of faces, etc. Thus the process of cell construction should be 'atomized' to make incremental construction (edge by edge) possible. This is possible with Euler operators, which are widely used in CAD for modifying b-rep objects (Mäntylä, 1988). They are 'atomic' operators that make only a minimal change in a model while preserving topological integrity (but do not provide rules to check or fix the topological consistency). Examples are: `Make Edge and Vertex` (MEV)—create a new edge and new vertex, and connect with the model at a given location; and `Make Edge Face` (MEF)—create a new edge between two vertices (that splits a face into two parts). Each operator is accompanied by a paired reverse operator (e.g. `Make Edge and Vertex`—`Kill Edge and Vertex`).

Construction of a single cell (without the dual) is a simple process using traditional Euler operators. For non-manifold models we need to be able to construct more complex structures: to create non-manifold cell complexes the standard Euler operators should be extended to manage connections between cells and to include operations like joining two cells by a shared face, edge or vertex. The idea of extended Euler operators was presented by Masuda (1993). For example (see Fig. 5), the `split_volume` operator splits a volume into two by adding a face that 'cuts' the volume in two. This face is shared by the volumes, thus the resulting model is non-manifold. The reverse operator `merge_volume` removes the face to merge the two adjacent volumes.

Masuda (1993) also presented an example of a lift operation based on a sequence of Euler operators. The lift operation is used to extrude 3D objects from their 2D footprints. The process is shown in Fig. 6. A face and a wire edge are an input 2D model (see Fig. 6a). Wire edges are generated from existing vertices (see Fig. 6b); and other wire edges are generated between newly created vertices (see Fig. 6c). In the next steps, five side faces are generated (see Fig. 6d); and one closing face on a top of an empty box (see Fig. 6e). In the last step a volume is defined (see Fig. 6f). The result is a non-manifold model—a solid (a box) with a laminar face attached.

Our set of construction operators is based on Euler operators and the extended version. This covers a "spanning set" (Braid et al., 1980) and is a part of a bigger set of all possible Euler operators. We have added the dual graph to these operators, so that our construction and manipulation operators also update the dual. All changes are made locally, thus there is no need for re-computation of the dual graph (graph of connections) in the whole model.

Using standard Euler operators we can easily build polyhedra of any shape. Fig. 7 shows the sequence used to create a cube from individual edge elements (this is one of many possible sequences). For clarity the dual is not shown, but it is present at each step, as with the external cell. The dual connects the internal and external cells together into one cell complex. Faces are defined automatically upon closure of the edges, and volumes are determined whenever a closed set of faces is completed.

It should also be added that holes and cavities (an example is shown in Fig. 8) are allowed: we use a so-called

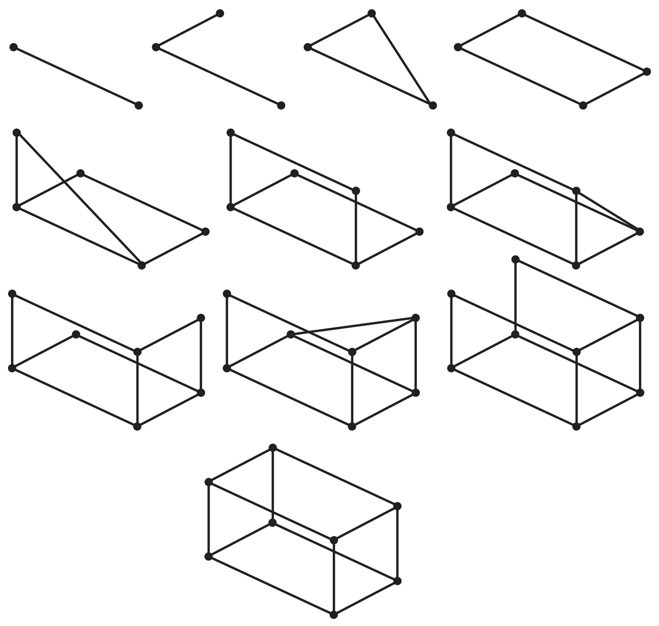
Figure 7: One of many possible ways to construct a cube using Euler operators.

'bridge edge' to connect internal rings (holes) to the outside ring (the face); and a bridge face to connect an internal cell (a cavity) with the outside cell. These permit us to represent many real-world situations.

With the extended set of Euler operators we can build a cell complex: we can join/separate or merge/split cells of the complex. Fig. 9 presents this idea. We perform a sequence of simple, atomic operations, the same as with standard Euler operators. Lee (1999) shows a similar example using standard Euler Operators (with no dual) for joining or splitting cells.

3.4 Navigation operators

Navigation in a model is possible with a set of *navigation operators* that use topological connections (pointers) set during the construction process.

A half-edge is a topological element, thus information about neighbouring edges, vertices, associated cells, and the immediate adjacent cell is easily accessible. With these DHE pointers it is possible to navigate: to the paired half-edge (opposite side of the edge); to the next counter-clockwise (looking at the cell from outside) half-edge around a vertex; to the next counter-clockwise (looking at the cell from outside) half-edge around a face; and to the associated dual half-edge (see Fig. 4). These basic operators are used to define the set of compound operators for navigation between cells, for example to go to the adjacent cell: first go to the dual half-edge, take the opposite side of the edge, and then go back to the primal. Using the full set of operators it is possible to develop more complex functions for a cell complex: for example finding all neighbours of a cell involves finding all the half-edges around a dual vertex representing the primal cell. The opposite ends of these dual edges point at vertices representing neighbouring cells.

Navigation between cells that are not directly connected is also possible, since a dual node represents a primal cell, and connections between cells are in the dual. Thus any graph traversal algorithm can be used on the dual graph to find a path between these two nodes (e.g. the shortest path between source and destination nodes is the result for the Dijkstra algorithm (Dijkstra, 1959)). In order to navigate between these two nodes we start from a half-edge associated with the source node. Then we can navigate to a half-edge associated with the destination node using simple pointer operators on the edges (nodes do not store topological connections). Consequently the path can be recorded as a sequence of topological connections starting from the source half-edge. In the more general case, the source and destination can be in either the primal or dual graph—these two graphs are connected by pointers, and the navigation is the same in both of them.

3.5 Attributes / semantic information

The DHE data structure and its construction and navigation methods are application independent; models can be used in many fields (e.g. engineering, mathematical modelling, etc.). However without semantic information they are of little value. We allow attributes to be assigned to each entity in a model. Although the DHE only

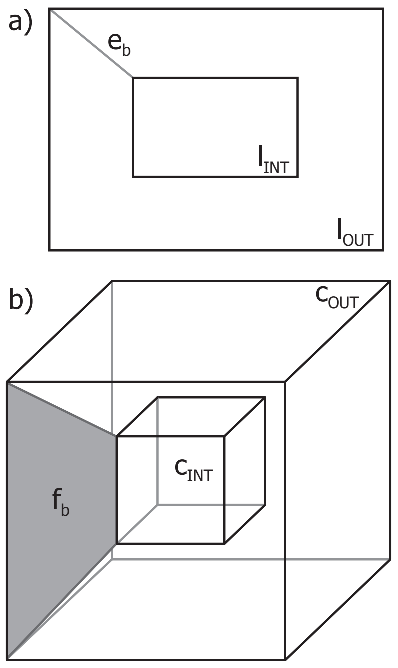

Figure 8: a) Hole in a face - bridge edge e_b connects internal loop l_{INT} with the outside loop l_{OUT}; and b) cavity in a cell - bridge face f_b connects internal cell c_{INT} with the outside cell c_{OUT}

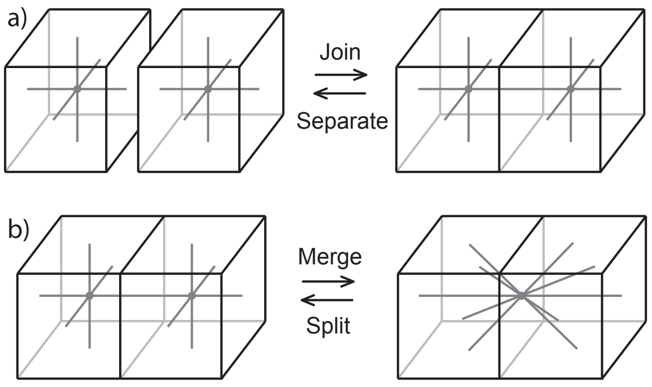

Figure 9: Operators for a cell complex construction: a) Join/Separate; b) Merge/Split.

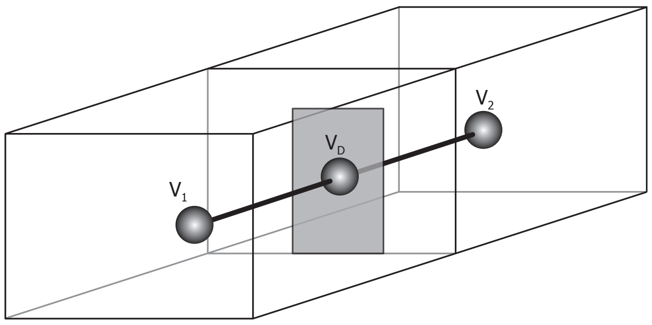

Figure 10: Two adjacent rooms V_1 and V_2 with the 'flat' door V_D in between. Only connection through the door is taken into account in an emergency management systems, however a direct connection through a shared wall between V_1 and V_2 exists in the model (not presented in the picture).

has node and edge entities, we exploit the dual to store attribute information to faces and volumes: the volume of a cell may be assigned to a dual node representing the cell; the wall colour to a dual edge representing a face. Attributes have meaning only for applications, and do not affect the geometry or topology of the model. For example in escape route planning the dual directed edge, describing communications between adjacent volumes (rooms) may have a cost/time value associated with its dual meaning. However, in a virtual museum this directed edge, in its primal mode, may be used to indicate the image or texture to be applied to one side of the wall in the building model.

4 Applications

We demonstrate in this section how our new data structure can be useful for the modelling, management and analysis of 3D buildings.

4.1 Doors

A building in general consists of several connected rooms that have volumes (corridors, office, storage spaces, etc. are considered as rooms too), so they are represented by primal cells. The geometry of a room can be easily modelled with the edges and nodes of a cell; relations between adjacent rooms can be represented with dual edges connecting cells. Relations can be described in terms of access level from one room to the adjacent one: access to the next room is easy by a door; the next room is not accessible because of a wall, but if the wall is thin a hole can be made. It may not be possible to get directly to the next room, if the wall is made of concrete. This is an example of a basic set of attributes that can be assigned to connections between rooms and then used as weights in graph traversal algorithms (e.g. Dijkstra).

Rooms are not the only objects in a building that are important. Walls, doors, windows, installations etc. are essential in many applications and can also be included in a model. They can also be represented as cells with geometry and volume, and attributes can be assigned to them. Further analysis can answer questions about a building structure: are there any pipes or wires in the wall between rooms A and B; is the door one- or two-leafed, etc.

Two approaches can be distinguished:

Type 1: Rooms are not the only objects in a model: walls, doors, windows, installations and other objects are represented with 'thick' cells too – non-zero volumes can be calculated from the geometry of the objects.

Type 2: Only rooms have a non-zero volume; other objects can be present in a model, but they are 'flat' (Fig. 10). Adjacent rooms are connected directly—there is no wall in between: they can also be connected by doors that are represented as double-sided flat faces. The volume of a flat object is zero, but there is still a dual node for this object.

4.2 Building model construction

To demonstrate the correctness of our methods we developed a prototype. We used the *Delphi* programming language to write it and an in-house graphics engine based on OpenGL for visualization. We reconstructed two

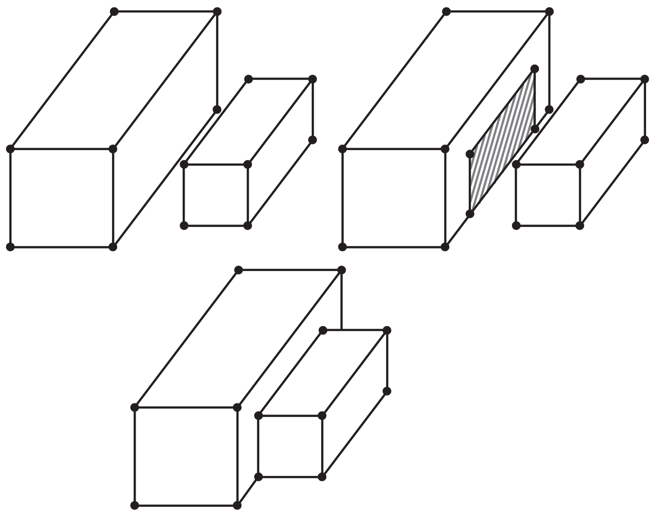

Figure 11: Boundary intersection module: a) two adjacent cells; b) new edges added to a bigger face create a new adjacent face; c) two cells connected.

building models with two different model types: *Type 1* and *Type 2*.

Three-dimensional building interior data sets are not easily available, but with the recent adoption of CityGML as a OGC standard, we should be able to obtain more and more buildings in that format (which uses GML for the three-dimensional geometry representation). As previously explained, the GML mechanism to represent topology has weaknesses and is seldom used: we use the DHE instead. We can detect if cells are adjacent based on their geometry, and we can easily connect cells by adjacent faces, but the real world is not so simple. Some cells need to be edited before they can be connected. Not every wall in a building is shared by exactly two rooms (e.g. one long corridor may have many adjacent rooms). Adjacent rooms can have walls of different shape or size. To solve this problem we had to develop a boundary intersection module. During the construction process we check the locations of two adjacent cells, add new edges if necessary, and then connect them (Fig. 11). We do not test if cells overlap; data sets have to be validated first using different methods.

The first model we reconstructed is a simple house (Fig. 12). This is an example of *Type 1*—with walls, ceilings and windows represented as thick cells. An original data set stored in CityGML format is available from the official CityGML webpage[1]. Good quality data in this set is valid and no extra cleaning is necessary. The CityGML LOD4 is included in the file; that means that data describing the interior with rooms, walls and even furniture is present. Besides the geometry there is also semantic information included: there are sections in the file representing single objects with their function (e.g. this cell represents a room; this set of faces represents a wall, window, door, stairs, etc.). Since holes and cavities are allowed in our models, we are able to import CityGML models and reconstruct the topology automatically. However this functionality is not implemented yet: in the current version of our CityGML format import application, holes and cavities are not detected automatically. The results are shown in Fig. 12 (it is possible to add bridge edges and bridge faces manually after a model is imported).

The second model—two buildings from the University of Glamorgan campus (Fig. 13a) connected by an overground passage (Fig. 13b)—is an example of *Type 2* (no walls between rooms; doors are flat). This is reconstructed from scanned paper plans. These plans were used as a raster background and all rooms were outlined and extruded in AutoCAD—the result was a set of 1300 cells that are not topologically connected. All the connections between adjacent rooms were set during the construction process using the DHE approach described in this paper. We want to use this model in emergency systems for finding escape routes from a building, thus only adjacency by a face is taken into account (only navigation through faces is possible); however cells can be connected by a shared edge or vertex. Because all adjacent cells are connected and we want to avoid navigation through solid walls, weights are assigned to connections between rooms; they describe how difficult is to move to the next room: an infinite value means no access, any other (positive) value is calculated from a geometric distance between the dual nodes representing adjacent cells. It is not easy in the plans to check a door's existence between rooms, and to input this information manually into the model. Doors were therefore modelled as well—but with zero thickness to put them in between two adjacent cells. Since doors are in the model we can calculate and assign weights only if two rooms are connected by a door.

[1]`www.citygml.org`

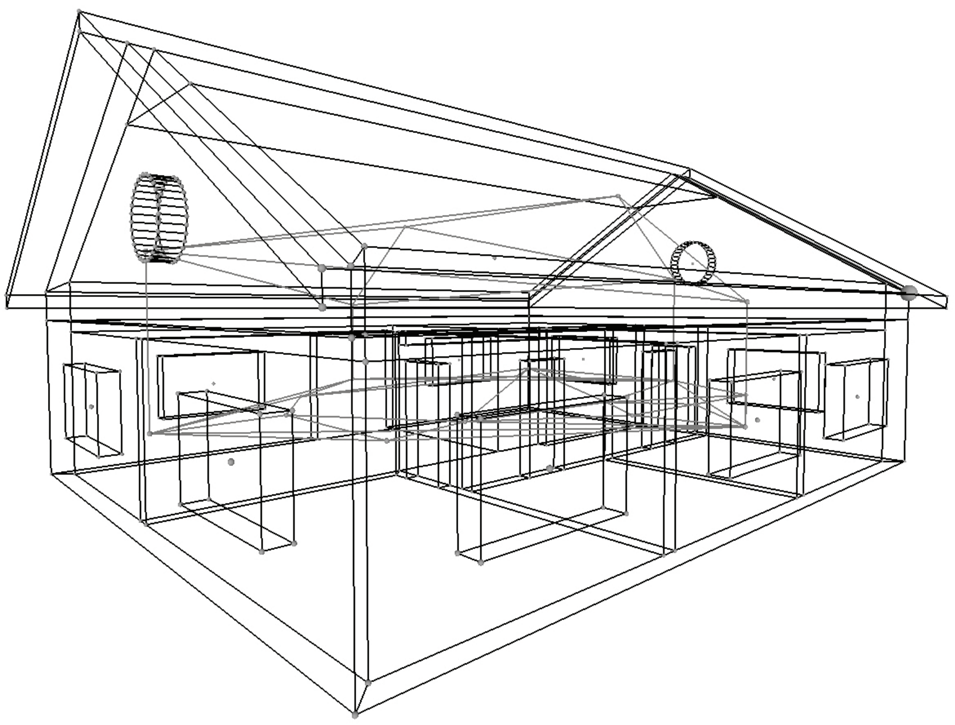

Figure 12: Model of a house reconstructed from the CityGML format (source: www.citygml.org) using the DHE data structure. Topological connections between cells are derived automatically from the geometry of the model. Walls, doors, windows are represented as thick cells (with non-zero volume). Holes and cavities are not reconstructed.

There is another element similar to a door: an "open space"—this flat cell is used to connect two parts of the same room; and has the same meaning in the weight calculation as the door element. It is useful when big rooms (e.g. long corridors, shopping malls) need to be modelled. It is easier to split them into smaller pieces than to analyse one room of a complex shape. It may be necessary to assign attributes (that were assigned to the original room) to two or more cells representing the same room after the split. Because only a reference to an attribute is associated with a cell, it is not necessary to store attributes redundantly—only references need to be multiplied; an attribute itself if stored only once. The complex feature handling method described by Gröger and Plümer (2010) may be also used. A hierarchical structure—a tree—is used for maintain aggregated solids. Atomic solids (whole rooms or parts of a room) are stored in tree leaves, while the root node represents the complex object (e.g. a building). Attributes could be assigned to the nodes of the tree or directly to the solids represented by leaves. Thus attributes could be assigned to any feature at any level of aggregation, while the feature is not represented explicitly in a model—for example, to a building, floor, corridor, room, etc.

In the current examples, building models represented by internal cells are enclosed by one external cell. Therefore all outside cells (at a boundary of a model) are connected to the exterior. However it may be required to represent external space above the ground (air) and below the ground (earth) as separate solids (Gröger and Plümer, 2010). A tessellation of the air and ground surrouning a modelled object was also presented in a full 3D data model by Penninga (2008). In our model, they would be represented as two separate cells with their dual volume vertices. This allows assigning different attributes not only to these volumes but also to connections between: underground rooms and the earth cell, and to overground rooms and the air cell.

4.3 Escape routes: shortest path analysis

The shape of the campus building is complex and for that reason the shortest paths between two cells are not always entirely inside. Sometimes it is faster to find a shortcut outside the building, and emergency assembly points are usually located outside buildings. Thus the surrounding terrain is included in the model and should improve the efficiency of rescue simulations. Terrain in our model is represented with cells (in the same manner as the building): they have a small thickness and they are connected to the building in the same way as building cells are connected together. Thus the same graph traversal algorithms can be used for the terrain and building part. An example is shown in Fig. 14—a projection of a ground floor with symbolic doors in a simple model

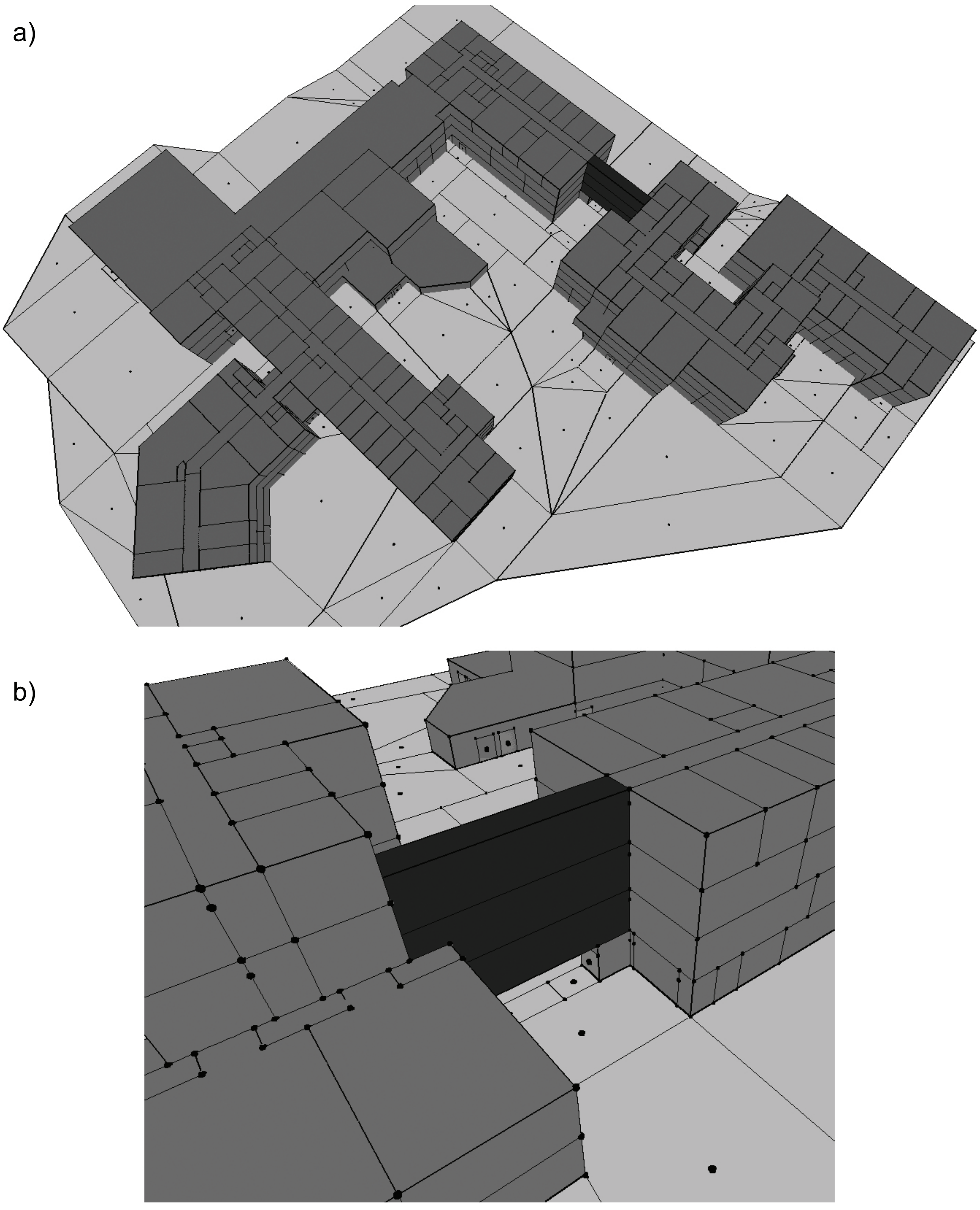

Figure 13: Two buildings from the University of Glamorgan campus connected by an overground passage modelled using the DHE data structure: a) light-grey cells represent terrain, grey cells represent rooms, dark-grey cells represent the overground passage between buildings; b) overground passage between two buildings - dark cells.

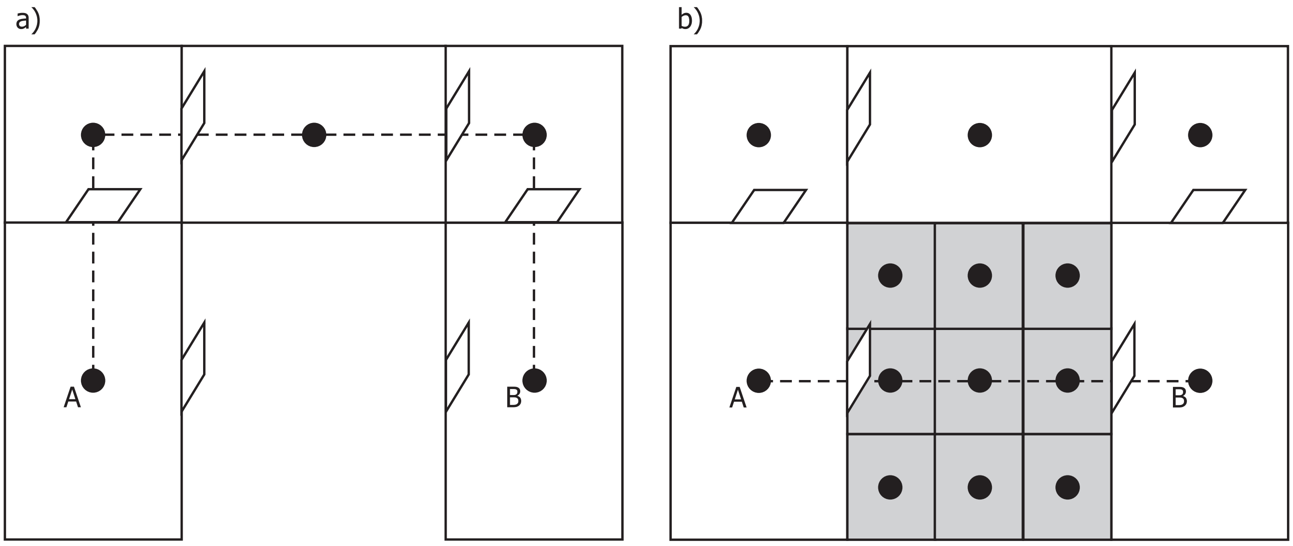

Figure 14: The shortest path (dotted line) between two rooms: A and B: a) No exterior terrain—the path is entirely inside the building: b) The building with the exterior terrain (gray mesh)—a shorter path exists in the model.

of a building is presented. Using the same algorithm a shorter path between two rooms can be found when the exterior terrain is present in a model.

We use the Dijkstra algorithm on the dual graph to find the shortest path between two specific rooms (unlike Lee (2007) our dual graph is automatically constructed). We use the same algorithm to find a route from a room to the nearest exit from a building—there is one source room and multiple exits. An exit can be any cell in a complex, but usually this is a cell representing a door connecting the building with the exterior. The locations of exits are not known at the beginning, thus they cannot be used as input parameter (which prevents us from using A* instead of Dijkstra).

The weights used in the Dijkstra algorithm are assigned to the dual edges connecting adjacent rooms. They are calculated based on the geometrical distances between the two nodes representing adjacent rooms only if there is a door or 'open space' between these rooms. Otherwise the weight is not calculated (or can be considered infinite) and this connection is not taken into account in the search process. All temporary attributes (e.g. cumulative distance, previous edge on the path, etc.) are assigned to nodes and half-edges of the dual graph. After the path is found this is visualized as a highlighted set of primal cells that occur on the way from the source to the destination.

5 Discussion and conclusions

The main thrust of this paper has been to demonstrate (but not to formally prove) that an understanding of the Pointcaré dual of a 3D cell complex—in particular of a 3D (geometric) building model—gives a clear template for a non-manifold data structure. All geometric elements (volumes, faces, edges, nodes) have representative entities, as do the dual elements, particularly the navigation network between adjacent cells or rooms. Indeed, the usual additional CAD entities of face-use, edge-use and vertex-use are handled easily, without additional structures. Volume entities are now introduced. Each of these may be given attributes appropriate to the application domain. The complete model may be represented with two fundamental element types: edges and vertices, providing great simplicity for conceptualization and implementation.

The value of the dual data structure for the navigation of cell structures has been mentioned before (Ledoux and Gold, 2007) but the new DHE data structure demonstrates its value in a more applied setting: incremental construction and modification of complex non-manifold structures is now much simplified. These results can be used for instance to build an ISO-19107- compliant topological data model from 3D datasets where only geometry is available.

A major contribution of this paper is the demonstration that Euler-type operators may be implemented with this mode, both 'true' Euler operators on a single shell, and related operators that manipulate multiple shells—or indeed split or join complete cell complexes. Building models are greatly facilitated if they may be constructed edge by edge, so intermediate cases with individual connected edges and faces that are not yet complete cells may be handled. In all cases, although difficult to visualise, the dual of each geometric edge is present, and if the connectivity of the geometric model can be achieved then all the information that is necessary to connect the dual structure is also available: while more pointers must be connected the conceptual 'cost' is unchanged.

As long as the primal structure is a connected graph then the dual graph is correct—for example, two separate cell complexes may be connected by a single edge using an Euler-type operator: the dual structure is complete and may be navigated, and additional edges immediately added. The same is true if the two cell complexes are initially connected only by a common vertex, or a common edge. Holes through a solid object are easily constructed by using a traditional bridging edge in each of the two faces penetrated—and even isolated cavities may be modelled by adding a 'bridging face'.

Because only vertices and edges are required in the structure, together with their associated primal and dual attributes, only two tables are required to store the model in a relational database. In this case a query is required to find each adjacency link. No attempt has yet been made to perform more efficient relational queries—this awaits future work. Similarly, more work is required to evaluate the functionality of 'stripped down' structures—for example it is possible to perform slightly limited navigation (and therefore construction) where the dual graph itself is removed, and replaced with a pointer directly to the adjacent shell: it may suffice for some applications, but not all queries are possible.

The availability of an incrementally-constructed non-manifold model may open up a set of applications not previously considered. One example is interior building surveying—a particularly difficult task in the absence of indoor GPS. The availability of relatively simple equipment for determining the position of some (non-reflective) point from a reference position permits rapid two-person surveying: one to take the reading and the other to express its relation to previous points (for example the next corner around a ceiling). Current equipment we have found has not the high precision of complete surveying systems, but it appears to be sufficient to express the relationships between adjacent rooms—the initial impetus for this project. Further applications, perhaps within BIM, CityGML or other full 3D applications are clearly possible, and await discovery. For example, Gröger and Plümer (2010) show how to decompose a 3D city model into simple solids with topological consistency checking rules. However no particular data structure is suggested. We believe the DHE data structure, which stores topological connections between elements, could be a useful supplement to their work. Also the modelling rules described there fit to the construction principles presented in this paper.

Role of the funding source The first author's research is supported by the Ordnance Survey and EPSRC funding of a New CASE award.

References

Baumgart, B. G., 1975. A polyhedron representation for computer vision. In: Proceedings National Computer Conference. AFIPS, pp. 589–596.

Becker, T., Nagel, C., Kolbe, T. H., 2009. A multilayered space-event model for navigation in indoor spaces. In: Lee, J., Zlatanova, S. (Eds.), 3D Geo-Information Sciences. Lecture Notes in Geoinformation and Cartography. Springer, pp. 61–77.

Boguslawski, P., Gold, C. M., 2010. Euler operators and navigation of multi-shell building models. In: Developments in 3D Geo-Information Sciences. Lecture Notes in Geoinformation and Cartography. Springer, pp. 1–16.

Braid, I., C., Hillyard, l., C., Stroud, I., A., 1980. Stepwise construction of polyhedra in geometric modelling. In: Brodlie, K. W. (Ed.), Mathematical Methods in Computer Graphics and Design. Academic Press, pp. 123–141.

Coors, V., 2003. 3D-GIS in networking environments. Computers, Environment and Urban Systems 27 (13), 345–357.

Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271.

Dobkin, D. P., Laszlo, M. J., 1989. Primitives for the manipulation of three-dimensional subdivisions. Algorithmica 4 (1–4), 3–32.

Ellul, C., 2007. Functionality and performance—two important considerations when implementing topology in 3D. Ph.D. thesis, University College London.

Ellul, C., Haklay, M., 2006. Requirements for topology in 3D GIS. Transactions in GIS 10 (2), 157–175.

Frank, A. U., 1992. Spatial concepts, geometric data models, and geometric data structures. Computers & Geosciences 18 (4), 409–417.

Gold, C. M., 1991. Problems with handling spatial data—the Voronoi approach. CISM Journal 45 (1), 65–80.

Gold, C. M., Ledoux, H., Dzieszko, M., 2005. A data structure for the construction and navigation of 3D Voronoi and Delaunay cell complexes. In: Proceedings 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. Plzeň, Czech Republic, pp. 21–22.

Gröger, G., Plümer, L., 2010. How to achieve consistency for 3D city models. GeoInformatica Online First, DOI 10.1007/s10707-009-0091-6.

Guibas, L. J., Stolfi, J., 1985. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. ACM Transactions on Graphics 4 (2), 74–123.

ISO(TC211), 2003. ISO 19107:2003: Geographic information—Spatial schema. International Organization for Standardization.

Kolbe, T. H., 2008. Representing and exchanging 3D city models with CityGML. In: Zlatanova, S., Lee, J. (Eds.), 3D Geo-Information Sciences. Springer, Ch. 2, pp. 15–31.

Ledoux, H., Gold, C. M., 2007. Simultaneous storage of primal and dual three-dimensional subdivisions. Computers, Environment and Urban Systems 31 (4), 393–408.

Lee, J., 2007. A three-dimensional navigable data model to support emergency response in microspatial built-environment. Annals of the Association of American Geographers 97 (3), 512–529.

Lee, J., Kwan, M.-P., 2005. A combinatorial data model for representing topological relations among 3D geographical features in micro-spatial environments. International Journal of Geographical Information Science 19 (10), 1039–1056.

Lee, K., 1999. Principles of CAD/CAM/CAE Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Lienhardt, P., 1994. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. International Journal of Computational Geometry and Applications 4 (3), 275–324.

Mäntylä, M., 1988. An introduction to solid modeling. Computer Science Press, New York, USA.

Masuda, H., 1993. Topological operators and boolean operations for complex-based nonmanifold geometric models. Computer-Aided Design 25 (2), 119–129.

Molenaar, M., 1992. A topology for 3D vector maps. ITC Journal 1, 25–33.

Muller, D. E., Preparata, F. P., 1978. Finding the intersection of two convex polyhedra. Theoretical Computer Science 7, 217–236.

OGC, 2007. Geography markup language (GML) encoding standard. Open Geospatial Consortium inc., document 07-036, version 3.2.1.

OGC, 2008. City geography markup language (CityGML) encoding standard. Open Geospatial Consortium inc., document 08-007r1, version 1.0.0.

Penninga, F., 2008. 3D topography : a simplicial complex-based solution in a spatial DBMS. PhD thesis, Delft University of Technology.

Schulte, C., Coors, V., 2008. Development of a CityGML ADE for dynamic 3D flood information. In: Proceedings Joint ISCRAM-CHINA and GI4DM Conference on Information Systems for Crisis Management. Harbin, China.

Tse, R. O. C., Gold, C. M., 2004. TIN meets CAD—extending the TIN concept in GIS. Future Generation Computer Systems 20 (7), 1171–1184.

van Oosterom, P., Stoter, J., Quak, W., Zlatanova, S., 2002. The balance between geometry and topology. In: Richardson, D., van Oosterom, P. (Eds.), Advances in Spatial Data Handling—10th International Symposium on Spatial Data Handling. Springer, pp. 209–224.

Weiler, K., 1988. The Radial Edge Structure: A topological representation for nonmanifold geometric boundary modeling. In: Encarnacao, J., L., Wozny, M., J., McLaughlin, H., W. (Eds.), Geometric Modeling for CAD Applications. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, pp. 3–36.

Yamaguchi, Y., Kimura, F., 1995. Nonmanifold topology based on coupling entities. IEEE Computer Graphics and Applications 15 (1), 42–50.

Zlatanova, S., 2000. 3D GIS for urban development. Ph.D. thesis, ITC, The Netherlands.

Zlatanova, S., Abdul Rahman, A., Shi, W., 2004. Topological models and frameworks for 3D spatial objects. Computers & Geosciences 30 (4), 419–428.