A Context-Aware Locomotion Assistance Device for the Blind

Christophe Jacquet, Yolaine Bourda and Yacine Bellik

September 2004

Abstract

In this paper, we present a study which aims at designing a locomotion assistance device that can deliver semantic information about its surrounding environment at any time. As a first step towards this goal, we introduce an original model suited for the description of building structure, and we present an algorithm that exploits these descriptions. Then, we explain how it is possible to link semantics to structure. Finally, we expose some research directions for user positioning and human-computer interface design.

Introduction

Over the past few years, several electronic travel aids for the blind have been developed. They provide warning signals about approaching obstacles, thus improving users' anticipatory capabilities.

To improve this kind of systems, we develop methods to give them the ability to provide *symbolic information* about pointed objects.

This paper discusses our preliminary results and future directions on this topic. The basic needs for our project are:

- a model and an associated formalism to describe and annotate architectural environments;
- algorithms to determine what relevant information to provide users with;
- a means to constantly know the 3D positions of the device and of the user.

After giving a short overview of the system, we present a model for the description of buildings and an associated algorithm. Next, we introduce semantic representations, and we show how to link them to building structure descriptions. This represents the work achieved so far. The last part of the paper presents directions for user positioning and human-computer interface design.

1 Towards New Locomotion Assistance Devices

1.1 History of Existing Systems

The basic principle underlying existing locomotion assistance devices for the blind is to measure the distances to obstacles (information capture), and to provide users with warning signals (information presentation).

Figure 1: Photo of the *Teletact*.

To measure distances, some devices use infrared sensors (like the Tom Pouce, developed by the LIMSI[1] and the LAC[2]) or ultrasonic sensors (like the Miniguide, see GDP Research (2003)). Both of these techniques detect obstacles within a wide range (roughly 30 degrees), a few meters ahead. To increase precision, other systems like the Teletact 1 (Farcy and Bellik, 2002) or the LaserCane N-2000 (Nurion-Raycal, 2002) use laser sensors. Their very narrow beams lead to very precise measurements, at a maximum distance of 10 meters, but they fail to detect narrow obstacles upon high incidences. The Teletact 2 combines an infrared and a laser sensor so as to yield good results at short range as well as at long range (see fig. 1).

To provide user feedback, existing systems use either sounds (for instance, the Teletact 1 uses musical notes corresponding to distance intervals) or tactile vibrations (like the Teletact 2).

Our goal is to build upon this kind of existing devices and augment them with contextual information (Salber et al., 1999). Developed in collaboration with Supélec, the new system will be aware of its current location, and then give relevant symbolic information to the user.

1.2 Overview of the Future System

The system will try to determine its position thanks to a GPS (Global Positioning System) receiver where GPS reception is possible, and otherwise thanks to an inertial unit. The position calculated from these devices will then be matched against structural and semantic information embedded in the environment description retrieved through WiFi connections (see section 5.1), so as to increase precision and compensate for positioning errors (see fig. 2).

Context-awareness will be enhanced thanks to telemeter data (as in existing devices the new system builds upon) and to light sensors that can provide additional information about light sources (sunlight, artificial light).

When the user's position has been determined, the system will give them context-related semantic information. And when they point at some specific object or location, the system will provide them with information *about* this object or location.

For instance, when a user points at their boss' door, current devices are only

[1]Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur
[2]Laboratoire Aimé Cotton

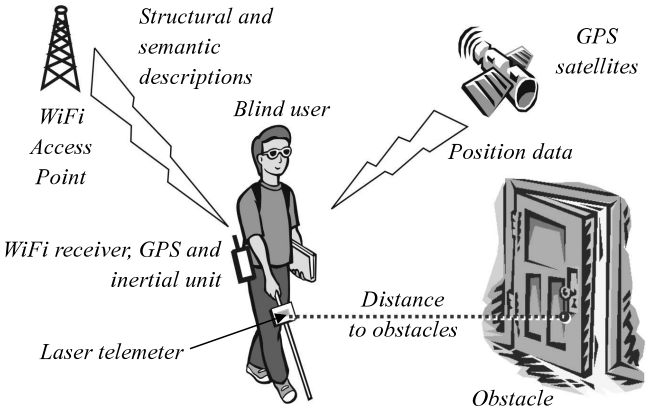

Figure 2: Overview of the future system.

able to tell them that "there is an obstacle three meters ahead". The device we are describing here will be able to add that "this obstacle is a door", and that "this door leads to the boss' office".

Such a system can significantly improve blind people's lives, by giving them precise information about their environment. They should therefore gain autonomy, being no longer compelled to rely on other people to find their way.

2 Structure Modeling

To build such a system, we must be able to model users' environments. Up to now, we have worked on building descriptions only. Of course, our model will eventually cover the full range of environment descriptions.

2.1 Existing Description Formats

Currently, formats like VRML (Virtual Reality Modeling Language, see Web 3D Consortium (1997)) or X3D (Extensible 3D, see Web 3D Consortium (2003)) are available to describe 3D scenes. However, they focus on describing the mere visual *appearance* of environments. Likewise, Computer-Aided Design (CAD) tools allow only the description of the *geometrical* characteristics of modeled objects. In contrast, we need to embrace both their *structure* and *semantics*.

Indeed, the *structure* of architectural environments determines how architectural elements are organized to compose buildings. Some elements of this structure may not be visible: we can imagine that a room be divided into two zones, a smoking one and a non smoking one. From a semantic point of view, a frontier exists between the zones. Although this is not a *physical* frontier, we need to model it. We call this a *virtual wall*.

Symbolic data bring semantics to the structure they are associated with. For instance, these data may contain information about the owners of rooms in a building, access restriction schemes, fire instructions, etc.

To put it short, virtual reality models target *sighted people* and try to describe scenes with many accurate visual details in order to be visually as close to reality

as possible. Conversely, we target *blind people* and thus we need to model the *organization* of environments.

For these reasons, most existing current 3D description languages do not suit our particular needs.

Geographical databases described in Hadzilacos and Tryfona (1997) represent a useful formalism when dealing with geographic data. But they do not allow the representation of strong structure, so we will not use them for building description.

Thus we define our own model to describe environments, but we aim at being able to perform conversions from existing descriptions (see section 5.2).

2.2 Modeling Building Structure

2.2.1 A Three-Tier Model

At first sight, it seems that building structure is intrinsically hierarchical. A building is composed of several floors, each one containing rooms, that in turn are delimited by walls, and so on. Therefore, the first modeling method that comes to mind is to use a tree structure.

But unfortunately, an only tree cannot account for the structure of a whole building: for instance, in a hierarchical model, a wall located between two different rooms would have to be a descendant of both rooms, which does not fit in a tree model.

To solve this issue, we use three trees, each one inducing a hierarchy over a category of architectural elements. Therefore, there are three such categories of elements, referred to as *tiers*:

- *first tier*: we call *lexical elements* the simple (elementary) architectural elements, such as walls, doors, flights of stairs, and so on;
- *second tier*: the so-called *syntactic elements* are complex (composed) architectural elements, constituted by putting together several lexical elements. For instance, a room is defined by its walls, a stairway is defined by several flights of stairs and landings, and so on;
- *third tier*: syntactic elements are further aggregated in what we call *aggregation elements*. For instance, several offices can be gathered in a *cluster* called, say, “sales department”.

It is possible to draw a parallel between our terminology and the structure of natural-language texts. At a low level, texts are simply made up with words: this is the lexical level. These words are then put together in sentences with respect to a defined syntax. In turn, sentences are aggregated in various units such as paragraphs, bulleted lists and so forth.

2.2.2 Building a Description

A description is composed of three tiers of objects, bound together as shown on fig. 3.

Elements are linked by two kinds of edges:

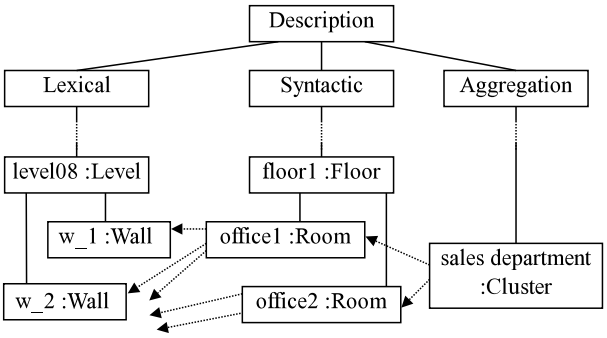

Figure 3: Excerpt of an example description. The *sales department* is composed of two offices, in turn defined by some walls. Intermediate layers of objects have not been represented for the sake of clarity, and have been replaced with dotted vertical lines.

- *inclusion links* (solid lines on fig. 3) represent inclusion between elements within a given tier. This gives a hierarchical structure to the tiers, that are represented by trees;
- *composition links* (dotted lines on fig. 3) enable objects of tier n to be composed of elements of tier $n-1$.

For instance, a room is on the one hand *composed* of several walls, and on the other hand *included* in a floor.

2.2.3 Concept hierarchy

The objects that appear in the description correspond to real architectural elements: a given wall, a given room, etc. These real objects can therefore be considered as instances of general concepts (or classes): a class representing walls, a class representing rooms, etc.

These classes take place in a concept hierarchy. On grounds of genericity, all classes derive from an abstract common ancestor called `Element`. This class has got three abstract subclasses, corresponding to the three tiers of our model: respectively `LexicalElement`, `SyntacticElement` and `AggregationElement`.

Concrete classes are then derived from one of these abstract classes, depending on what tier they belong to.

2.3 Representation of descriptions

This model has a strong hierarchical structure, so it is natural to use a representation format that makes this structure explicit. XML (eXtensible Markup Language) allows this type of explicit representation of structure, in addition to being widespread and universal. Therefore, we can use XML to store the three tiers of a description. For instance, the description of fig. 3 would lead to the following XML file:

```
<Description>
  <Lexical>
```

```
    ...
      <Level id="level-08" z="16m" height="2.20m">
        <Wall id="w-1" x1="2.3m" y1="4.2m" ... />
        <Wall id="w-2" x1="2.3m" y1="4.2m" ... />
        ...
        <Wall id="w-n" x1="12.3m" y1="13.2m" ... >
          <Door id="d-1" x="2.3m" width="1m" ... />
          <Window id="f-3" x="2.7m" y="0.5m" ... />
        </Wall>
      </Level>
    ...
  </Lexical>

  <Syntactic>
    ...
      <Floor id="floor-1">
        <Room id="office-1">
          <link ref="w-1" />
          <link ref="w-2" />
          ...
        </Room>
        <Room id="office-2" />
          ...
          <link ref="w-n" />
        </Room>
      ...
  </Syntactic>

  <Aggregation>
    ...
      <Cluster id="sales-department">
        <link ref="office-1" />
        <link ref="office-2" />
      </Cluster>
    ...
  </Aggregation>
</Description>
```

Inclusion links are represented implicitly through XML element imbrication, while composition links are represented explicitly thanks to XML `link` elements.

3 Beyond Structure: Semantics

3.1 Motivation

What are we able to do now? When the user points at an architectural element, the system is able to find it in its cartography. For instance, if the user points at a door, the system *knows* that it is a door, and that there is, say, an office behind.

However, our ultimate goal is to provide the user with *semantic* information. In the above example, the system would not only state that the user is pointing at a door leading to an office: it would also return the office owner's name, the office function, and so on.

To do this, we associate semantic information to the structure description. More generally, such information can be used:

- to add *normative* information to the structure, for example in order to tag restricted areas in a building,
- to identify objects, rooms, and zones;
- to represent connexity information;
- to add specific information to certain kinds of objects; for instance information about painters could be associated with sculptures in a museum.

3.2 Solutions – Linking Semantics to Structure

Information will be modeled using the Resource Description Framework (RDF) (Manola et al., 2003), an emerging W3C standard quite close to the theory of conceptual graphs (Sowa, 1976). In practice, objects of interest (called *resources*) are associated with each other by properties. Such associations (called *assertions*) are denoted by triples:

```
(subject, property, object)
```

Resources are instances of classes, the class hierarchy being expressed in OWL, the Web Ontology Language (McGuinness and van Harmelen, 2003).

Therefore, we end up with two worlds: on the one hand, a world of structure descriptions, and on the other hand, a world of semantic annotations. However, these annotations exist only with respect to the underlying structure of buildings.

This is the reason why semantic descriptions must be linked to the architectural elements of the structure description. This is done thanks to some instances of a class called `Place` in the RDF graph.

Indeed, instances of `Place` correspond to architectural elements, thus enabling semantic graphs to be *anchored* in the structure description. The actual correspondence is achieved through the use of a common identifier.

On fig. 4, we can see the role played by the instances of class `Place`: such instances belong to the RDF graph, but they correspond one-to-one with the structural elements.

In practice, the structure in fig. 4 looks like this:

```
...
  <Building id="building-34">
    <Floor id="floor-1">
      <Office id="office-210">
        ...
      </Office>
    </Floor>
```

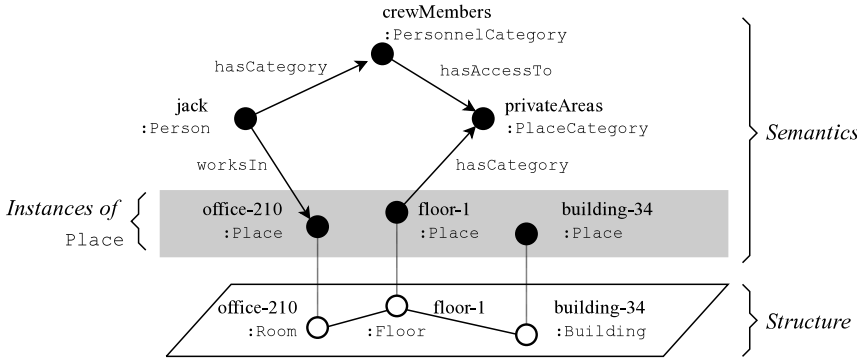

Figure 4: Linking between semantics and structure.

```
  </Building>
...
```

The triples of the associated semantic descriptions look like this:

```
(jack, hasCategory, crewMembers)
(crewMembers, hasAccessTo, privateAreas)
(jack, worksIn, office-210)
(floor-1, hasCategory, privateAreas)
...
(jack, rdf:type, Person)
(crewMembers, rdf:type, PersonelCategory)
(privateAreas, rdf:type, PlaceCategory)
(office-210, rdf:type, Place)
(floor-1, rdf:type, Place)
```

The structural elements `office-210` and `floor-1` have counterparts in the semantic description. The latter are instances of the class `Place`, and share the same identifier as their "structural" counterparts, thus linking structure and semantics.

3.3 Current Ontology for Structure Description

Currently, we have defined a basic ontology for structure description. The classes of this ontology take place in three categories:

- *Locations:* in addition to the basic class `Place` already described, we define two classes, `Function` that associates a function to a location, and `PlaceCategory` that allows categories of places to be defined (with respect to an access restriction scheme);
- *People:* likewise, a class `Person` represents human beings, `Role` represents roles played by people within an organization, and `PersonnelCategory` allows to specify categories of people (again with respect to an access

restriction scheme). Actually, we do not introduce a new class, but instead we use the existing class `Person` from the FOAF (Friend Of A Friend) ontology (Dumbill, 2002);

- *Schedules:* the classes `Schedule` and `Event` are used to associate events to locations.

The ontology also defines a whole set of RDF properties, used to specify relations between class instances. For example, `hasCategory` is used to associate a `PersonnelCategory` to a `Person` or a `PlaceCategory` to a `Place`; `hasAccessTo` states that a `Person` or a `PersonnelCategory` have access to a `Place` or `PlaceCategory`.

3.4 Reasoning

More than just enabling the description of complex relationships, a semantic description allows the definition of reasoning rules.

For instance, in the example of fig. 4 we have the following triples:

```
(jack, hasCategory, crewMembers)
(crewMembers, hasAccessTo, privateAreas)
(floor-1, hasCategory, privateAreas)
```

From that, it seems reasonable to deduce the additional triple:

```
(jack, hasAccessTo, floor-1)
```

Thanks to reasoning rules, it will be possible to implement such *common sense* behaviors, and thus enable the automatic deduction of new triples from the set of existing ones.

4 Determining Relevant Information

4.1 The Problem

From these architectural descriptions, the system will be able to determine:

- the position of the user;
- the object or location pointed at by the user.

However, we still do not know the *level of detail*, i.e. the *granularity* of information needed by the user.

At any moment, we must be able to provide users with relevant – not too detailed, not too general – information. Indeed, too general information is useless, and too detailed information might not be understandable if the user does not know the associated context. To illustrate this, let us look at an example (fig. 5).

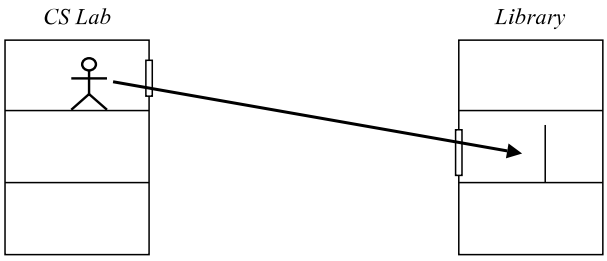

Figure 5: A user points from one building to another one.

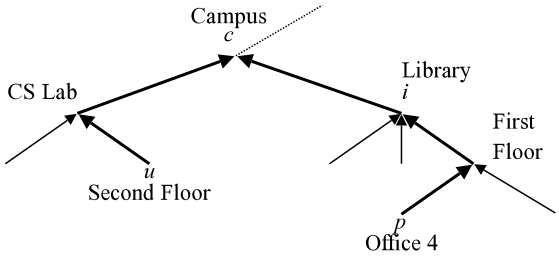

Figure 6: Tree representation of the scene.

4.2 Proposed Algorithm

Suppose that the user is located in u, on the second floor of the Computer Science (CS) Laboratory. He or she points through an open window at p, an office on the first floor of another building, the Library building, located next to the CS building. What information shall we return? Information attached to the room, the floor, the building, the campus...?

To solve this problem, we represent the scene as a tree. This tree is the syntactic sub-tree of our model (fig. 6).

First, let us find the deepest node that is common to both the path leading from u to the root, and the path leading from p to the root. This node is labeled c on fig. 6.

What happens if we return information located on c or above? Such information is too general, because it covers p as well as u. Thus, it will probably be useless for the user, because being in the CS building they already know that they are on the campus.

In consequence, we should return information located below c. However, the returned piece of information must correspond to p, so it must be located on the path leading from p to the root of the tree. That means that eventually we have to focus on the sub-path leading from p to c (c excluded).

We notice that this sub-path is outside of the context of the user. Therefore, if we return too precise information within this sub-path, the user would not understand because he or she would not know the context.

That is why we must return most general information inside the sub-path of interest. It means that we choose to return information located in i.

Note that there is a special case when u and p are on the same branch of the tree. In this case, it is not possible that p be above u, because the

user cannot point at an object that *surrounds* them. Therefore, we return information located on the child of u that is on the path leading to p.

In short, the above algorithm enables us to determine a default level of granularity when returning information about the pointed object. However, the user might wish to gain access to another level of granularity. Thus, the final user interface will have to offer some means of climbing up and down the tree.

5 Future Work

5.1 Tracking User Position: Semantic Map Matching

The whole system depends on its ability to track the position of the user. In the open, it should be quite easy thanks to the use of a GPS receiver. GPS positioning has become increasingly accurate over the past, especially since Selective Availability has been turned off (Selective Availability used to allow the US Department of Defense to introduce intentional errors in GPS signals in order to limit accuracy in non-US government user receivers). In consequence, GPS provides a reliable means to position the user *where GPS reception is possible.*

However, there are many place where GPS reception is not possible, for example in buildings and in dense urban areas. For instance Chao et al. (2001) reports very poor conditions in Hong Kong urban areas, regarding GPS and GLONASS (a Russian satellite positioning system similar to GPS). Unfortunately, these are precisely the places where our system would prove the most useful. Therefore we need a means to compute the position of users, even when GPS signal is unavailable.

The basic idea is to embed an inertial unit in the device, and then determine successive positions by means of dead reckoning: at every moment we try to determine our new position by estimating how far we have moved since the last computed position (this is done thanks to gyroscopes, magnetic compasses and accelerometers embedded in an inertial unit). We call this *relative positioning.* Conversely, when GPS reception is possible, we can perform *absolute positioning* because the GPS receiver can compute absolute positions from satellite signals.

In short, our system will use reliable GPS information to find its position *absolutely* when possible, and compute its position incrementally and relatively to the last GPS-acquired position when losing GPS signal.

Unfortunately, dead reckoning has the drawback of being very much error-prone (Fusiello and Caprile, 1997) : computed positions are likely to slowly deviate from real positions (cumulative errors). To overcome this shortcoming, the *map-matching* method has been proposed (Bernstein and Kornhauser, 1996; Kitazawa et al., 2000). The basic idea is to restrict the movements of people along well-defined paths on a map. Hence, it is possible to reduce deviation errors by permanently computing the most probable position of the user *along a path* and not in every possible direction.

We do not want to impose such restrictions on the users of our system, but it may be possible to perform some kind of map-matching anyway, because the device has got sensors that give much information about the environment. For instance, from telemeter data we can deduce whether the user is following a wall (and how far the wall is); from light sensors we can know if we are inside a building or outside, etc. These data, coupled with the structural and

semantic description of the environment we shall have, are likely to restrict users' probability of presence in some well-defined areas without imposing constraints on their movements.

We call this *semantic map-matching*. Still sketchy at the moment, this method seems to be an interesting research topic and will be further investigated in the future.

5.2 Acquisition of Descriptions

In this whole paper, we have assumed that we had environment descriptions at our disposal. Actually, these descriptions need to be constructed. We have listed three ways of obtaining environment descriptions:

- to write them from scratch, for example by using a graphical editor;
- to perform a conversion from an existing description, either automatically or semi-automatically. As architects are currently defining their own languages for building description (van Rees et al., 2002), it could be useful to be able to reuse their building description data at some point in the future. Similarly, we could use existing Geographical Information Systems to obtain geographical information;
- to scan environments with the device, and label objects on the fly.

The last method seems the most promising – and the most challenging, too. It would allow blind people to use their locomotion assistance devices even in places where there is no available description. We can imagine that they would tag the environment when first visiting a new place accompanied by some sighted person (as blind people usually do). From these data, the system would compute a partial model that could be re-used next time, as in the *map learning* method (Fusiello and Caprile, 1997). Each time the system would return to the same place, it would refine its model based on new information acquired.

It can even be imagined that blind people visiting a new place would be allowed to publish their partial model for others to use it and improve it in turn.

5.3 Presenting Semantic Data

In section 3 we have defined a way to associate semantic annotations to environment descriptions. However, our semantic graphs have to comply with a given ontology. So far, we have defined a test ontology that covers the fields of basic organizational relations. It is possible to extend this ontology, so as to cover wider fields. Unfortunately, it will never be possible to design an ontology wide enough to cover every situation.

Thus, description designers will have to define their own ontologies, or ontology fragments. As a result, the system will have to be able to deal with new (i.e. unknown) ontologies. In particular, it will have to know how to present users with information conforming to any given ontology. We could impose that every ontology has to be accompanied by a presentation sheet that would explain how to present instance data, within the framework of a multimodal system: for instance, a particular type of relations could be represented by speech synthesis or using a Braille display.

Conclusion

Our project builds upon electronic travel aids that have been developed recently. Already useful, these devices are nonetheless capable of only indicating distances to obstacles, and not of giving any higher-level information. To achieve our goal of being able to name objects and give additional information, we have proposed solutions for two critical issues in this paper:

- defining a formalism to model the structure of visited buildings;
- designing a model to represent semantic information associated with the structure. When the system has the ability to constantly know its geographical coordinates, it will therefore be able to determine candidate interesting information.

User position tracking, description acquisition and semantic data presentation will be among the topics of our future research work.

References

Bernstein, D. and Kornhauser, A. (1996), An Introduction to Map Matching for Personal Navigation Assistants, Technical Report, Princeton University, New Jersey TIDE Center, New Jersey Institute of Technology.

Chao, J., Chen, Y., Wu Chen, X. D., Li, Z., Wong, N. and Yu, M. (2001), "An Experimental Investigation into the Performance of GPS-based Vehicle Positioning in Very Dense Urban Areas", *Journal of Geospatial Engineering* **3**(1), 59–66.

Dumbill, E. (2002), Finding friends with XML and RDF, `http://www-106.ibm.com/developerworks/xml/library/x-foaf.html`.

Farcy, R. and Bellik, Y. (2002), Locomotion Assistance for the Blind, *in* S. Keates, P. Langdom, P. Clarkson and P. Robinson (eds.), *Universal Access and Assistive Technology*, Springer, pp.277–284.

Fusiello, A. and Caprile, B. (1997), "Synthesis of Indoor Maps in Presence of Uncertainty", *Robotics and Autonomous Systems* **22**(2), 103–114.

GDP Research (2003), Miniguide Home Page, `http://www.gdp-research.com.au/ultra.htm`.

Hadzilacos, T. and Tryfona, N. (1997), "An Extended Entity-Relationship Model for Geographic Applications", *SIGMOD Record* **26**(3), 24–29.

Kitazawa, K., Konishi, Y. and Shibasaki, R. (2000), A Method of Map Matching For Personal Positioning Systems, *in Proceedings of the Asian Conference on Remote Sensing ACRS 2000*.

Manola, F., Miller, E. and McBride, B. (2003), RDF Primer, Technical Report, W3C World Wide Web Consortium.

McGuinness, D. L. and van Harmelen, F. (2003), OWL Web Ontology Language Overview, Technical Report, W3C World Wide Web Consortium.

Nurion-Raycal (2002), LaserCane Home Page, `http://www.nurion.net/lasercane.htm`.

Salber, D., Dey, A. K. and Abowd, G. D. (1999), The Context Toolkit: Aiding the Development of Context-Enabled Applications, *in Proceedings of CHI'99*, ACM Press, pp.434–441.

Sowa, J. F. (1976), "Conceptual Graphs for a Database Interface", *IBM Journal of Research and Development* **20**(4), 336–357.

van Rees, R., Tolman, F. and Beheshti, R. (2002), How bcXML Handles Construction Semantics, *in CIB W98 Workshop*, Denmark, Århus.

Web 3D Consortium (1997), The Virtual Reality Modeling Language International Standard ISO/IEC 14772-1:1997, Technical Report. Available at `http://www.web3d.org/technicalinfo/specifications/vrml97/`.

Web 3D Consortium (2003), Extensible 3D (X3D) International Standard ISO/IEC FCD 19775:200x Final Committee Draft, Technical Report. Available at `http://www.web3d.org/fs_specifications.htm`.