International Journal of Geographical Information Science
Vol. 00, No. 00, Month 2014, 1–21

RESEARCH ARTICLE

Generation of navigation graphs for indoor space

Liping Yang[a]* and Michael Worboys[b]

[a]*School of Computing and Information Science, University of Maine, Orono ME 04469, USA*;
[b]*Department of Mathematical Sciences, University of Greenwich, London SE10 9LS, UK*

(*Received 00 Month 200x; final version received 00 Month 200x*)

This paper proposes a comprehensive approach to computing a navigation graph for an indoor space. It focuses on a single floor, but the work is easily extensible to multi-level spaces. The approach proceeds by using a formal model, based on the combinatorial map but enhanced with geometric and semantic information. The process is almost fully automatic, taking as input the building plans providing the geometric structure of the floors and semantics of the building, such as functions of interior spaces, portals, etc. One of the novel aspects in this work was the use of combinatorial maps and their duals to provide a compact formal description of the topology and connectivity of the indoor structure represented by a connected, embedded graph. While making use of existing libraries for the more routine computational geometry involved, the research develops several new algorithms, including one for computing the local kernel of a region. The process is evaluated by means of a case study using part of a university building.

Keywords: navigation graph; combinatorial map; algorithm; topology; geometry; semantics; indoor space

1. Introduction

It is generally recognized that the foundational data structure to support navigation in an indoor space is the navigation graph, whose edges represent paths of travel and whose nodes represent decision points, landmarks, or other places where information needs to be conveyed to or presented by the system. In this paper, we propose a comprehensive approach to generating a navigation graph taking as input the building plans providing

*Corresponding author. Email: liping.yang@umit.maine.edu

ISSN: 1365-8816 print/ISSN 1362-3087 online

DOI: 10.1080/1365881YYxxxxxxxx
http://www.informaworld.com

the geometric structure of the floors and semantics of the building, such as functions of interior spaces, and portals. Our approach focuses on a single floor, but the work is easily extensible to multi-level spaces.

The paper proceeds by systematically developing the stages required to move from building plans (what we call the *structure model* for the indoor space) to the navigation graph. Our work relies heavily on formal descriptions of the topology of spatial configurations, such as the combinatorial map (see (Worboys 2013) for a description of combinatorial maps in the context of spatial informatics). In this work the combinatorial map structure is extended to include the geometrical arrangement of the space. A straightforward construction transforms the extended combinatorial map into a dual map, and this provides a first approximation to the navigation graph. The dual map is then refined using the semantics of the indoor space (e.g., which walls contain doorways), and extra nodes and edges are introduced to take account of the geometry (e.g., junction nodes) and semantics (e.g., landmark nodes). It is particularly tricky to position nodes appropriately at complex corridor intersections, and in rooms with complex and often concave geometries. To handle this we develop several computational geometry constructs, such as kernel and local kernel, defined later in the paper. The result is the positioning of nodes and edges to form the navigation graph. We conclude with a case study, based on a university building, to demonstrate the ideas developed in this research.

The main contribution of the research reported in this paper is the development of a complete and unified process for the construction of navigation graphs of indoor spaces. The combinatorial map provides an elegant means of capturing the topology of the space and a one-step method for its conversion into its dual, thus providing the backbone of the navigation graph. Other contributions relate to the geometrical embedding of the graph, and its enhancement of the navigation graph using the semantics of the space.

2. Background

In this section we review some of the previous work that bears on our contribution. Surveys of models for indoor space are provided by Afyouni *et al.* (2014) and Worboys (2011), among others. Comparison of various graph-based models for navigation are provided in (Franz *et al.* 2005).

A hierarchical graph model for pedestrian navigation in indoor space that took into account some semantic features of the space was introduced by Stoffel *et al.* (2007). Stoffel *et al.* (2008) developed the concept of a hierarchical graph as a navigation graph and suggested an algorithm to automatically construct such a multi-level hierarchy from floor plans. In their graph structures, passages are usually represented by single nodes. This does not provide a sufficient level of detail in many typical navigation situations. Thus, as Becker *et al.* (2009) described, fine-grained subdivisions of corridors are needed.

An approach to indoor tracking using graphs was proposed by Jensen *et al.* (2009). This work emphasized the importance of semantically enriched models for developing intelligent devices for humans. It considered two categories of graph: the connectivity base graph (a labeled, undirected multi-graph), and the accessibility graph. The latter is a labeled, directed graph constructed to represent the movement permitted by doors or connections). This work also does not subdivide larger areas, such as corridors into subareas.

Although there is now a substantial body of work on navigation graphs (sometimes

referred to as route graphs), there is somewhat less on how they might be automatically generated. Most existing methods are based on dual structures and straight skeletons, as well as visibility graphs, and these are now reviewed.

Indoor structures and their navigation networks are closely related by duality (Lee 2004, Lee and Kwan 2005, Becker *et al.* 2009, Worboys 2011). From the computer gaming community, Borovikov (2011) introduced methods to generate navigation graphs based on Delaunay triangulations. A triangulation was constructed using the polygon structure of boundaries in the virtual space. The dual of the triangulation (i.e., Voronoi cells) formed the basis of the graph, which was simplified by removing edges intersecting with the structure (i.e., obstacles like walls). Because this generated navigation graph was aimed at game environments, many unnecessary nodes and connections between nodes were constructed, and this would not be practical for indoor space navigation in real world scenarios. A related method, targeted towards robotics and games, was introduced by Demyen and Buro (2006). Similar to Borovikov and Demyen, Wallgrün (2005, 2010) used the structure of obstacles in indoor space to build a topological route graph; he employed the methods introduced by Choset and Burdick (2000) to construct a generalized Voronoi diagram and then the corresponding generalized Voronoi graph from the obstacle structure. A coarse route graph was then computed using a simplification algorithm, which reduced many unnecessary nodes and edges. The OGC standard IndoorGML (Nagel *et al.* 2010) also uses dual graph to model network structures of indoor spaces, the basis of the duality in which is the node-relation graph (see Note 1). Boguslawski *et al.* (2011) employed a dual structure to model 3D buildings. Mortari *et al.* (2014) also used duality methods to generate their route graphs for indoor spaces. In particular, they used the dual of constrained Delaunay triangulation to generate the preliminary version of their route graph. Most of the duality based automatic approaches introduced above, especially those based on Delaunay triangulations, would generate many redundant nodes and therefore more unnecessary edges in corridors. Also, almost all the methods introduced above encountered the problem that some nodes, particularly those around corridor corners and intersections, were not positioned properly.

Straight skeleton based methods are another means of converting polygons into network structures (Haunert and Sester 2008, Worboys and Duckham 2004). Lee and his colleague (Lee 2004, Lee and Kwan 2005, Lee 2007) also used geometrical constructions based on the structure of the building. In their work, Lee (2004) developed a straight medial axis transformation algorithm (S-MAT) to create their geometric network model. For simple and relatively regular polygons, straight skeleton based methods, especially the S-MAT approach, can generate good results. However, in some complex cases, they would produce many unnecessary nodes and edges for human wayfinding, as other methods introduced above using duality.

A *visibility graph* for a navigational space is defined by a set of nodes and edges, where edges in the graph are those connecting two inter-visible nodes (Choset 2005). Using visibility graphs, Kneidl *et al.* (2012) constructed an algorithm to form a sparse navigation graph from a given spatial layout of buildings. The purpose of this work was micro-level pedestrian simulation in order to evaluate the practicality of various spatial designs. A *sparse navigation graph*, a subset of a standard visibility graph, is a spatially embedded graph that it is as sparse as possible but provides enough detail to serve as a basis for navigation. Navigation points in the sparse graph were placed at a certain distance from each convex corner of the given geometry. This distance depended on the scale of the space, and the specification of this distance was not discussed in this work. Edges were generated by connecting vertices based on a cone-based search method.

However, some edge connections brought unnecessary complexity to the resulting graph.

Much of the above research focuses on geometric and topological analysis of the structure of indoor spaces so as to construct navigation graphs. However, for effective assistance for human wayfinding, the navigation graph needs to be as simple as practical, and with nodes representing the salient points in the space, both from structural and human decision-point aspects. Thus we need to take account of the semantics of the entities in the space, and consider what might be salient points for human wayfinding. In particular, we are concerned with the affordances that entities in the indoor space provide. The term *affordance* was originally introduced by Gibson (1986), and denotes the action possibilities that entities provide, (for example, corridors provide the possibility of following a path between rooms). For our work we use the categories of entities in the indoor space that was developed in Yang and Worboys (2011a) (see Figure 1 in that paper). In particular, we take from that work as principal entities: Room, Corridor, Stairwell, Elevator, Wall, Door, and Doorway, with affordances containment, passage, connection, obstructing, portal, and portal covering. A fuller semantics is developed later in this paper.

3. Formal preliminaries

In this section we set out the formal apparatus used in this paper. The underlying construction is that of the combinatorial map, introduced by Edmonds (1960) and Tutte (1973) and developed in the context of spatial informatics by Worboys (2012, 2013). A combinatorial map provides a unique (up topological equivalence) formal representation of an embedded graph. More precisely, we have the following definition:

Definition 3.1: A *combinatorial map* $\mathbf{M}\langle S, \alpha, \tau\rangle$ consists of:

(1) A finite set S of elements, called *semi-edges*, where the number of semi-edges is even. We can write S as $S = \{a, \overline{a}, b, \overline{b}, \ldots, k, \overline{k}\}$
(2) A permutation α of S
(3) A permutation τ of S, which in cyclic form is $\tau = (a\overline{a})(b\overline{b})\ldots(k\overline{k})$

subject to the constraint that the collection of permutations $\{\tau, \alpha\}$ is transitive.

For a given combinatorial map $\mathbf{M}\langle S, \alpha, \tau\rangle$, every edge of the embedded graph is represented by a pair of semi-edges. Each cycle of permutation α defines the ordered sequence of semi-edges that represents the corresponding face of the embedding. A face is defined as the region on the left while traversing the semi-edges of a cycle of α. For example, in Figure 1 the combinatorial map of the graph embedding representing the indoor spatial scene is given by $\mathbf{M}\langle S, \alpha, \tau\rangle$, where $S = \{1, \overline{1}, 2, \overline{2}, \ldots, 30, \overline{30}\}$, $\alpha = (\overline{1}, 25, 21, \overline{20}, 18, 17, \overline{16}, \overline{12}, 11, \overline{9}, \overline{10}, 7, 4)(3, \overline{2}, 1, \overline{4})(6, \overline{5}, \overline{3}, \overline{7})\ldots(\overline{15}, 14, \overline{13}, 16)$, $\tau = (1, \overline{1})(2, \overline{2})\ldots(30, \overline{30})$.

To determine the nodes of a graph embedding from the combinatorial map, we calculate permutation β by the formula $\beta = \tau\alpha^{-1}$ where the product is a composition of functions, and α^{-1} denotes the inverse function of α. In the example of Figure 1, $\beta = (\overline{1}, \overline{4})(25, 1, 2)(\overline{2}, \overline{3}, 5)(4, \overline{7}, 3)\ldots(12, 11)$. Note that each cycle of permutation β represents a node, and the semi-edges in the cycle are arrayed in a counter-clockwise ordering of semi-edges around the node.

Definition 3.2: The *dual graph* of a graph embedded in the plane G is a graph such that each vertex denotes a face of G, and every edge of the dual graph links two adjacent

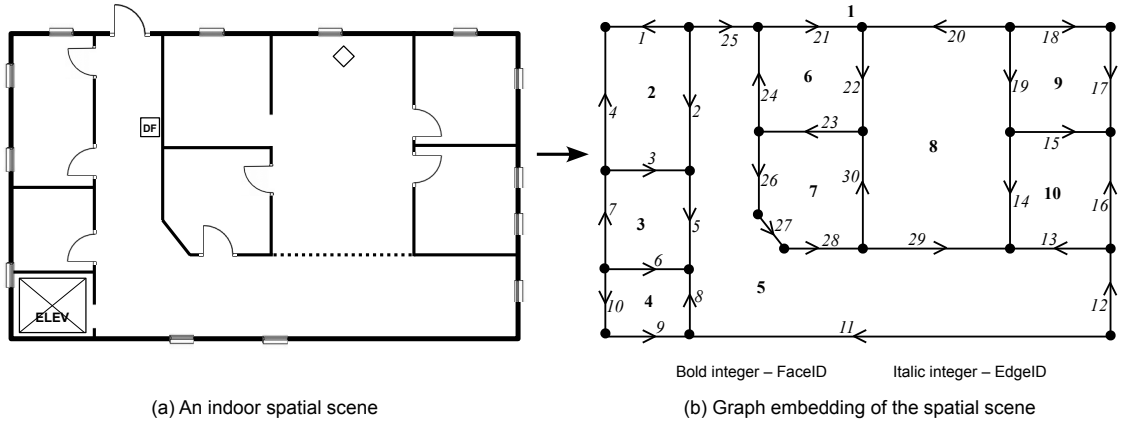

Figure 1. An indoor spatial scene and its combinatorial map

faces of G.

The navigation graph and the structure graph of an indoor space are closely related by duality (see Section 2 for details). The reason for this is that to move between two adjacent regions (faces) we need to take a path that crosses their mutual boundary, thus essentially traversing the dual graph. It turns out that the dual graph can very quickly be calculated from the combinatorial map. This is essentially achieved by swapping α and β. The following definition provides the solution as it gives a representation of the dual graph embedding as a combinatorial map, easily derivable from the original map.

Definition 3.3: Suppose given the *combinatorial map* $\mathbf{M}\langle S, \alpha, \tau\rangle$. Then the *dual map* $\mathbf{D(M)}$ of $\mathbf{M}$ is defined as the combinatorial map $\mathbf{D(M)}\langle S, \tau\alpha^{-1}, \tau\rangle$.

(An interesting technical wrinkle is that dualism is not quite symmetric in the context of combinatorial maps. In fact we have $\mathbf{D}^4\mathbf{(M)} = \mathbf{M}$.)

Combinatorial maps as defined above can only represent topology, and do not provide any geometrical information. To use them in navigation applications, it is necessary to bring in some geometry. We add the coordinate data to each node, and assume that all embedded edges are straight line segments. A combinatorial map enhanced with this additional data is termed an *extended combinatorial map (ECM).*

4. Approach

In this section, we present the general steps required for the generation of a navigation graph, given as input the structure and semantics of the indoor space, as may for example be found as part of the building plans or building information model (BIM). To aid the explanation we use the example shown in Figure 1 which we can imagine to be the floor plan of a doctor's surgery.

4.1. *Computing the ECM*

The first step is to use our input floor plan data to construct the ECM (shown schematically for our example on the righthand side of Figure 1). The algorithm to perform this was developed by the authors and is as follows: The input data are vertices and directed edges of the embedded, directed connected graph representing the floor plan, where each

vertex is labeled by a pair of coordinates and every directed edge is a directed straight line segment defined by its *from-vertex* and *to-vertex*. The algorithm generates an ECM containing the two permutations, α and β (see Section 3). Because it is easier to calculate permutation β, in the algorithm it is computed first by calculating and sorting the incident edges of each vertex separately in counter-clockwise order. The counter-clockwise ordered incident edges of a vertex form a cycle in the permutation β corresponding to that vertex. The next step is to compute the other permutation (i.e., α) using the formula $\alpha = \beta^{-1}\tau$, giving the faces. Each ECM has one external face and one or more internal faces. For example, face 1 in the righthand side of Figure 1 is the external face, and the rest are internal. The final step is to identify the face type (internal or external) of each face (i.e., cycle) in permutation α. According to the face definition introduced in Section 3, the vertices of each internal face are always ordered counter-clockwise and the external face clockwise. So we can use polygon orientation to identify the face type of each face in permutation α.

4.2. *Constructing the simplified dual map*

The ECM from the previous step is enhanced with semantic information, resulting in what we term the *structure graph*. The structure graph for our example is shown on the top left part of Figure 2. We note that to the topological and geometrical structure, computed using the methods of the previous section, has been added information about portals and terminals (i.e., corridor dead ends), information about which is captured from semantic input. We also make a distinction between a physical boundary (shown with a continuous line) and a functional boundary, for example, between a corridor and a waiting room (shown with a dashed line). As described in Section 3, it is a simple step to use the ECM to construct the dual map. This is shown in our example on the top righthand part of Figure 2 (For clarity, we have omitted most of the edges of the dual graph that connect to the external face.)

The next step is to use the structure graph to eliminate some of the edges of the dual map. Remembering that our final objective is to use the dual map to arrive at the navigation graph, we retain only edges of the dual map that relate to edges of the structure graph that contain portals. The result we term the *simplified dual map (SDM)*, and this is shown for our example in the bottom righthand part of Figure 2. To make a closer approximation to the navigation graph, we next add portal nodes and their associated edges to the SDM. The result we term the *skeleton navigation graph*, as shown by the example on the bottom lefthand part in Figure 2.

4.3. *Generating the supergraph from the skeleton navigation graph*

The skeleton navigation graph is now enhanced by adding a more detailed collection of nodes to represent the structure and semantics of each of the container polygonal faces. This results in what we term the *supergraph*, as described formally in the following definition.

Definition 4.1: Suppose given a graph G with node set N and a surjective function $f : M \to N$. Let H be a graph with node set M subject to the following condition: For all $x, y \in N$ with $x \neq y$, x, y are joined by an edge in G if and only if $\exists x' \in f^{-1}(x), y' \in f^{-1}(y)$ such that x', y' are joined by an edge in H. Then we say that H is a *supergraph*

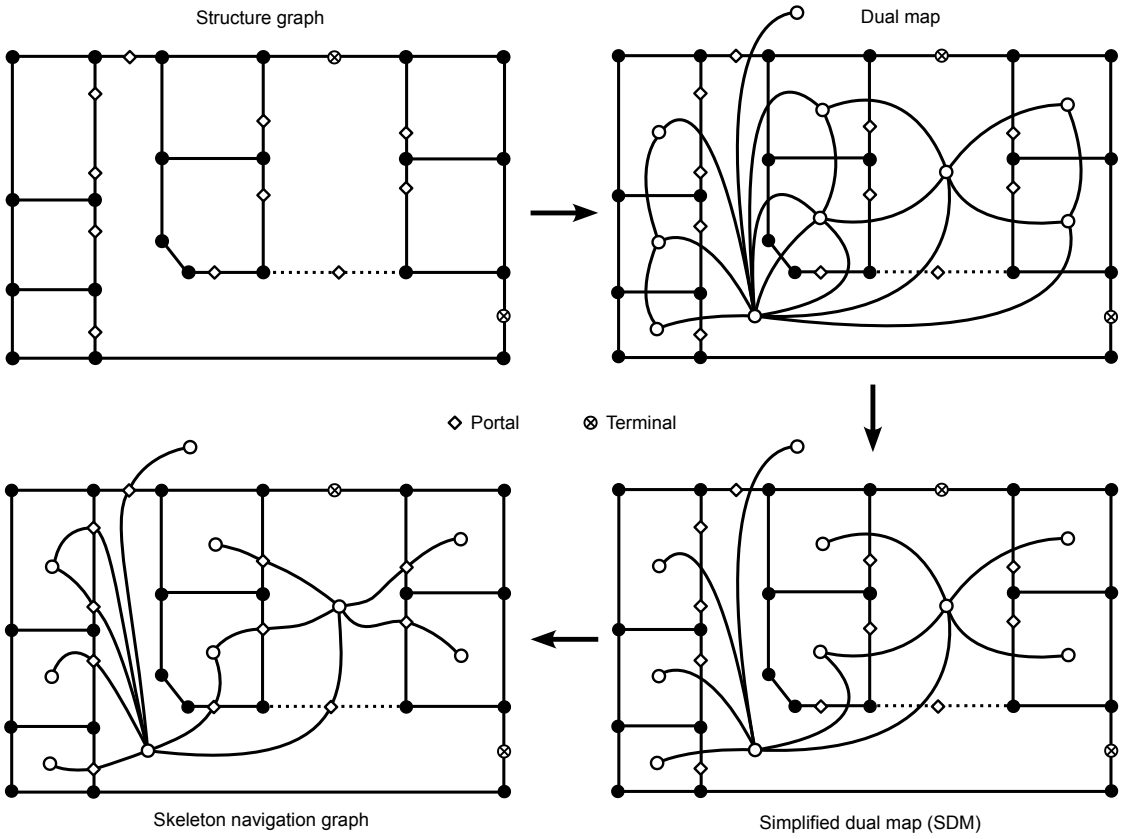

Figure 2. The stages in the construction of the skeleton navigation graph for the indoor space in Figure 1

of G. For each node $x \in N$ of graph G, we also say that the collection $f^{-1}(x)$ of nodes of graph H is a *supernode* of x.

Figure 3 provides an example of a supergraph derived from the skeleton navigation graph in Figure 2. The effect of the extension of a skeleton navigation graph to a supergraph is the addition of more nodes and edges while preserving the connectivity of the original. In this example, regions with a dotted boundary indicate supernodes. The corridor and larger room areas need to be expanded to take account of their extended geometries, while the smaller room and portal nodes remain as they are in the skeleton navigation graph.

4.4. *Node types and access constraints*

Our next step is to position the nodes and add edges. To do this, we need to move beyond the topology of the spatial scene and consider the geometry (e.g., shape and size of the containers) and semantics (nature of the spaces and the objects contained therein). Therefore, we now consider the classification of node types, their associated incident edges, and their access constraints. From now on, a node of a navigation graph is referred to as *navnode*. Each node type may be provided with one or more *access constraints*, which are constraints on passing through the node. For example, passage through a portal node may be limited only to those above a certain level of clearance or not possible for a person in a wheelchair, or access to an elevator may only be available when there is no

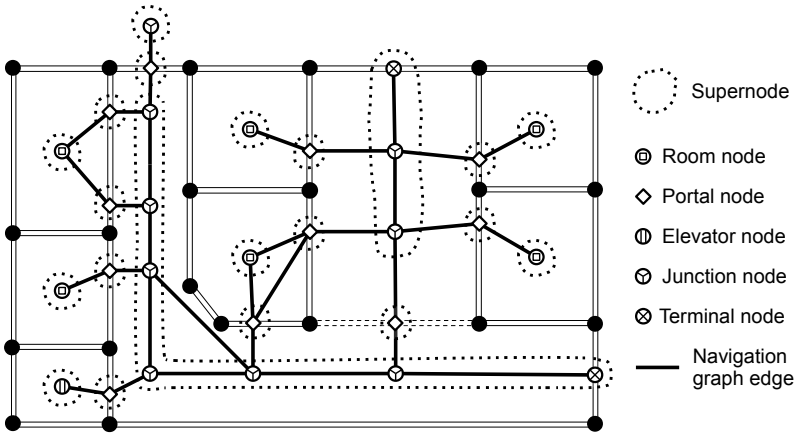

Figure 3. Supergraph generated from the skeleton navigation graph of Figure 2

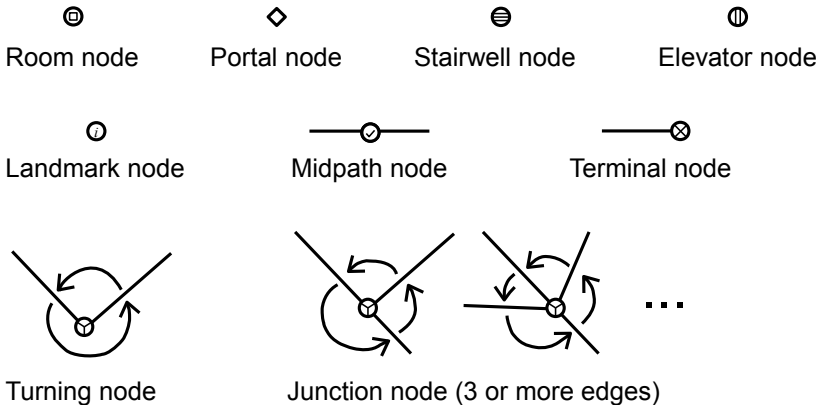

Figure 4. Types of navnodes and their associated edges in a navigation graph

emergency evacuation in progress. So, constraints may be context dependent and vary with time and type of user.

Other than constraints, navnodes fall into different categories. Figure 4 provides some of the principal navnode types, including all those used in Figure 3.

Navnode types are determined based upon topological, geometrical, and semantic considerations, and some principal categories are now considered.

(1) Some navnode types are determined by the *topological connectivity* of the structure. For example, *junction nodes* represent intersection points of several pathways and require a decision as to which route to take. In this case, further information about turning angles (usually given qualitatively) requires geometrical input, considered next.
(2) Any case where distance or angle information is required we label as *geometrical*. For example, *turning nodes*, occur when the path deviates significantly in angle. As mentioned above, geometrical information may be required in other cases, for example, associated with angles of turn at junction nodes, and labelling edges with metric information (traversal distance or time).
(3) Several types of navnode are derived from *semantic* considerations. In previous work, the authors have developed an extensive ontology of indoor spatial elements. Examples include *room node*, *internal portal node*, *external portal node* (which is an opening to the exterior space), *landmark node*, *stairwell node*, and

elevator node.

(4) Some other navnode types are *terminal node* (representing a physical dead end of a corridor, like dead ends on outdoor roads) and *midpath node* (which is used for reinforcement when there is a long corridor with no other types of nodes around).

4.5. *The navigation graph*

A navigation graph is an embedded graph that provides the underlying structure to support navigational guidance. Each navnode has a point location, usually given by two or three coordinates, the third being either its actually height above some datum or its floor level in the case that the building is clearly divided up into floors (the usual case). Each navedge is assumed to be embedded as a straight line between navnodes, and provides an approximation to a possible path between navnodes.

There are two main steps to generate a navigation graph from its skeleton graph: (1) add navnodes to each supernode, and (2) connect navnodes with appropriate navedges. In our work, most of this positioning can be done automatically, but there is some work to be done to make this happen. The next two sections deal with the navnodes and navedges, respectively.

4.6. *Constructing the navigation graph: The navnodes*

We use different techniques to add and position navnodes, depending upon their type and where they are in the structure. Some navnodes (e.g., turning node, junction node) are constructed using topologic and geometric information, while other navnodes (e.g., portal nodes added at portal locations) require semantic knowledge. An example of the positioning of such nodes in an indoor space is provided by Figure 3, and the details are now discussed.

4.6.1. Portals

Portal nodes have already been added at the skeleton navigation graph stage. It remains to position them according to the position of portals in the building plans, and to make the semantic distinction between external and internal portals, that provide links between interior and exterior, and interior and interior, respectively.

4.6.2. Simple rooms

In the case of relatively small, convex rooms, the existing room nodes from the skeleton navigation graph are positioned in the centroid of the room. More complex room spaces are discussed below.

4.6.3. Elevator wells and simple stairwells

Elevator wells (as with simple stairwells) are usually small convex spaces at any single level, and their nodes are positioned as with room nodes.

4.6.4. Landmarks

Indoor landmarks and the similarities and differences between indoor and outdoor landmarks are discussed in (Yang and Worboys 2011b,a). Once identified, their landmark nodes are positioned where the landmark is situated.

4.6.5. General considerations about corridors

As discussed above, we have identified junction nodes, turning nodes, terminal nodes, and mid-path nodes as of critical importance for navigation in indoor spaces. In general, the positioning of these types of navnodes are along the center-lines of linear spaces, such as corridors. Each junction node is positioned at the intersection of this line and a line perpendicular to a path to a destination, such as portal node. Terminal nodes are introduced at places along the center-line that result in dead ends for the navigator. Turning nodes are introduced at the turns of linear spaces, such as corridors, where no decision has to be made. Midpath nodes are introduced where the distance between any of the other lines of node is greater than a fixed length, and have the purpose to reassure the navigator that they are on the right path.

4.6.6. Simple rectangular corridors

We use Figure 5 as an example to illustrate the approach. There are some differences in the navnodes that are added on the longer and shorter edges of the rectangle, corresponding to corridor sides and ends, respectively. The longer and shorter edges can be automatically identified by computing distances between corner vertices (see circles in Figure 5). The corridor center-line is also computed. The next step is to add and position junction nodes associated with portals located on each of the two longer edges of the rectangular corridor. For each portal on the edges corresponding to corridor sides, a junction node is positioned on the center-line and opposite the portal. Examples of such junction nodes are shown in Figure 5. For each portal on the edges corresponding to corridor ends, a junction node is positioned opposite the portal and at a distance of half the corridor width. The figure also shows the positioning of a terminal node on a corridor end, and this is discussed in earlier subsections. The final step is to merge nodes that are closer than a specified tolerance. (Appropriate tolerances are discussed later in the paper). Examples of such mergers are shown in Figure 5, where d_1 and d_2 are less than the specified tolerance level.

4.6.7. Complex corridors and rooms

To add and position turning and junction nodes associated with the turnings and junctions of more complex, and in particular concave, polygons, we need some further constructions. We firstly provide a definition of a visibility polygon.

Definition 4.2: Let P be a polygon, and p, q be points of P. We say that p is *visible* from q if it is possible to draw a straight line segment from p to q entirely in P. The *visibility polygon* of a polygon P from a viewpoint p, written P_p, is defined as the set of all points in the interior or on the boundary of P that are visible from p.

Lee (1983) proposed an algorithm to calculate the visibility polygon of a point in a simple polygon.

Definition 4.3: The *kernel* of polygon P is the set of all points of P from which the entire interior of P is visible.

We note that the kernel of P may be empty. An important property of a non-empty kernel is that it is always a convex region. It is also clear that P is convex if and only if the kernel of P is itself P. An efficient algorithm for computing the kernel may be found in (Lee and Preparata 1979).

There are two categories of concave rooms to consider: (1) concave rooms that have non-empty kernels, and (2) concave rooms that have empty kernels. In case (1), because

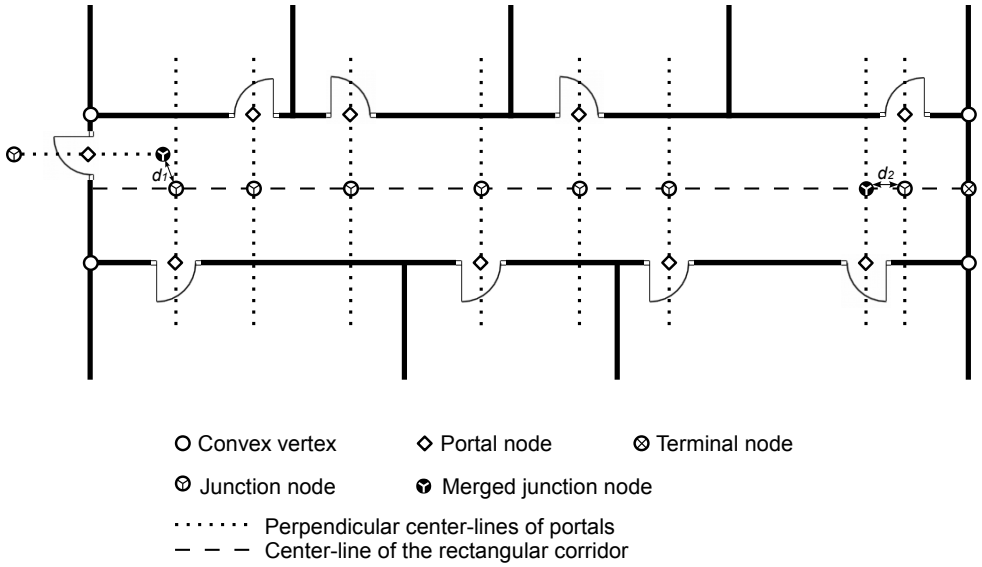

Figure 5. An example of addition and positioning navnodes in a rectangular corridor and in external face

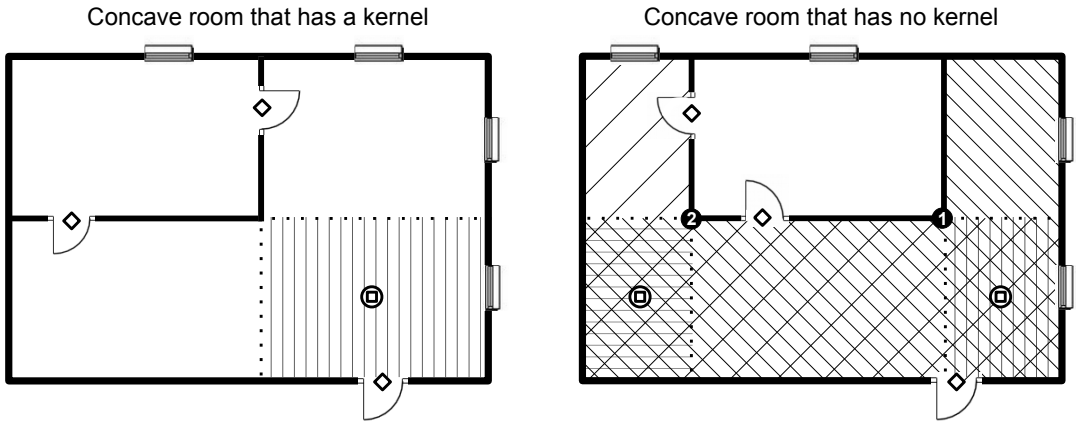

Figure 6. Two concave spaces

the kernel is convex, we can position the navnode at the centroid of the kernel. See left part of Figure 6 for an example of this case when the polygon is a room. The vertically hatched area is the kernel and the room node is positioned at the centroid of that hatched area.

For case (2), two or more room nodes may be necessary. To tackle this case we propose a new concept that allows kernels to be localised.

Definition 4.4: Let P be a concave polygon and p be one of the concave vertices of P. Let S be the set of all the concave vertices of the visibility polygon P_p except for point p itself. Partition P_p into a collection of convex polygons and exactly one non-empty

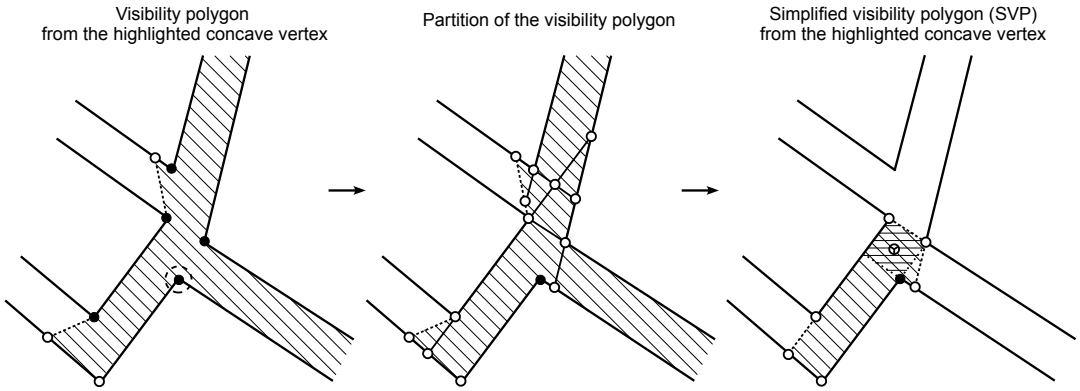

Figure 7. A complex corridor intersection

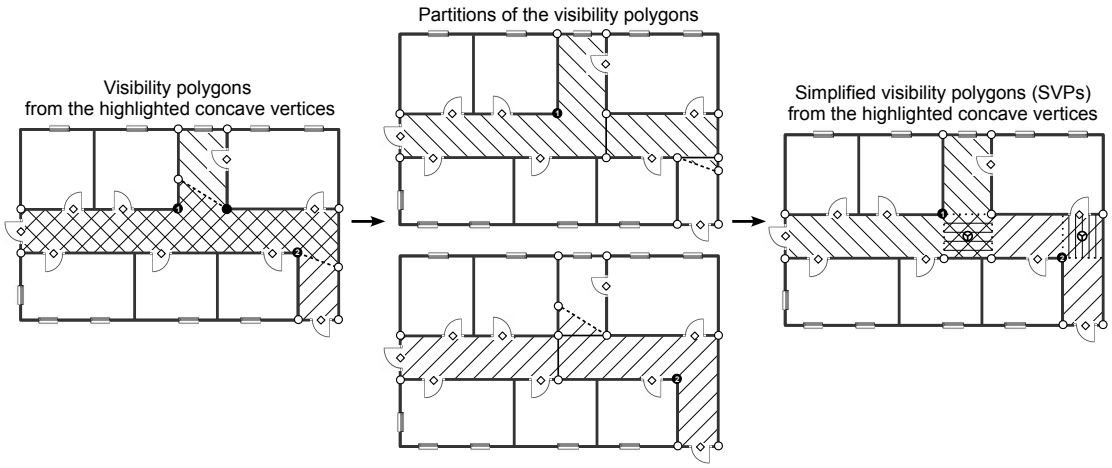

Figure 8. Addition and positioning of turning and junction nodes in a corridor

concave polygon, L, by extending all the edges that are incident with concave vertices $\in S$ and not collinear with point p. L is called the *simplified visibility polygon (SVP)* of P corresponding to point p. A *local kernel* of a concave polygon P with respect to point p is the kernel of L.

We may note that local kernels, because they are kernels, are always convex. We can therefore position a navnode at the centroid of each local kernel, corresponding to each concave vertex of the polygon. The next step is to merge navnodes positioned at centroids of local kernels of a corridor that are closer than the predetermined tolerance.

Figure 7 shows the construction of the local kernel of polygon P with respect to point p for a complex corridor intersection. The righthand side of Figure 6 shows the construction of all local kernels and positioning of multiple room nodes in the case of a concave room. Figure 8 shows the construction in the case of a complex corridor.

The final step is to add the remaining navnodes in the complex corridor. The junction

nodes associated with each portal along the complex corridor are positioned by firstly computing the SVP of each portal node (i.e., the viewpoint). Note that this SVP is slightly different from the one introduced above (Definition 4.4). Each portal node is a convex vertex, as no doors would be installed right on a corridor corner point, so its SVP would always be convex. Therefore the techniques introduced above for the addition of navnodes in a simple convex corridor apply to such a case. Finally, all the navnodes positioned in the corridor that are within the predetermined tolerance should be merged, including the previously merged navnodes positioned at centroids of local kernels. Proper merge tolerances will be discussed later (Section 5).

4.7. *Constructing the navigation graph: The navedges*

A navedge connecting two navnodes indicates a potential path from one to the other. In constructing the navedges a balance needs to be drawn between sparsity, and hence incompleteness, and unnecessary complexity. This section focuses on how to join navnodes with appropriate navedges. Once this is done, any direction constraints (e.g., one-way) on navedges can be derived from the access constraints of their associated navnodes. The construction of the navedges is divided into several cases, as below.

4.7.1. Type 1: Navedges connecting navnodes associated with portals

Such navedges are added by connecting each such portal node to its associated navnode in a corridor or in the external face. The topological connections identifying which portal node needs to connect to which corridor and which portal node needs to link to the external face, are already constructed in the skeleton navigation graph. Therefore, we can simply add geometry by making the edges straight line segments. For example, each topological edge in the skeleton navigation graph (the bottom left part in Figure 2) that links a portal to a corridor or to the external face results in one corresponding straight navedge in Figure 3. This navedge connects a portal node to its associated junction node in the corridor or the external face.

4.7.2. Type 2: Navedges connecting a room node and its single portal node

If a room supernode contains only a single room node, and if there is only one portal associated with the room, then there is only one navedge in the room, and it is added by connecting the room node and the portal node. See the navedge connecting the room node and the portal node at bottom left in Figure 3.

4.7.3. Type 3: Navedges in more complex spaces

Non-aligned case: This construction for this case works where no three or more navnodes lie in a straight line. Navedges are added by appropriately connecting any two room nodes contained in and or portal nodes linked to a room. The topological connections between each room and its associated portals are already in the skeleton navigation graph, so these are given a straight line segment geometry.

When there are two or more room nodes in a room or there are several portal nodes associated with the room, then we have to be careful to ensure that the straight line segments that are the navedges do not go through a structural boundary. To avoid this problem, we use Voronoi diagrams and their dual Delaunay triangulations (DT). The Voronoi diagram of a set of sites (Aurenhammer 1991, Fortune 1992) partitions space into a number of regions (also called Voronoi cells) such that each site corresponds to and is contained in a region that consists of all points closer to that site than to any other.

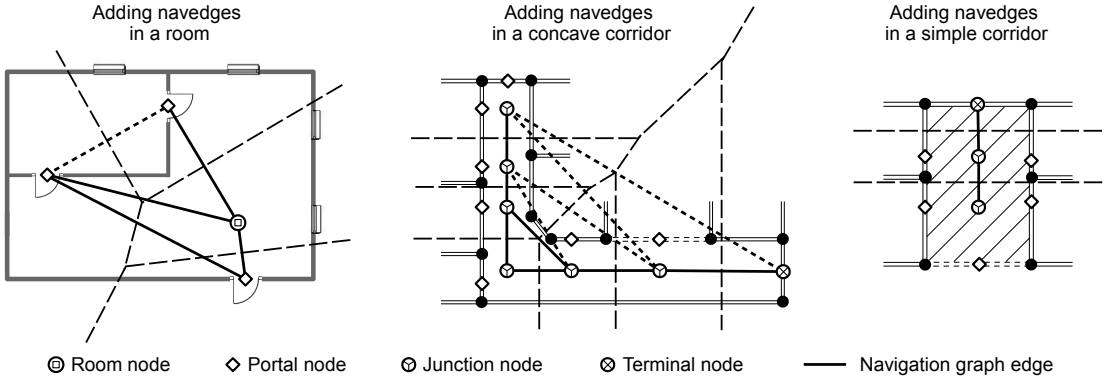

Figure 9. Examples of navedge addition

Its dual is the DT. To add navedges in a room, its associated room and portal nodes are used as sites to calculate Voronoi cells and the dual DT. The edges of the DT provide our candidate collection of navedges. However, not all edges in the DT are appropriate for navigation, so the following types of edges are removed: (a) edges that are not in the room; (b) edges that intersect the boundary of the room except for those that intersect only at portal nodes. Such edges are not retained because of visibility constraints due to room boundaries. The edges retained in the DT are the room navedges we need. The leftmost part in Figure 9 provides an example of navedge addition in a room, where the Voronoi cells are indicated by larger dashed lines and the corresponding DT by solid lines plus the smaller dashed line. The sites are the room node and three portal nodes that are accessible to the concave room. The smaller dashed edge is the improper edge eliminated, because it is not in the concave room.

The middle part of Figure 9 provides a corridor example. The junction nodes and the terminal node in the L-shaped corridor are the sites for the computation of Voronoi diagram (larger dashed lines) and DT (solid lines plus smaller dashed lines). The smaller dashed lines are eliminated because they intersect the corridor boundary and the rest of edges in the DT (i.e., solid edges) are the navedges we need.

Aligned case: The case where several collinear navnodes are disposed upon a straight line is particularly common in corridors, where navnodes are placed on center lines, and we use a corridor as our example. Unfortunately, in this case the triangulation becomes degenerate. We need to employ another technique named *constrained Voronoi diagram (CVD)*. Each CVD consists of a set of Voronoi cells resulting from the Voronoi diagram of sites clipped by a constraint polygon - in our case, a corridor. The sites used to construct the Voronoi diagram are those navnodes contained in a corridor. Each constrained Voronoi cell contains such a navnode, and thus the navedges can be added by connecting any two navnodes that are contained in adjacent Voronoi cells. However, as before we exclude edges that intersect the corridor boundary should not be added because of visibility constraints. The rightmost part in Figure 9 provides an example. Larger dashed lines represent Voronoi polygons, the sites of which are the two junction nodes in the corridor and the terminal node. The hatched corridor polygon serves as the constraint polygon. Three constrained Voronoi cells are constructed, where two of them each contains one junction node and on the boundary of the other lies a terminal node. Navedges added in the example are the two solid lines connecting two navnodes

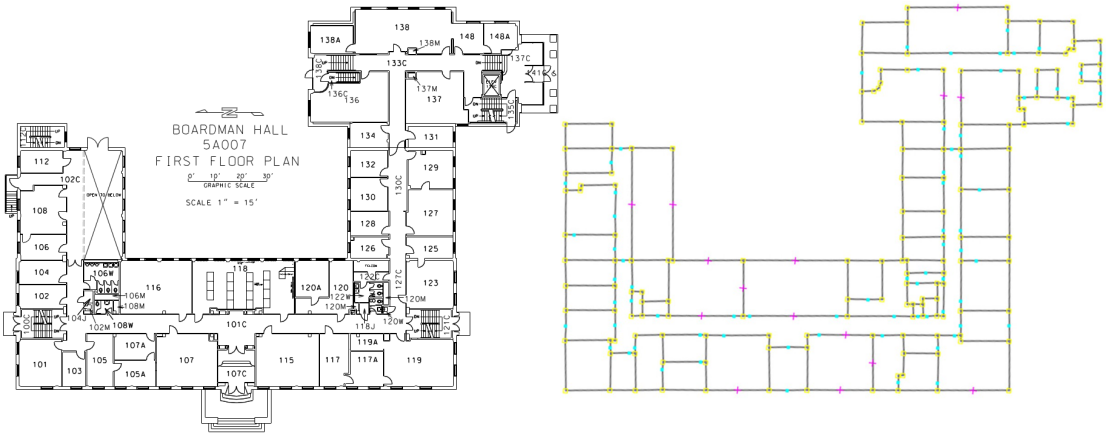

Figure 10. Floor plan and structure graph of the floor

contained or on the boundary of two adjacent constrained Voronoi cells.

5. Case study and evaluation

To illustrate and evaluate the approach introduced in Section 4, a case study was conducted with real world data: the first floor of Boardman Hall, University of Maine. The evaluation was primarily conducted by testing whether our methods could produce a navigation graph in a real-world case, and secondarily whether the resultant graph is correlated with how people might memorize and perceive the indoor space. Figure 10 shows the AutoCAD floor plan (left) and corresponding structure graph (right), where the blue points represent portals and corridor dead ends. This indoor space was embedded in OpenStreetMap (OSM) and references the IndoorOSM mapping scheme introduced by Goetz (see Note 2). Each floor level was modeled as a separate OSM relation, with separate room, corridor, stairwell, and elevator relations as its members. Code for our algorithms was developed in Java, using some external libraries (see Notes 3 and 4). However, most of the code (including code for kernel and local kernel) was generated from scratch.

As noted earlier, the setting of tolerances influences the navigation graph, because the larger is the tolerance the more nodes will be snapped together. For nodes positioned at centroids of local kernels in corridors, we found by experiment that a tolerance of half the average corridor width (in this case about 1m) was most appropriate. For navnodes in general positions in corridors, the tolerance was found to be slightly less at around 0.7m. The minimum width of a standard internal door is about 0.8m, so when two doors are next to each other it is proper to merge the two navnodes associated with them. Because we have a set of several different tolerances depending on the situation of navnodes, a merge priority was required. Nodes are assigned a priority, and nodes with the highest priority within a neighbouring group will not be snapped to nearby nodes. They remain in position and other nodes within the given tolerance and with lower merge priority will be snapped to them. In our case study, experiment showed that navnodes associated with corridor junctions (i.e., those navnodes put at the centroids of the kernel or local kernels of a corridor) should be given the highest priority. Figure 11 shows the

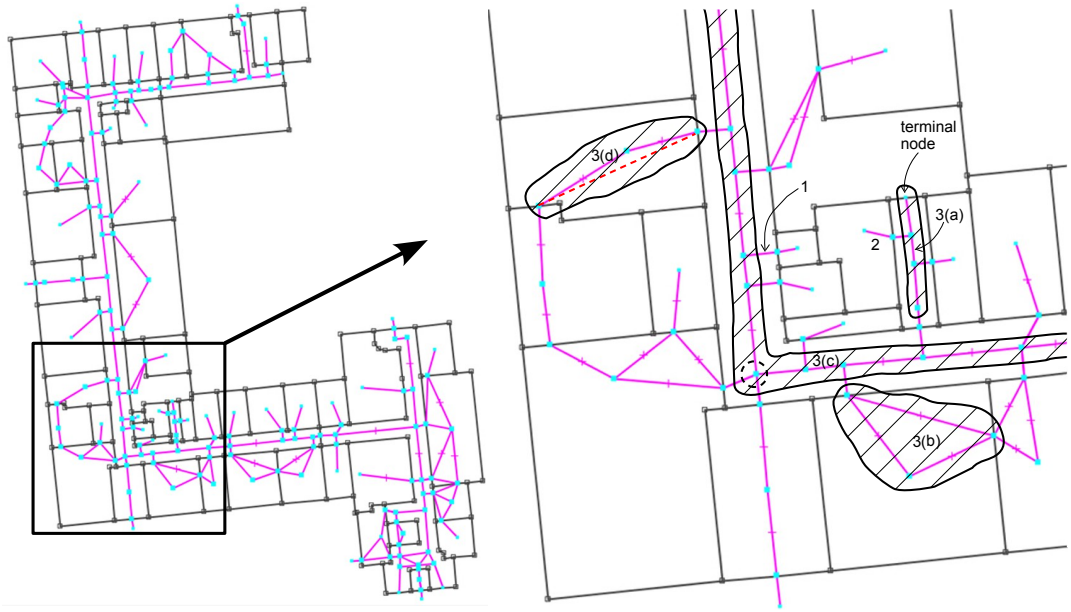

Figure 11. Computed navigation graph (The whole graph and a detail)

resulting navigation graph, where blue points represent navnodes, pink lines navedges, and the black graph is the structure graph. Navnodes have been positioned and merged as discussed above. The circled navnode in the figure detail shows an example of the positioning at the centroid of one of the local kernels of the corridor.

The next consideration is the navedges. We used the methods introduced in Section 4.7 to construct the navedges. In Figure 11, the edges labeled 1 and 2 provide examples of the first and second types. The navedges in the hatched areas labeled 3 in Figure 11 provide examples of the third type of navedge. (Navnodes in each hatched area indicated the sites used to compute the Voronoi diagram for that room or corridor). For the hatched simple corridor labeled 3(a), because all navnodes within it are aligned, the navedges were computed using CVD. For the navedges generated in the hatched room labeled 3(b) and in the hatched concave corridor labeled 3(c), it made no difference whether we used the Voronoi diagram and its dual DT, or the CVD method. However, for navedges in the hatched concave room labeled 3(d), using Voronoi diagram and its dual DT generated one more unnecessary navedge (the red dashed edge), and so in this case CVD was also more appropriate.

A successful navigation graph is one in which the types and positions of navnodes reflect the key functions and locations in the human navigational process. For example, we would expect a junction node to be located roughly where a human might require to make a decision during a navigation task. Sketch maps can provide the “window” into the human model of the indoor space. Therefore, to provide some evidence for or against the closeness of the correspondence between a computer-generated navigation graph and a human mental map, we performed a simple human subject experiment. The task was to draw a sketch map of the case study floor to help people navigate around it. Tversky and Lee (1999) found that limited schematic map and direction toolkits are sufficient to construct map directions. Based on this and to make the sketch maps drawn by different

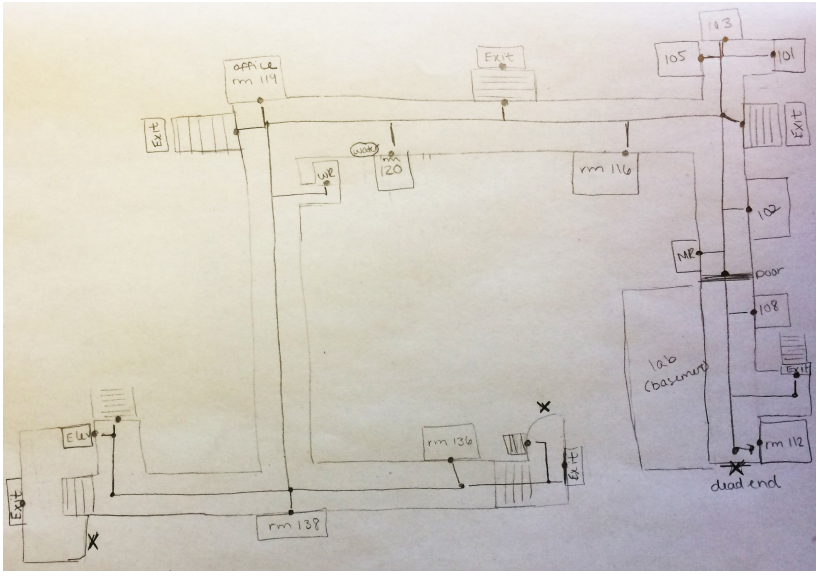

Figure 12. Sketch map drawn by one of the participants

participants comparable, we designed a sketch toolkit. Participants were constrained so that they could only use the symbols contained in the toolkit to draw their sketch maps. Ten human subjects (4 female and 6 male, ages in the range of 23 to 35) who are familiar with the floor at the University of Maine participated in the experiment. We also interviewed each participant after they finished their sketch map. They were asked how they drew or implied the points at corridor intersections. Most of them said that the points were produced by drawing centerlines of each corridor and the intersection points of those lines would be the junction points.

An example of a sketch map drawn by one participant is provided in Figure 12. Table 1 provides a summary of the experimental results. We compared the results of the sketch maps and our navigation graph, and found that all the subjects put a point at each intersection in the corridors. By comparing Figures 11 and 12, we can see that our approach generated all the junction nodes corresponding to each corridor junction in a very similar manner to that sketched by participants. Although this experiment is scientifically inconclusive, it provides some evidence for the alignment of our process with human wayfinding approaches.

6. Conclusions and future work

This paper proposed a comprehensive approach to computing a navigation graph for a single floor of a building, with the employment of new formal topological models enhanced with geometry and semantics. One of the novel aspects in this work was the use of combinatorial maps and their duals to provide a compact formal description of the topology and connectivity of the indoor structure represented by a connected, embedded graph. While making use of existing libraries for the more routine computational geometry involved, we developed several new algorithms, including one for computing the local kernel of a region.

Table 1. Experimental results

Validation items	**Participant percentage**	**Corresponding concepts in our navigation graph**
Put a point at each door/doorway location	90%	Portal node
Drew/implied a point at each corridor intersection	100%	Corridor junction node
Drew/implied a corresponding decision point in a corridor for each door along the corridor	100%	Junction node associated with portal
Noticed the closed but unlocked door in the corridor near room 104	90%	Portal node
Noticed the dead end in the corridor near room 112.	80%	Terminal node

To demonstrate our approach, it was fully implemented and applied in a case study to a floor of a university building. We also conducted a simple human subject experiment to get some idea whether the resultant navigation graph corresponded to what was required for human wayfinding in an indoor space. The results showed that this navigation graph provides a collection of appropriately positioned junction nodes associated with corridors and portals, as well as appropriate connecting navedges.

The method proposed in this work produces a navigation graph for a single floor of a building. However, it is not too difficult to extend this approach to a multi-level indoor environment by linking the navigation graphs of individual floors with navedges representing connecting stairwells and elevators. In addition, as we stated earlier in this section, each combinatorial map and its dual can provide topology and connectivity for the indoor structure represented by a connected graph. In the case of disconnected graphs, the combinatorial maps generalize to a new formal construction, which is our ongoing focus.

For very large regions (e.g., airports), the graph-based method introduced in this work is not appropriate. This requires further work to understand to what extent graph-based approaches can be used, and, if so, where and how to put navnodes and navedges for navigation in such regions. Also, in this paper, we have not considered presentation to users of the navigation graph once it is generated. This is outside the scope of our expertise, and requires investigation of visual, audio, tactile interaction. Another area where more work is required is the development of context-based navigation graphs that can take account of special conditions. For example, portal nodes related to elevators (and associated navedges) should not be active in emergency situations. Examples of other cases where context is required are access to areas requiring some level of security clearance, navedges that are not traversable by people in wheelchairs, and temporal constraints where certain areas are closed at certain times. Another area not considered in this paper is the imposition of directional and turn constraints, such as one-way routes and turn restrictions (more applicable to road networks than the indoor case). In fact, navigation graphs can be enhanced by the addition of a wide variety of semantics taking account of characteristics of users' tasks (e.g., exploration of the building or heading for a specific place) and of features of the indoor space (e.g., landmarks).

A major piece of research still to be done and only touched on in this paper is the integration of such navigation graphs with outdoor space networks, such as pedestrian

paths, transportation routes, and roads. In the combinatorial map model, the external face will provide the formal linkage to external spaces.

Acknowledgements

This material is partly based upon work supported by the US National Science Foundation under Grant number IIS-0916219. The authors are grateful to Jia Wang, Reinhard Moratz, Nicholas Giudice, and Lisa Walton for discussions relating to this work. We also thank all the experiment participants.

Notes

1. IndoorGML (http://indoorgml.net)
2. IndoorOSM (http://wiki.openstreetmap.org/wiki/IndoorOSM)
3. Visibility in polygons (http://www.geometrylab.de/applet-5-en)
4. JTS Topology Suite (http://www.vividsolutions.com/jts/jtshome.htm)

Figure captions

Figure 1. An indoor spatial scene and its combinatorial map
Figure 2. The stages in the construction of the skeleton navigation graph for the indoor space in Figure 1
Figure 3. Supergraph generated from the skeleton navigation graph of Figure 2
Figure 4. Types of navnodes and their associated edges in a navigation graph
Figure 5. An example of addition and positioning navnodes in a rectangular corridor and in external face
Figure 6. Two concave spaces
Figure 7. A complex corridor intersection
Figure 8. Addition and positioning of turning and junction nodes in a corridor
Figure 9. Examples of navedge addition
Figure 10. Floor plan and structure graph of the floor
Figure 11. Computed navigation graph (The whole graph and a detail)
Figure 12. Sketch map drawn by one of the participants

References

Afyouni, I., Ray, C., and Claramunt, C., 2014. Spatial models for context-aware indoor navigation systems: A survey. *Journal of Spatial Information Science*, (4), 85–123.

Aurenhammer, F., 1991. Voronoi diagramsa survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23 (3), 345–405.

Becker, T., Nagel, C., and Kolbe, T.H., 2009. A multilayered space-event model for navigation in indoor spaces. *3D Geo-Information Sciences*. Springer, 61–77.

Boguslawski, P., Gold, C.M., and Ledoux, H., 2011. Modelling and analysing 3D buildings with a primal/dual data structure. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66 (2), 188–197.

Borovikov, I., 2011. *Navigation Graph Generation* [online]. : . Available from: http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/navigation-graph-generation-r2805 [Accessed 2 June 2014].

Choset, H. and Burdick, J., 2000. Sensor-based exploration: The hierarchical generalized voronoi graph. *The International Journal of Robotics Research*, 19 (2), 96–125.

Choset, H.M., 2005. *Principles of robot motion: theory, algorithms, and implementation*. MIT press.

Demyen, D. and Buro, M., 2006. Efficient triangulation-based pathfinding. *In*: *AAAI*, Vol. 6, 942–947.

Edmonds, J., 1960. A combinatorial representation of polyhedral surfaces. *Notices of the American Mathematical Society*, 7, 646.

Fortune, S., 1992. Voronoi diagrams and Delaunay triangulations. *Computing in Euclidean geometry*, 1, 193–233.

Franz, G., *et al.*, 2005. Graph-based models of space in architecture and cognitive science-a comparative analysis. *In*: *Proceedings of the 17th International Conference on Systems Research, Informatics and Cybernetics*, 30–38.

Gibson, J.J., 1986. *The ecological approach to visual perception*. Psychology Press.

Haunert, J.H. and Sester, M., 2008. Area collapse and road centerlines based on straight skeletons. *GeoInformatica*, 12 (2), 169–191.

Jensen, C.S., Lu, H., and Yang, B., 2009. Graph model based indoor tracking. *In*: *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, 122–131.

Kneidl, A., Borrmann, A., and Hartmann, D., 2012. Generation and use of sparse navigation graphs for microscopic pedestrian simulation models. *Advanced Engineering Informatics*, 26 (4), 669–680.

Lee, D.T. and Preparata, F.P., 1979. An optimal algorithm for finding the kernel of a polygon. *Journal of the ACM (JACM)*, 26 (3), 415–421.

Lee, D., 1983. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22 (2), 207–221.

Lee, J., 2004. A spatial access-oriented implementation of a 3-D GIS topological data model for urban entities. *GeoInformatica*, 8 (3), 237–264.

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

Lee, J. and 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.

Mortari, F., *et al.*, 2014. "Improved geometric network model" (IGNM): A novel approach for deriving connectivity graphs for indoor navigation. *In*: *ISPRS Technical Commission IV Symposium, ISPRS Annals-volume II-4, Suzhou, China, 14-16 May 2014*.

Nagel, C., *et al.*, 2010. Requirements and space-event modeling for indoor navigation. *OpenGIS® Discussion paper (OGC 10-191r1), Open Geospatial Consortium, OGC*.

Stoffel, E.P., Lorenz, B., and Ohlbach, H.J., 2007. Towards a semantic spatial model for pedestrian indoor navigation. *Advances in Conceptual Modeling–Foundations and Applications*. Springer, 328–337.

Stoffel, E.P., Schoder, K., and Ohlbach, H.J., 2008. Applying hierarchical graphs to pedestrian indoor navigation. *In*: *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, p. 54.

Tutte, W.T., 1973. What is a map. *New Directions in the Theory of Graphs*, 309–325.

Tversky, B. and Lee, P.U., 1999. Pictorial and verbal tools for conveying routes. *Spatial information theory. Cognitive and computational foundations of geographic information science*. Springer, 51–64.

Wallgrün, J.O., 2005. Autonomous construction of hierarchical voronoi-based route graph representations. *Spatial Cognition IV. Reasoning, Action, Interaction*. Springer, 413–433.

Wallgrün, J.O., 2010. Qualitative spatial reasoning for topological map learning. *Spatial Cognition & Computation*, 10 (4), 207–246.

Worboys, M., 2011. Modeling indoor space. *In*: *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, 1–6.

Worboys, M., 2012. The maptree: A fine-grained formal representation of space. *Geographic Information Science*. Springer, 298–310.

Worboys, M., 2013. Using Maptrees to Characterize Topological Change. *Spatial Information Theory*. Springer, 74–90.

Worboys, M.F. and Duckham, M., 2004. *GIS: a computing perspective*. CRC press.

Yang, L. and Worboys, M., 2011a. A navigation ontology for outdoor-indoor space. *In*: *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, 31–34.

Yang, L. and Worboys, M., 2011b. Similarities and differences between outdoor and indoor space from the perspective of navigation. *Poster presented at COSIT*.