Towards a Semantic Spatial Model for Pedestrian Indoor Navigation

Edgar-Philipp Stoffel, Bernhard Lorenz, and Hans Jürgen Ohlbach

University of Munich, Oettingenstr. 67, 80538 Munich, Germany,
{stoffel,lorenz,ohlbach}@pms.ifi.lmu.de,
http://www.pms.ifi.lmu.de/

Abstract. This paper presents a graph-based spatial model which can serve as a reference for guiding *pedestrians* inside buildings. We describe a systematic approach to construct the model from geometric data. In excess of the well-known topological relations, the model accounts for two important aspects of pedestrian navigation: firstly, *visibility* within spatial areas and, secondly, generating *route descriptions*. An algorithm is proposed which partitions spatial regions according to visibility criteria. It can handle simple polygons as encountered in floor plans. The model is structured hierarchically - each of its elements corresponds to a certain domain concept ('room', 'door', 'floor' etc.) and can be annotated with meta information. This is useful for applications in which such information have to be evaluated.

1 Introduction and Related Work

Pedestrian guidance within buildings [6] differs from customary navigation based on networks. This is due to the following reasons:

1. Features like roads or railways can be commonly modelled by *one-dimensional* elements in a clearly defined network. In contrast, features of a building do not fit so nicely into this schema – a building exhibits a nested, *three-dimensional* structure. There are several ways to overlay a **floor plan** with a graph [5]. It is arguable which representation to choose: e.g. should a corridor be mapped to a node, a chain of connected nodes, or rather to an edge?
2. Pedestrians can roam *freely* between the interior boundaries of buildings. Their movement is less restricted than those of vehicles (which are often bound to their networks, e.g. trains). However, spatial orientation and visibility play a vital role for **human wayfinding** [16]: An *L-shaped* room, for example, cannot be perceived as a whole without moving around the corner. Although the primary concern of navigation is to determine a shortest (fastest) path, one should not underrate the importance of comprehensible **route descriptions** [13] – they tell us *how* to follow a path in a region.

Floor plans document the interior layout of a building in terms of boundaries. Particularly in computational geometry [3] and robot motion planning [2], so-called roadmap methods derive detailed navigational graphs from geometric information. At the other end of the spectrum, there are symbolic models which

can be classified [1] into topological models like Region Adjacency Graphs [10, 14] or Cell and Portal Graphs [11] and hierarchical models [8, 12, 15]. All these abstract away from geometric details; they merely represent qualitative spatial relations between regions [4]. However, as argued by Hu et al. [8], if the relations are too abstract or coarse, they are impractical – they cannot model reachability among regions via different entry and exit points. Finally, cognitive models of indoor spaces [16, 17] are interesting insofar as they cater for the representation of space from the perspective of humans. They are well-suited for generating route descriptions. What is missing is an elegant way to couple these different aspects into one coherent model, e.g. a high-level topological representation which can be refined to reveal the inner structure of a region if required.

The main contribution of this paper is the formalisation of a spatial model which can handle different levels of abstraction and provides a basis for the generation of route instructions. We describe a systematic approach to construct the model from floor plan data. It is worth noting that the notion of visibility is embedded into the model.

The paper is structured as follows: in Sect. 2.1 and 2.2, we formally define the basic elements of the model based on a floor plan's geometry. Sect. 2.3 explains the extesion to a hierarchy. The algorithm presented in Sect. 2.4 refines the hierarchy based on visibility criteria. Sect. 2.5 motivates the potential of annotating the model with meta information.

2 The Proposed Model

We assume that a set of floor plans of the environment are available in a vector-based format (these data originate e.g. from a CAD application). The geometric data structure consists of a (planar) **mesh of polygons**; each polygon encloses a spatial region R. In the following we use the terms polygon and spatial region interchangeably, although we are aware of the subtle difference [4] that a polygon is only a representation of the boundary ∂R of a spatial region.

2.1 Spatial Regions

Definition 1 (Spatial Region). *A **spatial region** $\boldsymbol{R} := C_1C_2..C_{z\geq 3}$ is geometrically represented as a list of corners $C_{i\in 1..z}$ (indices $i > z$ are calculated **modulo** z). Corners are ordered **counter-clockwise (ccw)**. Each corner takes up a position $C_i.pos = (x, y)$ in an underlying reference system. Two consecutive corners fix a **boundary line** $bLine(i) = C_iC_{i+1}$. Boundary lines do not intersect elsewhere: $bLine(i) \cap bLine(j \neq i \pm 1) = \emptyset$. **interior**$(R)$:= all points p with odd crossing number (number of times a ray starting from p intersects boundary lines of R).*

Two spatial regions R_p, R_q **touch** [4] if $\exists i, j$, $bl_p := bLine(i)$ in R_p, $bl_q := bLine(j)$ in R_q: $bl_p = bl_q$, i.e. they share at least one boundary line. A mesh of polygons is characterised more precisely as a **cell complex** (cf. Plümer et al. [14]) by the following axioms:

Condition 1 (pairwise disjoint) *Spatial regions may* ***not overlap****, but* ***touch****:* $\forall bl_p \in R_p,\ bl_q \in R_q$*:* $bl_p = bl_q \vee bl_p \cap bl_q = \emptyset \vee bl_p \cap bl_q = \{C \| C \in R_p \cap R_q\}$.

Condition 2 (jointly exhaustive) *All n spatial regions* ***make up*** *the environment to be modelled:* $\bigcup_{i \in 1..n} R_i = FloorPlan$.

Whereas convex polygons are not critical for route descriptions, we have to treat **non-convex polygons** specially (see Sect. 2.4). This is simply because humans can't see what is beyond a corner.

Definition 2 (Concave Corner, Chain). *A corner X enclosing an internal angle greater than* $180°(\pi)$ *is called* ***concave****. A non-convex polygon has at least one concave corner.* ***Concave chains*** *are maximum sequences* $X_i X_{i+1}..X_{i+k}$ *of consecutive concave corners.*

2.2 Boundary Nodes

For the purpose of navigation, we have to define connectivity among spatial regions. Even though two adjoining spatial regions are touching, they can still be physically separated by walls or other divisions (loges in a theatre, platforms on a station). Access and egress are only possible through specific points on their shared boundary (i.e. `doors` and other openings), called **boundary nodes**. In the literature these elements are also referred to as *gateways* [17] or *exits* [8]. A prototypical setting with two boundary nodes is depicted in Fig. 1.

Definition 3 (Boundary Node). *A* ***boundary node*** $B := (id, t, R_p, R_q, w, \Omega)$ *is a waypoint for a path between two adjoining spatial regions* R_p, R_q*:* $\exists a, b, bl$*:* $bl = bLine(a)$ *in* $R_p \wedge bl = bLine(b)$ *in* $R_q \wedge B$ *in* bl*. It has exactly one type t and a* ***unique*** *identifier id. Furthermore, w denotes its total width and* Ω *the angle/orientation perpendicular to its boundary line bl.*

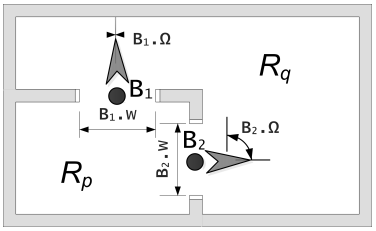

Fig. 1. Two Exemplary Boundary Nodes

Remarks:

1. Boundary nodes are key points for navigation. Their type t is a conceptual representation of the underlying architectural feature, e.g. of a door, a window or an opening.

2. Two spatial regions can have more than one boundary node in common (see Fig. 1), thus their connection has a certain multiplicity. The *unique* identifier *id* guarantees that each boundary node can be distinguished as a separate entity.

A boundary node can be regarded, from a **dual** perspective, as a transition relation (or connection) between regions. In Region Adjacency Graphs [12, 14] or Cell and Portal Graphs [11], they are represented as *multi*edges (remark 2) between the nodes R_p and R_q.

2.3 Hierarchical Graphs

Apart from coinciding boundaries, there is another fundamental relation between spatial regions: they can be nested. Premises are inherently organised into constituent floors, sections, rooms, and so forth. Therefore it makes sense to define a relation for containment. This relation can be used to extend the aforementioned flat graphs into **hierarchical graphs** [1, 12, 15]:

Definition 4 (Child Relation $\lhd$). *A spatial region R_c is **child** of another spatial region R_p, denoted as $R_c \lhd R_p$, if $\exists(C \in R_c) : C \in interior(R_p)$ and $\forall(R_a \neq R_p) : [C \in interior(R_a) \Rightarrow \exists(C_p \in R_p) : C_p \in interior(R_a)]$. R_p, R_a are called **parent** and **ancestor** region, respectively.*

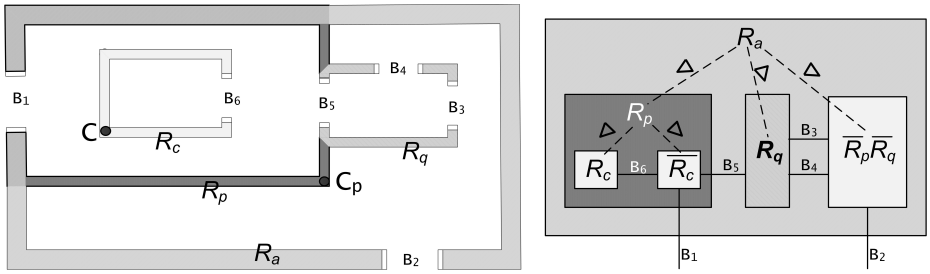

Fig. 2. An Example for a Hierarchy

Remarks:

1. According to axioms 1 and 2, the first condition of Def. 4 implies that the other corners $C_{other} \neq C$ of R_c lie either in the interior of R_p or on a boundary line shared by R_p and R_c. The second condition guarantees that R_p is indeed the minimal region containing R_c - there is no other region R_a between R_p and R_c that also contains R_c (illustrated in Fig. 2).
2. A further consequence is that only one region R_p can be parent of the region R_c. However, an important question arises: How can, for example, rooms R_c be represented which belong to a floor R_{p1} and a wing R_{p2} at the same

time? There is a way to fit them into the model: one can simply substitute R_{p1} and R_{p2} by the three regions $R_{p1} \setminus R_{p2}$, $R_{p1} \cap R_{p2}$, and $R_{p2} \setminus R_{p1}$. This method works in general for n overlapping parent regions. They are replaced by at most $2^n - 1$ non-overlapping regions.

Definition 5 (Region Graph). *$G_R := (N_R, E_R, t)$ is the **region graph** of a spatial region R. The type t of a region graph represents concepts like floors, sections, rooms, etc. All region graphs G_Q with $Q \triangleleft R$ are nodes in N_R. Comprised in the edge set E_R are the boundary nodes of R as well as those of all child regions Q. **Local edges** of G_R are boundary nodes between Q_1, $Q_2 \in N_R$. Otherwise, if one $Q_{i \in \{1,2\}} \notin N_R$ they are **interface edges** of G_R.*

One can easily see that the relation $\triangleleft$ is anti-symmetric and irreflexive. It thus defines a partial ordering on spatial regions. This ordering represents the multiple levels of the hierarchy, from coarse- to fine-granular spatial regions:

Definition 6 (Level, Root, Leaf).
***level**(G_{R_r}) := 0 if $\nexists R_p : R_r \triangleleft R_p$. Such R_r are called **roots**.*
***level**(G_{R_c}) := 1 + level(G_{R_p}) if $R_c \triangleleft R_p$, i.e. $G_{R_c} \in N_{R_p}$.*
*All R_l with $\nexists R_c : R_c \triangleleft R_l$ are **leaves**. They lie at the bottom of the hierarchy.*

Pragmatic considerations speak in favour of using hierarchical graphs as underlying navigation model: Experiments conducted e.g. by Jing et al. [9] and Shekhar et al. [18] have shown a gain in the processing time of shortest path queries. Besides, humans can rather make sense of hierarchical structures than of coordinates returned by a positioning system since the former are qualitative. The hierarchy is always coined by the floor plans. However, it requires some reorganisation with respect to *reachability* between child regions:

- $G_{Q_i} \in N_R$ may *not* be *mutually* reachable by a sequence of boundary nodes $B_N \in E_R$. For instance, two rooms Q_1, Q_2 in fully separated sections of a floor R can only be reached via another floor. It makes sense to split G_R into its **connected components**, each becoming a new region graph.
- The removal of one **articulation edge** from E_R (say B_5 in Fig. 2 is locked), leads to having two connected components. Knowing this, one can split G_R in advance at all articulation edges.

2.4 Partitioning Algorithm and Navigation Process

Hierarchical graphs reflect a building's topology. Nevertheless, their resolution is too coarse for navigation: The interior of *non-convex* leaf regions may be complex so that several route instructions are necessary (e.g. for a door around two corners). The path between two boundary nodes (e.g. B_1 and B_2 in Fig. 1) in a leaf region (R_q) is not always the line of sight, unless the region is convex.

Visibility depends on the shape of a leaf region. In the following, we present an algorithm which partitions leaf regions according to visibility criteria: The principal idea is to connect corners in a non-convex leaf region in such a way

that they partition the region into non-overlapping **convex sub-regions** (see Fig. 3). The partitioning is not arbitrary (we could use any triangulation then), but concave corners play a major role. The actual process of partitioning is described in the main algorithm:

```
1  List<Polygon> convexPartitioning(Polygon p)
   {   List<Polygon> subPolys = new List<Polygon>(); /* store sub-polygons */
3      for (ConcaveCorner r in p.concaveCorners())         /* ccw list */
       {   ConcaveCorner rNext = p.nextConcaveCorner(r);   /* ccw from r */
5          if (rNext == null)  /* (1) the only concave corner of p is r */
               subPolys.add(matchOneConcaveCorner(p,r));
7          else                /* (2) more than one concave corner in p */
           {   if (p.index(r) + 1 == p.index(rNext)) continue;
9                  /* (2.1) r, rNext in concave chain: skip one iteration */
               Polygon sub =
11                 p.createSubpolygon(range(p.index(r), p.index(rNext)));
                  /* puts all corners from r to rNext in sub, rNext points to r */
13             if (not(sub.isConvex())) //(2.2) match concave in sub with corner
               {   List<Polygon> subSplits = convexPartitioning(sub);
15                 if (not(p.nextConcaveCorner(rNext) == r))
                     subSplits.remove(polygons with bLine rNext to r);
17                 subSplits.remove(polygons with concave corner of p inside);
                   subPolys.add(subSplits); }
19             else        /* (2.3) sub-polygon is convex */
               {   if (no concave corner of p inside sub)
21                 {   p = p.splitSubpolygon(sub);
                           /* bypass all corners between r, rNext */
23                     subPolys.add(sub);  } } } }
       p.updateConcaveCorners();   /* some can fall away now */
25     if (not(p.isConvex()))      /* still non-convex */
           subPolys.add(convexPartitioning(p));    /* recursive call on p */
27     return subPolys;    }
```

In order to cut off a sub-polygon *sub* (line 10,11), the algorithm tries to connect each concave corner r with its next concave corner $rNext$ in ccw (line 4). If this succeeds, a new boundary line is created between r and $rNext$. In case r is the only concave corner (line 5,6), it has to be matched to (ideally) one or two convex corners (see next listing). If r and $rNext$ are already connected (line 8), the algorithm skips r and proceeds to $rNext$.

The polygon *sub* is only cut off if it is convex and no other concave corners of p are enclosed in its interior (line 19-23). As a counterexample, the cut r_2r_3 on the left of Fig. 3 would enclose r_1. However, assuming a cut is allowed, all corners between r and $rNext$ are removed from p (line 21).

If *sub* is non-convex (line 13-18), it implies that r and/or $rNext$ are also concave in *sub*. The algorithm is called recursively on *sub*. The result of this call is *subSplits*, a convex partitioning of *sub*. All polygons of *subSplits* with a concave corner of p inside have to be removed (line 17), also the polygon with the *bLine* from r to $rNext$ if this cut wouldn't be done by $rNext$ (line 15,16). The rest of the polygons in *subSplits* are then cut off from p (line 18). After one complete tour around the polygon p, it might have become smaller due to some successful cuts. The concave corners of p are updated (line 24); if there are still concave corners in p, the algorithm is called recursively (line 25,26).

The partitioning of a polygon with only *one* concave corner (line 6) is illustrated on the right hand side of Fig. 3 by the description of cut 2, as well as in the following listing:

```
1  List<Polygon> matchOneConcaveCorner(Polygon p, ConcaveCorner r)
```

```
{   int ix = p.index(r); List<Polygon> subPolys = new List<Polygon>();
3   List<Corner> Cmatches = all corners c of p with
        c leftOf bLine(ix-1) and c leftOf bLine(ix);        /* LL: ideal */
5   if (Cmatches == null)   /* can't reduce with one ideal cut (LL) */
    {   Cmatches = corners c,d of p such that d = p.nextCorner(c) and
7        c rightOf bLine(ix-1) and c leftOf bLine(ix)   /* RL */
         and d leftOf bLine(ix-1) and d rightOf bLine(ix);  }   /* LR */
9   for (Corner c in Cmatches)
    {   Polygon sub = p.createSubpolygon(range(p.index(r),p.index(c)));
11      sub.addBoundaryNodeBetween(c, r);
        if (no concave corner of p inside sub)
13      {   p = p.splitSubpolygon(sub);
            subPolys.add(sub); } }
15  return subPolys; }
```

A corner in the area 'LL' would be ideal to connect to (line 4), since the concave corner would become convex. But if there is none, one can pick the last corner in the area 'RL' (line 7) and the first one in the area 'LR' (line 8).

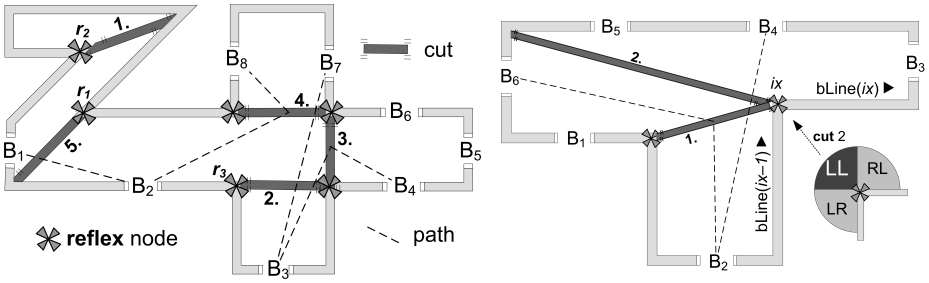

Fig. 3. Applying the Partitioning Algorithm

On the basis of the partitioning, one can define a navigational graph for representing also **paths between boundary nodes**. This is exemplified, too, in Fig. 3: paths are indicated by the small dashed lines. Each sub-region is convex, so all boundary nodes in the *same* sub-region are *per se* mutually visible. They can be directly connected by a path whose distance is simply the Euclidean distance. For route descriptions, one can make use of the orientations $B.\Omega$ encoded in boundary nodes and divide the space into front, left and right. This is working also in cases where boundary nodes lie on the same boundary line (e.g. B_5 and B_4 on the right hand side of Fig. 3): With the angles enclosed between the path B_5B_4 and $B_5.\Omega$ (resp. $B_4.\Omega$) one can find out that a left turn is required starting at B_5. A route description such as "Turn left [at B_5] and move along the wall until you reach the first door [B_4] on your left" can be obtained.

Boundary nodes in *different* sub-regions can be directly connected by a path if they intersect *all* cutting lines between their sub-regions (on the left hand side of Fig. 3: B_1B_2 with cut 5, B_3B_7 with cuts 2 and 4). Otherwise, they either lie outside the considered leaf region L (left hand side: B_7B_6), or intersect some boundary line of another sub-region of L (left hand side: B_2B_8, B_4B_3). This means, in any case, that they are not mutually visible in L. However, they can be connected by a chain of paths which additionally run through points, e.g.

the centres, of the cutting lines between the involved sub-regions. The centre of cut 4 (left hand) is added as an intermediary point along the path from B_2 to B_8. When connecting B_3 with B_4, the center of cut 2 is superfluous because the path between B_3 and the centre of cut 3 intersects cut 2 – this means that using only cut 3 is sufficient.

2.5 Evaluation of Constraints

Although the focus of this paper is on the spatial model, it is worth noting that the model can be annotated with meta information. Especially the types t of boundary nodes and region graphs could contain further attributes, e.g. in form of a list of key-value-pairs. This could be very useful in practise, for applications which require a more detailed processing of *context* information. The notion of distance could be understood in a variety of different ways, depending on the **semantics** [19, 20] of the application and its context (encoded in these attributes). Consider the following examples:

- Doors (boundary nodes) can be locked or, more general, access requires *authorisation* (key, card, biometric scan etc.).
- Admission of entry can be limited in *time*, e.g. opening hours of an office.
- Certain sections of a public building (all interface edges into a region graph) may be *restricted* in access ("staff only", high-security wings, laboratories).
- Special exits and base level windows can be used for *emergencies*.

The examples from above can be modelled as **boolean** (hard) **constraints** of the form $\bigwedge^* (attr = value \vee attr \in valuePartition)$ on boundary nodes and/or region graphs of a certain type. After evaluation, such a constraint yields a truth value. Boolean constraints can, hence, be used to determine under which conditions motion is physically possible ('can') or admitted ('may') in the environment [7]. The environment can be filtered only for the relevant parts which fulfil these binary constraints *before* the actual navigation process. A rich indoor model should take these kinds of constraints into account, but not exclusively. *Person-related* properties like roles, privileges, or preferences also have a significant impact on navigation:

- Imagine a person inside a building, pushing a pram. She intends to get from the ground floor to an upper floor. This person opts for a path with an `elevator` (in case the pram fits in), deliberately accepting a detour.
- In the same building a second person on business has an appointment in an office. Say it is on the *second floor*. Rather than waiting for the elevator, this person uses the `staircase` in order to arrive timely. Now let us assume a slightly modified situation: The appointment takes place on the *ninth floor*. In this case the person may instead be willing to use the elevator.

Although in both situations, the topology of the building is exactly the same, there are *two* interpretations of distance. The *personal* context of the wayfinder matters. The trade-off described in these situations can be modelled by **soft**

constraints: $\forall path \in G_R[of\ type\ t] : path.cost = [time_{wait} +]\ bonus/penalty * path.time$. They alter the costs of traversing certain regions of the environment (e.g. $t = stairs \rightarrow penalty = 4$) in favour of others (e.g. $t = elevator \rightarrow time_{wait} = 20$).

3 Conclusion

In this paper we presented a hierarchically structured model of an indoor environment which accounts for different entry and exit points of regions. Defined upon concrete geometries, the model is not abstract but can be implemented and provided with real data from floor plans. Furthermore, we presented an algorithm which partitions regions according to visibility criteria, so that route descriptions can be given for their interior. It would be interesting to study deeper the ties between the spatial entities in the model and their linguistic counterparts for route descriptions. Another point for research is the further development of constraints: Semantic Web technologies are appealing for their specification (annotation of maps in a wiki-like style), and especially their processing. It would be also worthwhile to examine inhowfar constraint processing and hierarchical planning could be intertwined.

Acknowledgements

This research has been co-funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net).

References

1. C. Becker and F. Dürr. On location models for ubiquitous computing. *Personal Ubiquitous Comput.*, 9(1):20–31, January 2005.
2. H. Choset and J. Burdick. Sensor-Based Exploration: The Hierarchical Generalized Voronoi Graph. *International Journal of Robotics Research*, 19(2):96 – 125, February 2000.
3. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Visibility Graphs: Finding the Shortest Route. In *Computational Geometry: Algorithms and Applications*, chapter 15, pages 307–317. Springer, 2000.
4. M. J. Egenhofer and R. D. Franzosa. Point Set Topological Relations. *International Journal of Geographical Information Systems*, 5:161–174, 1991.
5. G. Franz, H. Mallot, and J. Wiener. 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*, pages 30–38, 2005.
6. P.-Y. Gilliéron and B. Merminod. Personal Navigation System for Indoor Applications. In *Proceedings of the 11th IAIN World Congress on Smart Navigation, Systems and Services*, Berlin, 2003.

7. M. D. Hendricks, M. J. Egenhofer, and K. Hornsby. Structuring a Wayfinder's Dynamic Space-Time Environment. In *Proceedings of the International Conference on Spatial Information Theory*, volume 2825 of *LNCS*, pages 75–92. Springer, 2003.
8. H. Hu and D. L. Lee. Semantic Location Modeling for Location Navigation in Mobile Environment. In *Proceedings of the 5th IEEE International Conference on Mobile Data Management*, pages 52–61. IEEE Computer Society, 2004.
9. N. Jing, Y.-W. Huang, and E. A. Rundensteiner. Hierarchical Encoded Path Views for Path Query Processing: An Optimal Model and Its Performance Evaluation. *Knowledge and Data Engineering*, 10(3):409–432, 1998.
10. B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the Hybrid Spatial Semantic Hierarchy. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 5 of *ICRA*, pages 4845–4851. IEEE Computer Society, 2004.
11. S. Lefebvre and S. Hornus. Automatic cell-and-portal decomposition. Technical Report 4898, INRIA, July 2003.
12. B. Lorenz, H. J. Ohlbach, and E.-P. Stoffel. A Hybrid Spatial Model for Representing Indoor Environments. In *Proceedings of the 6th International Symposium on Web and Wireless Geographical Information Systems*, volume 4295 of *LNCS*, pages 102–112. Springer, 2006.
13. H. J. Ohlbach, M. Rosner, B. Lorenz, and E.-P. Stoffel. NL Navigation Commands from Indoor WLAN fingerprinting position data. REWERSE Deliverable A1-D7 (http://rewerse.net), 2006.
14. L. Plümer and G. Gröger. Nested Maps – a Formal, Provably Correct Object Model for Spatial Aggregates. In *Proceedings of the 4th ACM International Workshop on Advances in Geographic Information Systems*, pages 76–83. ACM Press, 1996.
15. A. Poulovassilis and M. Levene. A Nested-Graph Model for the Representation and Manipulation of Complex Objects. *ACM Trans. Inf. Syst.*, 12(1):35–68, 1994.
16. M. Raubal and M. Worboys. A Formal Model of the Process of Wayfinding in Built Environments. In *Proceedings of the International Conference on Spatial Information Theory*, volume 1661 of *LNCS*, pages 381–401. Springer, 1999.
17. U.-J. Rüetschi and S. Timpf. Using Image Schemata to Represent Meaningful Spatial Configurations. In *OTM Workshops*, volume 3762 of *LNCS*, pages 1047–1055. Springer, 2005.
18. S. Shekhar, A. Fetterer, and B. Goyal. Materialization Trade-Offs in Hierarchical Shortest Path Algorithms. In *Symposium on Large Spatial Databases*, pages 94–111, 1997.
19. C. Stahl and J. Haupert. Taking Location Modelling to New Levels: A Map Modelling Toolkit for Intelligent Environments. In *Proceedings of the 2nd International Workshop on Location- and Context-Awareness*, volume 3987 of *LNCS*, pages 74–85. Springer, 2006.
20. V. Tsetsos, C. Anagnostopoulos, P. Kikiras, and S. Hadjiefthymiades. Semantically enriched navigation for indoor environments. *International Journal of Web and Grid Services*, 2(4):453–478, 2006.