Knowledge Representation and Management in Indoor Mobile Environments

Imad Afyouni

THESE / UNIVERSITE DE BREST
sous le sceau de l'Université européenne de Bretagne

pour obtenir le titre de
DOCTEUR DE l'UNIVERSITE DE BREST
Spécialité : Informatique
Ecole Doctorale des Sciences de la Mer

présentée par
Imad AFYOUNI

Préparée à l'Unité Mixte de recherche (n°3634)
Naval Academy Research Institute

« Knowledge Representation and Management in Indoor Mobile Environments »

Thèse soutenue le 17 septembre 2013
devant le jury composé de :

Thierry Delot
Professeurs des universités - Univ. de Valenciennes - INRIA Lille / *rapporteur*
Hervé Martin
Professeurs des universités - Directeur du Laboratoire d'Informatique de Grenoble / *rapporteur*

Jacques Tisseau
Professeurs des universités - Directeur de l'ENIB Brest / *examinateur*
Karine Zeitouni
Professeurs des universités - Univ. de Versailles, Paris / *examinateur*

Sergio Ilarri
Associate professor - University of Zaragoza / Invité

Christophe CLARAMUNT
Professeurs des universités - Directeur de l'IRENav - Ecole Navale, Brest / Directeur de thèse
Cyril RAY
Maître de conférences - IRENav - Ecole Navale, Brest / Co-directeur de thèse

N^o d'ordre : \NumeroOrdre

Thèse

présentée à

l'Université de Bretagne Occidentale

pour obtenir le titre de

Docteur de l'Université de Bretagne Occidentale

Mention *Informatique*

par

Imad AFYOUNI

Knowledge Representation and Management in Indoor Mobile Environments

soutenue le 17 septembre 2013 devant la commission d'examen :

Composition du Jury

Rapporteurs :	Pr. Thierry Delot, Université de Valenciennes – INRIA Lille
	Pr. Hervé Martin, Université de Grenoble
Examinateurs :	Pr. Jacques Tisseau, ENIB Brest
	Pr. Karine Zeitouni, Université de Versailles (UVSQ)
Invité :	Dr. Sergio Ilarri, University of Zaragoza
Co-Directeur :	Dr. Cyril Ray, IRENav Brest
Directeur :	Pr. Christophe Claramunt, IRENav Brest

Naval Academy Research Institute (IRENav)

To my parents, Taha and Karima

To my wife Naamat

Acknowledgement

One of the joys of completion this Ph.D. is to look over the journey past and remember all the friends and family who have helped and supported me along this long but successful adventure.

I would like to express my sincere gratitude to Professor Christophe Claramunt and Dr. Cyril Ray, who are not only advisors but dear friends. I could not have asked for better role models, each inspirational, supportive, and patient. You have also given me the freedom to pursue various research directions without objection. Christophe, I hope that I could be as lively, enthusiastic, and energetic as you and to someday be able to command an audience as well as you can. Cyril, I am very grateful for the many insightful discussions and suggestions; you were always my primary resource for getting my science questions answered.

I would also like to thank the members of my Ph.D. committee, Professor Thierry Delot, Professor Hervé Martin, Professor Jacques Tisseau, Professor Karine Zeitouni, and Associate Professor Sergio Ilarri, who provided encouraging and constructive feedback. It is not an easy task to review a thesis, and I am grateful for their thoughtful and detailed comments. Special thanks go to Dr. Sergio Ilarri for his patience, understanding, guidance, and most of all the encouragement he has given me during my thesis. Your hard work is very much appreciated! Keep up the great work!!

Regarding funding, my thanks go to the French Ministry of Defense, and to the Naval Academy represented by the Rear Admiral Philippe Hello. I thank all the present and past members of the MoTIM group of the Naval Academy Research Institute (IRENav): my gratitue goes especially to Abdel (extremely knowledgeable, helpful, and friendly), Rémy (supportive, very pleasant, and inspiring), Éric (a friendly and knowledgeable expert, I enjoyed teaching by your side), David (a nice and helpful friend who has been pretty supportive during my search for jobs and writing), and Joseph (a great personality, I also enjoyed hearing from you about your beautiful missions overseas). I also thank Rozenne, Laurent, Delphine, and Valérie from the Underwater Acoustics part of our team and the members of the M2EN group (Jacques-André, Jean-Yves, Jean-Pierre, Patrick, Jean-Frédéric, François, Céline, Thomas, Frédéric, Franck, and Hocine) who have been so helpful and made me feel very welcome when I first arrived to IRENav. I?ll also not forget the institute administrative assistants (Marie - great personality, great performance!), Capitaine Savary, Magalie, Christine, Luc, Jacques, and the SAGA team) whose friendly attitude and willingness to carry out

all manner of tasks in the shortest possible time and with the highest possible levels of enthusiasm are greatly appreciated.

I also thank my friends and colleagues; I list a few from IRENav: Georges (great friend! I'll never forget your help and support when I broke my leg), Emerry (cooool! best luck :)), Laurent Etienne (so friendly and I'm glad to have you as a friend), Pierre-Luc (great coach!), Géraldine (very supportive), Ali (best friend!), Nicolas (carambar, nutella, etc.), Mathieu and Lamia (funny, friendly, and very supportive), Dieudonné, Zhibin, Ashraf, Clément, Jingya, Arnaud, Ines, Sofiane, Jean-Baptiste, Alex, Julien, etc. A few from outside: Ramez, Ayman, Ali, Kamal, Mustapha; I keep great and fun moments with you.

Outside of Brest, I thank all staff of the department of Computer Languages and Systems Engineering at the University of Zaragoza for the opportunity to meet with you. It was exciting to be part of the Distributed Information Systems (SID) team - a very knowledgeable, fun, and friendly group of people. Special thanks to Roberto for helping me get used to life in Zaragoza quickly.

Most sincerely, I thank my parents, Taha and Karima, for providing me unconditional love and care. Without you, I never would have made it here; thank you for everything and I miss you so much. My thanks also go to my brother and sisters. You have been my best friends all my life; I love you dearly and thank you for all your advice and support. I am also grateful to my family-in-law, especially my mother-in-law Halima for warm-heartedness. Finally, the best outcome from these past several years is finding my best friend, soul-mate, and wife Naamat. I praise Allah for sending me you my love, there are no words to convey how much I love you. These past several years have not been an easy ride, both academically and personally for both of us. I feel that we both learned a lot about life and this strengthened our commitment and determination to each other. Thank you my dear sweet love for sharing my dreams.

Ahead of all, all praise is due to Allah, without whom I could not have accomplished anything. No amount of gratefulness will be enough to thank Allah for the blessings and guidance he have given me. I pray for Allah's support and guidance in the rest of my career and my life.

Contents

List of Figures

List of Algorithms

List of Tables

Abstract

The range of applications in ambient information systems progressively evolves from large to small scale environments. This is particularly the case for applications that assist humans in navigation-related activities in indoor spaces (e.g., airports, museums, office buildings). The research presented by this Ph.D. dissertation develops a data and knowledge representation of an indoor environment that takes into account user-centred contextual dimensions and mobile data management issues. We introduce a hierarchical, context-dependent, and feature-based indoor spatial data model. This model takes into account additional contextual dimensions such as time, user profiles, and real-time events. The model is based on a tree structure in which location information is represented at different levels of abstraction. The hierarchical design favours performance and scalability of location-dependent query processing. A query grammar is developed and implemented on top of that model. This query language supports continuous location-dependent queries and takes into account user preferences at execution time. The concept of location granules is introduced at the query execution and presentation levels.

This modelling approach is complemented by the development of a generic architecture for continuous query processing. Several algorithms for location-dependent query processing over indoor moving objects have been designed and implemented. These algorithms include path searches and range queries applied to both static and moving objects. They are based on an incremental approach in order to execute continuous location-dependent queries. The operators and constraints introduced in the query language and the algorithms for location-dependent query processing have been implemented as a database extension of the open source DBMS PostgreSQL, and where the hierarchical network-based indoor data model has been developed at the logical level. Several experiments have been conducted to evaluate the scalability and performance of the whole framework.

Keywords

Indoor data models, context-aware systems, mobile data management, continuous location-dependent queries, moving objects

INTRODUCTION

Research context

The last decades have seen a tremendous growth in mobile computing and advanced wireless technologies and services. Mobile computing systems take advantage of rapidly evolving wireless technologies to provide end users with personalised and innovative services. Thanks to the portability and processing power of new mobile devices such as Smartphones and Tablets (e.g., iPhone, iPad, and Android-based devices), as well as their extended battery life, a wide range of regularly available and computationally intensive mobile services can be developed and offered to end users. The availability of mobile information systems is driven by the increasing demand of services that provide data access to users *anytime* and *anywhere*. For instance, a report recently published by the International Telecommunication Union (ITU) estimates that about 6 billion mobile phone subscriptions at the end of 2011, out of which 1.08 billion are Smartphones (Go-Gulf, 2012). By 2016, eMarketer expects that 2.1 billion active mobile users worldwide will be using Android/iOS applications alone.

A further step in this development has been foreseen by Marc Weiser in the early 1990s and was referred to as *Ubiquitous Computing* [Weiser, 1993] (also called *Pervasive Computing* [Satyanarayanan, 2001]), in which interconnected smart devices are embedded unobtrusively in everyday environments, and are able to communicate and cooperate in order to offer personalised and adaptive services to human users. In his seminal work on the next generation of computing technologies for the 21st century, Marc Weiser stated that [Weiser, 1991]:

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it".

A more precise definition of Ubiquitous and Pervasive Computing can be derived from the Oxford Dictionary, where Ubiquitous is defined as:

"Adjective: present, appearing, or found everywhere".

And that of Pervasive:

"Adjective: (especially of an unwelcome influence or physical effect) spreading widely throughout an area or a group of people".

Ubiquitous computing becomes a reality thanks to emerging seamless environments of computing, where smart artefacts and sensors, enabled by recent developments in sensor technologies, can collect huge amounts of data and interact among themselves to offer advanced and adapted services. The miniaturization of smart devices helps users in their every-day life activities by providing them with computational abilities in a smart way, and with minimum user involvement.

Another key enabler of pervasive computing environments, apart from the networking infrastructure, relies in the successful integration of *semantics* and *advanced reasoning techniques* into mobile information systems. Those features have been recently characterized under the so-called *Ambient Intelligence (AmI)* technologies [Aarts and Wichert, 2009]. An ambient system is expected to be *sensitive*, *responsive*, *adaptive*, *transparent*, *ubiquitous*, and *intelligent* [Cook et al., 2009]. Applications of ambient systems mainly include *"smart spaces"* such as *smart cities*, *smart homes*, and *smart offices*, which seek to seamlessly support people in their daily lives.

The range of applications in ambient information systems is progressively evolving from large outdoor environments (e.g., *Smart Cities*) to small scale indoor environments (*Smart Homes*, *Smart Offices*). Indoor spaces are closely related to ambient systems where sensors provide real-time data without direct human interaction. An indoor space can be informally defined as a built environment where people usually behave [Li, 2008] (e.g., buildings, commercial malls). Amongst many factors, recent developments of ubiquitous computing and indoor positioning techniques have largely favoured mobile application development to indoor spaces [Kolodziej and Hjelm, 2006]. Nowadays, techniques currently available for *indoor positioning* range from radio-based technologies (WLAN, RFID, Bluetooth) to non-radio technologies (Infrared and Ultrasound) and inertial navigation systems (INS) [Kolodziej and Hjelm, 2006]. Moreover, hybrid approaches appear as promising solutions providing reliable, continuous, and accurate location information [Torres-Solis et al., 2010].

Mobile location-aware services have attracted recent research attention as their development is expected to have significant impact for users in both indoor and outdoor environments [Schiller and Voisard, 2004; Yu and Spaccapietra, 2010]. Location-aware services provide the user with the ability to interact with his/her physical surroundings in order to achieve some tasks. More generally, *context-aware systems* exploit contextual dimensions such as user-centred dimensions (e.g., user profile, user's physical/cognitive capabilities), environmental context (e.g., location), temporal context, and context of execution (e.g., network connectivity, nearby resources). This allows to anticipate user's needs and to customize his/her navigation experience [Baldauf et al., 2007].

A successful integration of indoor knowledge representation (i.e., model of space, objects of interest, user profile and activities, etc.) and ambient systems still requires the development of appropriate spatial data structures and data management facilities. We believe that this is a mandatory development for the delivery of intelligent-based context-aware systems applied to indoor spaces. More specifically, the research presented in this thesis focuses on *"context-aware indoor navigation systems"* in the context of mobile location-aware services.

Motivation and research challenges

There is a growing need for applications that assist humans in their navigation-related activities in indoor spaces. Real-time, continuous, and reliable user guidance in a time-dependent manner is required not only in transit structures such as train stations or airports, but also in museums, hospitals, campuses, and buildings. Moreover, diverse kinds of services can be provided by enabling real-time integration of contextual dimensions into an underlying data model and an associated query processing engine. This includes not only everyday activities and tasks, but also emergency situations and evacuation processes where real-time and reliable responses are particularly crucial, as well as services that support collaborative tasks such as healthcare activities in a pervasive hospital environment [Bardram, 2005; Oliveira et al., 2010].

Despite the continuous development and improvements made in mobile computing and the variety of technologies that can be used to enable ambient intelligence environments, there are still many research challenges that need to be addressed [Cook et al., 2009]. Amongst the many challenges facing the development of context-aware indoor navigation systems, knowledge representation and management issues in indoor mobile environments are studied in this thesis.

Many research questions on how to design a flexible, context-dependent, and efficient indoor data model that incorporates the dynamic properties of the environment still need to be addressed (Figure 1). Additional issues on the incorporation of context-awareness into the data model require further investigation. Notwithstanding their smaller size with respect to outdoor environments, effective modelling of indoor environments can be fairly complex. Thus, data models and methods for outdoor navigation cannot be directly applied. For instance, an indoor data model should take topological configurations into account (e.g., connectivity or adjacency relationships between rooms or floors in a building). In addition, location information, which is often defined with absolute coordinates in outdoor environments, is more likely to be described in relative terms for both physical and functional spaces in indoor environments (e.g., John's office) [Walton and Worboys, 2012]. Moreover, interactions among humans and/or other objects in an indoor space are usually more

frequent than in an outdoor one, which is usually beyond the range of humans' physical interactions. Furthermore, in contrast to car navigation, the main differences of human navigation relies in the degrees of freedom in movement, the velocity of movement, and the level of granularity.

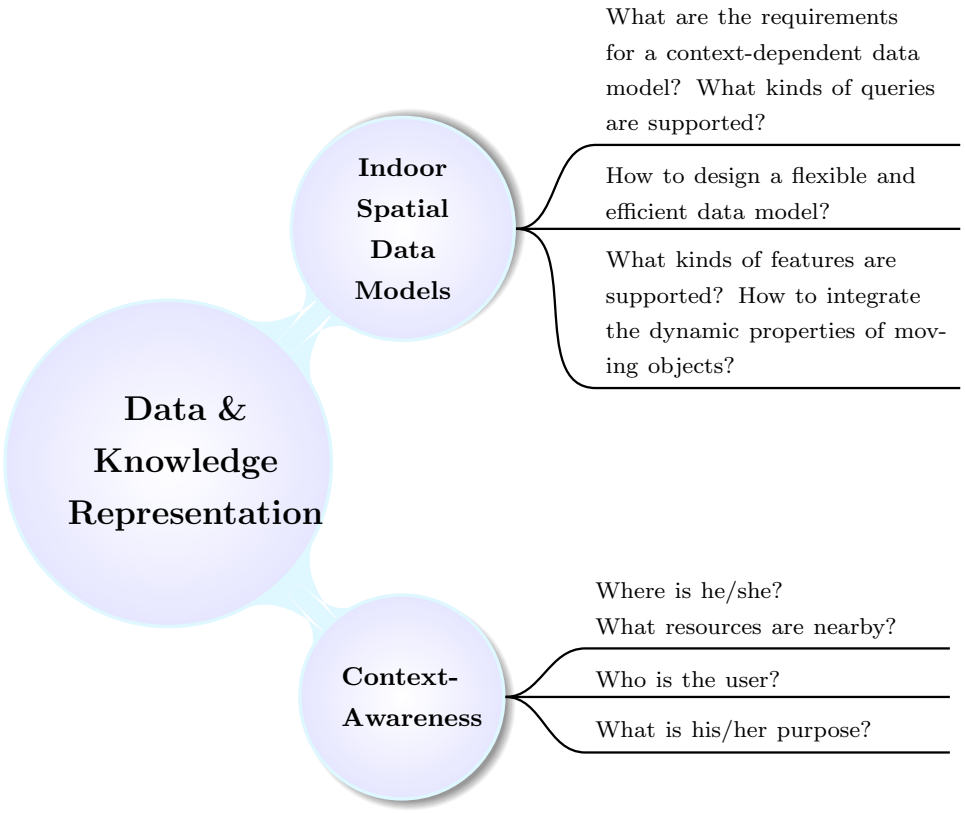


Figure 1: Challenges in data & knowledge representation covered in the thesis

A knowledge-based representation of an indoor environment should take into account the hierarchical layout of the indoor space and: (i) all *static/moving features* that populate an indoor environment, where a feature can refer to either a person (i.e., a mobile user or any other social entity of interest[1]) or an object/event of interest (e.g., sensors, exits, tables, continuous phenomena such as a fire, etc.); (ii) the spatial properties associated with those features (e.g., location and extent); and (iii) the behaviours that emerge from them (i.e., how these objects can interact and communicate within the environment).

Issues regarding data management and processing of location-dependent queries are also of particular interest in this context (Figure 2). *Location-Dependent Queries (LDQs)*, also referred to as *location-based queries* [Zhang et al., 2003; Mokbel, 2005], constitute a key building block in

[1]Human beings that are located in the vicinity and are of interest to the query are referred to as social entities.

context-aware indoor navigation systems. A location-dependent query is a query whose answer depends on the locations of the objects of interest. From a query processing perspective, indoor environments have brought special features and constraints (e.g., multiple floors, stairs, lifts, access privileges, etc.) that should be considered when processing location-dependent queries. Moreover, taking advantage of contextual information favours more appropriate query answers and real-time adjustments according to user preferences and contextual constraints. Those specific properties and requirements for a knowledge-based representation of an indoor mobile environment raise different questions on (Figure 2): 1) how to represent and manage static/moving objects and other entities of interest? 2) how those objects are taken into account during query processing? and 3) what kind of processing architecture do location-dependent queries need?

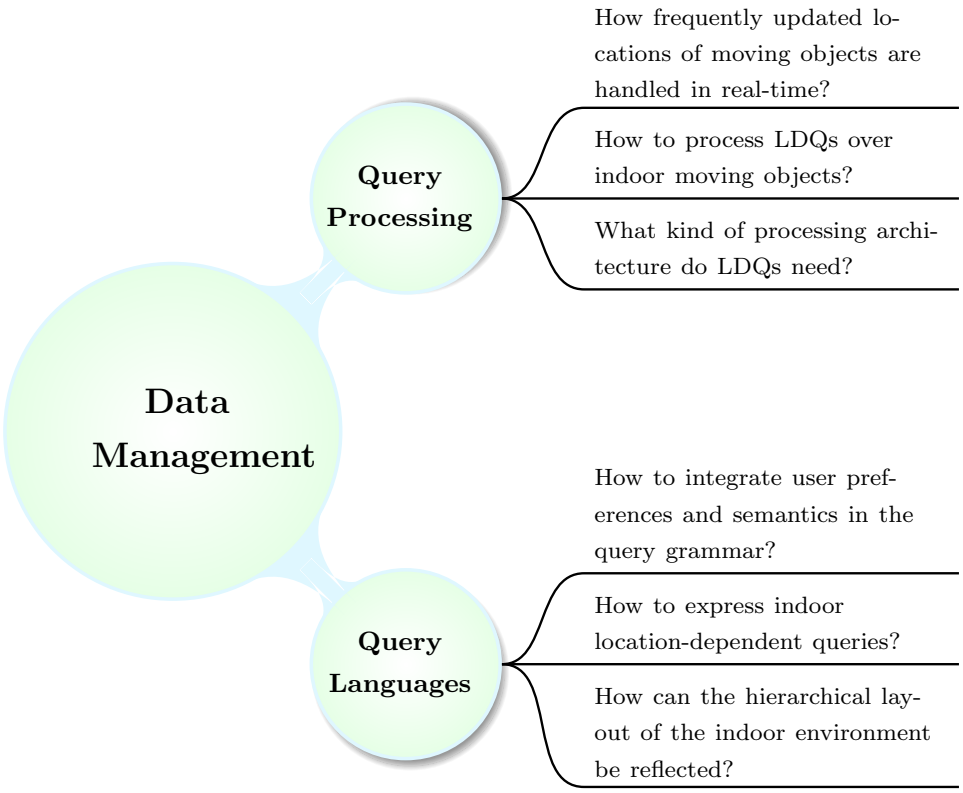


Figure 2: Challenges in data management discussed in the thesis

An appropriate management of static and dynamic data is a key issue for processing these queries, since the result of a query is only valid for a particular location of the query issuer and for certain locations of the objects of interest. As those queries are time-sensitive and location-dependent, they may be valid for only a given period of time (e.g., shops in a mall have certain opening hours and are not available outside that schedule), and within a given area. Therefore, those

queries are expected to be processed as *continuous queries* [Terry et al., 1992], which means that the system should continually keep the answers up-to-date over a certain period of time (usually, until the query is explicitly cancelled by the user). Dataflow processing applications often have a monitoring or filtering aspect in which queries are continuously active. Therefore, continuous location-dependent queries appear to have considerable impact for the development of different categories of location-based and context-aware services.

Location-dependent queries [Ilarri et al., 2010; Wang and Zimmermann, 2011], and more generally context-dependent queries, represent typical examples of *pull-based* services (i.e., requests are triggered by the user), and which are needed in such context-aware systems, as well as a key building block to detect situations of interest for *push-based* services (i.e., services initiated by the service provider without having been requested by the user). The context-dependent character of these queries means that any change in the context (e.g., changes in the locations and/or profiles of the objects that are involved in the query) may significantly affect the answer. For example, if a user wants to find out his/her friends within a range of 100 meters while navigating a shopping centre, the answer depends on both the user's current position and the location of the nearby friends. This type of query is particularly challenging because, in most cases, the user and the entities relevant for the query (e.g., the friends of the user) are moving. Most work on location-dependent query processing has been developed for outdoor environments [Ilarri et al., 2010]. For indoor spaces, approaches for query processing based on *spatial networks* instead of *free Euclidean space* are preferred and more realistic. However, existing approaches for network-based query processing usually assume an outdoor environment [Deng et al., 2009; Lee et al., 2005; Papadias et al., 2003], where for instance hierarchical networks do not naturally appear.

Query languages designed for location-dependent queries over moving objects are strongly related to the data model underneath, and on how moving object data is managed [Erwig and Schneider, 1999]. Therefore, additional challenges on how to express the semantics behind different kinds of location-dependent queries over indoor moving objects, and how to incorporate user preferences and other semantics into the query model are also of interest in this context (Figure 2). This requires the design of a query grammar that supports navigation-related queries, and which takes into consideration other properties such as the hierarchical layout of the indoor space for presentation of query results as well as for query processing purposes.

Research objectives

The research presented in this thesis studies location- and context-aware services and queries in indoor environments, with a special focus on navigation-related queries. A unique combination of challenges arises, as the proposal must be able to represent different kinds of location-dependent queries in a flexible manner, and to take into account additional contextual information, time-dependency, and the hierarchical layout of the indoor environment. Some of the challenges that are presented and addressed in this thesis are covered within different fields of research. The major research fields covered and/or related to this thesis are illustrated in Figure 3. Research on context-aware navigation in indoor mobile environments inherits challenges of both research fields: *Mobile and ubiquitous computing*, and *location-based services*. Challenges in knowledge representation and management are mainly addressed in this thesis. Particularly, *indoor spatial data models, context-awareness, query languages, and query processing* parts are presented along the contributions of this thesis from Chapter 4 to Chapter 7.

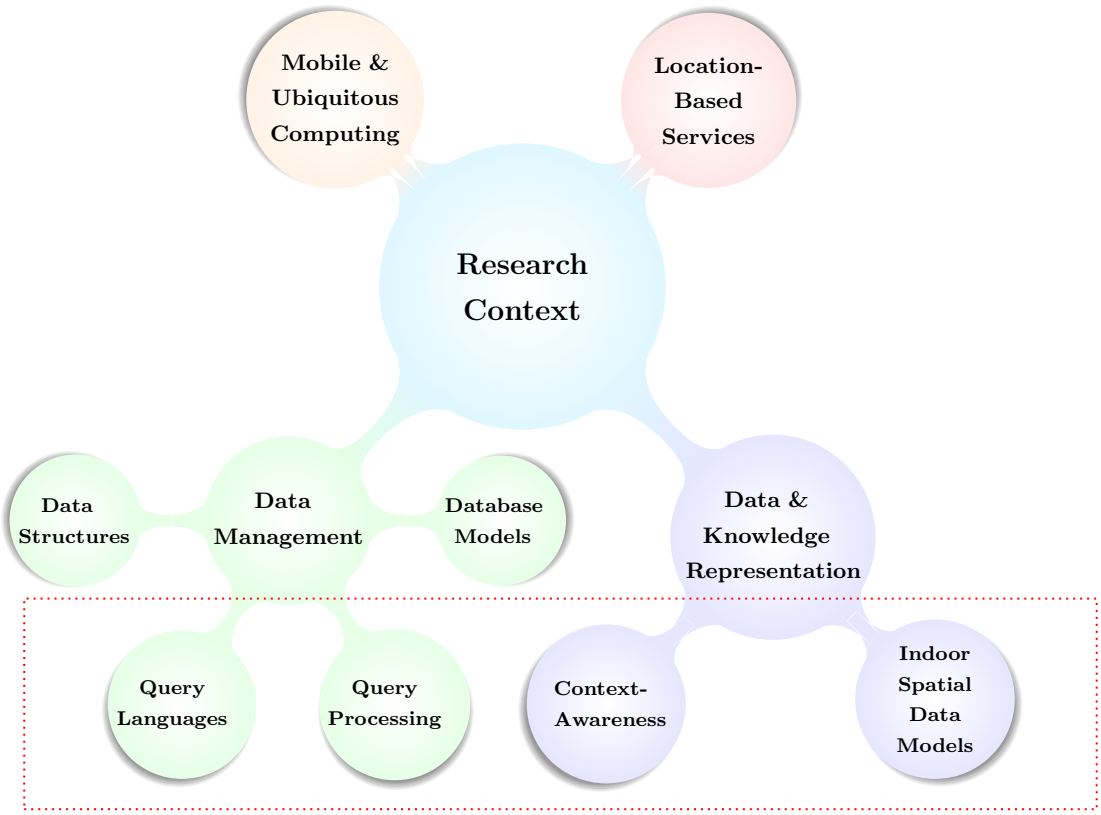


Figure 3: Major fields covered and/or related to this thesis

The main objective of this work is to investigate:

"A data and knowledge representation of indoor mobile environments incorporated with user-centred context-awareness, and to address data management issues in indoor context-aware navigation systems. The underlying principles that represent a context-aware indoor data model are introduced, as well as a query language and algorithms for the continuous location-dependent query processing in indoor environments."

To achieve these goals we propose the following approach:

1. To model and design a hierarchical and context-dependent indoor spatial and feature-based data model that can be viewed as a tree structure in which location information is represented at different levels of abstraction. This hierarchical design alleviates performance and scalability issues in location-dependent query processing. In addition, time-dependent functions that compute network distances and travel times are introduced. Furthermore, a classification of user profiles is presented in order to perform an offline filtering of the multi-level data model, thus reducing the amount of data that need to be processed in real-time.

2. To design a generic architecture used for the continuous processing of location-dependent queries in indoor environments. Navigation-related queries are processed in accordance with this architecture, and are continually executed while the request is not explicitly cancelled by the user. Moreover, a query grammar is also presented to express location-dependent queries. This grammar supports navigation queries and incorporates some other preferences and semantics in the query model. It also supports the hierarchical data model by using the concept of locations granules to represent different levels of abstraction.

3. To develop algorithms for the continuous processing of location-dependent queries. Particularly, algorithms for hierarchical path searches and range queries applied to both static and moving objects are introduced. Those algorithms take advantage of the hierarchical data model of the indoor environment, and apply an incremental approach in order to efficiently execute continuous location-dependent queries, thus avoiding solving each search problem independently from scratch.

4. To design and implement a database extension based on the open source DBMS PostgreSQL. This prototype handles continuous path searches and range queries on top of the hierarchical network-based indoor data model. The main parts of the prototype developed are: (i) a hierarchical network-based data model of indoor environments; (ii) operators and location-dependent constraints introduced in the query grammar; and (iii) algorithms to process continuous location-dependent queries over moving objects.

Clearly the scope of this thesis does not cover all issues and challenges that face the design and development of context-aware indoor navigation systems. Although many challenges regarding recent techniques for indoor positioning are of great interest for context-aware indoor navigation, and are strongly related to our research field, they are not further addressed in detail here. Indeed, the goal pursued in this work is to abstract our approach from specific positioning techniques as far as possible, while considering that a hierarchical data model can be suited to different kinds of positioning techniques (cf., Chapter 1 for more details).

Additional research fields that cover issues on database models and challenging problems in data structures, and indexing methods are discussed in Chapters 3 and 7, so that technical decisions are made for the prototype implementation, without being involved into those research challenges.

Thesis outline

This thesis is organized into three parts and seven chapters. The first part establishes a common ground for the development of a flexible and efficient context-aware indoor navigation system. The second part acts as a literature review of indoor spatial knowledge representations and location-dependent query processing techniques, respectively. The third part describes our methodology by presenting the main contributions in four different chapters. The structure of the thesis is organized as follows:

Chapter 1 presents the basic concepts of context-aware indoor navigation, and introduces preliminary recommendations for the development of indoor spatial data models from a context-aware systems perspective. Moreover, the role of contextual information is discussed, and how it can be incorporated into the data model. From a data management perspective, a set of requirements are also discussed with the aim of designing a real-time system for ambient intelligence environments.

Chapter 2 surveys indoor spatial data models developed for research fields ranging from mobile robot mapping to indoor location-based services (LBS), and most recently to context-aware navigation services applied to indoor environments. A taxonomy of models is presented and assessed with the aim of providing a flexible spatial data model for navigation purposes, and by taking into account the contextual dimensions.

Chapter 3 discusses related work on location-dependent query processing in both outdoor and indoor mobile environments. Different models and systems for handling moving objects in databases are reviewed. A special focus on the continuous processing of navigation-related queries, namely path, range, and nearest neighbour queries, is then made by considering the algorithmic aspects and

the different query processing techniques. Moreover, languages for querying location-dependent and other contextual data are also presented.

Chapter 4 introduces a hierarchical and context-dependent data model that represents an indoor space at different levels of abstraction. Other components such as the *feature component*, which represents mobile persons and objects of interest, and the *action component* are also presented. The management of user profiles and real-time events are further highlighted.

Chapter 5 first proposes a generic architecture for the continuous processing of location-dependent queries, and describes the main components considered in the architecture, and then discusses some data distribution issues. Secondly, the semantics of a continuous-based query language are introduced, which incorporates navigation-related queries along with some other preferences in the query model. Finally, some motivating sample queries are illustrated.

Chapter 6 presents several continuous processing algorithms for handling hierarchical path searches and range queries over mobile objects in indoor environments. The principles of an incremental and hierarchical-based approach are introduced for both kinds of queries. In addition, descriptions along with pseudocodes of the corresponding algorithms are provided.

Chapter 7 discusses the design and implementation of a database extension based on the open source DBMS PostgreSQL. This prototype handles continuous path searches and range queries on top of the hierarchical network-based data model of the indoor environment. The main parts of the prototype developed are presented. Results of the experiments, that have been conducted to investigate the scalability and performance with respect to the intrinsic properties of the proposed solutions, are also reported.

A summary of the contributions presented in this thesis is finally given, and some perspectives are drawn in order to extend the scope of the approach that has been developed by our research.

Part I

CONTEXT-AWARE INDOOR NAVIGATION

CHAPTER 1

Requirements for Context-Aware Indoor Navigation Systems

Contents

The first part of this dissertation includes one chapter that lays the groundwork for the thesis by examining different aspects and key building blocks for developing a context-aware indoor navigation system. This chapter first discusses the role of context-awareness in mobile computing. Recommendations for designing an efficient and flexible indoor spatial data model are then introduced, based on *service-oriented* and *efficiency-related* requirements. Furthermore, data management issues that face the design and development of context-aware indoor navigation systems are presented.

This chapter is organized as follows. Section 1.1 highlights the role of context-awareness in mobile information systems. Section 1.2 introduces a methodological framework for the design of an indoor spatial data model, while Section 1.3 presents challenges in data management from a query processing perspective. Finally, Sections 1.4 and 1.5 discuss general observations about current indoor navigation systems and draw some preliminary conclusions, respectively.

1.1 On the role of context in mobile computing

Indoor navigation services are set to become among the ubiquitous services that will make our living and working environments intelligent. Context-aware indoor navigation systems inherit characteristics and challenges of both mobile information systems, and location-based services. An additional component that is essential in such systems is the presence and the incorporation of context-awareness.

Mobile technologies have made research on the *lifelog*[1] and *user-centric contextual information* feasible. Diverse kinds of services delivered to the users can be provided by enabling real-time integration of contextual dimensions. Indeed, various sensors in mobile devices collect information on our daily life. This has triggered in-depth investigation on the *context-awareness* research field [Dey and Abowd, 1999; Dey, 2001; Filho et al., 2010]. The notion of context has been recognised as a key element in the development of mobile information systems [Chen and Kotz, 2000]. Context is literally described by the Oxford Dictionary as:

"The circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood".

In mobile computing, contextual information can be defined as: *Any information that is gathered and can be used to enrich the knowledge about the user's state, his/her physical surroundings, and capabilities of his/her mobile device(s)* [Dey and Abowd, 1999; Schmidt et al., 1999]. Context varies according to application constraints, and by taking into account the way users act in the environment, as well as the interfaces to interact with. Two generic concepts with three main categories have been introduced as super-classes that encapsulate contextual dimensions [Satyanarayanan, 2001; Baldauf et al., 2007; Petit, 2010] - Figure 1.1:

- The *context of use* includes two main categories:

  (1) *user-centred dimensions* of context such as the user profile, his/her preferences as well

[1] Lifelogging services keep a complete record of users' day-to-day activities (i.e., *lifestream*), thanks to small wearable gadgets or computers that captures large chunks of the user's daily life.

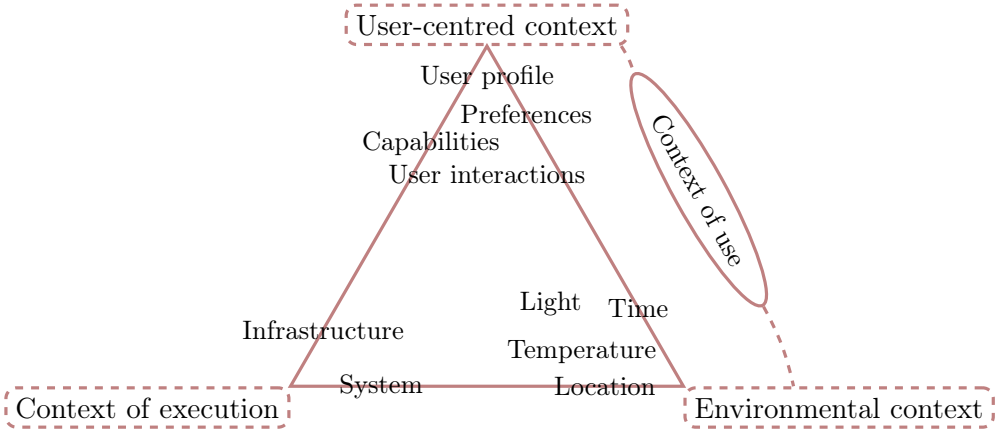


Figure 1.1: Classification of contextual dimensions

as his/her physical/cognitive capability, and the user-system interactions (i.e., input data manipulation through the user interface and output communication operations).

(2) the *environmental context* which refers to the parameters that influence the user (e.g., location, time, temperature, light, etc.).

- The *context of execution* models the behaviour of an information system and encompasses: (i) the infrastructure dimension and the topological distribution of the system components and (ii) the system dimension that evaluates resource utilization (e.g., memory, processor and network) of the system components and capabilities of the user's mobile device(s).

This chapter first discusses service-oriented and efficiency-related requirements in order (1) to reconsider current indoor spatial data models from a context-aware systems perspective so that a reference data model can be proposed; (2) to take into consideration dynamic properties and activity-oriented interactions of moving objects with their physical surroundings, and (3) to examine the role played by smart devices and ubiquitous positioning sensors. Secondly, the need for an adaptive query processing paradigm as well as for new ways to express and evaluate continuous location-dependent queries in indoor environments is highlighted.

1.2 Challenges in context-dependent indoor data models

Whether location information should be handled as any other contextual information or managed differently is a key issue. A better understanding of location information and the relationships that might exist among spatial entities, either acting or located in the environment, should be taken into account. Therefore, the integration of an indoor spatial data model into a context-aware system is

considered as a key building block for designing advanced navigation services.

This section presents a classification of the requirements needed to design a context-dependent spatial model into two categories [Afyouni et al., 2012]: *service-oriented* and *efficiency-related* requirements. The first group supports real-time and delayed services, and includes: (i) *localisation*, (ii) *navigation*, (iii) *location-aware communication*, (iv) *activity-oriented interaction*, and (v) *simulation and behavioural analyses*. The second group examines efficiency issues and includes: (i) *modelling effort*, (ii) *flexibility*, and (iii) *performance and scalability*. Those are generic and application-independent requirements for the development of a reference spatial data model, and are hereafter used to assess existing approaches.

1.2.1 Service-oriented requirements

A context-dependent spatial model is assumed to represent the locations of objects of interest appropriately, and with additional semantic descriptions so that advanced services can be provided. It also needs to support navigational services a user is likely to ask. Potential communication between located entities should also be supplied. Moreover, a user navigating an indoor space should be able to interact with his/her physical surroundings. Other deferred services can also be offered by the model thus enabling a better understanding of users' behaviours.

1.2.1.1 Localisation

Several studies have been performed in order to design and build positioning systems in GPS-less indoor environments [Kolodziej and Hjelm, 2006]. Indeed, indoor spaces are constrained by the architectural components such as doors, floors, corridors, and walls. On the one hand, these constrained environments provide meaningful location information at the logical and topological levels, but on the other hand, the task of achieving accurate localisation is not always straightforward.

Spatial data representation

Two different representations of location information can be delivered according to either a geometric or a symbolic representation of space:

- **Geometric information** gives a quantitative representation of moving objects in the form of coordinates (e.g., Cartesian or latitude-longitude-altitude) that are delivered according to a

given coordinate system. Whereas systems like Global Positioning System (GPS), used outdoors, provides geometric coordinates with respect to a *global reference system*, others such as the Active Bat system [Ward et al., 1997] and MIT Cricket [Priyantha et al., 2000] use ultrasonic technology and provide three-dimensional positions with respect to *local coordinate systems*. Those are developed based on a set of local landmarks (e.g., beacons) that are distributed in an indoor space. Topological relationships like *"overlap"*, *"inside"*, *"intersect"*, and *"disjoint"* can be inferred, thus enabling an interpretation at a higher level of abstraction. As an example of overlapping relationships, floors can be shared between several wings within a building. In such a situation, floors overlap with wings, while a room may belong to a floor and a wing.

- **Symbolic information** provides qualitative human-readable descriptions about moving objects based on structural entities and/or points of interest (e.g., room or floor identifier, building name, etc.). For instance, the Active Badge system handles values that represent the symbolic identifiers of fixed IR sensors [Want et al., 1992]. In contrast to geometric information, symbolic descriptions allow to explicitly model topological relations (e.g., spatial containment) between entities in the environment. This can be done by means of symbolic spatial models such as set-based or graph-based models (see Chapter 2). Symbolic descriptions enable spatial and semantic reasoning at an abstract level, thus favouring interaction between spatial entities and within space.

Sensor-based positioning

Nowadays, the techniques currently available for indoor positioning range from radio-based technologies (WLAN, RFID, and Bluetooth) to non-radio technologies (Infrared and Ultrasound) to inertial navigation systems (INS) [Kolodziej and Hjelm, 2006]. One challenge for all these techniques is to achieve a precise positioning of a given mobile user, together with some context-aware data that integrate the location of predefined sensors, and more important, a sense of the environment in which users are acting. This leads us to not only locate the users in the environment, but also the sensors available, and the spatial structures underneath. As these systems should be real-time based, in order to take into account the fact that mobile users behave in the environment, there is a general agreement on the fact that there is so far no perfect technique which is capable of reaching the desired accuracy continuously.

Recent approaches have adopted a hybrid approach combining several positioning technologies. For instance, Ray et al. [2010] proposed an infrastructure-less positioning system for both indoor and outdoor environments. This system combines GPS receivers with inertial sensors (MEMS) and an indoor map-matching algorithm. The interest of MEMS sensors is that they usually provide acceleration and angular velocity. The principle behind this approach is that user location data

have to be provided by the sensors continuously. When GPS data are not available, this being the common case in indoor environments, inertial sensors give alternative location data. However, inertial sensors are error-prone that causes sharp performance degradation. This implies to integrate additional rectification processes.

Similar techniques have been introduced elsewhere still for the design of ubiquitous location-based systems. In particular, when using WLAN or RFID positioning techniques, an integrated approach combines one (or both) of them with an inertial navigation system (INS) [Evennou and Marx, 2006; Fu and Retscher, 2009]. On the one hand, INS helps to overcome the lack of coverage of radio-frequency (RF) signals when an insufficient number of WLAN antennas or active RFID tags are deployed in an indoor environment. On the other hand, when both technologies (INS/Wi-Fi or INS/RFID) are available, a more precise position measurement can be produced by applying an appropriate multi-sensor data fusion technique [Retscher and Fu, 2007]. Therefore, appropriate multi-sensor data fusion techniques and map matching algorithms need to be used in the filtering process [Ray et al., 2010].

Consequently, a spatial model should be capable of representing the coordinate system by which the location information is expressed and, when necessary, transforming that location information in order to get all sensor data in a common format.

1.2.1.2 Context-aware, adaptive navigation

Navigation in an environment can be defined as a scheduled and goal-oriented movement made by humans or robots [Montello, 2005]. While navigating, a user may be technically assisted by sensory devices embedded in the environment that provide relevant information to maintain orientation, and other suggestions to encourage him or her to interact within the environment.

As an extension of typical navigation tasks, contextual dimensions need to be integrated into querying tasks, thus offering opportunities to develop advanced services. A context-aware navigation task is carried out in two phases [Champandard, 2003]: the static phase, which is generally known as *"path planning"*, encompasses a multi-criteria path selection process that generates an unbroken path from the current location to the destination. This process is context-sensitive as it aggregates multiple criteria (e.g., user preferences and capabilities, distance, time, etc.) passed as function parameters to evaluate the cost value of each step and then to select the most appropriate path, that is, the *optimal* path that allows, for instance, to reach a destination while avoiding threats [Disser et al., 2008]. The dynamic phase lies in a dynamic framework that implements event-triggered controllers needed to monitor the user's progression in order to avoid deviations from the planned

path.

Although very few works have discussed the integration of such a dynamic framework, researchers agree on some general requirements [Delling et al., 2009]. First, this framework should comprise a next-step selection algorithm that keeps continuous track of the user's position and tries to recover from deviations by providing additional information. Secondly, the framework should adapt the predetermined path if it detects any significant event that may affect the user's movement. In such a situation, a path to the nearest emergency exit should, for instance, be recomputed. Recent studies have proposed algorithms for shortest and/or fastest path searches with improved tracking strategies [Berger et al., 2010; Sun et al., 2009; Xu et al., 2010]. The main focus of these algorithms is to keep real-time tracking of moving objects. However, each of them deals with either time or distance constraints without incorporating other elements such as user preferences or events that may significantly influence the answer.

1.2.1.3 Location-aware communication

In contrast to internet telecommunications which enable communication between computing devices based on IP or MAC addresses, communication among objects in smart environments is often based on their location [Beigl et al., 2002]. Location-aware communication models in distributed systems can be classified according to who initiates information exchange and how information is then forwarded to the specified receiver [Aitenbichler, 2008]. One can make a distinction between direct (i.e., point-to-point) and indirect communications [Baldoni and Virgillito, 2005]. As an example of indirect communication, a provider can disseminate information about events to invoke *remote callback methods* from potential subscribers. Similarly, a consumer can request information from a known (i.e., direct message exchange) or anonymous (i.e., indirect message exchange) provider. Moreover, collaborative activities can take place between communicating objects working together to achieve some common goals.

Although spatial models do not affect users' physical abilities to communicate, the selection of a specific spatial model has a direct effect on the quality and format of data exchanged between users and other entities located in space. Moreover, the way a user perceives its ability to communicate with other entities may be significantly disturbed due to inadequate representation of space. For instance, a fined-grained model can represent a sensor range more accurately than a coarse representation of space. Thus, the user can be aware of the exact communication range for a given sensor.

A dynamic and symbolic location model that supports location-aware communication among rigid entities (i.e., sensors and users) in smart spaces has been suggested in Satoh [2007], which gives

an example of what can be achieved. Rigid entities and places are represented as components in a hierarchy of symbolic labels (i.e., user-room-floor-building hierarchy) based on the containment relationship. Each component in the hierarchy can act as a service provider, a service consumer or both. Different types of communication can then be established between these components depending on their location in the hierarchy. An example of the dynamicity of this model might be the case where users navigate between places (e.g., rooms). A component migration mechanism is then used to update the hierarchical structure of the components and thus keeps track of users' movements. Two types of communication are then established between these components depending on their location: *vertical communication* that supports the publish/subscribe approach; this means that events are published by parent components to address their children. Child components that are interested in those events need to be subscribed so that they can receive runtime notifications. *Horizontal communication* encompasses communication between same-level components of the hierarchy (i.e., components that belong to the same parent in the hierarchy). As an example, interactions between a user and its computing devices (e.g., stationary computer) within a room can be represented, both considered as components that are contained in a parent component which is, in this case, the room.

1.2.1.4 Activity-oriented interactions

A context-dependent model should also support human activities with the aim at achieving some objectives. An activity can be defined as a collection of goal-oriented and context-dependent actions an entity can perform [Kofod-petersen and Cassens, 2006; Wu and Fan, 2009]. Actions comprise a sequence of location movements, interactions with other neighbouring entities and artefacts, and requests for some services in order to achieve a predefined goal. An activity can also be made of a set of primitive activities or be part of a larger collective activity [Christensen and Bardram, 2002]. Activity theory has focused attention on the usefulness of spatial models in the design process to enrich consciousness and interaction within space [Kaptelinin and Nardi, 2009; Nardi, 1996].

The number of artefacts has increased and been distributed into ubiquitous environments. Artefacts can be physical (e.g., chair, door, heating, etc.) or virtual (e.g., 2D/3D image of a physical artefact, digital user interface, recommendation/information, etc.) [Pederson, 1999]. Physical artefacts can also be augmented with various kinds of sensors or tags (e.g., RFID) so that they can perceive the environment and provide additional information [Rivera et al., 2011]. Those are referred to as *digital* or *sentient* artefacts [Beigl et al., 2002; Kawsar et al., 2007]. Spatial representations serve as a mediator to relay relevant information to humans about artefacts in their surrounding. This allows users, who are engaged in a certain activity, for both collecting knowledge and understanding physical surroundings, and furthermore, for reconfiguring and manipulating physical/virtual artefacts

to anticipate or produce changes in the environment. Consequently, embodied interaction defined by [Dourish, 2004] can take place by means of spatial reconfigurations that may influence the context by affecting existing activities and/or initiating new ones. A main challenge that designers face and should consider is to efficiently represent artefacts of interest located in the environment. Unfortunately, most of existing indoor spatial models are not designed for that purpose and thus do not supply interaction with these artefacts and the tasks they might participate in.

1.2.1.5 Spatial & behavioural analyses

An essential issue in the development of analyses and simulations lies in identifying an appropriate spatial representation with respect to the phenomenon or behaviour being explored. Several typical scenarios for planning purposes are introduced in [Li et al., 2010]. These scenarios present application-dependent constraints, so each of them needs an appropriate level of granularity. Some examples of spatial analyses and scenarios applied on a fine-grained spatial model are [Li et al., 2010]:

- Route analysis scenarios that aim at finding shortest paths between two given locations or all pairs shortest paths applied on a given floor architecture and with a given data structure [Werner et al., 2000; Zhang et al., 2011].
- Diffusion analysis defined as a dynamic process where the spatio-temporal evolution and extent of a phenomenon within an indoor space are explored [Batty et al., 1999].
- Centrality measures that characterize the architectural design and the spatial distribution of objects of interest in a built environment [Bonacich, 1987], and are largely applied by space syntax studies [Jiang and Claramunt, 2004; Kim and Jun, 2008; Kostakos, 2009].

The granularity parameter is an important aspect of these analyses that enables the exploration and understanding of spatial data. A fine granularity can certainly reflect the indoor space delicately, and is especially needed when simulating a physical process or when considering fine-grained tracking of human mobility to get accurate understanding of the environment. In contrast, topological analyses often require a coarser granularity to support adjacency and connectivity relations between spatial units (e.g., room), and thus deriving a more abstract view of the topology of a complex indoor space. In particular, topological analyses have been largely developed and applied by space syntax studies to evaluate human-environment interactions and to examine different spatial and structural configurations [Jiang and Claramunt, 2000; Turner and Penn, 2002].

Behavioural analyses are, on the other hand, of special interest in context-aware computing, and have been recently developed in numerous fields for activity-recognition purposes [Albinali, 2008].

Activity-recognition systems can support different application scenarios in many areas including safety control, medical healthcare, and other monitoring activities in virtual environments that aim at customizing the system to end-users' situation or context [Bardram, 2005; Christensen and Bardram, 2002]. This helps, for instance, to reduce occurrence of hazardous situations by monitoring and correcting human error during the execution of critical tasks (e.g., administering medication in healthcare, ensuring proper execution of tasks in safety applications) or to exploit user interactions with an application (e.g., mouse clicks) to infer his/her activities as suggested in [Albinali and Gniady, 2006]. Moreover, knowing the locations and preferences of the users in an indoor context-aware navigation system can lead to optimise the knowledge about the user's behaviour in the considered scenario (e.g., inferring the most bought products in a supermarket by the loyal users) [Porros Pérez et al., 2011]. Activity monitoring combined with the delivery of behaviour-related context-aware reminders for elderly persons have been also discussed in [Zhang et al., 2011].

1.2.2 Efficiency-related requirements

An offline requirement regarding the modelling effort needs to be considered in the evaluation process. A spatial model should also be flexible as much as possible so that a wide range of applications can be applied. Efficiency is closely related to the performance and scalability of the system being developed. Those requirements are discussed in the following subsections.

1.2.2.1 Modelling effort

The modelling effort can be evaluated by the cost and complexity of the model design effort. Some modelling techniques are sophisticated and take considerable effort [Thrun, 2003]. Some others are closely dependent on objects within space and need to be periodically maintained or even to be rebuilt from scratch if these objects change their position or shape. Recently, some authors have discussed several methods for automated construction that minimize manual intervention by designers during the modelling phase [Stoffel, 2009].

1.2.2.2 Flexibility

A flexible spatial model should support a large spectrum of applications that can be developed at different levels of abstraction, as well as different kinds of positioning sensors that might coexist to achieve better accuracy. This enables human reasoning about space, robot-based activities, and even sensor-based and object-oriented interactions within the environment. In brief, may a simple unmixed spatial model assist and deal with a very diverse intelligent navigation tasks efficiently? As

will be shown later (cf. Section 2.3), it might be difficult to find a single solution that meets these seemingly contradictory requirements. Consequently, we propose to assess current models based on the range of queries and services that can be supported.

1.2.2.3 Performance and scalability

A context-aware indoor navigation system requires to efficiently execute users' queries such as shortest paths between two given locations, k-nearest neighbour objects or all accessible locations starting from their current position [Lyardet et al., 2008]. When considering contextual information, performance is no longer restricted to simply deriving position queries or distance functions, but some other functionalities are required, thus leading to a significant increase in workload. Furthermore, dynamic updates of the spatial data model stored on mobile devices should be periodically performed in order to maintain accuracy and quality of the stored data. For instance, one challenge facing spatial model designers is the cost of updating the location of every moving object continuously. Existing approaches try to overcome this problem by performing periodic, deferred (i.e., demand-based) or immediate (i.e., event-driven) update queries to keep an incremental view maintenance of the database [Iwerks, 2004].

A related approach focuses on real-time map updates on mobile devices by considering the use of a spatial model with different levels of detail [Kang et al., 2007]. Updates in this approach are handled at the basic level on the server side, and then an update propagation process through the other levels of the spatial model is continuously performed at the client side. This framework considers the use of a spatial model with different levels of detail. Hence, it first handles updates at the basic level on the server side, and then manages the update propagation process through the other description levels of the spatial model. Further, it transfers update messages that contain updated spatial objects to subscribed clients thus replacing original objects with matching identifiers.

The scalability requirement denotes the ability of a spatial model to scale up to a large indoor environment while keeping tolerable resource consumption, which mainly affects the memory storage capacity and the behaviour of processing workloads. In context-aware applications, heterogeneous mobile devices as well as a huge number of embedded sensors have to be handled in an efficient and scalable way. Therefore, there is a need to establish a mechanism that eliminates superfluous information to reduce processing and communication costs, so that the system can handle a higher number of real-time queries.

1.3 Data management issues in location-aware services and queries

While representing location information is a key issue for building context-aware indoor navigation systems, other challenges regarding the management of *location-dependent data* and the continuous processing of navigation-related queries are also of key interest for designing an efficient and flexible enough indoor navigation system. This section covers major challenges in that area by first discussing location-based services and queries in an indoor context. Secondly, issues related to the modelling and querying of moving objects and to the continuous query processing paradigms are presented. Finally, different aspects of designing languages for querying location-dependent data with semantically enriched operators are also highlighted.

1.3.1 Location-based services and queries

A wide range of location-aware services can be applied to indoor environments. Examples of such services include continuous crowd monitoring within a given area, location-based alerts (e.g., continuously send E-coupons to all customers within 200 metres of my store), report any congestion in an emergency situation, and location-based friend finder (e.g., let me know if I am near to a restaurant while any of my friends are there). The main goal of these services is to provide the user with the ability to interact with his/her physical surroundings in order to achieve some objectives. Location-aware, user-centred services can be distinguished according to two modes of data access: *pull* mode and *push* mode [Schiller and Voisard, 2004], which are defined as follows:

- Pull-based location-aware services comprise requests triggered by the user with the aim of pulling some location-dependent information from the service provider [Ilarri et al., 2010; Zhang et al., 2003].
- Conversely, push-based services are initiated by the service provider without having been requested by the user. The service provider takes into account location information of subscribed users to trigger alerts or contextual advertising, and push them to the user's device [Friedman, 2004; Navas and Imielinski, 1997].

Push-based services: Several push-based, real-time services that deliver valuable information and allow for communication have been reviewed in [Kolodziej and Hjelm, 2006; Lee et al., 2005; Navas and Imielinski, 1997]. This includes, for instance, location-based instant messaging in which outgoing messages are not just stamped by the local time-zone of the sender but also by the sender's current location [Burak and Sharon, 2004; Friedman, 2004]. Similarly, a user can instead manually

specify a certain location specification so that other users whose specifications meet those given by the sender will receive the message [Hakkila and Hexel, 2003]. Other kinds of push-based services implement the principles of *geocast* messaging in order to provide location-based advertisements and alerts [Basagni et al., 1999; Navas and Imielinski, 1997]. Geocast messaging can be described as a location-based multicast where messages are delivered to users located in a specific area instead of those subscribed to a given group.

In particular, location-based alerts are not necessarily time-related and are generally used in the case of an emergency to warn people to go around a dangerous zone. It might also be useful, for example, to remind a user navigating a shopping centre to buy some food or fruit stock when he or she is located next to a supermarket. Location-based advertisements (also called proximity-triggered advertisements) generally target nearby consumers to provide them with information about stores' offers, discount coupons, etc. [Aalto et al., 2004]. This kind of push-based services should provide users with a subscription-based mechanism that allows for relevant and non-intrusive advertising.

Pull-based services: Location-dependent queries are considered as typical examples of pull-based services, and are needed in any context-aware navigation system [Ilarri et al., 2010; Zhang et al., 2003]. Queries are location-dependent, meaning that any change of the locations of objects that are involved in the query may significantly affect the answer. For example, if a user asks to find out his/her friends within a range of 100 meters while navigating a shopping centre, this answer will depend on the user's current position as well as on the location of the nearest friends. Some particularly relevant location-dependent queries in indoor contexts are briefly described as follows:

1. *Position queries* determine the locations of mobile and static objects, and are processed according to either a geometric or symbolic model of space. Location-dependent queries cannot be carried out without up-to-date information on the locations of objects of interest [Becker and Durr, 2005].

2. *Path queries* encompass all the queries that directly help the users to find and reach points of interest, by providing them with navigational information while optimizing some criteria such as the traversed distance or the travel time. Examples of such queries are: (i) discovering *optimal* paths to a nearest point of interest (e.g., landmark, place), and (ii) planning a path to a destination.

3. *Range queries* find and retrieve objects or places of interest within a user-specified range or area [Wu et al., 2006]. Those queries support navigation by continuously updating relevant details according to the users' movements. Ranges may be characterized by a circular or rectangular-shaped window in which objects of interest must be located.

4. *k Nearest Neighbour (kNN) queries* search for the k closest qualifying objects to the moving user with respect to his/her current location [Tao et al., 2002; Zhang et al., 2003]. As opposed to range queries, kNN queries are range-independent, except in the case of *constrained nearest neighbour queries* [Ferhatosmanoglu et al., 2001], where the search is constrained to a given region. The user initiates a request by specifying some characteristics about objects of interest, so that the k closest objects whose specifications meet these characteristics are retrieved (e.g., the closest available colour printer or the k nearest friends).

As those queries are time-sensitive and location-dependent, they may be valid only for a given period of time and within a given area (i.e., data returned are only spatio-temporally valid). Therefore, they are expected to be processed as *continuous queries* [Terry et al., 1992], which means that the system should continually keep the answers up-to-date over a certain period of time. The following section takes a closer look at the continuous query processing paradigm, and highlights different issues related to the management and execution of those queries.

1.3.2 Continuous and adaptive query processing paradigms

As mentioned in Section 1.2.1.2, a context-aware navigation task needs to establish a multi-criteria path selection process along with a continuous real-time mechanism that allows for route adaptation depending on the movement of the objects of interest. Similarly, both pull-based and push-based, real-time services require the continuous monitoring of moving objects so that adaptive answers to users' requests can be provided. Therefore, an adaptive query processing paradigm as well as new methods for the continuous processing of location-dependent queries over indoor moving objects are required to achieve those tasks. This paradigm should be generic enough to support different kinds of queries, but without loss of efficiency.

One important goal of this thesis is to process different kinds of location-dependent queries in a continuous manner, and to take into account additional contextual information, such as time-dependency and user profiles, and the hierarchical layout of the indoor environment.

Different criteria can be used in order to classify continuous location-dependent queries. Particularly, *query time* and *object mutability* criteria are considered as follows:

- *According to the query time:* whether it refers to the past (*"historical spatio-temporal queries"*), present (called *"instantaneous or now queries"*) or future states (referred to as *"future or predictive queries"*) [Mokbel, 2005]. An example of continuous historical query is: "*Continuously*

extract the locations of a certain moving object in the last 20 minutes", and of a continuous now query: "*What are the k nearest friends to my current location*". A continuous predictive query may have a style similar to this example: "*Alert me if a certain person is going to enter my current room in the next 5 minutes*".

- *According to the mutability of both the query point and the objects of interests:* Continuous location-dependent queries may have a static or dynamic starting *query point*[2] [Gratsias et al., 2005; Vargas-Solar et al., 2010]. Similarly, a continuous location-dependent query can be applied on static or dynamic data, depending on whether the objects that are the target of the query are moving or not; those are classified into three categories (Table 1.1): (1) *Stationary queries over moving objects* where the query point is stationary, while objects of interest are moving. For example, one could ask to *"retrieve the identifiers of persons in a given room"*; this implies monitoring the moving objects within a given "stationary" area. (2) *Moving queries over stationary objects* in which the query point is moving, while objects of interest are static. An example of this category is the *guide-me services*, or one can ask to: *"retrieve all the communicating entities (fixed sensors) accessible at a distance smaller than 100 meters of the user identified by userID"*. (3) *Moving queries over moving objects* where both the query point and objects are moving. For example: *"find the* shortest *route from person userID1 to person userID2"*.

Data object / **Query point**	**Stationary**	**Moving**
Stationary	N/A[3]	Category 1 (e.g., monitor persons in a given room)
Moving	Category 2 (e.g., *guide-me* to a destination)	Category 3 (e.g., *get-together* services or track a moving object)

Table 1.1: Taxonomy of continuous location-dependent queries

Different challenges regarding the management of moving objects and the continuous evaluation of location-dependent queries are discussed in Sections 1.3.2.1 and Section 1.3.2.2.

[2]The *query point*, also referred to as the *focal point* in [Gedik and Liu, 2006] and the *reference object* in [Ilarri et al., 2006a], is the point where a given location-dependent query starts (e.g., for a range query, this indicates the centre of the range specified by the user).

[3]This classification of continuous queries is not applicable to *stationary queries over stationary objects*, as both the query point and the target objects are static.

1.3.2.1 Managing moving objects

Efficient and appropriate management of static and dynamic data is a key issue for processing time-sensitive, location-dependent queries. Particularly, mobile environments have a significant impact on the system design, especially for modelling and querying moving objects. This design should allow to model the movements of entities and to ask queries about such movements. In some cases, only time-dependent locations need to be managed, leading to the moving point abstraction, while in other cases, a time-dependent shape or extent is also of interest which is referred to as a moving polygon. Conventional systems do not provide facilities for update operations in a mobile computing environment.

Two major trends can be identified for modelling and querying moving objects in databases, which have extended the classic database paradigm in order to achieve the so-called *"mobile database"* technology: *Moving Object Databases (MOD)* [Wolfson et al., 1999b; Pelekis et al., 2004; Güting and Schneider, 2005], and *Data Stream Management Systems (DSMS)* [Chandrasekaran et al., 2003; Arasu et al., 2004a].

Moving Object Databases (MOD): Research on moving object databases has been classified by Güting into two main categories:

- The *location management* approach introduced by Wolfson and his colleagues in a series of papers [Sistla et al., 1997; Wolfson et al., 1998, 1999a,b], in which the current and (near) future movement of entities are represented. They have designed an interesting data model called MOST in which *dynamic attributes* are introduced in order to represent the highly changing properties of the moving objects (i.e., locations) as *motion vectors* instead of moving points. Furthermore, the Future Temporal Logic (FTL) query language has been proposed to help expressing continuous queries by specifying temporal relationships between objects that are of interest to the current queries.

- The second major approach in moving object databases was explored by Güting and his colleagues, and was referred to as the *spatio-temporal database* perspective [Guting et al., 2000; Güting and Schneider, 2005; Güting et al., 2005, 2006]. This approach manages time-dependent geometries (i.e., continuously moving points, lines, and regions) in database in order to capture complete histories of movements of the entities of interest. An abstract data model that comprises spatio-temporal data types and a comprehensive set of operations, which forms an algebra representing moving objects, along with a related query and data manipulation language have been developed [Güting et al., 2006]. An Open Source and extensible database system

called SECONDO has been further implemented as a research prototype to study spatial and moving objects databases, network models, fuzzy spatial data types, and other optimization techniques [Güting et al., 2005].

Another important issue in mobile data management is to appropriately and efficiently index moving objects in databases. With the highly dynamic character of mobile location-aware environments, several attempts have been proposed with the aim of tuning traditional spatial indexing mechanisms, such as R-tree [Guttman, 1984], R+-tree [Sellis et al., 1987], R*-tree [Beckmann, 1990], and X-tree [Berchtold et al., 1996], in order to support frequent updates (e.g., *LUR-tree* [Kwon et al., 2002], FUR-tree [Xiong and Aref, 2006], and the *CTR-tree* [Cheng et al., 2005]).

The main idea behind the *Lazy Update R-tree (LUR-tree)* is to build a *minimum bounding rectangle (MBR)*, based on the moving object's speed vector, within which no action is taken other than updating the location. Once the object moves out of the MBR, either the object is deleted and then reinserted or the MBR is extended in case the object is still not so far from the MBR. The *Frequently Updated R-tree (FUR-tree)* extends the LUR-tree by performing enhanced bottom-up update methods to accommodate frequent updates of the moving objects. While both the LUR-tree and FUR-tree assume that moving objects are following known patterns, the *Change-Tolerant R-tree (CTR-tree)* does not put any restrictions on the object movement. Many other variants for *spatio-temporal access methods* have been reviewed in [Mokbel et al., 2003; Chen et al., 2008; Nguyen-Dinh et al., 2010].

Data Stream Management Systems (DSMS): The data stream technology focuses on continuously changing data that arrive as *streams* in real-time and in large (or even unbounded) amounts. For instance, GPS receivers and ambient sensors are continuously measuring specific parameters and characteristics of the environment; those have to be processed in real-time [Babcock et al., 2002; Golab and Özsu, 2003]. The management of data streams represents different challenges [Ilarri et al., 2010]:

1. Impossibility of managing and storing the whole history of the data streams; this has led to use the concept of *sliding windows* to limit the search scope, and to support processing queries in real-time [Ghanem et al., 2007; Hammad et al., 2003].

2. Supporting continuous queries which are typical in data stream applications. Those are not supported by default in traditional stored data sets [Arasu et al., 2004a]. An adaptive approach for query execution that requires dynamic query plans is essential for good performance, and for the continuous monitoring of the incoming data streams.

Several data stream management systems have been proposed to cope with these challenges and to address continuous queries, such as the *STREAM* prototype [Arasu et al., 2004a] , *Nile* [Hammad et al., 2004; Mokbel et al., 2005], and *TelegraphCQ* [Chandrasekaran et al., 2003].

From a data stream management perspective, [Mokbel, 2005] considers that these index structures are not well suited in highly dynamic environments, and their performance degrades dramatically for high update frequencies. Consequently, these authors have proposed a grid-based index structure for indexing both frequently updated moving objects and queries, along with an in-memory technique that limits the search scope to the object that are of interest to at least one active continuous query [Mokbel et al., 2005].

1.3.2.2 Continuous evaluation of location-dependent queries

Continuously updating queries implies a communication overhead and additional processing cost at the server side. Different approaches to process location-dependent queries in real-time mobile environments have been proposed [Cai et al., 2006; Mouratidis et al., 2006; Yuan and Schneider, 2010]. These approaches depend not only on the way moving objects are managed, but also on the types of query considered, on how the environment where those objects evolve is represented, on the assumptions made on the movement and patterns of the moving objects, and also on the amount of cooperation required from the moving objects, among others. For instance, some works focus only on one type of query [Yuan and Schneider, 2010]. Others deal only with stationary queries over moving objects [Cai et al., 2006; Mouratidis et al., 2006], or with moving objects following predictable trajectories [Wolfson et al., 1999a; Stojanovic et al., 2008]. Many variants of these queries are summarized in [Ilarri et al., 2010]. Mainly, three different approaches have been investigated in literature order to make repetitive evaluation of continuous queries more efficient:

1. **Update policies:** some works assume specific update policies that aim at abstracting any continuous query into a set of *snapshot queries* executed at different time instants. A concept of *valid region* or *valid time period* of the query result has been introduced to lessen this problem by determining a safe area or a time interval around the initial user position in which the result of the query is always valid [Zhang et al., 2003; Hu et al., 2005]. This can be computed by estimating the (maximum) velocity of the moving objects so that an extended range around them is built within which the result to the query is considered to be valid.

2. **Preprocessing:** precomputing the (or part of the) result is another way to reduce communication overhead and the number of evaluations of a continuous query. For instance, location-dependent searches can be abstracted to different levels of detail, such that answers to

specific queries are preprocessed at a higher level, and the resulting 'abstract' answers can be refined in real-time [Botea et al., 2004].

3. **Incremental processing:** some techniques employ an incremental approach in order to efficiently process continuous location-dependent queries, thus avoiding solving each search problem independently from scratch. This implies caching the previous query results either at the client or at the server side. Upon triggering of the new evaluation of the current query, the cached results are used to prune the search in order to obtain the new updated result of the query [Song et al., 2008; Sun et al., 2009].

Most work on location-dependent query processing has been developed for outdoor environments. However, indoor environments bring some special features and constraints that should be considered during query processing (i.e., the constraints that emerge from the architectural layout of space). Although many research studies have discussed location-dependent queries and location-based services, a few works have addressed the issue of incorporating contextual dimensions into query processing, particularly those related to *user-centric* and *environmental contexts*. [Mokbel and Levandoski, 2009] has recently highlighted these issues and discussed some of the challenges to be considered in order to carry out context-aware queries and services. Some of these challenges include (i) designing a spatial and user-oriented model, (ii) supporting multi-objective and possibly contradictory queries, and (iii) analysing efficiency and scalability when dealing with context-aware continuous queries.

1.3.3 Query languages for location-dependent queries

Further discussions on location-dependent data management have brought other research challenges related to what kind of query language can be provided to answer location-dependent queries [Sistla et al., 1997; Dunham and Kumar, 1998; Ren and Dunham, 2000; Güting and Schneider, 2005]. The query language is generally based on the concepts (i.e., represented as abstract data types) described in the database model, and allows asking requests in a straightforward manner and formulating predicates about those concepts. Several works related to moving object databases have presented query language extensions that support spatio-temporal data types and allows to formulate any kind of questions about such objects' movements, such as the FTL language [Sistla et al., 1997], and the approach developed by [Güting et al., 2006].

Such a query language should also support user preferences (e.g., optimization criteria for path planning), and other application-related constraints. Some semantically enriched query languages for path planning in outdoor environments have been proposed in [Booth et al., 2009; Mokhtari,

2011]. A query model for multimodal transportation systems has been presented in [Booth et al., 2009], which provides users with the ability to choose between different modes of transportation and applies spatio-temporal restrictions adapted to the user's preferences. Another approach based on fuzzy logic theory that helps identifying ambiguous and possibly contradictory preferences have been proposed in [Mokhtari, 2011]. However, the authors do not provide a mechanism for continuous location-dependent query processing.

A query language is closely related to the mobile environment in which objects evolve. This means that all concepts which represent the environment and/or related to the application should be integrated in the query grammar. Particularly, in the context of indoor navigation, a query language should improve the expressiveness of location-dependent queries by considering the hierarchical layout of the indoor environment. The approach presented in [Ilarri et al., 2011] covers the use of location granules in outdoor environments from both a query processing as well as a result presentation points of view. Location granules formulate queries with a location resolution which is appropriate for the intended application.

1.4 Discussion

A general evaluation of mobile indoor navigation systems presented in [Huang and Gartner, 2010], shows that most of the existing systems do not support context-awareness. There are only a few works that integrate some contextual dimensions other than location, especially the semantics behind the user profiles, and provide context-dependent adaptation according to these dimensions [Kritsotakis et al., 2009; Lyardet et al., 2008; Tsetsos et al., 2006]. C-NGINE [Kritsotakis et al., 2009] supports an ontology-based modelling approach along with a rule-based reasoning technique to develop a navigation system adapted to the user's needs and preferences. The major shortcoming of such a semantic approach is the lack of geometric details about objects of interest and places represented in space. On the other hand, OntoNav [Tsetsos et al., 2006] is based on a hybrid data model, which combines an indoor navigation ontology with a geospatial model (i.e., GIS layers representing a building blueprints), and a user model that helps processing path queries adapted to the user context. CoINS [Lyardet et al., 2008] is another indoor navigation system that supports navigation queries. It integrates a hybrid (i.e., symbolic and geometric) spatial data model, as well as a user model with access permissions to enable adaptive pathfinding. Nevertheless, these three systems aim at providing a navigation service, and thus they do not support other location-dependent queries such as range and nearest neighbour queries. Moreover, none of these systems has proposed a generic paradigm for the continuous processing of location-dependent queries.

1.5 Summary

This chapter presented the main challenges for the design of context-aware indoor navigation systems, and introduced recommendations for building an appropriate and sufficiently flexible data model of the indoor environment. Service-oriented and efficiency-related requirements have been established to achieve that purpose. Furthermore, mobile location-aware services and queries have been discussed and classified based on different criteria. Challenges and issues on modelling, querying, and indexing moving objects, continuous evaluation of location-dependent queries, and location-dependent query language support have been introduced. Those challenges and recommendations are considered as essential features that allow to asses indoor data models and continuous query processing techniques as will be shown in Chapters 2 and 3, respectively.

Part II

RELATED WORK

CHAPTER 2

Spatial Models for Indoor Context-Aware Navigation Systems

Contents

2.1 Introduction

Indoor spatial models have been studied and developed in many areas, ranging from mobile robot mapping to Geographic Information Systems (GIS) and ubiquitous computing [Abowd and Mynatt, 2000; Thrun, 2003; Becker and Durr, 2005]. A successful integration of indoor spaces (e.g., houses, commercial malls, etc.) and context-aware systems still requires the development of dynamic and flexible spatial models that provide appropriate services to mobile users acting in the environment.

This chapter surveys indoor spatial models developed for research fields ranging from mobile robot mapping to indoor location-based services (LBS), and most recently to context-aware navigation services applied to indoor environments [Thrun, 2003; Becker and Durr, 2005; Nagel et al., 2010].

Over the past few years, several studies have evaluated the potential of spatial models for robot navigation and ubiquitous computing. This chapter takes a slightly different perspective, considering not only the underlying properties of those data models, but also to which degree the notion of context can be taken into account when delivering services in indoor environments.

A classification of service-oriented (i.e., localisation, navigation, location-aware communication, activity-oriented interaction, and simulation and behavioural analyses) and efficiency-related (i.e., modelling effort, flexibility, performance, and scalability) requirements needed to design a context-dependent indoor data model has been presented in Chapter 1. A taxonomy of models is presented in Section 2.2 and assessed based those recommendations, with the aim of providing a flexible spatial data model for navigation purposes, and by taking into account the contextual dimensions. Section 2.3 discusses trends in designing hybrid spatial data models that favour different kinds of applications at different levels of abstraction.

2.2 A taxonomy of indoor spatial models

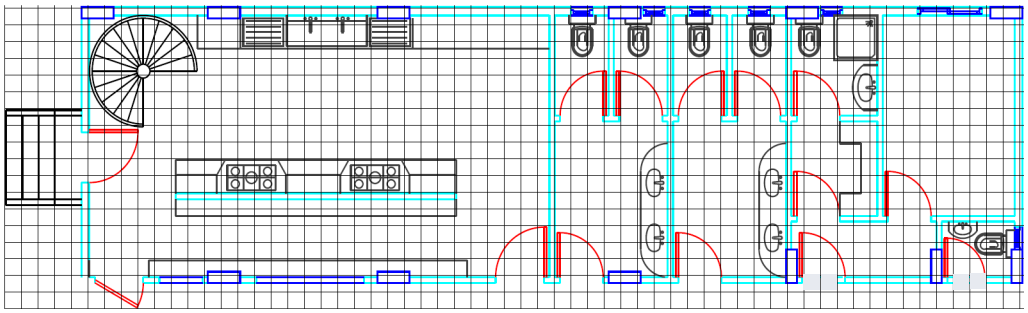
In order to deliver navigation-oriented context-aware services applied to indoor spaces, a navigation system requires an appropriate data model that is capable of representing the locations of objects either situated or actively acting within the environment. Modelling approaches are classified into two main classes: *symbolic* and *geometric* spatial models. Geometric-based approaches (otherwise referred to as metric or coordinate-based approaches) consider that locations are represented as points, lines, areas or volumes. In contrast, symbolic-based approaches provide qualitative human-readable descriptions about objects based on symbolic points of interest (e.g., room or floor identifier, building name, etc.). In this section, the strengths and weaknesses of both classes of models are assessed by describing and evaluating different spatial representations while having in mind the above-mentioned requirements, which are necessary for strengthening the spatial data structure and thus the entire context-aware system being developed.

2.2.1 Geometric-based approaches

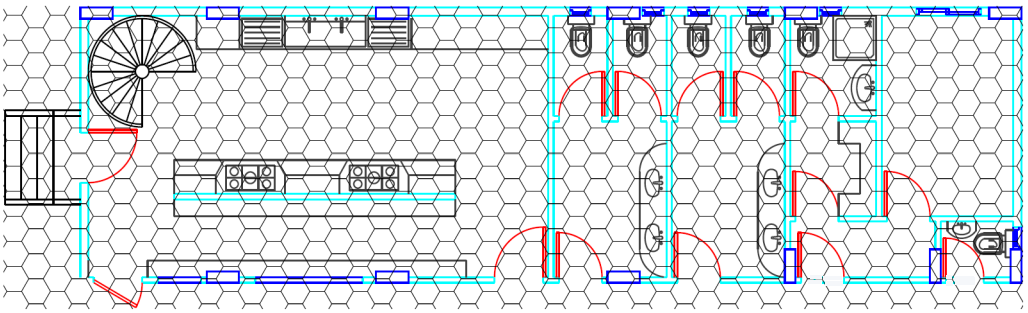
Geometric spatial models (otherwise referred to as metric or coordinate-based approaches) consider space as continuous or discrete, and mainly comprise cell-based and boundary-based geometrical representations. Table 2.1 presents an assessment of geometric models according to the requirements previously defined.

2.2.1.1 Cell-based models

The *cell-based* approach decomposes the physical space into a finite number of non-overlapping areas, thus building a partition that covers the entire space. This approach provides an implicit modelling ability to capture adjacencies between neighbouring cells. Two main types of tessellations can be distinguished [Ledoux, 2006; Mekni, 2010; Siegwart and Nourbakhsh, 2004]: regular tessellations decompose space into cells that have the exact same shape and size (e.g., primarily, square- and hexagonal-shaped cells as illustrated in Figure 2.1). Irregular tessellations aim at providing an adaptive decomposition of space that is suitable to exactly represent the complexity of the environment being studied (e.g., to accurately represent obstacles). Cells forming the irregular partition of space can be with different shapes and sizes (i.e., arbitrary polygons in 2D or polyhedra in 3D).



(a) Square-shaped grid



(b) Hexagonal grid

Figure 2.1: Grid tessellation of a floor plan based on either (a) square or (b) hexagonal cells

Regular tessellations A well-known regular tessellation is the grid-based model. Indeed, grid-based techniques are trivially implemented and can simply represent navigable and impassable regions in space by associating different cell states. Particularly, the *occupancy-grid* approach provides a probabilistic-based regular tessellation representation, which was firstly introduced in Elfes [1989]; Moravec and Elfes [1985], to address mobile robot mapping issues. In an occupancy grid, a high

probability value is assigned to cells within accessible space, while a low probability is assigned to cells occupied partially or completely by objects or obstacles.

Grid-based approaches provide spatial models with continuous geometric properties, thus favouring different kinds of geometric-based queries as well as cell-level interactions. The extent and the level of granularity are two mandatory parameters that have to be determined *a priori* for the derivation of the grid. The accuracy of the resulting grid depends on the cell resolution. Hence, a first trade-off arises between preserving high precision information retrieval and its impact on memory and time consumption, especially when dealing with large environments. A fine-grained grid provides accurate location data, but could introduce heavy processing workloads.

Dealing with a huge number of cells may exponentially increase query processing time (e.g., shortest path queries, real-time updates), thus leading to performance and scalability problems. Moreover, regular tessellation techniques do not precisely represent objects with arbitrary shapes. Hence, object boundaries are toothed, and it is possible for narrow pathways to be missed in the modelling process, especially in areas burdened by spatial objects.

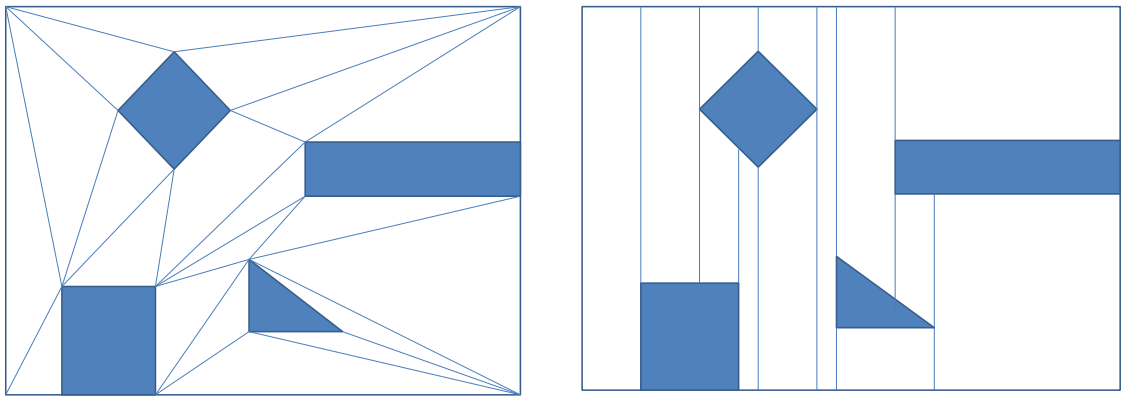
The aforementioned inefficiency has motivated the development of hierarchically-organized grid-based structures known as *quadtrees* for two-dimensional spaces [Samet, 1984]. For instance, the *region quadtree* (also referred to as *PR quadtree*) structure is commonly used when less details are required for some paths of the represented space. This also allows to recover missed pathways by repeatedly creating smaller squares to capture more details when necessary. However, the key disadvantage of this approach lies in its lack of flexibility especially when dealing with a highly dynamic environment. Indeed, dynamic environments encompass moving users, sensors, and transient obstacles that can be moved or distributed diversely in space. This means that whenever the distribution of these objects changes, a significant update may affect the whole quadtree in real time. Moreover, a quadtree model delivers quadrant-based location data so that accuracy is closely related to the size of the quadrant being identified. Localisation in free areas, where the size of the quadrant can be excessively large, is therefore significantly disturbed and may not be sufficiently accurate for navigational purposes. A large quadrant representing a part of free space may also disturb the perception of communication ability of the user. A quadtree variant called *skip quadtree* is proposed in Eppstein et al. [2005], which allows dynamic insertion and deletion of points, and search operations in logarithmic time.

Irregular tessellations Given a set of polygonal obstacles described as line segments, two main techniques have been proposed to irregularly tessellate a space:

- *free-space tessellations* that take into account obstacles and decompose free areas into convex

polygonal cells (e.g., triangles, trapezoids) - Demyen and Buro [2006]; Mekni [2010].

- *Voronoi tessellations* subdivide space into a set of special cells called *"Voronoi cells"* [Aurenhammer, 1991; Choset and Burdick, 2000].



(a) Triangulation tessellation (b) Trapezoidal-based tessellation

Figure 2.2: Examples of free-space tessellation techniques

Free-space tessellations Two kinds of decomposition can be described in this category: *trapezoidal-* and *triangulation-based* tessellations. Both spatial models are constructed based on the endpoints of the line segments that compose the boundaries of the obstacles. A trapezoidal decomposition is built by projecting a vertical line from each end-point through the free space until it hits another barrier, thus forming trapezoidal cells of different thicknesses (Fig. 2.2(b)). Although this decomposition provides a light model, resulting cells tend to be long and thick, which is not suitable for localization. In addition, navigation is performed by finding the intermediate cells to find the destination.

Triangulation-based tessellations can be generated by introducing edges between boundaries' endpoints without any edge crossings, until no more edges can be inserted [Demyen and Buro, 2006]. This technique supports path finding by hopping between triangles. Still, it may result in very thin triangles. Another technique known as *Delaunay triangulation* $\mathcal{DT}$ overcomes the shortcomings of the last mentioned approach. A $\mathcal{DT}$ of the set of endpoints is a set of triangles that decompose space such that no endpoint is inside the *circumscribed circle* of any triangle of the $\mathcal{DT}$.

Free-space tessellation methods support object-oriented analysis and allows for object-oriented interactions within space. However, they suffer from multiple paths between a source and a destination inside the chosen channel (i.e., the set of qualifying polygons), because accurate location information

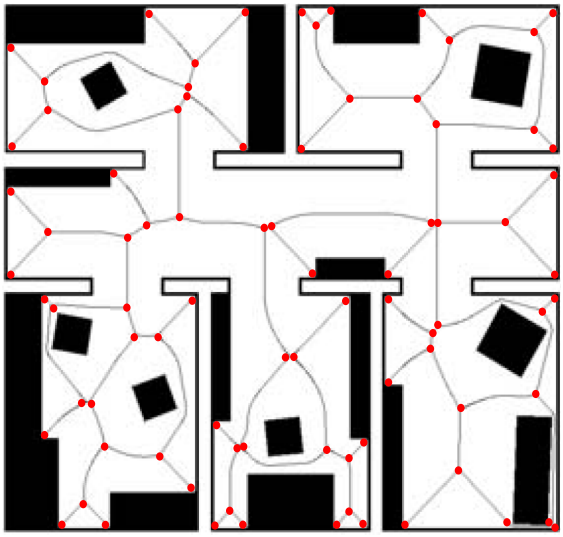


Figure 2.3: A generalised Voronoi diagram of an indoor space [Wallgrun, 2005]

within each polygon cell is unavailable. Moreover, dynamic insertion and deletion of objects is rather difficult.

Voronoi tessellation The *Voronoi Diagram* VD of a set of k points `S` is another approach that represents a built environment as a network of one-dimensional curves which concisely exhibit pathways across these points thus forming an irregular tessellation of space [Aurenhammer, 1991]. The VD can also be applied to convex objects such as line segments or obstacles rather than points. This extended approach, called the *Generalised Voronoi Diagram* (GVD), uses a function to measure the distance from a point in space to the closest point on the geometric object (Figure 2.3). Choset et al. [Choset, 1997; Choset and Burdick, 2000] have formally discussed how to incrementally construct the GVD by operating with line-of-sight sensors so that they can derive a *Generalised Voronoi Graph* (*GVG*) used for motion planning as described in Section 2.2.2.

GVDs represent a fundamental data structure for spatial modelling and have been widely used in many research areas such as robotic motion planning, computational geometry, computer graphics, GIS, etc. With a restricted number of objects, Voronoi tessellation as other irregular tessellation approaches can be more compact than grid-based approaches. However, the construction and maintenance of GVDs is still not straightforward. Additionally, localisation is often mentioned as a real problem since Voronoi diagrams force mobile users' positions to be shifted along Voronoi edges which may yield to suboptimal trajectories. As a result, GVDs cannot directly fulfil our requirements with respect to context-aware navigational services, but lend themselves perfectly to extract the GVGs that, per se, can be used as operational route-based graphs suitable for many uses.

		Service-oriented requirements				Efficiency-related requirements			
		Localisation	**Navigation**	**Location-aware communication**	**Activity-oriented interaction**	**Spatial & behavioural analyses**	**Modelling effort**	**Performance & scalability**	**Flexibility**
Geometric approaches	**Grid-based** [Elfes, 1989; Moravec and Elfes, 1985]	⊕ accurate location data	⊕ suitable for navigation ⊖ no symbolic instructions	⊕ geometric-based data exchange ⊕ high quality data	⊕ cell-level interactions ⊖ no object-based interactions	⊕ continuous analysis ⊖ no object-oriented analysis	⊕ easy to design and maintain	⊖ consumes high memory and processor time in large spaces	⊕ good for navigation queries as geometric information is accurate
	Quadtree [Samet, 1984]	⊕ quadrant-based location data ⊖ constrained by its structure and the size of quadrants	⊕ optimises navigable space ⊖ no symbolic instructions	⊕ geometric data exchange ⊖ disturb the perception of communication ability[a]	⊕ quadrant-based interactions	⊕ quadrant-based analysis ⊖ no object-oriented analysis	⊕ medium effort to build the tree ⊖ dynamic insertion and deletion of objects is difficult	⊕ more compact ⊖ poor in highly dynamic environments	⊕ good for navigation queries ⊖ not flexible in dynamic environments
	Free-space tessellation [Demyen and Buro, 2006; Mekni, 2010]	⊕ location data based on an irregular tessellation ⊖ not always suitable for localization	⊖ paths might be not optimal	⊕ geometric data exchange ⊖ disturb the perception of communication ability	⊕ object-oriented interactions	⊕ object-oriented or empty space related analysis	⊕ easy to tessellate space ⊖ dynamic insertion and deletion of objects is not straightforward	⊕ efficient because more compact ⊖ poor in highly dynamic environments	⊕ basic navigation services
	Generalised Voronoi diagram [Aurenhammer, 1991; Choset and Burdick, 2000]	⊕ location data based on an irregular tessellation ⊖ no accurate location information	⊖ paths might be not optimal	⊕ geometric data exchange ⊕ ability to communicate about objects	⊕ interactions with objects within cells	⊕ analysis of objects within Voronoi cells	⊖ dynamic insertion and deletion of objects is not straightforward	⊕ efficient because more compact ⊖ poor in highly dynamic environments	⊕ basic navigation services
	Boundary-based [Chatila and Laumond, 1985; Crowley, 1989]	⊕ geometric location data	⊖ very limited	⊕ geometric data exchange	⊕ boundary-based interactions	⊕ basic analysis	⊕ easy to design and maintain a CAD model ⊖ model matching can be costly in robotic applications	⊕ efficient with basic operations	⊖ do not support navigation services

Table 2.1: Assessment of geometric-based approaches

[a] A large quadrant may be partially covered by a sensor, but this covered zone can not be perceived by the user, thus weakening its ability to communicate

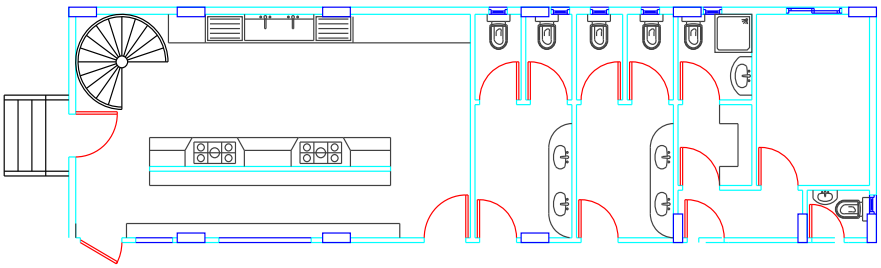


Figure 2.4: Example of a floor plan designed using a CAD system

Irregular tessellation techniques generally offer several advantages over regular techniques including smooth modelling of arbitrary objects, recognition of narrow valid paths, and the fact that it can be markedly compact in unburdened environments. However, most of these techniques do not support highly accurate localisation of mobile objects within the polygonal cells.

2.2.1.2 Boundary-based models

Boundary-based models represent the obstacles' boundaries of an indoor space with sequences of primitive geometries such as points, lines, curves, etc. In most cases, a boundary-based map is constructed by concatenating a set of lines segments either extracted from sensor data and representing obstacles as polygons [Chatila and Laumond, 1985; Crowley, 1989; Thrun, 1997], or designed using a CAD system (Figure 2.4). In many robotic applications that assume no prior knowledge of the environment, sensor data are obtained and the resulting extracted objects are matched to a geometric map. However, model matching can be computationally expensive [Thrun, 1997]. Indeed, the line extraction sensors need to be extremely accurate so that lines representing walls, for instance, can be properly positioned at the right location and the right angle on the map. On the other hand, a geometric floor plan can be designed using a CAD system so that various spatial entities (e.g., doors, windows, rooms, etc.) are represented as sequences primitive geometries. But Those approaches are mainly oriented towards architectural design.

In contrast to cell-based approaches, boundary-based geometric models form the most direct way to represent an indoor space and can be significantly compact. However, such an approach lacks the capability to incorporate additional object-based semantics so that a deeper knowledge of the represented spatial entities can be achieved. Moreover, boundary-based models are less suitable for navigational services, like path planning, or for communication because they do not lend themselves very well to applying standard techniques and algorithms for spatial search, finding most appropriate paths, and many other services.

2.2.2 Symbolic-based approaches

Symbolic approaches have frequently attempted to model indoor environments using topological-based structures [Durr and Rothermel, 2003; Hu and Lee, 2004], graphs by capturing the connectivity and reachability between spatial units [Franz et al., 2005; Jensen et al., 2009a; van Treeck and Rank, 2004], and hierarchies [Becker et al., 2010; Choset and Burdick, 2000; Stoffel et al., 2008]. The main advantage of these approaches is that object location is provided semantically by using human-readable descriptions (e.g., room identifier), and topological relationships are also represented. Depending on their nature, symbolic location models can reveal *containment*, *connectedness*, *closeness*, and *overlapping* relationships [Durr and Rothermel, 2003; Hu and Lee, 2004; Jensen et al., 2009a]. Using the containment relationship, for instance, one can derive a hierarchical structure of the indoor space in order to take into account application constraints and properties. Accuracy of location information in symbolic space depends on the level of abstraction of the indoor data model. For instance, a place-based symbolic model can provide location information at the structural-entity level. However, symbolic models are often unable to support a highly accurate indoor positioning system, and are less suitable for determining the Euclidean distances between spatial entities. Moreover, a symbolic model depends on the application domain and needs to be created and managed accordingly. Hence, managing a very large number of location symbols requires an important modelling effort.

Symbolic models are classified into two categories: set-based models and graph-based models. The set-based approach gathers object or place identifiers into sets and subsets; these are used to define spatial relations between elements of a set or, for instance, the intersection between two subsets, etc. [Becker and Durr, 2005; Bhatt et al., 2009]. These sets can be further hierarchically organised to form a tree- or lattice-based structure [Durr and Rothermel, 2003; Li and Lee, 2008b; Hu and Lee, 2004]. Graph-based approaches represent a space as a topological graph where nodes symbolize predefined landmarks (e.g., place, gate, sensor range, object, etc.) extracted either manually or automatically from the environment, and edges stand for the connections that make it possible to move through these landmarks [Choset and Burdick, 2000; Remolina et al., 1999; Remolina and Kuipers, 2004; Werner et al., 2000]. These two categories will be discussed further as follows.

2.2.2.1 Set-based symbolic models

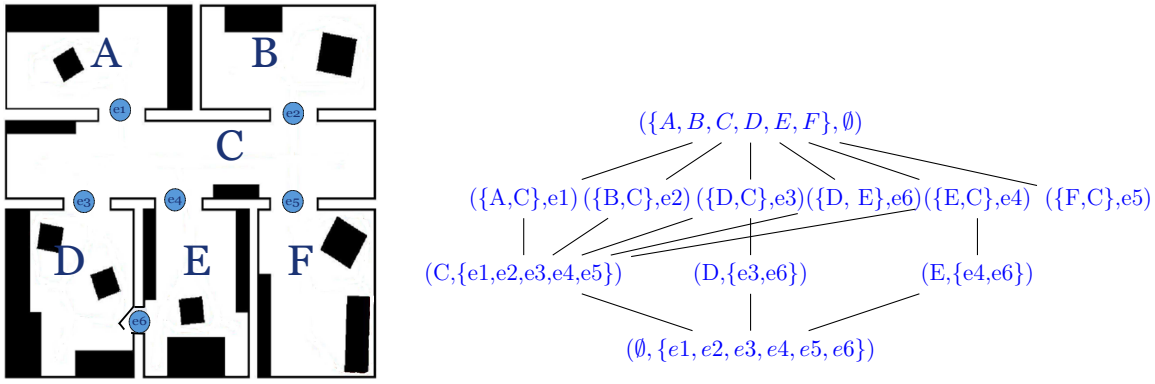
Set-based models identify places or objects of interest within an indoor space, and then gather these symbolic identifiers into sets. Two kinds of set-based models are distinguished: *place-based* and *object-oriented* models. The former considers a set of place identifiers based on the architectural properties of an indoor space [Becker and Durr, 2005; Li and Lee, 2008a], while the latter deals with all entities that may contribute to build a smart environment [Bhatt et al., 2009]. Indeed, the

major difference between these two approaches is the level of abstraction. Place-based models deal with places and build its hierarchy based on the containment relationship. Whereas, object-oriented models consider not only places, but all other objects of interest (e.g., walls, doors, sensors, etc.) in order to build the hierarchy.

Place-based sets Current place-based approaches model an indoor space by creating sets and subsets of place identifiers based on the architectural properties of space [Durr and Rothermel, 2003; Kainz et al., 1993]. A typical example considers places of a building such that each floor is contained within a building, and each room is contained within at most one floor. A superset is likely to be defined as the set of floor numbers, and at a lower level, a subset related to each floor that aggregates all room numbers has to be created. Such a model is hierarchically organised and can be viewed as a tree structure in which location information is represented at different levels of abstraction. Moreover, when considering places and their neighbours, one can reflect adjacency relations between them as well as some qualitative notions of distance, i.e., one can infer that the distance between two neighbouring rooms is smaller than the distance between two distant rooms [Becker and Durr, 2005].

Place-based sets models deliver room-level location data, and allow for place-based data exchange as well as for interactions with places. However, the main shortcoming of those approaches is their inability to model connectivity between places as, for instance, two neighbouring rooms may or may not be interconnected. Also, it is often unable to determine quantitative distance efficiently. Furthermore, such a tree-based structure is built with respect to the containment relationship, and it does not allow for an element to inherit from multiple parents; this means that a place cannot be contained within two overlapping subsets. This is the case where floors are shared between several wings within a building. In such a situation, a room may belong to a floor and a wing at the same time. This problem has been dealt with by using a lattice structure instead of a conventional tree [Li and Lee, 2008a].

As defined in Kainz et al. [1993], a fine-element lattice is a partially ordered set in which every subset can have a least upper bound and a greatest lower bound. These characteristics of lattices are then applied on a set of places `P` along with the containment relationship $\leq$ (i.e., $p_1 \leq p_2$ means p_2 spatially contains p_1) in order to construct the model. This model overcomes the last mentioned problem regarding overlapping subsets by allowing for multiple inheritance, and it is appropriate for range queries as used in Durr and Rothermel [2003] to send *geocast* messages. However, this model does not lend itself to other tasks like navigation, because it still does not represent connectedness between places. Another lattice-based semantic model has been presented in Li and Lee [2008a], that uses the exit-location concept. The exit-location approach models a given space with two types of entities: a location described as a bounded geometric area bordered by one or more exits, and an



(a) An exit-location area in a building

(b) The corresponding lattice-based model

Figure 2.5: Example of a place-based set model

exit as a boundary gate that allows moving in or out of a location [Hu and Lee, 2004; Lee et al., 2005]. The hierarchical structure of the lattice is mainly based on the containment relationship between locations (Figure 2.5). This model is, for instance, applied to find optimal paths based on nearest neighbour relations.

Object-oriented model An object-oriented model introduces an approach that annotates objects of interest with semantic information according to the properties of the environment. Entities are thus represented as objects, and are characterised by specific attributes and relations. For instance, the *Industry Foundation Classes* (IFC) provide a specification of an object-oriented data model, which serves as a data exchange format reflecting building information [Froese et al., 1999; Liebich et al., 2006]. In contrast to geometric plans (e.g., 2D or 3D CAD drawings), IFC represent all entities of space (e.g., walls, windows, etc.) as objects. IFC-based models are used in planning and management processes to improve quality of data exchanged and to perform complex management-oriented analysis (e.g., electronic checking, energy effort, etc.)

An object-oriented model based on the IFC specification has been proposed in Bhatt et al. [2009]. The indoor space is then defined as the combination of three complementary layers:

- A conceptual layer models terminological and functional characteristics on the architectural entities (e.g., sensor, sliding door, meeting room, computer science laboratory, etc.) regardless of how and where they will be used.
- A quality module gives qualitative descriptions about spatial dependencies and relationships with other entities. For instance, the authors use relations as provided by the spatial calculus RCC [Grütter et al., 2008] (e.g., *proper part of*, *partially overlaps*, etc.)

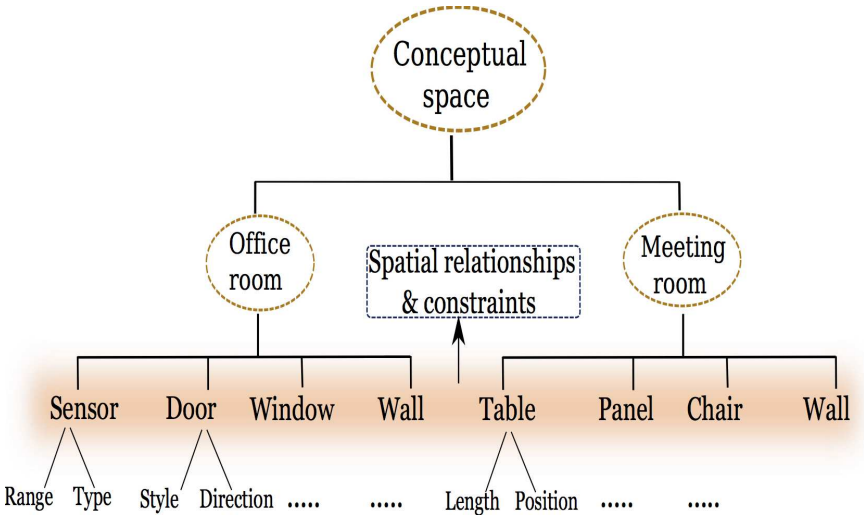


Figure 2.6: Example of an object-oriented model of an indoor space

- Finally, architectural entities are quantitatively described in the quantity module.

The advantage of this approach is that it qualifies the different roles played by the objects in the environment, these roles being spatially qualified. A example of such a conceptual space is illustrated in Figure 2.6. Such a conceptual space models all entities of interest along with quantitative and qualitative properties attached to them, as well as spatial relationships and constraints between represented entities.

Object-oriented data models provide symbolic location data at the object level, and allow for semantically-enriched data exchange about objects of interest. Moreover, object-oriented spatial and behavioural analyses can be performed. Nevertheless, object-oriented approaches are not directly suitable for navigational tasks, since geometric details about represented objects as well as connectivity and adjacency relationships are not directly supported.

2.2.2.2 Graph-based models

Graph-based approaches represent an indoor space as a graph where nodes model predefined locations (e.g., place, gate, point of interest, etc.) extracted either manually or automatically from the environment, and edges stand for the connections that make it possible to move through these locations [Choset and Burdick, 2000; Remolina et al., 1999; Remolina and Kuipers, 2004; Werner et al., 2000]. For instance, a topological graph that directly reflects the architecture of a floor plan represents rooms as nodes and doorways as edges; this can simply express connectedness relationship between the architectural entities [Franz et al., 2005]. In this category, two main modelling concepts

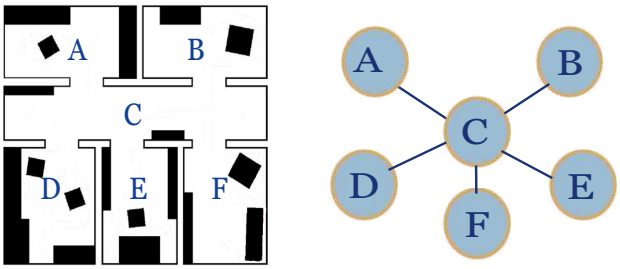


Figure 2.7: An place graph representing structural entities of an indoor space

are discerned: *layout-based* and *layout-independent* models. The difference between these two categories is shown as follows.

The layout-based representations rely on graphs where nodes are derived or extracted from the structure of space. This can be exhibited by a basic graph model that can be referred to as a *place graph*, as well as by *visibility graphs*. *Voronoi-based graphs* are also constructed by extracting meet points and boundary points directly from space as described earlier. These points refer to the nodes of the generalised Voronoi graph. *Fine-grained graphs* preserve indirectly structural properties of the environment since nodes are evenly distributed over the entire space. In contrast, some other approaches have adopted graph models that are layout-independent [Becker et al., 2009; Jensen et al., 2009a]. Nodes are then not directly derived from the structure, but instead are extracted by means of a sensor deployment strategy within space.

Place graphs In their simplest form, place graphs clearly materialize topological properties of space. In this approach, nodes stand for places such as rooms and/or hallways, and doorways that connect these places appear as edges (Figure 2.7). Besides the connectivity relationship artlessly presented, other variants of topological relations between structural entities can be inferred such as adjacency and containment properties by annotating nodes and edges and/or supporting a graph with multiple levels of granularity. So far, this modelling concept has been widely used since it allows to efficiently navigate between places, plan routes to destinations, and is well-suited for nearest neighbour queries. In addition, it supports symbolic data exchange and interactions with places. Nevertheless, this approach still does not consider interacting objects. It also has a less accurate location information that does not meet specific application requirements. Geometric properties of space disappear, and it is still difficult to model a semantic distance function which helps, for instance, determining the shortest path. A semantic exit-location based model has been presented by Hu and Lee [2004]. The aim of this modelling approach is not only to preserve the advantages of the classic place graph model, but also to integrate geometric information such as a distance model to

overcome some of the aforementioned problems. The exit-location approach models the space with two types of entities: a location described as a bounded geometric area bordered by one or more exits, and an exit as a boundary gate that allows moving in or out of a location. The authors have introduced a series of algorithms for automatically constructing the location and exit hierarchies to derive the symbolic location model. Moreover, this symbolic model is built on some geometric information that maintains distance semantics to support services such as nearest neighbour search, shortest path, and location-aware navigation. Although this approach has dealt with many critical problems, other advanced contextual queries were still not addressed because the model cannot fully support object movement and can provide only basic types of services.

Visibility graphs Based on a triangulation tessellation, one can derive a visibility graph that materialises edges between mutually visible endpoints representing specific locations and obstacles, and extracted from space [De Berg et al., 2008]. Visibility graphs are useful to predict some behavioural and spatial analyses to favour spatial reasoning as suggested in space syntax [Turner, 2001; Franz et al., 2005]. However, the disadvantages of triangulation tessellations remain the same as accuracy of location information, optimality of path queries, interaction, and flexibility are not dealt with. Furthermore, dynamic insertion and deletion of objects may lead to changes in the graph topology, i.e., the locations and number of nodes can change, and additional edges could be drawn.

Generalised Voronoi Graph (GVG) The GVD is an appealing approach that represents a built environment as a network of curves which concisely exhibit pathways suitable for navigational purposes. A route-based graph can be extracted from the GVD previously described. Such a route graph is referred to as the *Generalised Voronoi Graph* (GVG), which directly reflects pathways through obstacles [Choset and Burdick, 2000]. Voronoi-based approaches are suitable for navigation services such as finding a collision-free path towards a destination. In addition, a GVG inherits all advantages of graph-based representations so that it can, for instance, annotate its nodes by additional information thus favouring location-aware communication and object-oriented interactions. Specific techniques have been suggested to further prune and remove irrelevant nodes and edges so that the whole graph can be more compact [Wallgrun, 2005, 2010]. However, an indoor space, such as an office building, can be populated with a huge number of objects of interest, which may significantly increase the number of nodes and edges that constitute the graph. Furthermore, the location accuracy problem is raised in specific situations when moving in free-areas, which means that the model does not lend itself to completely support ubiquitous positioning sensors.

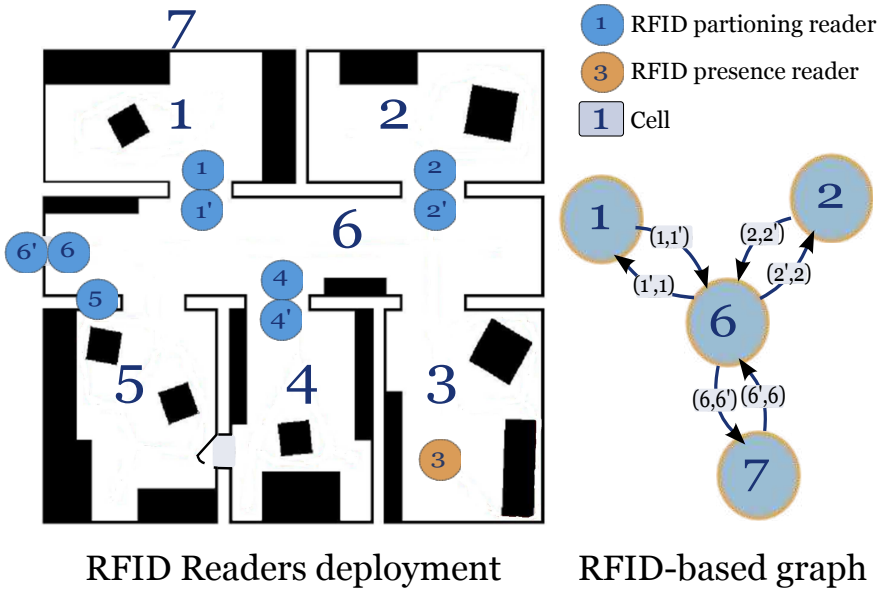


Figure 2.8: Example of a sensor-based graph

Fine-grained graphs Although previous approaches consider an indoor space at a more abstract level and so have a more compact representation, they appear to be badly-suited to context-aware navigation services since interactions at a fine level within space are increasingly needed. In a companion paper [Li et al., 2010], the authors have presented a graph-based model at a fine level of granularity that retains continuous and structural-based properties of space. Nodes in this model represent cells within an occupancy grid, while connections between cells are materialized by edges (Figure 4.3). Nodes and edges are labelled according to their belonging to a given spatial unit such as a room or a connecting space. One advantage of this modelling approach is that it achieves a maximum coverage of space so that accurate positioning sensors used for indoor navigation can be supported. Indeed, the geometrical properties are implicitly represented by the continuous layout of the graph and reflect indirectly the shapes of places being considered. This approach allows for high quality data exchange, and for fine-grained and continuous analysis. However, a major difficulty that still needs to be confronted regarding efficiency-related criteria and specially when considering performance and scalability issues. A large indoor space could comprise thousands of nodes and edges so that managing and executing real-time queries could be an excessive time- and memory-consuming process.

Sensor-based graphs Many approaches have adopted a sensor-oriented point of view where the objective is mostly oriented to the tracking and monitoring of mobile objects [Becker et al., 2009; Jensen et al., 2009a]. For instance, the model presented in Becker et al. [2009] describes

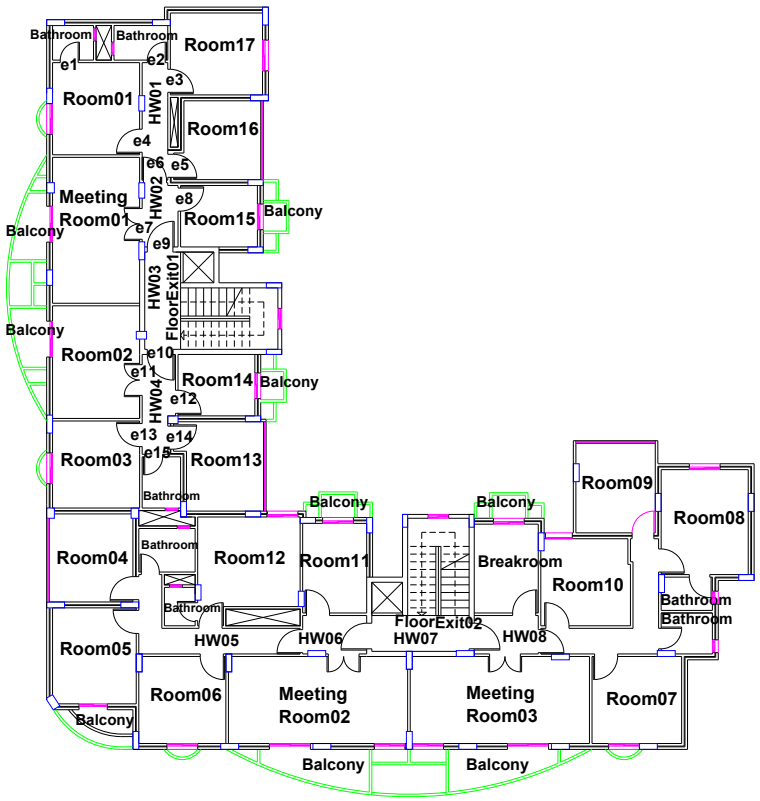


Figure 2.9: Example of a fine-grained graph of a floor plan

space as a set of layers, the topographic space and the sensor space, and establishes connections between layers based on the sensor coverage. Deployment graphs have been proposed in Jensen et al. [2009a] by using different types of positioning sensors in order to improve indoor tracking accuracy (Figure 2.8). A classification of positioning sensors in indoor space into presence sensors and partitioning sensors can be found in Jensen et al. [2010]. Navigation between cells representing ranges of the deployed sensors is supported, which also allows for range-based analysis. However, accuracy of location information in such techniques is relatively low since it depends on the sensor range. Object-oriented interaction is also not supported since objects are not directly represented. Furthermore, an optimised deployment strategy of the sensors is needed so that a more compact and more efficient graph can be created. A more detail discussion on sensor-based graphs along with a method for an optimised deployment of sensors can be found in Kang et al. [2010].

2.2.3 Discussion

The taxonomy presented in this section relies upon the model of location information being delivered. Other classifications of indoor models have been proposed in Becker and Durr [2005]; Worboys [2011].

		Service-oriented requirements				Efficiency-related requirements			
		Localisation	**Navigation**	**Location-aware communication**	**Activity-oriented interaction**	**Spatial & behavioural analyses**	**Modelling effort**	**Performance & scalability**	**Flexibility**
Symbolic approaches	**Place-based set** [Becker and Durr, 2005; Durr and Rothermel, 2003; Li and Lee, 2008a]	⊕ location data at room & connection level	⊖ no information about connectivity between places ⊖ not suitable for metric functions	⊕ symbolic-based data exchange ⊖ no geometric data exchange	⊕ place-based interactions ⊖ no object-oriented interaction	⊕ room-level analysis	⊕ easy to design and maintain	⊕ efficient and scalable thanks to the hierarchical structure	⊕ suitable for range queries ⊖ no support for navigation services
	Object-oriented set [Bhatt et al., 2009]	⊕ location data at object level (e.g., sensor, ...)	⊖ needs information about adjacency and connectivity between places ⊖ not directly suitable for navigational tasks	⊕ symbolic-based data exchange ⊕ object-oriented communication ⊖ no geometric data exchange	⊕ object- and place-based interactions	⊕ semantic & object-oriented analysis	⊖ medium effort as the number of objects to manage is high	⊕ scalable thanks to the hierarchical structure	⊕ semantic-based applications ⊖ navigation services are limited
	Place-based graph [Hu and Lee, 2004; Li and Lee, 2008b]	⊕ graph-based, room & connection level	⊕ suitable for navigation as connections between places are materialised ⊖ no support for metric functions	⊕ symbolic data exchange ⊖ no geometric data exchange	⊕ place-based interactions ⊖ no object-oriented interaction	⊕ room-level analysis	⊕ easy to design and maintain	⊕ efficient due to its compactness	⊕ navigation services
	sensor-based graph [Becker et al., 2009; Jensen et al., 2009a]	⊕ graph-based (accuracy depends on sensor range)	⊕ range-based navigation	⊕ symbolic & range-based data exchange	⊖ no object-oriented interactions	⊕ range-based analysis	⊕ medium effort to design the graph ⊖ optimized deployment strategy of sensors is needed	⊕ efficient due to its compactness	⊕ tracking services
	Visibility graph [De Berg et al., 2008]	⊕ graph-based (accuracy depends on the number and location of nodes)	⊕ suitable but for basic navigational tasks	⊕ symbolic data exchange ⊕ object-oriented communication	⊕ object-based interactions	⊕ visibility and spatial analyses	⊕ easy to design ⊖ dynamic insertion and deletion of objects is not straightforward	⊕ efficient because more compact ⊖ poor in highly dynamic environments	⊕ applications for spatial cognition and human reasoning
	Generalised Voronoi graph [Choset and Burdick, 2000; Wallgrun, 2005]	⊕ graph-based (accuracy depends on the number and location of objects)	⊕ suitable but for basic navigation services	⊕ symbolic data exchange ⊕ object-oriented communication	⊕ object-based interactions	⊕ navigation-based analysis	⊖ sophisticated, takes considerable time	⊕ efficient because more compact ⊖ poor in highly dynamic environments	⊕ navigation services
	Fine-grained graph [Li et al., 2010]	⊕ graph-based (accurate location data)	⊕ suitable for navigation	⊕ symbolic-based data ⊕ high quality data exchange	⊖ no object-oriented interaction	⊕ fine-grained and continuous analyses	⊕ easy to design and maintain	⊖ consumes high memory and processor time ⊖ not scalable	⊕ navigation services

Table 2.2: Assessment of symbolic approaches

The distinction between geometric and symbolic (also referred to as *topological* in Worboys [2011]) spatial models is similarly emphasized in both classifications as explained earlier in this section. A distinction is also made between spatial and semantic models since the latter kind of models is object-oriented and not necessarily spatially-dependent [Worboys, 2011]. However, neither of these proposals has reviewed and assessed different modelling approaches from the particular perspective of context-aware indoor navigation systems. A summary of different modelling approaches is given below, along with a discussion of their use from an application perspective.

2.2.3.1 Geometric-based approaches

Grid-based models can efficiently integrate metric properties, thus allowing precise locations, direction information, and distances (cf. Table 2.1). Quadtree is a hierarchically-organized grid-based structure that allows for space optimization, but lacks flexibility in dynamic environments. Free-space tessellation and Voronoi-based diagrams are less suitable for localisation, but they are more compact. Boundary-based models are less suitable for navigational services but provide accurate location data. In addition, geometric models require an integration of semantic annotations to achieve a higher degree of location- and context-awareness.

2.2.3.2 Symbolic approaches

As shown in Table 2.2, symbolic models are generally less accurate but context-awareness is easier to achieve as they favour human-recognizable descriptions. Thanks to their hierarchical structure, set-based models achieve a good level of efficiency and flexibility, but lack of topological relationships such as connectedness. Graph-based models are widely used in applications at a coarse-grained level of abstraction thanks to their richness and variety. The major shortcoming of symbolic models is the lack of geometric details on entities and places represented in space.

2.2.3.3 Application perspective

Symbolic-based approaches are often preferred, from an application perspective, over conventional geometric-based approaches and have been recently used in many application scenarios [Becker and Durr, 2005], because they can capture the semantics of entities and places represented in an indoor space. In particular, graph-based and semantic models constitute the most common approaches used, so far, in many application areas ranging from emergency management and safety control in micro-scale environments [Kwan and Lee, 2005; Lee, 2007; Lee and Zlatanova, 2008; Park et al., 2009] to indoor context-aware navigation services, and especially those adapted to users with special needs

[Becker et al., 2009; Cagigas and Abascal, 2004; Dudas et al., 2009; Fernández-Madrigal et al., 2004; Kritsotakis et al., 2009; Tsetsos et al., 2005]. Applications that aim at providing intelligent emergency responses mainly employ simple place graphs, which capture topological relations between structural entities, because they are more concerned with network-based models that allow to discover nearest or optimal exits [Kwan and Lee, 2005; Lee, 2007]. In contrast, applications that support contextual elements, such as user preferences and capabilities, tend to favour semantically-enriched data models either by designing an ontology-based model, or by employing a hybrid model that combines a graph-based with a semantic model of space as discussed in Section 2.3 [Cagigas and Abascal, 2004; Dudas et al., 2009; Kritsotakis et al., 2009; Tsetsos et al., 2005].

2.3 Towards hybrid spatial models

Numerous spatial data models have been presented in different research fields (e.g., robotics, GIS, ubiquitous computing) with the aim of combining advantages of geometric and symbolic approaches [Buschka, 2005; Jiang and Steenkiste, 2002; Leonhardt, 1998; Thrun, 1998; Wallgrun, 2005]. Those approaches have complementary strengths and weaknesses since they have different purposes. Whereas geometric models can efficiently integrate metric properties to provide highly accurate location and distance information that are necessary elements in most of context-aware applications, symbolic models maintain a more abstract view of space by providing users with easily-recognizable information and by materializing more complex relationships between entities. Neither approach is, therefore, directly suitable for fulfilling all requirements of context-aware services. Clearly, integration of geometrical and topological representations implies considering qualitative and quantitative points of view, thus favouring a large spectrum of interacting representations and applications, as suggested by the Spatial Semantic Hierarchy (SSH) introduced in Kuipers [2000]. This enables human reasoning and human- or robot-centred activities, and even sensor-based interactions within the environment.

Accordingly, the idea of integrating different coexistent models of space appeared really as a promising alternative. Hybrid spatial models can be produced in different manner by applying various kinds of organization [Becker and Durr, 2005; Buschka, 2005; Wallgrun, 2005]: (1) parallel models (also referred to as overlays) aim at using different spatial models (usually a combination of geometric and symbolic models) that cover the entire space [Becker et al., 2009; Li et al., 2010; Stirbu, 2009; Thrun, 1998; Wallgrun, 2005], (2) patchwork-based approaches represent a space with several local, usually geometrical, models that are linked together to form a global, usually symbolic, model of space [Kuipers, 2000], and (3) hierarchical models that embed different layers with different levels of abstraction [Fernandez and Gonzalez, 1998, 2002; Jiang and Steenkiste, 2002].

Recent studies on mobile robot navigation have focused on how to extract a topological graph from a basic geometric map such as a grid-based map or a Voronoi diagram. The resulting graph is then pruned by applying some algorithms that select the more relevant nodes, thus implicitly providing a hierarchical representation as those encompass knowledge of the environment at different levels of abstraction [Thrun, 1998; Wallgrun, 2005]. Earlier, the SSH model has been presented [Kuipers and Byun, 1991; Kuipers, 2000], and further extended in Beeson et al. [2005]; Kuipers et al. [2004]. The SSH consists of a hierarchy of representations that are inter-dependent. This hierarchy is mainly made of local geometrical maps that correspond to the human or robot's sensory horizon, and are merged together based on topological relationships which, in turn, are derived thanks to causal state-action-state schemas.

A relevant example of a hybrid spatial model designed to handle location-dependent queries is presented in Jiang and Steenkiste [2002]. This model is a combination of a hierarchical, set-based representation of space and a geometric representation of places, sensor ranges, and objects of interest. These geometric annotations are attached to the corresponding elements in the symbolic set. The model achieves a good trade-off between geometrical and symbolic approaches by combining the benefits from both sides. Although this model handles some relevant location-dependent queries such as *"find the nearest object of interest"*, it lacks a clear process description of how to acquire an accurate location of the mobile user continuously. In addition, queries are handled based only on the distance parameter while most of context-aware applications require integration of contextual dimensions that aim to find the optimal solution not always the nearest one. In Stirbu [2009], the author proposed to symbolize a physical space with a set of layers that correspond to different location models designed to meet various activities performed by the users. At the basic layer, the model embeds a quadtree by considering some points of interest. On top of this layer, various topological models were added depending on users' activities and formed a lattice model. The lattice model has been used together with a simple graph model, extracted to materialize connections, to perform relevant location-dependent queries such as position, range, and path queries.

The Annotated Hierarchical graph model (AH-graph), presented in Fernandez and Gonzalez [1998, 2002], constitutes a comprehensive framework for efficient mobile navigation. The AH-graph consists of multiple topological layers defined at different levels of abstraction and linked together based on abstraction functions developed for nodes and arcs. An annotation function is also defined at each level so that information can be attached to nodes and arcs. This allows performing basic queries (e.g., path searching) hierarchically, thus achieving better performance. An extension of the model was then presented by adding multiple hierarchies produced based on several scenarios specified at the application level, thus favouring a large spectrum of applications and achieving a high level of flexibility. The model has been further used in Remolina et al. [1999] to implement the

SSH at topological level thanks to its efficiency and flexibility. In Galindo et al. [2005], the AH-graph hierarchy was appended to a semantic hierarchy to further improve human-robot communication.

Generally, parallel models that combine, for instance, a graph-based model with a geometrical model underneath, tend not to scale well to large environments since they could not get rid of weaknesses of geometric models with respect to efficiency and scalability. Hierarchical models, in contrast, scale very well to large environments since queries such as path search are performed hierarchically by switching from finer to coarser levels and vice versa. In addition, a specific level of granularity can be used in specific situations with respect to application constraints and users' preferences. Approaches that integrate hierarchical organization require, however, to maintain connections between levels and to integrate a fine-grained geometric model that guarantees accurate localisation for specific navigational purposes.

Most of the principles presented in this chapter has inspired our work whose objective is to develop a context-dependent multi-granular indoor data model whose objective is to provide a flexible representation of an indoor space, and also will take into account the objects located and acting in the environment. This modelling approach integrates different levels of granularity, that is, a fine-grained layer at the first level and an exit hierarchy as well as a location hierarchy at the second level of abstraction, and considers other contextual dimensions besides the location of the involved entities, such as time and user profiles (see Chapter 4). The main advantage of the indoor data model relies on its hierarchical and context-dependent design, which allows a large spectrum of applications to be developed at different levels of abstraction, while alleviating performance and scalability issues in location-dependent query processing. Moreover, this model supports different categories of location-dependent queries, either in continuous or discrete modes, and two algorithms for continuous processing of path and range queries on top of this modelling approach are proposed in Chapter 6.

CHAPTER 3

Continuous Location-Dependent Query Processing

Contents

3.1 Need for incremental and adaptive query processing

Many innovative mobile applications need to incorporate a mechanism for the continuous processing of location-dependent queries over moving objects. Examples of such services include continuous crowd monitoring within a given area, location-based alerts (e.g., continuously send E-coupons to

all customers within 200 metres of my store), crowd notification in an emergency situation, and location-based friend finder (e.g., 1- *"let me know if I am near a restaurant while any of my friends are there"*; and 2- *"if I continue moving towards this direction, which will be my closest restaurants in the next 10 minutes?"*).

A continuous query remains active over a period of time and has to be continuously evaluated during that period as the objects of interest are moving. Efficient processing of such queries is a complicated task in location-based services due to its expensive consumption of memory and computational resources. A straightforward approach to deal with this kind of query would be to periodically recompute the best answer from scratch for all queries upon arrival of every delay update. However, such a naïve approach performs poorly in highly dynamic environments, where the locations of both the query point and the target objects change over time. In addition, this approach scales poorly when there are many concurrent users who are asking queries at the same time.

Consequently, there is a growing need for approaches that provide *adaptive* and *incremental* query processing techniques to enhance efficiency of location-dependent queries over moving objects [Hu et al., 2005; Mokbel et al., 2005]. On the one hand, *adaptive query processing* addresses the problems of unpredictable costs and dynamic data by using *runtime feedback* in order to *tune* query execution in a way that provides appropriate answers with better response time or memory utilization [Deshpande et al., 2007]. On the other hand, an *incremental execution paradigm* deals with continuous query processing by reusing information from previous searches to speed up current searches. Therefore, new answers for series of similar search problems can be computed faster than by reevaluating each query independently from scratch [Sun et al., 2009]. Both adaptive and incremental execution paradigms are required in order to achieve sufficiently flexible and more efficient continuous query processing.

This chapter surveys approaches for querying and managing moving objects in dynamic environments. Both outdoor and indoor contexts are considered, since indoor-oriented strategies are rarely discussed in the literature, and some outdoor strategies can be applied to indoor settings with adjustments, when necessary. The remainder of this chapter is organized as follows. Section 3.2 presents strategies and architectures for location-dependent query processing, mainly, in moving object databases, over spatio-temporal data streams, and in specific indoor settings. It then discusses preference-aware query processing techniques and their applicability to continuous queries. Section 3.3 takes another perspective and discusses algorithmic issues with the aim of highlighting adaptive and incremental approaches for the continuous processing of navigation-related queries in spatial network databases. Languages for spatio-temporal queries and more specifically for preference-aware location-dependent queries are discussed in Section 3.4.

3.2 Architectures for location-dependent query processing

Different architectures, indexing strategies, and update methods have been proposed to efficiently deal with location-dependent queries over moving objects [Lam et al., 2001; Lazaridis et al., 2002; Kalashnikov et al., 2002; Prabhakar et al., 2002; Cao et al., 2003; Gedik and Liu, 2004; Jensen et al., 2004; Hu et al., 2005; Cai et al., 2006; Güting et al., 2006; Gedik and Liu, 2006; Ilarri et al., 2006a; Zhao et al., 2010]. This section takes a closer look to those approaches and mainly classifies them into three categories: (i) approaches for continuous query processing in *moving object databases*; (ii) approaches for continuous processing of *spatio-temporal data streams*; and (iii) approaches that incorporate *user preferences* and other *contextual dimensions* in query processing. It should be noted that most of those strategies have been developed for outdoor environments. The only works we are aware of regarding architectures for mobile data management in indoor environments are presented in [Jensen et al., 2009b; Lu et al., 2012; Xie et al., 2013; Alamri et al., 2013]. Therefore, this section describes the intrinsic properties of those approaches applied to both outdoor and indoor environments, while highlighting their advantages and disadvantages with respect to our context.

3.2.1 Continuous query processing in moving object databases

Many features and criteria can be specified in order to classify and assess approaches for location-dependent query processing over moving objects such as the incremental evaluation support, whether static and/or moving queries are considered, the kinds of queries supported, whether a centralised or a decentralised strategy is proposed, communication overhead, and update policy, among others [Deshpande et al., 2007; Ilarri et al., 2010]. [Ilarri et al., 2010] have classified location-dependent query processing approaches into three categories: (i) approaches that require cooperation from moving objects; (ii) approaches that assume known objects' trajectories; and (iii) generic query processing approaches.

1. **Cooperation from moving objects:** In this category moving objects are considered as the main processing units that can communicate location updates to a centralized server according to a certain update policy. They also have to monitor the regions they may affect with respect to the currently performed (i.e., *outstanding*) queries, and perform local query processing tasks. Examples of such approaches are *MQM (Monitoring Query Management)* [Cai et al., 2004, 2006], *MobiEyes* [Gedik and Liu, 2004, 2006], and the *SRB (Safe-Region-Based) framework* [Hu et al., 2005]. A summary that shows advantages and disadvantages of these proposals is illustrated in Table 3.1.

   (i) MQM focuses on *continuous static range queries over moving objects* and uses a partially distributed strategy (i.e., with a *mediator* server) in which moving objects monitor and

contribute to processing queries that are part of or overlap with their *resident domains*, that is, a monitoring region assigned to each moving object depending on its location. MQM aims at reducing the frequency of location updates reported to the central server, by communicating a change of location only when it leaves its *resident domain* (i.e., similar to the concept of safe region). For that purpose, a spatial index structure, referred to as *BP tree (Binary Partitioning tree)* has been proposed for storing information about queries of interest for each portion of the partitioned workspace. This approach assumes that moving objects' devices have considerable processing and communication capabilities, which is not available for all kinds of devices. Moreover, only static query points are considered, thus moving range queries as well as other kinds of navigation-related queries are not directly supported. Furthermore, the result to a given query should be completely reevaluated on each update, and only the set of identifiers of objects of interests are returned. Therefore, no information about the locations of those objects or the paths towards them is provided.

(ii) MobiEyes deals with *moving range queries over moving objects*. A slightly different technique than MQM has been proposed, which partitions the workspace into regular grid cells, and assigns a set of cells, called *monitoring region*, to each moving query point. Each moving object covered by one or more monitoring region of certain queries should then communicate its current cell to the server in order to determine the identifiers of queries that might have been affected. Similarly to MQM, MobiEyes does not consider an incremental evaluation mechanism in query processing, but rather tries to reduce the cost of query reevaluation by minimizing the number of updates. MobiEyes does not support different kinds of queries, and does not provide information about the locations of the objects of interest but rather just their identifiers. An optimization of this approach that aimed at providing the continuous evaluation support has been presented in [Gedik and Liu, 2006]. The concept of *motion-sensitive bounding boxes* has been introduced so that moving objects as well as moving query points are stored along with boxes which are adapted to their motion (i.e., speed and direction) in order to reduce the cost of query reevaluation.

(iii) The *SRB (Safe-Region-Based) framework* approach introduced the concept of *active location update* that considers two types of updates [Hu et al., 2005]: *source-initiated updates* that occur only if a moving object identifies itself within the range of one or more outstanding queries; and *server-initiated updates* where the server explicitly asks an object for a location update due to changes of other objects' locations that might have affected that object's safe region. This approach was only applied to static range and kNN queries over moving objects.

2. **Approaches with known objects' trajectories:** Those approaches mainly consider moving objects that follow certain routes or trajectories. This allows to estimate or predict near-future

	Proposal	Advantages	Disadvantages
MQM	- Static range queries over moving objects - Resident domains - BP tree	- Partially distributed - Reduces location updates	- Considerable processing at mobile devices - No moving queries - Reevaluation on update
MobiEyes	- Moving range queries over moving objects - Monitoring regions on queries - Motion-sensitive bounding boxes	- Partially distributed - Reduces location updates	- Considerable processing at mobile devices - Does not return objects' locations - Reevaluation on update
SRB	- Active location update - Static range and NN queries over moving objects - Safe region	- Saves location updates	- No moving queries - No interest in locations - Reevaluation on update

Table 3.1: Cooperation from moving objects [Adapted from [Ilarri et al., 2010]]

locations of objects, thus reducing the frequency of location updates to be communicated to the server. However, this could present some limitations depending on the applications they are designed for. For instance, although the movements of objects in indoor environments are constrained by the structure of space (i.e., walls, stairs, etc.), objects are assumed to be moving freely in halls, rooms or malls. Therefore, a prior knowledge of objects' trajectories in such situations is not realistic. Relevant works in this category are discussed as follows (cf., Table 3.2).

(i) The *Domino* project, presented in [Sistla et al., 1997; Wolfson et al., 1999a,b], has focused on modelling and querying moving objects in databases. The concept of *dynamic attribute* has been proposed, which implies representing and storing a moving object location as a motion vector that depends on the speed of the object. A time-dependent function has been designed to compute near-future positions following a trajectory on the road network. An update policy, called *"dead-reckoning policy"*, that consists in triggering a location update request upon reaching a certain accuracy threshold. Although Domino have provided relevant answers to some crucial questions in modelling and querying moving objects, techniques for continuous processing have not been discussed in detail.

(ii) A comprehensive data model, a query language, and an architecture for managing and querying objects moving in road networks have been presented in [Güting et al., 2005, 2006]. This approach supports handling and querying historical spatio-temporal queries over moving objects, thanks to an extensible framework called *Secondo*. Abstract data types, operators, and algorithms have been introduced to enable more efficient network-based models, so that objects' movements can be represented as trajectories. Based on this prototype, several proposals for

efficient indexing and query processing have been presented [De Almeida and Güting, 2005, 2006; Güting et al., 2010]. As an example, the work presented in [Güting et al., 2010] addresses the problem of finding the continuous k nearest neighbours to a query trajectory in a large set of stored trajectories. In contrast, *MON-Tree* has been proposed in [De Almeida and Güting, 2005] as an index structure for moving objects on networks. The index structure is composed of a 2D R-Tree for indexing polyline bounding boxes and a set of 2D R-Trees for indexing objects' movements along the polylines. Despite the usefulness and extensibility of this framework, the actual goal behind this design was always about querying histories of movements. There has been no direct work on addressing continuous query processing applied to current movements. Nevertheless, adjustments can still be made to adapt its behaviour to deal with such kinds of queries.

(iii) *Hermes*,on the other hand, has been designed as a system extension that provides spatio-temporal functionality to ORDBMS (beyond those spatial functionalities presented in Oracle Spatial and PostGIS) [Pelekis et al., 2006, 2007, 2008, 2010]. Hermes provides data management infrastructure for handling histories of movements with several types of queries in free Euclidean space. Examples of queries are nearest neighbour searches on moving object trajectories and trajectory similarities [Frentzos et al., 2007; Pelekis et al., 2007]. Similarly to Secondo, this approach focuses only moving object trajectories and does not cover the area of continuous processing of location-dependent queries in real-time systems.

	Proposal	**Advantages**	**Disadvantages**
DOMINO	- MOST model - Dynamic attribute - Predictive queries	- Dynamic motion vectors - Temporal properties - Spatial networks	- No continuous processing
SECONDO	- Historical spatio-temporal queries - Trajectories of movements - MON-tree	- Efficient processing of historical queries - Network-based model	- No continuous processing of current movements
HERMES	- Historical spatio-temporal queries - Trajectory similarity	- Adds spatio-temporal functionality to ORDBMS	- No continuous processing of current movements

Table 3.2: Approaches that assume known objects' trajectories

3. **Generic query processing approaches:** This category encompasses techniques that require a minimum contribution from moving objects, and with few assumptions regarding, for instance, the prior knowledge of objects' trajectories or patterns. Some of those architectures are described as follows (See Table 3.3 for a summary of these approaches).

(i) The Monash University project considers, on the other hand, a distributed strategy to process range queries over static objects [Jayaputera and Taniar, 2005a,b]. Amongst the limitations of this approach is that it does not consider moving target objects, but only moving query points. Moreover, query points can only detect objects that match the same horizontal or vertical directions.

(ii) *LOQOMOTION (LOcation-dependent Queries On Moving ObjecTs In mObile Networks)* is an agent-based platform that focuses on distributed processing of location-dependent queries over moving objects [Ilarri et al., 2006a,b]. A generic architecture that could be applied to different kinds of queries (mainly moving range queries have been tested), which consists in deploying a hierarchy of agents with different roles in a completely distributed environment for monitoring purposes. Those agents can move freely on a network of fixed computers in order to track the locations of relevant moving objects, correlate partial results of local queries, and continuously adapting the final answer and presenting it to the user. No assumption has been made about storing and retrieving location data, and the current locations of the resulting objects may be returned. However, LOQOMOTION assumes a free space and Euclidean distances are applied for range queries, which could present an important limitation in road-network-based applications or in indoor environments. Moreover, this generic architecture could result in a less efficient query processing when compared to specialised approaches that focus on a specific type of queries.

(iii) Finally, a general multi-threading based framework for continuous queries over moving objects has been proposed in [Zhao et al., 2010]. This proposal adopts a multi-threading computing paradigm and assumes that continuous queries are periodically reevaluated. Three types of threads were used: (1) *update threads* in charge of periodically updating the locations of moving objects and query points; (2) *re-order threads* in charge of resorting the grid indexes used for both the objects and the queries, with the aim of optimizing the cache access; and (3) *query threads* that contribute to the processing of a uniformly partitioned working set and to writing query results. A kNN query algorithm with two variants: *query index* and *object index* has been designed to validate this proposal. Nevertheless, there has been no incremental approach for handling continuous queries, and only Euclidean distances were considered in query processing.

3.2.2 Approaches for location-dependent query processing over data streams

Another category of works proposes to handle continuous location-dependent queries based on *Data Stream Management Systems (DSMS)* [Chandrasekaran et al., 2003; Mokbel et al., 2005; Nehme and Rundensteiner, 2006; Chow et al., 2009]. In contrast to moving object databases, the locations

	Proposal	Advantages	Disadvantages
Monash University	- Range queries - Static objects	- Distributed architecture	- No moving target objects - No continuous queries - Limitations on user's directions
LOQOMOTION	- Generic agent-based architecture - Mainly moving range queries over moving objects - Hierarchy of agents for monitoring	- Distributed architecture - Continuous queries - No assumptions on user's movements	- Free Euclidean space - Difficult optimization for special purposes
[Zhao et al., 2010]	- Multi-threading framework - Continuous queries over moving objects	- Optimized cache acess - Periodic reevaluation	- No incremental processing - Free Euclidean space

Table 3.3: Generic query processing architectures in moving object databases

of moving objects that are received in real-time can be considered as data streams, where new tuples are continuously released and processed *without necessarily being stored permanently in the database*. Several data stream management systems have been recently proposed and were used later for location-dependent query processing over data streams, such as NiagaraCQ [Chen et al., 2000], Telegraph/TelegraphCQ [Chandrasekaran et al., 2003], STREAM [Arasu et al., 2004a], and Nile [Hammad et al., 2004]. Some of the approaches that use the concept of data streams for defining the semantics of continuous queries in mobile environments are summarized in Table 3.4 and described as follows.

1. The *PLACE (Pervasive Location-Aware Computing Environments)* project was developed on top of a data stream management system called *Nile* [Hammad et al., 2004; Mokbel et al., 2005], and provided functionalities for the processing of continuous spatio-temporal streams in highly dynamic environments. PLACE is meant to be as generic as possible so that different category of continuous queries can be supported. For instance, the *SINA* algorithm has been especially designed for the continuous processing of moving range queries over moving objects [Mokbel et al., 2004], in which *grid indexes* have been used for storing both the query points and the moving objects. This algorithm continuously applies an in-memory hash-based join technique between query points and objects as location updates arrive. An approach for continuous kNN queries on road networks has been also presented in [Xiong et al., 2005]. An incremental memory-based evaluation paradigm has been further introduced, which focuses on only computing and reporting changes to the last evaluated answer. This is mainly achieved through the concept of negative/positive updates that

tell whether an object should be removed/added from/to the answer. Moreover, the concept of predicate-based sliding windows has been introduced, which allows objects that satisfy a certain query predicate in real-time to be part of the current window.

2. A *Scalable Cluster Based Algorithm (SCUBA)* has been proposed in [Nehme and Rundensteiner, 2006], with the aim of processing a large set of continuous queries over spatio- temporal data streams. SCUBA processes concurrent moving range queries by grouping moving objects and query points into moving clusters based on common spatio-temporal properties. An incremental clustering mechanism is performed mainly by maintaining two spatio-temporal thresholds: the distance speed thresholds. This optimizes query execution at run-time, and thus favours scalability. SCUBA is implemented within the stream processing system *CAPE* [Rundensteiner et al., 2005]. This approach is well suited for scenarios where groups of objects are moving together, such as groups in tourism trips, traffic jams, and animal and birds migrations, etc. However, in other scenarios where moving objects do not present similar spatio-temporal patterns, this technique is not suitable.

3. Other research studies have focused on managing trajectories of moving objects and on real-time monitoring over streaming traffic data by building techniques upon the TelegraphCQ DSMS [Patroumpas and Sellis, 2004, 2012]. The Open-Source *TelegraphCQ* stream engine has been built on top of PostgreSQL RDBMS [Chandrasekaran et al., 2003]. Hence, it comes readily equipped with built-in spatial operators, functions and data types, offering a great benefit for expressing continuous queries over geospatial streams. [Patroumpas and Sellis, 2004] proposed to represent the continuous location data flow of moving objects as *trajectory streams* evolving in space and time. Whereas in [Patroumpas and Sellis, 2012], an abstract model for representing traffic data streams in congested road networks was presented. Applications of these *geostreaming* techniques include online monitoring of *location-aware vehicles*.

The aforementioned works developed on top of data stream management systems suggest that such a data stream paradigm can be powerful enough to manage data about moving objects. This trend needs further investigation regarding the appropriate continuous window operators to be used (e.g., time-based, tuple-count, and predicate-based sliding windows), the indexing schemes, and the types of queries that can be supported. [Mokbel and Aref, 2008] have presented the *scalable on-line execution (SOLE)* algorithm for continuous evaluation of concurrent spatio-temporal queries over data streams. This approach claims to be bridging the areas of moving object databases and data stream management systems. SOLE favours in-memory processing of concurrent continuous queries and uses an incremental evaluation paradigm, a shared grid structure as a basis for shared execution, and a hash table to index moving objects. SOLE is also assumed to be able to deal with range queries as well as kNN queries. However, SOLE only returns the set of qualifying objects without

	Proposal	Advantages	Disadvantages
PLACE	- Continuous queries over data streams - SINA & SEA-CNN - Grid index - Predicate windows	- Incremental evaluation - Moving range and NN queries - No assumptions on user's movements	- No interest in locations - No support for path queries
SCUBA	- Concurrent moving range queries - Spatio-temporal clusters - Data streams	- Scalable and optimized query execution - Incremental evaluation	- Not suitable for generic purposes
[Patroumpas and Sellis, 2004]	- Managing traectories of moving objects - TelegraphCQ - Geostreaming	- Real-time monitoring of traffic data - Network-based model	- No continuous processing of current movements
SOLE	- Concurrent queries over data streams - In-memory processing - Shared execution	- Incremental evaluation - Bridging links between MOD and DSMS - Range & kNN queries	- No interest in locations - No support for path queries

Table 3.4: Approaches for query processing over spatio-temporal data streams

being concerned about the exact locations or paths towards those objects.

3.2.3 Approaches for managing and querying indoor moving objects

Few research studies have addressed the specific requirements to manage and query indoor moving objects [Jensen et al., 2010]. Two main approaches have been recently proposed in this context [Jensen et al., 2009b; Lu et al., 2012; Xie et al., 2013], and are described as follows:

1. An infrastructure for the continuous range monitoring of indoor moving objects has been introduced in [Yang et al., 2009]. This proposal is developed on top of a graph data model, and deploys a set of sensors to continuously monitor the users' movements, thus maintaining the query result up-to-date. Indoor moving objects are classified either within *active* or *inactive* subspace depending on the coverage of sensor devices. Several hashing-based indexing schemes have been designed, which mainly map positioning devices to the set of active objects that are currently located within their respective ranges, and for maintaining objects' current states. An incremental maintenance of range queries is proposed which aims at determining only new observations detected by sensor devices in order to maintain the query result. Whenever an object enters or leaves the range of a given *critical* sensor (i.e., the one from which a new observation can potentially change the query result), a query update is performed. Experimental results show that the underlying data

model is flexible, since it allows for different kinds of queries to be performed, and the solutions are efficient and scalable. However, the aim of this approach was to monitor indoor moving objects, so it only processes static range queries over moving targets, without taking into account moving query points. In addition, the model underneath relies on sensor-range-based positioning techniques, which is not perfectly suitable for navigation queries that may require fine-grained location information. Moreover, no information about the optimal path to the starting query point is obtained.

2. A framework for managing and querying indoor moving objects has been recently proposed in [Lu et al., 2012; Xie et al., 2013]. This approach proposes to manage indoor geometries (i.e., spatial entities), indoor topologies (i.e., relationships among spatial entities and between space and moving objects), as well as indoor moving objects (modelled with uncertainty regions around), with a composite index scheme in order to support efficient distance-aware queries. Geometries that represent indoor partitions are indexed by an R*-tree structure. The leaf nodes of that structure are associated with the set of moving objects currently located within the corresponding partitions. A topological layer that consists of a *door-to-door-based graph* is also used to compute paths towards moving objects. Upper and lower bounds for indoor distances are established, where the lower bounds represent the Euclidean distance that prunes away disqualified objects, whereas the upper bounds are determined based on topological constraints. This framework has been applied to indoor range and kNN query processing. Query evaluation is performed in four phases: 1) A *filtering phase* that locates the source spatial partition containing the initial query point, and retrieves candidate objects; 2) A *subgraph phase* that builds the doors graph in order to compute shortest paths towards the candidate objects; 3) A *pruning phase* that applies the upper/lower bounds on indoor distances to prune away disqualified objects; and finally 4) A *refinement phase* that computes the actual indoor distances towards the last candidates to return the qualifying objects. A similar but adapted process is applied to process kNN queries. However, only static query points were considered, and this approach did not discuss how to incrementally reevaluate those queries.

3.2.4 Towards context and preference-aware location-dependent queries

Over the past few years, several research studies have discussed the integration of some contextual dimensions in query processing [Trajcevski et al., 2004a; van Bunningen et al., 2006; Mokbel and Levandoski, 2009; Levandoski et al., 2010b; Bosc et al., 2009; Yu and Spaccapietra, 2010; Wenzel et al., 2011]. Users' preferences have traditionally been exploited in query personalization to better anticipate their needs and customize their experience. In order to deliver personalized query answering in mobile environments, context- and preference-aware query processing techniques are increasingly required. A designed query processing strategy should provide sufficient flexibility and adaptiveness

to location-based services. However, there is still a lack of approaches addressing context-awareness in continuous location-dependent query processing. Some the relevant works that might inspire further research in this area are described as follows (cf., Table 3.5).

1. *CAT (Correct Answers of continuous queries using Triggers)* is a framework that deals with continuous queries in moving object databases with a known trajectory model and taking into account certain real-time spatio-temporal events [Trajcevski et al., 2004a]. The CAT project relies on the use of triggers in databases in order to detect the relevant changes such as real-time traffic conditions. In such cases, the server requests for location updates in order to keep the data structures and the answer to active queries up-to-date. Some relevant contextual changes can be taken into account either at the server side by detecting events such as traffic jams, accidents, etc., or by the moving objects based on changes in their plans [Trajcevski et al., 2005a, 2006]. An extended version called *CAT++* has been proposed based on a new paradigm for expressing reactive behaviour in moving object databases, and referred to as *Evolving Context-Aware Event-Condition-Action* $(ECA)^2$ [Trajcevski et al., 2005b]. This approach allows for reevaluating queries when changes occur to the database, and for the processing of dynamic topological predicates such as *moving along* and *moving towards*, which is believed to be of interest for applications such as emergency response, digital battlefield, and for event notification systems.

2. An ontology-based preference-aware model for database querying in an ambient intelligent environment has been proposed in [van Bunningen et al., 2006]. The context model is based on a variant of Description Logics (DL) for knowledge representation. This model claims to support pull and push-based queries. The model has been implanted on top of a DBMS as an OWL-based knowledge base for reasoning. However, this approach is not directly applied to the continuous processing of location-dependent queries and for handling moving objects. [Yu and Spaccapietra, 2010] presented another ontology-based and application-independent knowledge framework for location-based services querying. This knowledge is dynamically maintained and used to reformulate user's queries via personalized and contextualized rewriting. The architecture is made of ontology modules that manage user profiles, preferences, and service accessibility; and a mechanism for information profiling and semantics matching. Nevertheless, real-time event management have not been directly addressed, and the architecture was not applied to location-dependent queries over moving objects.

3. Mokbel and his colleagues recently proposed two prototypes called *FlexPref* and *CareDB* with the aim of incorporating context-awareness in DBMS to provide adaptive answers to spatio-temporal queries [Mokbel and Levandoski, 2009; Levandoski et al., 2010a,b]. FlexPref was first presented as a framework implemented in PostgreSQL for preference evaluation in database systems. The framework

allows new preference-based methods to be plugged into it. Once integrated, the preference method lives at the core of the database, enabling the execution of preference queries involving common database operations. CareDB has further extended FlexPref by integrating a taxonomy of preference types and contextual data (e.g., user profile, traffic conditions, weather) in query processing, and by handling data uncertainty [Levandoski et al., 2010b]. Although this approach appears to be very promising, it has not yet been applied to continuous location-dependent queries.

4. Within the context of *fuzzy queries* that deal with *"soft"* criteria or constraints trying to fulfil them as closely as possible, several research studies has been conducted such as *PreferenceSQL* [Kießling and Köstler, 2002; Kießling et al., 2011], and the approach developed in [Bosc et al., 2009; Mokhtari, 2011]. PreferenceSQL extends standard SQL with a preference model, whose keywords and operators are discussed in Section 3.4, as well as an optimizer in charge of translating preference queries into standard queries. [Bosc et al., 2009] have considered the interpretation and processing of fuzzy queries by incorporating contextual predicates on attributes such as *low*, *medium*, and *high*. However, those approaches were not tested in location-dependent query processing. [Mokhtari, 2011] has recently proposed an approach based on fuzzy logic theory for expressing and processing route planning queries in roads networks. This approach identifies ambiguous and possibly contradictory preferences and to provide a selection mechanism that retrieves the tuples which satisfy as closely as possible those fuzzy constraints. This proposal does not provide an mechanism for continuous processing of location-dependent queries.

Consequently, there is a promising trend to discuss and deal with preferences and other contextual data beyond space and time. Approaches that address this issue are still not completely applied to highly dynamic environments, and to process continuous location-dependent queries. Real-time event management still appears to be an important challenge to cope with in such environments, since an entire dedicated sensor-based infrastructure is required to communicate changes in real-time, and to perform adaptation to query answers accordingly.

3.3 Continuous processing of navigation-related queries

While Section 3.2 focuses on architectures that deal with location-dependent query processing over moving objects, this section takes a different perspective to discuss the incremental execution paradigm at the algorithmic level to process location-dependent queries. The incremental execution paradigm implies reusing information from previous searches in order to obtain the current result adaptively without having to reevaluate everything from scratch.

As suggested in Chapter 2, graph-based data models are well suited to compute an optimal and

	Proposal	Advantages	Disadvantages
CAT	- Continuous queries using triggers - Active rules - $(ECA)^2$	- Periodic reevaluation - Detection of real-time events - Predctive queries	- No incremental evaluation - No moving queries
[van Bunningen et al., 2006]	- Ontology-based model - Preference-aware queries	- Contextualized queries	- Not applied to continuous LDQs
FlexPref & CareDB	- Preference evaluation in DBMS (FlexRef) - Contextual data (CareDB) - Data uncertainty	- Context-awareness in DMBMS	- Not applied to continuous LDQs
PreferenceSQL	- Fuzzy queries - Preference-aware queries	- Fuzzy preferences in DBMS	- Not applied to location-dependent queries
[Bosc et al., 2009; Hadjali et al., 2012]	- Fuzzy queries - Preference-aware queries - Route planning ([Hadjali et al., 2012])	- Fuzzy preferences - Preference-aware route planning	- No support for continuous LDQs

Table 3.5: Context- and preference-aware query processing approaches

realistic route to a destination by taking into consideration architectural constraints and dynamic changes in the environment. Particularly, the Euclidean distance is meaningless to compute routes in road networks or in indoor spaces, due to path constraints. Therefore, approaches for query processing based on the network distance are preferred and more realistic. However, existing approaches for network-based query processing usually assume an outdoor environment (e.g., [Deng et al., 2009; Lee et al., 2005; Papadias et al., 2003]), where for example hierarchical networks do not naturally appear and there are no accessibility rules based on user profiles. Nevertheless, recent works have studied location-dependent queries in indoor environments [Yang et al., 2009, 2010; Yuan and Schneider, 2010].

Navigation-related queries, and more specifically, queries for computing optimal paths between static and/or moving objects are very relevant for routing services (e.g., "guide-me", "get-together", "find-me"). Range queries can also be used in many indoor location-based services for monitoring purposes (e.g., "what-is-around" services, crowd monitoring within a given area, location-based alerts, etc.). Other location-dependent queries such as k nearest neighbour and reachability queries are also considered (e.g., find the nearest OOI). This section first discusses foundations of query processing in spatial network databases (cf., Section 3.3.1). Related work on continuous path and range queries over moving objects is presented in detail in Sections 3.3.2 and 3.3.3, as algorithms to address those queries in indoor mobile environments are introduced in Chapter 6. Section 3.3.4 presents works on

processing kNN and reachability queries.

3.3.1 Query processing in spatial network databases

Due to the importance of spatial networks in real-life applications, intensified efforts have been made in the last decade to bring up new foundations for query processing in spatial databases [Papadias et al., 2003; Deng et al., 2009]. These approaches have mainly focused on efficient query processing in spatial networks, but only with static query points and/or over static data. Related work on static query processing in spatial networks is discussed hereafter.

1. [Papadias et al., 2003] have emphasized the importance of using spatial networks instead of Euclidean spaces for location-based query processing. They have introduced the *Incremental Euclidean Restriction* (IER) and the *Incremental* as well as *Range Network Expansion* (INE and RNE, respectively) mechanisms for the processing of different kinds of queries such as *nearest neighbour, range, and closest pair* queries on road networks. IER performs pre-processing in Euclidean space first in order to retrieve valid candidates. IER works with restrictions on data points rather than nodes of the network. These candidate objects are then subject to network distance computation to decide whether they are of interest or not. In INE, a *wavefront* is gradually expanded in networks from the initial query point such that the data points closer to the query point are visited earlier than the others. These foundations have been afterwards adopted in many research studies on spatial network databases [Mouratidis et al., 2006; Demiryurek et al., 2009; Xuan et al., 2010, 2011]. However, these techniques were not applied to the processing of such queries over moving objects, since only static query and data points were considered. The incremental approach was used to apply a multi-step methodology that gradually responds to a given query, but without discussing a continuous reevaluation strategy to deal with moving objects.

2. Performance optimization of query processing in spatial networks has been the focus of the work presented in [Deng et al., 2009]. A framework that extends IER and INE techniques has been proposed. INE computes the shortest network path from each candidate (i.e., target object) to the query point which is very costly. An incremental lower bound constraint technique has been introduced which extends INE by restricting network expansion to only the paths towards candidate target objects. Those candidates are in turn determined by performing a prior step of incremental Euclidean restriction on data points. Approximate static RER and RNE solutions have been presented in [AL-Khalidi et al., 2013], which tried to reduce the number of false hits, and the number of network distance computations of the corresponding techniques.

Although these mechanisms only assume static query points and applied only to static data, they constitute a key building block in other research studies that focus on the continuous processing of location dependent queries.

3.3.2 Path queries

Shortest path algorithms have been the subject of extensive research for many years resulting in a large number of algorithms for various scenarios and in different environments [Orda and Rom, 1990; Sommer, 2010; Wu et al., 2012]. Most of these algorithms deal with fixed graphs (i.e., fixed topology and fixed link weights). Path queries imply finding an optimal route to a specified place or an object of interest. Both static and/or moving target objects can be considered in this kind of query. This section first presents approaches for *multi-criteria and hierarchical path search* considering static objects. Techniques for *continuous processing of path queries* over moving objects are then discussed while taking into account different types of environment (i.e., Euclidean spaces with obstacles as well as spatial networks).

3.3.2.1 Multi-criteria & hierarchical path searches

As discussed in Chapter 1 (cf., Section 1.2), A path search needs to aggregate multiple criteria (e.g., user preferences and capabilities, distance, time, etc.) passed as function parameters to evaluate the cost value of each step and then to select the most appropriate path, that is, the optimal path that allows, for instance, to reach a destination while avoiding threats [Disser et al., 2008; Delling et al., 2009]. On the other hand, hierarchical routing algorithms became important tools in querying databases for the shortest paths in time-critical applications like Intelligent Transportation Systems (ITS), due to the growing size of their spatial networks [Shekhar et al., 1997].

Existing approaches on multi-criteria and multi-modal path searches consider dynamic environments in which edge weights can change dynamically according to time (e.g., opening hours), real-time events (e.g., traffic conditions) or to the different kinds of transportation modes available for a given path search [Peytchev and Claramunt, 2001; Bielli et al., 2006; Ding et al., 2008; Berger et al., 2010; Sommer, 2010]. Particularly, *time-dependent* shortest paths in dynamic networks have been proposed in [Ding et al., 2008; Berger et al., 2010]. [Ding et al., 2008] designed an edge-delayed function, associated with each edge and stored in database, which computes the *best travel time* from a source to a destination by taking into account the departure time. This favours queries such as the minimum-travel-time path, with the best departure time to be selected from a time interval. [Berger et al., 2010] introduced a technique for time-dependent shortest paths, which allows

to handle dynamic changes. A dynamic timetable that stores departure and arrival events (i.e., mainly departure and arrival times in train networks) is maintained. Edges can be delayed or even become temporarily unavailable to reflect events derived from real-time traffic data.

Hierarchical path search approaches have also been explored in many research studies [Shekhar et al., 1997; Jung and Pramanik, 2002; Botea et al., 2004; Cagigas and Abascal, 2004; Cagigas, 2005; Guerra-Filho and Samet, 2006; Geisberger et al., 2008; Rice and Tsotras, 2010; Seder et al., 2011]. They are mainly applied to two-dimensional occupancy grid maps representing the environment. Specific so-called *bridge* nodes (i.e., nodes that connect submaps to a parent map) are then determined to create the hierarchy. The key issue lies in choosing the optimal placement of such bridge nodes. For instance, the *HiTi* graph model provides a mechanism for structuring a topographical road map in a hierarchical fashion, to efficiently compute the optimal minimum cost path [Jung and Pramanik, 2002]. On the other side, a hierarchical but static variant of A*, referred to as *Hierarchical Path-finding A** (HPA*), has been proposed in [Botea et al., 2004], which decomposes a grid map into linked clusters and pre-computes optimal distances for crossing clusters at an abstract level. A bottom-up approach applied on a two-level hierarchy has been proposed. The technique has been used for path planning computations applied to static data only, and not to moving objects. An off-line hierarchical path planner module of a smart wheelchair-aided navigation system has been described in [Cagigas and Abascal, 2004]. This approach uses hierarchies of abstraction and was applied for indoor navigation of a wheelchair user between floors of a building. However, on-line path planning and obstacle avoidance are not described in this proposal.

The *Hierarchical D* (HD*)* [Cagigas, 2005] and the *Focused Hierarchical D* (FHD*)* [Seder et al., 2011] both deal with hierarchical path searches in dynamic indoor environments. These techniques are based on the *D** (Dynamic A*) [Stentz, 1994] and the *Focused D** [Stentz, 1995], which assume dynamically changing environments (e.g., transient obstacles) and partially unknown environments to be explored by robots. The HD* algorithm precomputes the set of paths between bridge nodes based on A*. HD* tries to to find the best path between the submaps where the start and goal nodes are located. FHD* ensures optimality of te global path by identifying the optimal placement of the bridge nodes, and decreases computational complexity of path replanning in real-time by reducing the search scope to the area around the optimal path.

An iterative (non-recursive) algorithm to create a hierarchy of networks, and to find a shortest path through all levels of the hierarchy has also been proposed by [Guerra-Filho and Samet, 2006]. An approach for indexing road networks to enable efficient path query processing has been presented in [Rice and Tsotras, 2010]. This approach uses a bidirectional Dijkstra algorithm and extends the hierarchical graph indexing approach known as Contraction Hierarchies [Geisberger et al., 2008]. The idea behind is to identify a limited set of possible constraints and changes that might affect the

shortest path search on road networks, and then to incorporate those constraints directly into the graph index construction in order to avoid the overhead of reconstructing the index for each possible constraint scenario at query time.

It should be emphasized that the above described approaches work only on static objects (i.e., nodes of the graph), and some have discussed path planning in dynamic environments meaning that multiple criteria could be mixed together in order to compute a cost function that delivers more appropriate paths. Nevertheless, a few research studies have lately been presented for the processing of path queries over moving objects. These techniques are highlighted in the following section, especially those that employ an incremental and continuous query processing approach.

3.3.2.2 Continuous path search algorithms

Efforts have been recently made to deal with the challenge of continuously processing path queries over moving objects located on a spatial network [Chon et al., 2003; Lee et al., 2007; Yin et al., 2008; Song et al., 2008; Sun et al., 2009, 2010a,b; Malviya et al., 2011]. Some of them considered moving objects that are following known paths such as vehicles, and designed a distributed architecture to maintain a global view of the resulting route for a given query, so that any dynamic change reported from local views can be returned to the user [Chon et al., 2003]. This approach maintains two queues for continuous monitoring of updates (e.g., routes congestion) that can affect the path for each moving object until that object reaches its destination. However, this approach does not handle moving targets and the continuous monitoring was performed by a set of periodical snapshots of the system.

The approach presented by [Lee et al., 2007] employs a mechanism to monitor the specified area for the continuous evaluation of fastest path queries. This mechanism eliminates unnecessary reevaluations of fastest path queries, and reduces the search space for finding the fastest path. In addition, a grid-based index has been proposed to increase the efficiency of multiple query processing. This technique handles changes in location of the query point, but was only applied to static destination. Similarly, the problem of building a traffic aware dynamic route planning service was considered in [Malviya et al., 2011]. The authors have introduced the *k candidate paths* technique which maintains the set of *pre-designated* or preprocessed routes between a given source and target, and updates those routes as traffic delays on road segments change while trying to avoid complete reevaluation of the query.

Sun et al. have proposed a series of A*-based algorithms referred to as *Fringe-retrieving A** -FRA*- [Sun et al., 2009], *Generalized FRA** [Sun et al., 2010a], and *Moving target D* lite* [Sun

et al., 2010b], that aim at providing an efficient incremental approach for moving target path search in dynamic environments. In particular, FRA* is an incremental version of A* that is applied on moving targets in grid maps, and aims at repeatedly finding the shortest path without having to process each search iteration independently from scratch. This algorithm regularly transforms the previous search tree to an updated tree based on the new locations of the objects of interest. The current search tree is always rooted at the current location of the query point. Each cell within the search tree maintains a pointer to its parent cell, so that a shortest path to the root can be directly obtained by traversing the tree in reverse following the ancestor nodes. Although this algorithm performs well on moving objects, its scalability to a large environment (i.e., a large campus with multi-storey buildings) is still an issue to address, since fine-grained grids have been adopted, which have not be proven to be scalable to large spaces. Moreover, FRA* does not take into account the hierarchical structure of an indoor environment, and thus cannot handle continuous path searches in multi-storey buildings. GFRA* is a generalized approach that further solves moving target search problems on arbitrary graphs, including the state lattices used for robotics applications [Sun et al., 2010a].

Both the incremental processing provided in FRA* and the hierarchical approach of HPA* have inspired our work on the continuous processing of hierarchical path searches over moving objects described in Chapter 6. Our approach transforms an initial search tree to an updated tree depending on the movements of the objects and the changes in the environment.

3.3.3 Range queries

Range queries are used to retrieve information about objects or places within a specified range or area[Zhang et al., 2003]. Some range queries have a static query point and others have a moving query point. Similarly, the target objects of the queries can be static or moving. While approaches on static range query processing in spatial network databases have been discussed in Section 3.3.1, this section presents techniques for an efficient evaluation of continuous range queries on moving objects. It should be also emphasized that, in contrast to Section 3.2 that describes architectures and strategies, this part deals with algorithmic challenges and related work regarding the continuous processing of such queries.

A technique presented in [Lazaridis et al., 2002] as an attempt to process *non-predictive dynamic queries*, mainly covers moving range queries, as well as *predictive dynamic queries* by employing a motion indexing technique for storing moving objects. A multi-dimensional R-tree-based data structure has been further used to index the bounding boxes representing objects' motion. This implies saving disk access since motion segments that do not intersect with the query are not updated

even though their bounding boxes do. However, this work proposes a periodic reevaluation of continuous queries from scratch.

Query Indexing [Prabhakar et al., 2002; Kalashnikov et al., 2004] is a technique for the continuous processing of multiple static range queries over moving objects. The objective was to overcome disadvantages of traditional indexing techniques by mainly indexing static query points instead of objects, since they have an update frequency that is much lower than moving objects. An incremental evaluation mechanism that consisted of determining, for each change of object's location, the set of queries in the index that might be affected. This is implemented by adopting a similar concept of *safe region* in which an object does not need to update its location. Another variant of this approach called *Velocity Constrained Indexing (VCI)* was introduced, which consists of an R-tree index that stores the minimum bounding boxes of each moving object taking into account the maximum speed of that object. Although Q-index appears to have outperformed traditional indexing schemes, it suffers from expensive operations upon the arrival of new queries and it is only applied to static query points.

Related work on continuous processing of range queries in spatial network databases can be found in [Stojanovic et al., 2008; Xuan et al., 2010; Cheema et al., 2011; Wang and Zimmermann, 2011; Xuan et al., 2011]. The *ARGONAUT* prototype is an approach built based on a known trajectory model [Stojanovic et al., 2008]. This project has been developed on top of a model for representing and querying moving objects called *mSTOMM (Spatio-Temporal Object Modeling and Management)*, and provides data management and query processing facilities, especially to perform *continuous moving range queries in road networks*. An in-memory indexing scheme based on R*-tree for storing segments of the spatial network has been further developed. Moreover, an incremental evaluation paradigm is proposed for continuous range monitoring, which include a *filter step* that results in a set of candidate moving objects by pruning objects whose *minimum bounding rectangle (MBR)* of the current segment overlaps with the query range, but their actual network distance is greater than the query range. A *refinement step* is then periodically performed for the continuous query reevaluation. This approach is not applied to other kinds of queries, and does not return the locations of resulting objects, neither the paths towards them.

A time constrained continuous range search algorithm applied to moving objects has been proposed in [Xuan et al., 2010, 2011], which aims at retrieving all objects of interest within the specified range and in a given time quantum. Their approach is based on double weighted *Network Voronoi Diagram* representing time and distance, and built around the query point and by taking into account the data points in neighbouring components. It makes use of the *Voronoi-based Range Search (VRS)* on top of the road network to reduce the expansion area and further enhance efficiency. The continuous range search proposed in this approach considers a moving query point, while data

points (i.e., target objects) are static. No incremental approach has been discussed, but rather a policy to decrease the frequency of updating.

[Cheema et al., 2011] have studied the continuous monitoring of moving range queries over static data objects applied to Euclidean spaces as well as to road networks. This approach employs the concept of a safe zone which guarantees that the results of the query do not change as long as the query remains within that zone. [Wang and Zimmermann, 2011] have designed a specific structure called *Shortest-Distance-based tree (SD-tree)* hat preserves the network connectivity and distance information in continuous query processing, thus speeding-up the network expansion process. This approach considers moving query points on moving target objects. An algorithm has been proposed that can rotate, truncate, and extend the edges of the SD-tree with regard to the query point movements. It uses periodic sampling on moving object positions, so that a number of object updates are received and stored during each cycle. This technique does not require prior knowledge of the query point's destination during query processing. Although this technique appears to be very close to our proposal described in Chapter 6, it does not consider dynamic networks and does not return the locations of resulting objects, neither the paths towards them.

A recent work has studied continuous range queries in indoor environments [Yuan and Schneider, 2010]. A specific graph data model that represents an indoor space has been designed, thus allowing the processing of specific kinds of queries on top of the generated spatial network. In [Yuan and Schneider, 2010], the authors have introduced an approach to support range queries based on a virtual cell-based network generated for each query. Besides, an extension of this method has been proposed in the same paper to continuously process range queries whenever the query point moves. However, this approach deals only with range queries, and is only applied to static data (i.e., static points of interest). Moreover, for each query, a new virtual network that connects the query point to predetermined points of interest is required, and additional computations are also needed to update the network each time the query point leaves its safe area. Furthermore, qualifying data points that are within the specified range are returned, without information about optimal paths towards those points.

3.3.4 Other kinds of location-dependent queries

This section briefly surveys the state-of-art of other kinds of location-dependent query processing over moving objects, such as the well known k Nearest Neighbour queries and other navigation-related queries. Although kNN query processing is as much important as the previously described queries (i.e., path and range queries), this thesis does not cover the continuous query processing of such queries due to lack of space and time. Nevertheless, those queries are represented and taken into

account in the whole system and the query language described later in Chapter 5. Moreover, the indoor data model introduced in Chapter 4 is sufficiently flexible to support the continuous processing of those queries as will be shown later. Therefore, this section covers the most relevant research studies in this area, and highlights the pros and cons of currently available solutions for further development.

3.3.4.1 Nearest neighbour queries

The basic concept behind k Nearest Neighbour (kNN) queries is to search for the k closest qualifying objects to a given query point in space [Roussopoulos et al., 1995]. Although nearest neighbour queries have been analysed extensively, the implications of properties and constraints in highly dynamic environments still require more investigation. Particularly, continuous nearest neighbour searches over moving objects have increasingly attracted the interest of researchers in the field [Tao et al., 2002; Cao et al., 2003; Mouratidis et al., 2006; Demiryurek et al., 2009; Güting et al., 2009; Demiryurek et al., 2010; Gao et al., 2011; Sistla et al., 2011; Elmongui et al., 2013].

[Tao et al., 2002] addressed continuous moving nearest neighbour queries on static data, where a moving query point follows a known segment in an Euclidean space. This approach tries to avoid *false misses* and the high processing cost, and uses R-tree for continuous query processing. Several heuristic methods are also used to avoid accessing nodes, if they cannot contain qualifying data. [Cao et al., 2003] have proposed a solution for kNN query processing with an approach to index static target objects within a *reachability graph* for supporting *attainability* (i.e., real driving distance in road networks). The reachability graph index is built based on actual driving distance from a moving vehicle to nearby static objects. Graph exploration techniques are used to retrieve nearest target objects from the index. An architecture for location-dependent query processing has been proposed whose main components aim at managing the mobile client's current location and predicting its future position during query execution. Again, this approach only handles static target objects and for only one type of queries.

The continuous monitoring of moving nearest neighbour queries applied to moving objects in dynamic road networks has been discussed in [Mouratidis et al., 2006]. An in-memory monitoring algorithm based on the previously described incremental network expansion (INE) mechanism is proposed. This algorithm computes the shortest paths from the source towards nodes of the expanded network to form an *expansion tree*, which is used to effectively determine the relevant updates that might affect the answer. A shared execution paradigm is also applied to reduce processing time. it groups together the queries that fall in the path between two consecutive intersections in the network, and produces their results by monitoring the NN sets of these intersections.

According to [Demiryurek et al., 2009], the approach presented in [Mouratidis et al., 2006] fails to scale in real-world scenarios where the spatial network is large and the data objects moving on the network are numerous. Consequently, a hierarchical approach for network expansion might be of great interest to efficiently process continuous queries. In order to increase the efficiency of the algorithms described in [Mouratidis et al., 2006], two techniques referred to as *guided search*, that makes use of the Euclidean restriction mechanism (described in Section 3.3.1), and *localized mapping*, that only maps a location update to the network if it is relevant to the result of the query, have been proposed in [Demiryurek et al., 2009]. Time-dependent edge weights in road networks were further considered in [Demiryurek et al., 2010], but on static kNN queries. Other solutions have addressed continuous k nearest neighbour query processing on moving object trajectories [Güting et al., 2009]. The impact of obstacles on visibility between objects for continuous nearest neighbour search but in Euclidean space has been further considered in [Gao et al., 2011].

The only work we are aware of in the context of indoor environments for supporting k nearest neighbour search has been provided in [Yang et al., 2010]. This approach is developed on top of a door-to-door graph data model. This technique deploys a set of sensors to continuously monitor users' movements, thus maintaining the query result up-to-date. It also uses a probability estimation mechanism to prune unqualified candidates from the candidate set, so that the most probable k nearest neighbours are retrieved. The results show that the underlying data model is flexible, since it allows for different kinds of queries to be performed, and the solutions on top of these foundations are efficient and scalable. However, this approach only considers static query points, and the model underneath depends on sensor-range-based positioning techniques, which are not perfectly suitable for navigation queries that may require fine-grained location information. Moreover, other contextual dimensions such as time and user profiles are not considered in query processing. Furthermore, in the case of large indoor spaces, a generic architecture that allows distributing and managing data over several pieces of a database would still be required.

3.3.4.2 Reachability Queries and Reverse Range and kNN Queries

Other kinds of location-dependent and navigation-related queries have been recently studied [Bao et al., 2010; Shang et al., 2011; Shirani-Mehr et al., 2012]. Finding the most accessible locations given a trajectory dataset in road networks has been explored by [Shang et al., 2011] and was referred to as *Reverse Path Nearest Neighbor (R-PNN)*. Similarly, [Shirani-Mehr et al., 2012] studied *Reachability queries* in spatio-temporal databases, which evaluate whether two objects are *reachable* through an evolving network of contacts among trajectories of moving objects. Constrained nearest neighbour queries were also proposed by [Ferhatosmanoglu et al., 2001], which refer to nearest neighbour queries

that are constrained to a specified region. On the other hand, a *k-Range Nearest Neighbour* query in road networks finds the k nearest neighbours of every point on the road segments within a given query region based on the network distance [Bao et al., 2010]. The goal of these approaches is not to deal with the continuous processing of those queries, but rather to provide new efficient solutions to different kinds of queries required in different scenarios.

Continuous Reverse k Nearest Neighbour queries (RkNN) in both Euclidean space and spatial networks were the focus of the work presented in [Cheema et al., 2012]. A reverse k nearest neighbour (RkNN) query retrieves all the data points that have the initial query point as one of their k nearest neighbours. Continuous aggregate nearest neighbour queries for moving objects in road networks were discussed in [Elmongui et al., 2013], in which the set of k moving objects that have the smallest aggregate distance function (e.g., max, sum or min) are retrieved.

3.4 Languages for location-dependent queries

A strongly related field of interest when considering location-dependent query processing is the representation of the semantics behind those queries [Sistla et al., 1997]. Most of the query languages designed for representing location-based queries in moving object databases or those that are compliant with the spatio-temporal data stream paradigm are strongly related to the data model underneath, and on how moving object data is managed [Erwig and Schneider, 1999]. Some data models define a flexible collection of type constructors and operations, that are used to create and access the underlying data structures, and to express all queries using a few manipulation primitives [Güting, 1994]. This section highlights previous work on query languages for location-dependent queries, and focuses on the characteristics of the corresponding grammars with respect to the semantics reflected, and the preferences and constraints that might be specified by the user or inferred from his/her request and/or from his/her current context. Query languages in moving object databases are first discussed in this section, and then a brief survey on languages for handling spatio-temporal data streams is presented. The section ends by discussing approaches for querying preference-aware and contextual data.

3.4.1 Query languages in moving object databases

Future Temporal Logic (FTL) is the query language associated with the MOST model introduced by [Sistla et al., 1997] and described in Section 3.2.1. Specific temporal operators (such as *until*, *nexttime*, *eventually*, *sometimes*, *within*, and *always*) were introduced in order to handle time sensitive data, and to enable *present* and *future* queries over moving objects. Additional spatial operators such

as *inside* and *dist* have been also proposed. *Instantaneous* queries (i.e., predicates on the current database state) and *continuous* queries are considered. An answer to a continuous query is provided by using a time interval *begin* to *end*, and both *immediate* and *delayed* strategies for transmitting the result to the user are employed. [Vazirgiannis and Wolfson, 2001] further defined other modifiers such as *ALONG EXISTING PATH* and *ALONG SHORTEST PATH*, and the operands *DISTANCE* and *TRAVELTIME* for handling network-based distances and travel times. Finally, [Trajcevski et al., 2004b] were concerned with managing uncertainty in moving object databases, and proposed two related operators called *possibly* and *definitely* to determine the object may *possibly* satisfy a given condition or it should *definitely* do so.

A query language for handling time-dependent geometries such as *moving point* and *moving region* has been introduced in [Erwig et al., 1999; Guting et al., 2000]. A large number of auxiliary spatio-temporal data types and operations for handling and querying moving object trajectories (e.g., *deftime*, *rangevalues*, *locations*, *trajectory*, *traversed*), and for representing spatial relationships between geometries (e.g., *intersects*, *touches*, *attached*, *overlaps*, etc.) have been presented. Aggregate functions (*min*, *max*, *avg*) have also been applied to moving object data types. An extension to this approach which aimed at representing those moving objects in spatial networks has been proposed in [Güting et al., 2006]. Static and/or moving objects are represented on top of a network model for supporting the description and querying of complete histories of movement. Two data types *gpoint* and *gline* are defined, which represent a position in a given network and a region within that network, respectively. *moving(gpoint)* and *moving(gline)* represent the corresponding time-dependent geometries. The basic concept of *route* (corresponding to roads or highways and to paths over a graph) and some special operations for networks are also presented such as *shortest_path* and *trip*. A generic data model for representing the complete movement of a moving object (i.e., indoor and outdoor movements), where the roads, streets and rooms are considered as constituting entities, has been further described in [Xu and Güting, 2011]. Additional data types have been provided such as *mpptn*, for representing moving points in public transportation network (i.e., buses, trains and underground trains), and *groom*, that describes a room by a 2D area plus a value denoting the height above some ground level of the building. An indoor graph has been designed based on *groom* and *door* data types to support indoor trip planning at the room level. Doors in this graph represent nodes and edges correspond to rooms. Searches for optimal routes is made available through this model, and based on a preprocessing step that computes paths between all pairs of doors. However, this model does not represent objects' movements at a fine-grained level. In addition, this approach does not deal with the continuous processing of current movements but rather with histories of movements.

Moving GeoPQL is a visual query language in spatio-temporal databases described in [D'Ulizia

et al., 2012]. A temporal layer is added on top of the Geographical Pictorial Query language (GeoPQL) [Ferri and Rafanelli, 2005], thus allowing to specify and visualize the spatial configuration of moving objects in a given time interval. GeoPQL algebra basically consists of topological, logical, and metric operators, but does not handle temporal and spatio-temporal queries. This set has therefore been enlarged in Moving GeoPQL by adding specific temporal (*T-before*, *T-meets*, *T-overlap*, *T- starts*, *T-during*, *T-finishes* and *T-equals*) and spatio-temporal operators (*Geo-growing*, *Geo-shrinking*, *Geo-merging*, *Geo-splitting*, *Geo-appearing* and *Geo-disappearing*). However, this algebra is also designed for expressing queries on histories of movements and does not consider continuous present queries.

3.4.2 Data types and operations for spatio-temporal data streams

The other category of query languages deals with data streams to represent abstract continuous semantics with specific data types and operators. As previously mentioned, time-based as well as tuple-based execution models are considered in classical data stream management systems, so that the associated continuous sliding-window semantics should be reflected by the corresponding query language [Jain et al., 2008]. For instance, the TelegraphCQ system [Chandrasekaran et al., 2003] proposes a declarative language for continuous queries with a particular focus on expressive windowing constructs. Several types of queries are supported such as *snapshot*, *landmark*, and *sliding* queries, which correspond to the type of window used in expressing the query. A snapshot query executes exactly once over one window, whereas a landmark query has a fixed beginning point in the timeline, and a forward moving endpoint. Finally, a sliding query is carried based on forward moving beginning and end points.

The *Continuous Query Language (CQL)* is another SQL-based declarative language for representing continuous queries over data streams [Arasu et al., 2004b, 2006]. Abstract semantics are defined based on two data types *stream* (i.e., a multiset of timestamped elements) and *relation* (i.e., an unordered set of tuples at any time instant within a given interval), and three classes of operators: *stream-to-relation*, *relation-to-relation*, and *relation-to-stream*. Stream-to-relation operators in CQL are based on the concept of a sliding window over a stream. Three relation-to-stream operators are also defined: *Istream* (input stream), *Dstream* (delete stream), and *Rstream* (relation stream) with time-based sliding windows. The whole approach has been illustrated with a road traffic management application, which simulates real-time computation of vehicle tolls based on traffic conditions.

3.4.3 Languages for querying preference-aware and context data

While Section 3.2.4 presented trends towards context-aware processing of location-dependent queries, this section focuses on query languages for expressing and representing contextual data in such location-dependent queries. One of the main considerations in representing context is to select and apply an appropriate and expressive query language for defining user's preferences and needs [Haghighi et al., 2006]. Relevant works in the area are discussed as follows.

1. A preference model for database querying that delivers personalized answers in ambient intelligent environments has been proposed in [van Bunningen et al., 2006]. An ontology-based language has been developed as a variant of Description Logics (DL) to represent the user's context. However, this approach does not represent location-dependent queries and does not apply to moving objects. Similarly, [Reichle et al., 2008; Benyelloul et al., 2010] have proposed to query contextual data by representing different static/dynamic contextual elements, but also without being concerned about querying location-dependent data.

2. Preference SQL extends SQL by a preference model to deal with fuzzy predicates based on a soft selection constraints mechanism [Kießling and Köstler, 2002; Kießling et al., 2011]. Preferences are expressed inside an SQL query with the *PREFERRING* keyword. Other operators such as *AROUND*, *BETWEEN*, *LOWEST*, *HIGHEST*, *POS*, *NEG* are defined to enable expressing *wishes* in a declarative manner. A location-based preference constructor for querying such spatial databases in outdoor environments has been developed as an extension of Preference SQL [Wenzel et al., 2011]. The NEARBY spatial operator has been added to retrieve objects in an *approximate* range. This approach has not been applied to moving objects.

3. Another fuzzy model for representing contextual preferences has been presented in [Bosc et al., 2009; Hadjali et al., 2010]. Fuzzy predicates on attributes such as *low*, *medium*, and *high* have been first defined in [Bosc et al., 2009], and are evaluated within a *context* clause that results in a given constraint with a certain degree of satisfaction. Contextual preferences were further considered in [Hadjali et al., 2010], where the goal was to infer a set of relevant preferences and their semantics regarding the user context from a fuzzy rule base modelled as a set of contextual preferences. This approach has been applied to route planning queries on road networks while taking into account a set of user-defined preferences [Mokhtari, 2011; Hadjali et al., 2012]. A distinction between *constraints* and *wishes*, which are not compulsory, is made. A typology of users preferences is also proposed, where *spatial*, *global*, and *spatio-temporal* preferences are distinguished. *Routes* and *Segments* are defined as two relations to describe the paths and elementary segments in a road network, respectively. A series of keywords (e.g.,

FIND k ROUTES FROM ... TO ..., *PREFERRING*) and modifiers (e.g., *all*, *most*, *such that*, *passes [near/inside/outside]*) are also defined to express those route planning queries. Despite the relevance of this approach in representing user preferences based on fuzzy predicates, it does not consider real-time location-dependent queries over moving objects.

4. The TRANsportation QUerY Language (TRANQUYL), presented in [Booth et al., 2009; Booth, 2011], encompasses a set of relations and operators on top of a graph-based relational model for supporting multimodal transportation systems. Relations such as *road_ edge*, *freeway_ edge*, *bike_ edge*, *sidewalk_ edge*, and *train_ edge*, along with other vertex relations are defined, thus allowing an intuitive expression of advanced trip planning queries in spatial networks (via *leg*, *transfer*, and *trip* high level relations). The language extends the SQL structure with new clauses (e.g., WITH MODES [null|bus|train|pedestrian|etc.], WITH CERTAINTY, WITH STOP VERTICES, and MINIMIZE|MAXIMIZE clauses), operators (e.g., All-Trips), and specific spatio-temporal certainty, and facility conditions in the WHERE clause to support user preferences. Although temporal operators such as *starts*, *ends*, *arrival*, and *departure* are defined to support dynamic planning queries, this rich query language assumes static origins and destinations in the All-Trips operator, and cannot be directly applied to location-dependent queries over moving objects.

5. The query grammar presented in [Ilarri et al., 2011] enhances the expressiveness of location-dependent queries by considering the granularity of moving objects' locations. A *location granule* is a data type defined as a set of elementary locations in a given reference system (e.g., GPS locations). For instance, a granule could represent a *city*, a *region* or a *country*, and used to express locations at different levels of abstraction. This approach covers the use of location granules based on the user preferences from both a query processing as well as a result presentation points of view. Several operations are then presented to handle those granules such as *inGranule*, *contains*, *getGranules*, *getNearestGranule*, *getGranulesObject*, etc. The results shown from using the location granules in query processing are considered very satisfactory. Granule-based query constraints such as *inside* constraint is added to express range queries over moving objects. Indeed, the experiments show that the advantages of location granules do not come at the expense of performance. However, this approach considers moving objects only in Euclidean space, and does not handle other navigation-related queries such as path queries.

The last two approaches have inspired our work presented in Chapter 5, which introduces a query language that favours continuous location-dependent queries in indoor mobile environments. Several types of queries are supported such as range, nearest neighbour, and path queries. User

preferences, optimization criteria, and management of location granules in indoor environments are incorporated into this query grammar.

3.5 Conclusions

This chapter covered related work on location-dependent query processing in both outdoor and indoor environments. Indoor-specific techniques have been rarely addressed, but some outdoor architectures and strategies might be applied to indoor configurations with adjustments, when necessary. Architectures for continuous processing of location-dependent queries were first studied from different perspectives (i.e., *moving object databases* and *data stream management systems*), and in indoor and outdoor settings. Trends for integrating user preferences and other contextual data were then presented. Secondly, a focus was put on *adaptive* and *incremental* paradigms for processing navigation-related queries especially in spatial networks. Algorithms for *path* and *range* query processing were discussed, as *hierarchical and incremental execution paradigms* for both type of query are introduced in Chapter 6. Finally, languages that introduce data types and operations for querying moving objects were highlighted, along with a discussion on expressing preference- and context-aware queries. Related work in this chapter supports our contributions presented in Chapters 5, 6, and 7, where an architecture for continuous location-dependent query processing, a query language, algorithms for continuous path and range searches, and a prototype encapsulating all these elements are introduced.

Part III

DATA MODEL, QUERY LANGUAGE, ALGORITHMS, AND SYSTEM DESIGN

CHAPTER 4

A Hierarchical and Context-Dependent Indoor Data Model

Contents

4.1 Need for a hierarchical and context-dependent data model

The modelling approach described in this chapter has been introduced in [Afyouni et al., 2013], and represents a hierarchical graph representation of an indoor system that can be integrated into a context-aware system architecture. The preliminary requirements for the development of indoor spatial models have been surveyed in Chapter 1 from a context-aware system perspective. An indoor data model should meet service-oriented (i.e., localisation, navigation, location-aware communication, activity-oriented interaction, and simulation and behavioural analyses) and efficiency-related (i.e., modelling effort, flexibility, performance, and scalability) requirements. Satisfying those requirements creates the conditions for representing context-awareness in indoor mobile environments.

Hybrid spatial models (i.e., with both geometric and symbolic representations) provide a good trade-off to efficiently integrate metric properties, while maintaining a more abstract view of space

with easily-recognizable information about relationships between entities [Afyouni et al., 2012]. A hierarchical data model embodies knowledge of the environment at different levels of abstraction. A hierarchical design can support a large spectrum of applications, and offers a solution to alleviate performance and scalability issues in location-dependent query processing. Hierarchical models usually scale very well to large environments since queries such as path searches can be performed hierarchically by switching from finer to coarser levels and vice versa.

This chapter presents a hierarchical and context-dependent indoor data model that tries to bridge the gap between supporting context-awareness and alleviating performance and scalability issues required for location-dependent query processing. This data model is hierarchically organised and can be viewed as a tree structure in which location information is represented at different levels of abstraction. The remainder of this chapter is organized as follows. Section 4.2 introduces our approach for representing indoor environments as well as the features located and/or acting in those environments. Particularly, *indoor moving objects* are represented, and the methodology for managing user profiles and real-time event management is also presented. Finally, a discussion on the data model is provided in Section 4.3.

4.2 Modelling approach

This section presents a modelling approach of an indoor-oriented system that takes into account the spatial component representing the environment, as well as other components for handling static/moving objects and the actions that emerge from them. The integration of an indoor spatial data model into a context-aware system lies in considering a dynamic environment that should represent (Figure 4.1):

1. All the *features* that populate an indoor environment, where a feature can refer to either a person (i.e., a mobile user or any other social entity of interest[1]) or an object/event of interest (e.g., sensors, exits, tables, continuous phenomena such as a fire, etc.).

2. Their spatial properties (e.g., location and extent), as well as other static (e.g., for users: profile, interest, etc.) and dynamic (e.g., status) properties.

3. The behaviours that emerge from them (i.e., how these objects can interact and communicate within the environment). It should be noted that an *object of interest (OOI)* may or may not have communication capabilities, be mobile or static, physical or virtual, and attractive or

[1]Human beings that are located in the vicinity and are of interest to the query are referred to as social entities.

repellent (i.e., depending on whether the user may want to reach it or to avoid it, respectively)[2].

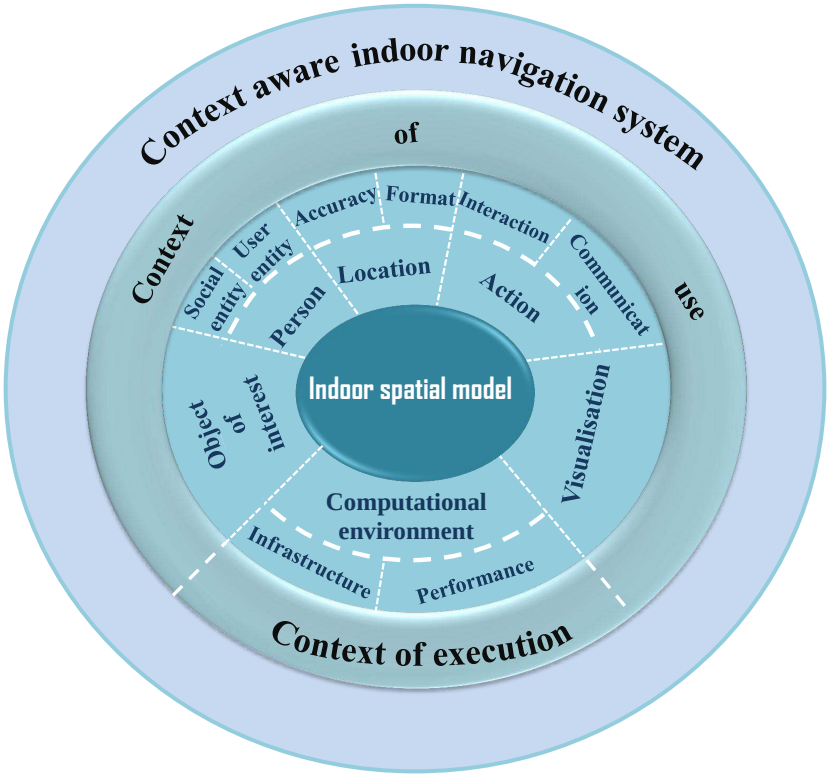


Figure 4.1: Indoor spatial model as a core component of a context-aware indoor system

As shown in Figure 4.1, many components that contribute to the design of a context-aware system should be reflected by the spatial data model underneath. We assume that a user navigating an indoor space should be able to communicate with other nearby social entities as well as with sensors deployed in the environment. Moreover, an indoor navigation system should be aware of the situation and the context of the user, and have to be adaptive and to provide answers in an efficient and appropriate way. Furthermore, specific properties such as the granularity of the spatial data model retained for the representation of an indoor space might affect the manipulation, visualization, and computational capabilities of the system. Those characteristics will serve as a basis for the development of our study.

Let us formally present the main concepts of the indoor data model. This context-dependent data model represents an indoor environment with three complementary components $\langle \mathcal{S}, \mathcal{F}, \mathcal{A} \rangle$, where:

[2]The distinction between attractive and repellent events is similar to the one suggested in [Delot et al., 2011] regarding attraction and repulsion events.

- The spatial component $\mathcal{S} = \bigcup_{i=1...|\mathcal{S}|} \mathcal{S}_i$ is made of a set of layers ($\mathcal{S}_i$) hierarchically organised and representing the indoor space, and thus defining the multi-granular spatial structure of the model.
- The feature component $\mathcal{F} = \bigcup(\mathcal{P}, \mathcal{OOI})$ encompasses the features (i.e., persons ($\mathcal{P}$) and objects of interest ($\mathcal{OOI}$)) located in the environment.
- The action component $\mathcal{A} = \bigcup(\mathcal{FA}, \mathcal{SA})$ represents actions that are either predefined and triggered automatically by the system in form of informative, context-aware messages ($\mathcal{SA}$), or generated by a given feature acting in the environment ($\mathcal{FA}$).

These three components are hereafter discussed in more detail.

4.2.1 Spatial component

The *spatial component* of an indoor-based system takes into account different levels of spatial granularity, and this regarding not only the way sensors are deployed in the environment, but also considering the fact that different models (i.e., geometric and symbolic representations) of space may coexist. The approach developed is a hierarchical spatial representation of an indoor environment that can be integrated into a context-aware system architecture. The spatial data model is hierarchically organised and can be viewed as a tree structure in which location information is represented at different levels of abstraction. This hierarchical design can support a large spectrum of applications that can be developed at different levels of abstraction, and allows to alleviate performance and scalability issues in location-dependent query processing [Afyouni et al., 2013]. The approach also takes into account the large range and trends of positioning techniques to offer a context-dependent spatial model that supports different applications.

This conceptual approach is embodied in a hierarchical model that implicitly embeds different levels from *micro* to *macro*, from *continuous* to *discrete* structures of space. One assumption of this model is that a coarse-grained model can be derived from a finer grained representation depending on the application and context-aware constraints and capabilities. The approach assumes that a given user acting in an indoor space can be continuously located in real-time, using for instance a MEMS sensor, thus providing a *fine-grained* representation for this user. In contrast, and when WLAN or RFID positioning systems are deployed in the environment, a *coarser level* of granularity might be provided to locate users in the environment. Similarly, the spatial representation to consider in order to relate mobile users to the environment are chosen appropriately by taking into account some application constraints and properties. For instance, if one asks to know how many users are located in a given room, it may be inappropriate to display the precise locations of those users and their

trajectories. In such a case, a representation at the *room level* is likely to be sufficient. Alternatively, a finer level of granularity might be appropriate when one is interested by the relative location of some users in a given room, and also with respect to the location of some sensors in that room.

A spatial component contains a set of spatial layers hierarchically organized. Figure 4.2 describes a part of a scenario (i.e., first and second floor are illustrated) where a user navigates inside a three-story laboratory building. The ground floor of the building comprises two teaching labs and some administrative staff offices. Offices of the Linguistics Department are located on the first floor. Specifically, Figure 4.3 illustrates the second floor of the building where the Computer Science Department is located. The core layer is firstly presented. Then, other coarser layers that can be incorporated into the hierarchical data model are discussed.

4.2.1.1 Core spatial layer

The core layer $\mathcal{S}_1$ (referred to as $\mathcal{S}_{micro}$) of the indoor data model is made of a fine-grained graph $G_{micro} = (V_{micro}, E_{micro}, W_{length}, W_{time})$ embedded within a spatial grid with a regular cell size, and which covers the indoor space (Figure 4.3). The extent and the level of granularity are two mandatory parameters that have to be determined *a priori* for the derivation of the grid. The accuracy of the resulting grid depends on the cell resolution. A fine-grained grid supports accurate location data, but could introduce heavy processing workloads (details on the cell-based approach are provided in Chapter 2, Section 2.2). For example, the spatial resolution selected in the scenario illustrated in Figures 4.3 and 4.4 is 50 cm, which roughly corresponds to the human spatial extent [Raubal, 2001]. This means that a 50 cm distance is set between horizontal and vertical neighbour nodes, while $1/\sqrt{2}$ cm is set for diagonal distances. This fine spatial resolution assumes highly accurate location data, but other coarser resolutions can be used depending on the application constraints. A coarser resolution results in a less accurate representation of space as well as a distorted perception of the objects' movements. The resulting grid graph encompasses vertices (i.e., nodes) that represent cells within the grid, and connections between cells are explicitly materialized by edges. Each node is located in the centre of a cell, and is connected to its eight neighbours (not only the four ones located in the boundary) with horizontal, vertical, and diagonal edges.

This modelling approach achieves a maximum coverage of the indoor space. An indoor environment is represented as a continuous space that supports continuous positioning techniques used in indoor navigation. Besides, the cell- and graph-based representation supports the modelling of structural properties (i.e., connections and relationships between nodes) at different levels of granularity, while keeping geometrical properties. Nodes of the grid graph are labelled according to their membership to a given spatial unit such as a room or a connecting space (i.e., a hallway).

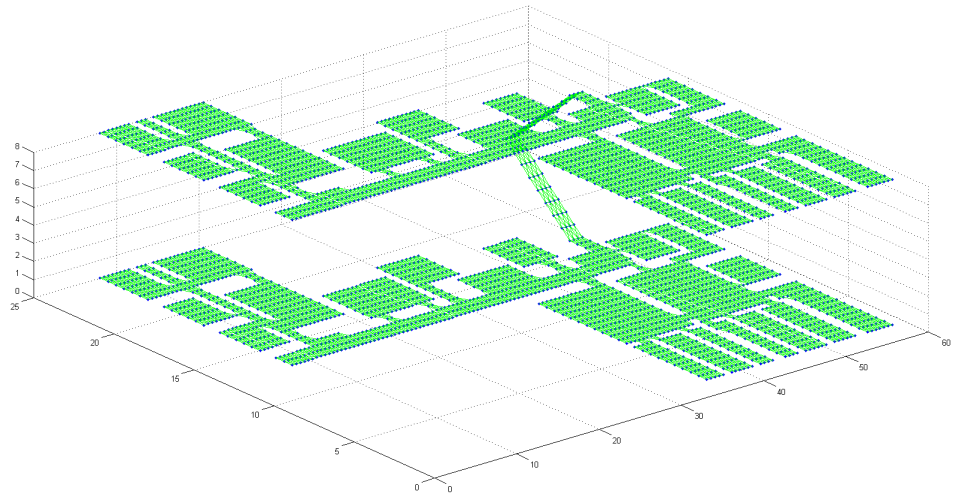


Figure 4.2: A fine-grained network of a two-storey building: first level of the hierarchical spatial data model

Therefore, each node has one and only one membership value since it belongs to one and only one spatial unit, whereas an edge might have multiple membership values when it intersects several spatial units. Nodes and edges can be labelled with impedances defined at the application level (i.e., node's and edge's accessibilities). Figure 4.4 illustrates a closer view of the fine-grained network, and shows that a room at the abstract level may contain multiple nodes of the fine-grained graph (green points represent nodes, and links between them depict edges). The set of exits (i.e., brown polygons) illustrated in Figures 4.3 and 4.4 are considered afterwards (Section 4.2.1.2) in order to constitute an abstract layer as a part of the hierarchical data model.

In the definition of $\mathcal{S}_{micro}$, $V_{micro} = \{v_i\}$ is the set of vertices and $E_{micro} \subseteq V_{micro} \times V_{micro}$ is the set of edges. For each edge $e = (v_i, v_j) \in E_{micro}$, there exist two time-dependent cost functions $\omega l_{i,j}(t) \in W_{length}$ and $\omega t_{i,j}(t) \in W_{time}$ that compute the *length* and *travel-time* from v_i to v_j, respectively, if traversal is started at instant t. Besides time, this model also takes into account other contextual dimensions such as *user profiles* and *real-time events*, to further associate impedances with edge weights. User profiles are handled by considering adapted graphs that are derived from the base graph G_{micro} and which correspond to predefined categories of users (Section 4.2.2.2). Effects of real-time events on edge weights will be discussed later in Section 4.2.2.3.

Each node $v \in V_{micro}$ has a set of attributes that describe its physical location or state (i.e., whether it is accessible or not). A node v is formally defined by the tuple $\langle v_{id}, \mathtt{x}_v, \mathtt{y}_v, \mathtt{s}_v, \mathtt{L}_v, \mathtt{A}_v \rangle$. v_{id} is the node identifier, $(\mathtt{x}_v, \mathtt{y}_v)$ denotes the geometric location of v according to a reference system, and $\mathtt{s}_v \in \{\textit{free}, \textit{occupied}\}$ determines whether or not the node v is physically occupied by an

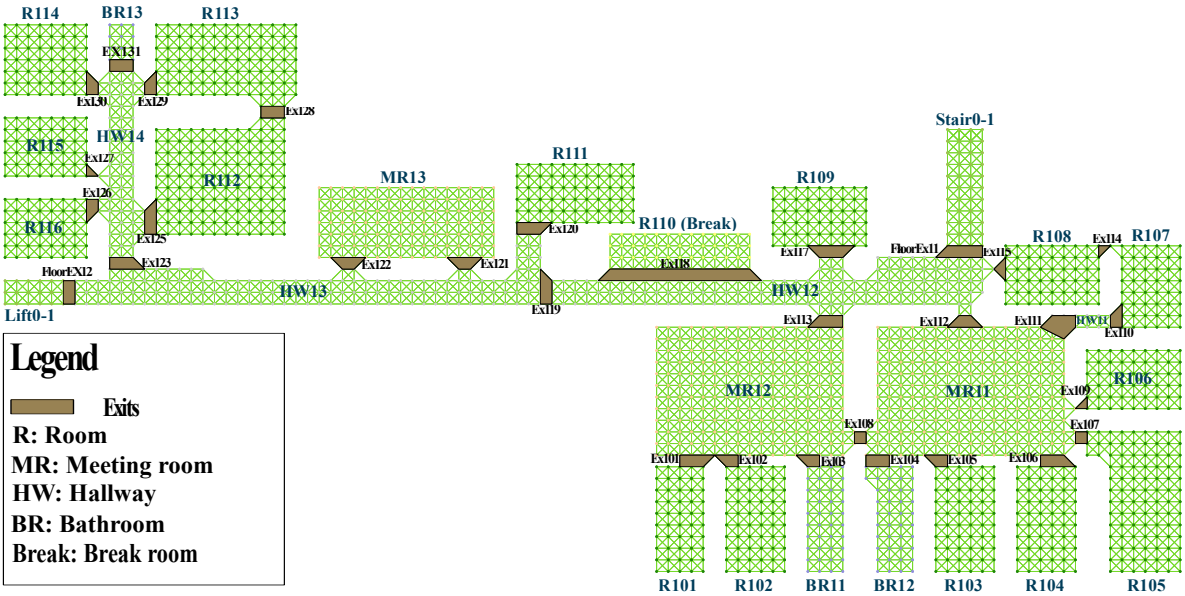


Figure 4.3: A fine-grained network of a two-storey building: first level of the hierarchical spatial data model

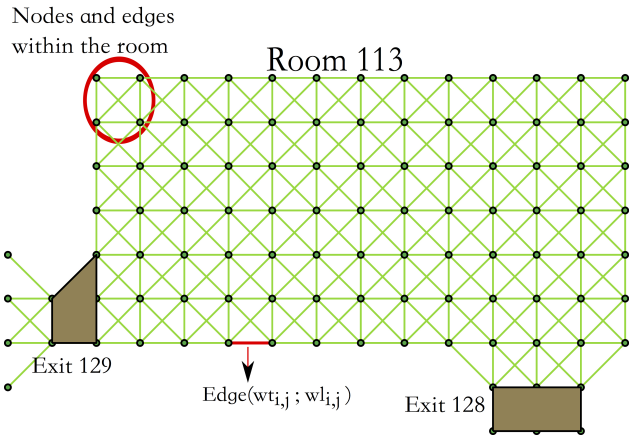


Figure 4.4: A closer view of the fine-grained graph (V_{micro}, E_{micro}) at Room 113 (R113) of the first floor

object at that moment. Nodes that are occupied by static objects (or affected by real-time events that indicate that the node is inaccessible) are assumed to be unusable for path planning[3]. Let $\Sigma_{label} = \{\Sigma_{fine-grained} \cup \Sigma_{room} \cup \Sigma_{floor} \cup \Sigma_{building}\}$ be a set of labels or symbolic values that consists of all the identifiers of the topological hierarchy (i.e., local identifiers of nodes at the fine-grained level, as well as room, floor, and building identifiers) for a given space. Hence, $\mathrm{L}_v \subset \Sigma_{label} = \{$*local-id*, *room-id*, *floor-id*, *building-id*$\}$ is a set of labels assigned to v, where *local-id* denotes its local identifier

[3]Moving objects are usually not considered as obstacles, and even if they are obstructing the path (e.g., a cleaning machine blocking a pathway) they are expected to move in a short time. Nevertheless, there is no problem to model this kind of situations in our modelling approach, as a closed pathway can be considered as a real-time event that temporarily prevents passing by.

at the fine-grained level, and the others are associated according to their belonging to the topological hierarchy. We assume at this level that v belongs to one and only one room, and one building. In contrast, *floor-id* is a subset of the set of floor identifiers since, for instance, a node located on a staircase may belong to several floors. Finally, $\mathtt{A}_v \subset \mathcal{A}$ is the set of triggered actions, i.e., contextual messages or notifications that are predefined and can be executed according to some contextual constraints (e.g., to remind a user navigating a shopping centre to buy some food or fruit stock when he/she is located next to a supermarket). Implementation issues and integration of those actions in query processing are discussed in Section 4.2.3.

An edge $e \in E_{micro}$ is defined by a tuple $\langle (v_i, v_j), \mathtt{L}_e, \omega l_{i,j}(t), \omega t_{i,j}(t) \rangle$, where $v_i, v_j \in V_{micro}$, $v_i \neq v_j$, and $\mathtt{L}_e \subset \Sigma_{label}$ is a subset of the set of labels (e might have multiple labels when it intersects several spatial units -e.g., rooms-). Besides, $\omega l_{i,j}(t)$ and $\omega t_{i,j}(t)$ are time-dependent functions associated with the traversal of e. The traversal of some edges may be constrained by a temporal interval defined at the application level, and within which the traversal is possible; otherwise the corresponding edge cannot be traversed. These functions are defined as follows:

$$\omega l_{i,j}(t) = \begin{cases} Ed(v_i, v_j) & if\ \mathtt{t} \in [t_{start}, t_{end}] \\ \\ \infty & otherwise \end{cases}$$

where $Ed(v_i, v_j)$ is the Euclidean distance between v_i and v_j, and t_{start} and t_{end} are defined at the application level (for example, $[08:00,\ \ 17:00]$ could be specified for an office building).

$$\omega t_{i,j}(t) = \begin{cases} f(\omega l_{i,j}(t)) & if\ \mathtt{t} \in [t_{start}, t_{end}] \\ \\ \infty & otherwise \end{cases}$$

where $f(\omega l_{i,j}(t))$ is a length-dependent time function that further associates impedances to compute the travel time between v_i and v_j.

Temporal intervals can be partitioned into different subintervals to satisfy other application-dependent constraints. The network distance and the travel time from v_s to v_d are computed as indicated in Definitions 1 and 2, respectively. These functions take the Euclidean distance derived from the fine-grained network in order to compute the optimal navigational network-based path, depending on either the distance and/or time criteria, as well as other semantic constraints.

Definition 1 ***Fine-grained and time-dependent network distance:*** *Let $p = \langle v_{start}{=}v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k{=}v_{goal} \rangle$ be a path that contains a sequence of nodes $v_i \in V_{micro}$, i=1, ..., k. The time-dependent network distance of p is given by $length_{start,goal}(t_{start}) = \sum_{i=1}^{k-1} \omega l_{i,i+1}(t_i)$, where $t_i = t_{i-1} + \omega t_{i-1,i}(t_{i-1})$ represents the estimated time instant at node v_i, $\forall$ i=2, ..., k, and $t_1 = t_{start}$.*

Definition 2 ***Fine-grained and time-dependent travel time:*** *Let* $p = \langle v_{start}{=}v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k{=}v_{goal} \rangle$ *be a path that contains a sequence of nodes* $v_i \in V_{micro}$*,* $i{=}1, \ldots, k$*. The time-dependent travel time of* p *is given by* $time_{start,goal}(t_{start}) = \sum_{i=1}^{k-1} \omega t_{i,i+1}(t_i)$*, where* $t_i = t_{i-1} + \omega t_{i-1,i}(t_{i-1})$ *represents the estimated time instant at node* v_i*,* $\forall\ i{=}2, \ldots, k$*, and* $t_1 = t_{start}$*.*

This core spatial layer is built in an offline phase and a subsequent online phase is in charge of updating potential changes and time-dependent data. For instance, in the offline phase, nodes that are covered by static objects (e.g., a wall, a table, etc.) are marked as occupied whereas the rest are considered initially free. Furthermore, the state of a node depends also on the user profile, since different kinds of users may have a completely different set of accessible nodes (e.g., a certain node may be apparently free but correspond to a room that can only be entered with a key card). This can also be statically managed with the use of a user access model, as discussed in Section 4.2.2.2.

4.2.1.2 Coarser spatial layers

Thanks to the flexibility of the spatial data structure, a coarse-grained representation can be derived by aggregating nodes and edges according to some membership value, whereas edges will be derived accordingly. This favours data manipulation at a coarser level of granularity. For instance, *topological queries* can then be applied to find out emerging spatial and temporal properties and mobility behaviours at the level of the rooms and connections of an indoor space (e.g., objects located and passing through a given room for a period of time). Moreover, relationships between the sensors located in the environment and the resulting graph can also be derived as suggested in [Becker et al., 2009] to achieve inter-layer interaction.

A node v at a coarser layer $\mathcal{S}_i \in \{\mathcal{S}_2, \ldots, \mathcal{S}_{|\mathcal{S}|}\}$ is defined as an aggregation of a subgraph of the finer graph, and is denoted by $\langle \mathtt{L}_v, \mathtt{A}_v \rangle$, where $\mathtt{L}_v \subset \Sigma_{label}$ is the set of labels assigned to v, which is adapted accordingly to fit the corresponding level of abstraction, and $\mathtt{A}_v \subset \mathcal{A}$ comprises the set of triggered actions that are predefined at the corresponding node. Cost functions are derived and processed based on the edge weights of the fine-grained level. Formal definitions of the relevant layers considered in the data model are described hereafter, along with an explanation of how these layers can be derived.

Exit hierarchy. Exits represent connections between rooms at the abstract level. An exit is an important element of the data model used for query processing, through which a user can leave or

enter a place (e.g., doorways or staircases). An exit is represented as an abstract node that belongs to two different spatial units, and is derived by aggregating boundary nodes of both units whose adjacent node lists contain at least one neighbour that belongs to the other spatial unit (Figure 4.5). Accordingly, an exit can contain multiple nodes and edges at the fine-grained level (see Figure 4.3 and 4.4). Based on these exits, a coarser network (at a higher level of abstraction) can be designed, in which nodes depict those exits and links represents optimal navigational paths between directly reachable exits. Optimal network distances and travel times between relevant pairs of exits are pre-processed and cached in order to reduce on-the-fly computation of hierarchical path searches. An exit ex' is relevant for a given exit ex if and only if ex' is directly reachable from ex (i.e., there is an accessible passageway for pedestrians from ex to ex' which does not involve any other exit). An exit hierarchy is constructed at a higher level of abstraction, which allows computing optimal distances between locations to be used later for processing diverse kinds of queries.

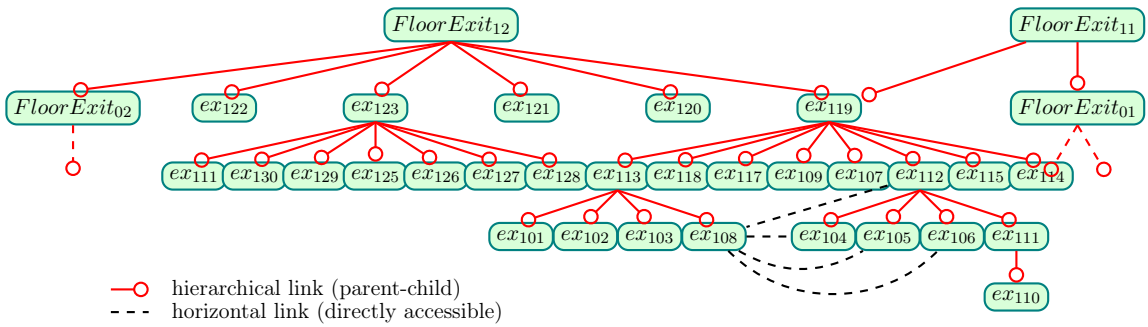


Figure 4.5: Part of the exit hierarchy derived from the fine-grained graph (Floor-01, Building-1)

A link between two directly reachable exits is represented by a path (i.e., a sequence of nodes and edges in (V_{micro}, E_{micro})) at the fine-grained layer. More formally, let r, $r' \in \Sigma_{label}$ be the labels of two connected rooms, the exit representing the doorway between r and r' is given by: $ex_{r,r'} = \{v_i, v_j \in V_{micro} \mid \exists ex \in E_{micro}, ex = (v_i, v_j) \wedge r \in ex.\mathtt{L}_{ex} \wedge r' \in ex.\mathtt{L}_{ex}\}$. Regarding its belonging to the topological hierarchy, an exit is also characterised by: $\mathtt{L}_{ex_{r,r'}} = \{$*local-id*, $\{r, r'\}$, *floor-id*, *building-id*$\}$. $FloorExit_{11}$ is an example of an exit depicted in Figure 4.5, which belongs to two structural units: $Stair_{01}$ and HW_{12}. Therefore, $\mathtt{L}_{FloorExit_{11}} = \{$*FloorExit-11*, $\{Stair_{01}, HW_{12}\}$, *Floor-1*, *Building-1*$\}$. Similarly, one can derive the abstract nodes of the second layer representing all exits on a given floor. An abstract edge ($ex_{r,r'}$, $ex_{r',r''}$) in the exit hierarchy is a path made of a sequence of nodes and edges of the fine-grained level that compose the optimal network distance from a node $v_{start} \in e_{r,r'}$ to a node $v_{goal} \in ex_{r',r''}$. An edge of the exit hierarchy is referred to as *exit-path* and is denoted by $\langle source\_exit\_id, target\_exit\_id, length, time\rangle$. The optimal network distance and travel time are computed by applying $length_{start,goal}(t_{start})$ and $time_{start,goal}(t_{start})$, and the resulting values are associated with each edge of the exit hierarchy, thus forming the second

layer of the data model[4].

Moreover, exits are organised in a hierarchical manner since a flat graph does not reflect their significance from a semantic navigation point of view [Hu and Lee, 2004]. As illustrated in Figure 4.5, this hierarchical structure allows to distinguish between a *room exit* and a *floor exit*, which is represented at a higher level of abstraction due to its importance, so that a direct path from a current position to the nearest floor/building exit can be easily determined. Other edges between exits of the same level are also materialized according to their connectivity (horizontal links illustrated as dashed lines in Figure 4.5)[5]. Consequently, the final representation of this layer preserves the connectivity between directly accessible exits while emphasizing their importance for navigation purposes. A generalisation of this hierarchy that covers a multi-storey building is used for path planning. Consequently, an exit of a ground floor has a *building exit* as a parent node, and a *first-floor exit* as a child node since both are parts of a staircase.

It should be emphasized that exit-paths in this hierarchy are assumed to be undirected. However, this model can be adapted for specific scenarios where a one-way access to several areas is required. This can be done by either replicating edges in the opposite direction or associating a property to each edge that adds impedances to the path weight depending on the travel direction. Adjustments of the query processing algorithms are needed accordingly.

Location hierarchy. Incorporating information about exits into the topological hierarchy enables the modelling of optimal paths at an abstract layer. Those are used to facilitate hierarchical path searches and to alleviate performance issues raised while traversing the fine-grained graph. Although connectivity relationships between those elementary structural units can be computed from the exit hierarchy, an adjacency relationship needs to be associated to each unit in a separate abstraction layer. A room consists of a set of nodes at the fine-grained layer as illustrated in Figure 4.4. An abstract view of an indoor space considers rooms as abstract nodes and connections between rooms as links. Such topological properties are not explicitly materialised in the exit hierarchy, even though information representing their belonging to the topological hierarchy has been incorporated. Consequently, a location hierarchy that is based on a connectivity graph, which represents rooms as nodes and doorways as edges, can be automatically derived from the fine-grained graph as an additional layer in order to preserve topological relationships (Figure 4.6).

A room in the location hierarchy is characterized by $\langle room\_id, room\_type, Adj\_room\_list, \mathtt{L}_r, \mathtt{A}_r \rangle$, where $room\_type$ describes whether this unit is a room, a meeting room, a hallway, etc.,

[4]The hierarchy is not fully illustrated in Figure 4.5, since the right part rooted at $FloorExit_{02}$ is developed similarly.

[5]For clarity's sake, not all the edges that depict connectivity between exits are shown in Figure 4.5.

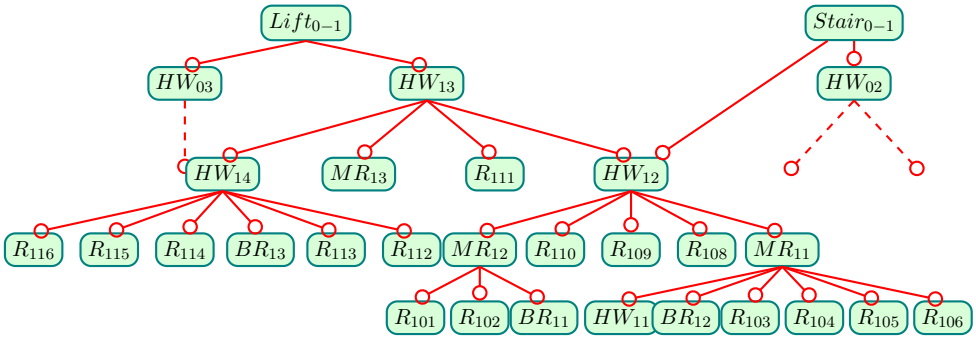


Figure 4.6: Part of the location hierarchy derived from the fine-grained graph (Floor-01, Building-1); "HW" stands for Hallway, "MR" for Meeting Room, "R" for Room, and "BT" for Bathroom

Adj_room_list denotes the list of identifiers of the adjacent units, and $\mathtt{L}_r$, $\mathtt{A}_r$ are introduced in a similar way as in the fine-grained level. Such a location hierarchy can be directly derived from the fine-grained layer, but can also be generated from the exit hierarchy since information about the belonging of exits to their respective structural units is stored. A staircase that connects a given floor to another is represented as a room that belongs to the two corresponding floors, and which is bounded by two floor exits. An elevator is represented in a similar way to stairs. A multi-floor elevator consists of several stages that correspond to the number of floors of the building. Each stage of the elevator is modelled as a room that belongs to the two corresponding floors and bounded by exits/entrances to/from the corresponding floors.

From the fine-grained graph, a typical clustering process results in an abstract layer as illustrated in Figure 4.6. Graph partitioning is thus carried out based on the set of room labels associated to the nodes of the base graph. Consequently, this process consists of: (1) extracting and aggregating nodes whose room labels are identical to form the new abstract nodes of the location hierarchy; and (2) creating abstract edges between connected structural units, thus favouring topology-based queries. These steps are as follows:

- ***Step 1.*** Based on the set of room labels, the fine-grained graph is partitioned into subgraphs. Let $\varphi = \bigcup_{i=1\ldots|\Sigma_{room}|} \varphi_{\ell_i}$ be the set of subgraphs of S_{micro} such that $\ell_i \in \Sigma_{room}$, and where $\forall i \in \{1, \ldots, |\Sigma_{room}|\}$, $\varphi_{\ell_i} = (\mathtt{V}_{\ell_i}, \mathtt{E}_{\ell_i}) \subset S_{micro}$ is a subgraph extracted from the fine-grained graph according to node and edge labels, and where $\bigcap_{\ell_i \in \Sigma_{room}} \mathtt{V}_{\ell_i} = \emptyset$. An abstract node that represents each subgraph is then created, having ℓ_i as its *local-id*.

- ***Step 2.*** The set of outgoing edges between connected subgraphs is defined by: $\mathtt{E}_{\ell_i,\ell_j} = (\varphi_{\ell_i}, \varphi_{\ell_j})$ $\forall i, j \in \{1, \ldots, |\Sigma_{room}|\}$, $i \neq j$. It is worth noting that, for geometric-based queries (e.g., navigation, range, and nearest neighbour queries), the exit hierarchy is more likely considered, as it lends itself to more accurate and more realistic pre-processing techniques. In contrast, the

location hierarchy is more suitable for topology-based queries (e.g., connectivity, adjacency, etc.) or when one looks for the optimal path that contains the smaller number of rooms. Therefore, there is no need to associate precomputed network distances to edges of the location hierarchy.

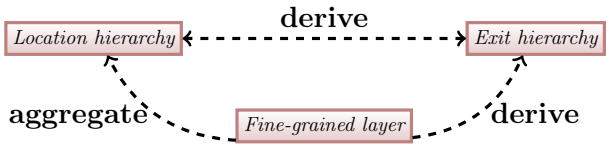


Figure 4.7: Links between neighbouring layers of the hierarchical data model

Similarly, there exists a relationship between exit and location hierarchies since exits belong to structural units. For instance, by retrieving the list of room labels associated to all exits, one can derive connected rooms and rebuild the corresponding location hierarchy. Accordingly, switching between a location hierarchy and an exit hierarchy is always possible, thus covering a larger range of queries (Figure 4.7). Three spatial layers at two levels of abstraction (i.e., the fine-grained layer at the first level, and the exit and location hierarchies at the second level) are employed and used in this work. However, the data model can be generalized to introduce higher levels of abstraction in order to cover a wider range of applications, and with more flexibility.

4.2.2 Feature component

The feature layer models different types of objects located or evolving in an indoor space, which present an interest from an application point of view. These features are either attached to the infrastructure (i.e., static objects like tables, doors, walls, fixed sensors, ...) or evolving in the environment (i.e., moving objects, continuous phenomena). This section first presents the feature component principles, and secondly highlights the management of user profiles and real-time events.

4.2.2.1 Principles

A feature component $\mathcal{F}$ models *persons* and *objects of interest* in an indoor space. We make a difference between features that represent a *rigid entity* (e.g., spatial objects, sensors, and moving users) and features that model a *continuous phenomenon* (e.g., gas leak or noise diffusion). A feature may be discrete or continuous. By continuous feature we mean any kind of phenomenon that is likely to continuously spread over space. An *entity* may be static or dynamic and is modelled as an object. An *object* is identified and characterized by its *static properties* (i.e., attributes) and potentially other *dynamic properties* such as the *interaction spaces* attached to it [Bhatt et al., 2009].

In addition, an object can perform a selected list of *actions* that can be triggered according to some contextual constraints which are application-dependent.

Formally, a *feature* $\mathtt{f} \in \mathcal{F} = \bigcup(\mathcal{P}, \mathcal{OOI})$ is defined by a tuple $\langle \mathtt{Id}, \mathtt{Ct}, \mathtt{S}, \mathtt{FD}, \mathtt{A}, \mathtt{IS} \rangle$, where: `Id` is the feature identifier, `Ct` denotes the feature class type, `S` is the set of states a feature can hold, `FD` is the set of values that describe `f` (i.e., typically, a set of *string* values describing static properties of the feature, such as profile properties for a given user), $\mathtt{A} \subset \mathcal{A}$ is the list of context-dependent actions associated with `f`, and `IS` is the list of interaction spaces associated with the feature [Bhatt et al., 2009]. The sets of states and actions available for a given feature are specified depending on the feature class type. A feature class type `Ct` is associated with a pair $\langle \mathtt{S}, \mathtt{A} \rangle$ where $\mathtt{S} = \bigcup_{i=1...|\mathtt{S}|} \mathtt{s}_i$ and `A` denote the set of states and actions, respectively. As an example, a user $\mathtt{u} \in \mathcal{P}$ may have some static descriptions about his/her profile and some predefined preferences. Besides, `u` can execute an action $a \in \mathtt{A}_u$ at an instant t when, for instance, he/she is in state $s \in \mathtt{S}$ and located on a node v. In contrast, an object of interest can be characterized by some qualitative and quantitative descriptions (e.g., its spatial extent), and Boolean parameters that determine whether the object is able to communicate or not, whether it is mobile or static, physical or virtual, and attractive or repellent.

Moreover, each feature is associated with specific dynamic properties, referred to as *interaction spaces*, that cover some semantic information used for interaction purposes. The notion of interaction spaces, first introduced in [Bhatt et al., 2009], is generalized and applied to dynamic and continuous features as illustrated in Figure 4.8. The component `IS` is a quadruple $\langle \mathtt{ps}, \mathtt{os}, \mathtt{fs}, \mathtt{rs} \rangle$ that refers to the *physical*, *operational*, *functional*, and *range* space. At the fine-grained level, the interaction spaces are formally defined as sets of nodes dynamically updated in real-time (see Figure 4.8):

- The *physical space* is represented by the set of nodes covered by the feature at a given time instant. For a moving object, the physical space corresponds to the node where the user is currently located.
- The *functional space* denotes the nodes on which another feature can physically interact with the considered feature. For instance, the functional space of a moving user refers to the interaction field around his/her physical space; it is defined according to some specific user properties. In the example shown in Figure 4.8, a user is localised on a given node, so its functional space will comprise the subset of the eight neighbour nodes at a given instant.
- The *range space* is a specific parameter only assigned to sensor objects and designates the set of nodes covered by the sensor (i.e., detectable nodes). The range space of a moving user is indeed the region covered by the sensor that is either integrated in the mobile device or attached to that user.

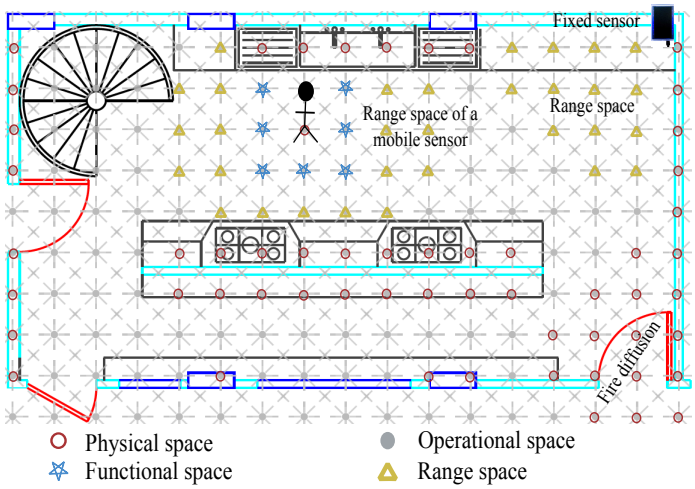


Figure 4.8: Interaction spaces of features evolving or located in space

The notion of *operational space* has been introduced to cover all features of the space (static and/or moving objects). However, the definition of the operational space varies significantly depending on whether this feature is a (pseudo)-static[6] or a moving object. The difference between the two definitions is emphasized as follows:

- The operational space of a (pseudo-)static object can be represented by the union of all the potential nodes and edges an object may cover when it performs an action in the environment. For example, the operational space of a door comprises all the potential nodes this window may cover when opening and closing.

- The operational space of a moving user denotes the set of nodes accessible to the user at a given time instant. The operational space of a mobile user strongly depends on the contextual information gathered. For example, the user profile directly influences the operational space according to whether the user is a *security guard*, a *firefighter*, a *user with special needs* or a *user with restricted privileges*, etc. Time is another important dimension that might have an impact when visiting a shopping centre or entering a laboratory building (i.e., if the current time is in the morning, at night, during the weekend, etc.). Continuous phenomena such as a gas leak or a fire that breaks out inside a building may also have a significant impact on the operational space of the user. While the fire is spreading progressively within space, a subset of nodes that are covered by the fire are removed from the operational space. Instead, additional subsets which correspond to emergency exit routes will build up the new operational space. Due to the complexity of modelling and implementing such a process, tracking of fire diffusion

[6] A door or a window is an example of a pseudo-static object, since it can either open or close (as illustrated in Figure 4.10), but it cannot move elsewhere.

is more likely to be abstracted to higher levels (i.e., *room* or even *floor* level). This results in temporary unavailable rooms or floors for user navigation.

It is worth noting that functional, range, and operational spaces are computed based on a user request, and are considered as specific continuous location-dependent queries. In particular, the range space of a given mobile sensor is considered as a typical range query, by taking into account the sensor range as the maximum threshold needed by the query. Moreover, a continuous evaluation of the operational space for a given user requires to retrieve all the spatial units accessible to this user at a given instant; this is typically done in the case of reachability queries.

4.2.2.2 User profiles

One assumption of this approach is that the user model, which encapsulates knowledge about the users' personal data and preferences, classifies users into groups according to their privileges to access restricted areas. The aim of this classification is to derive an adaptive representation of space based on access control information associated with the user. This filtering process allows to derive *adaptive graphs* from the generic *base graph* by eliminating the set of nodes that are actually inaccessible for a specific type of user, thus reducing the amount of data that need to be processed in real-time for each query and supporting the retrieval of more accurate answers based on user profiles. A similar process takes place to update the time-dependent accessibility of some nodes, for instance, abstract nodes corresponding to rooms that are closed at specific times.

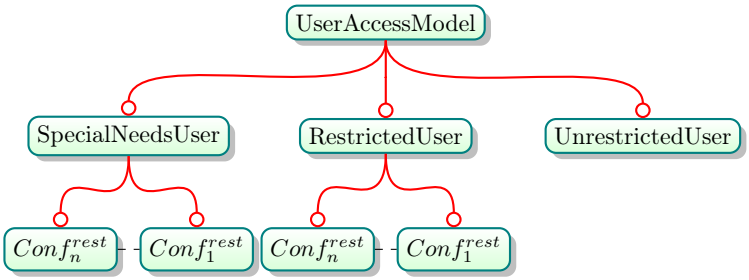


Figure 4.9: Classification of user profiles based on access control information

Users are therefore classified into three main categories: (i) unrestricted user, (ii) restricted user, and (iii) user with special needs (Figure 4.9). Additional types could be obtained by referring to these basic categories and incorporating further restrictions, thus yielding different configurations. An *unrestricted user* has full access privileges and so he/she can navigate through all areas of the building, that is, the generic graph representing all the floors of the building. A example of an unrestricted user is a firefighter or a security guard, who should have a complete knowledge of all the emergency exits in a building. A *restricted user* category includes *staff members*, *guests*, and *visitors*.

Usually, staff users have premium member privileges, but with some restricted areas, and can also have different configurations, whereas visitors have access to all the public places in a building. *Users with special needs* follow the same rules as restricted users except that additional penalties might be added to edge weights so that the most appropriate routes can be selected (e.g., using the elevator instead of the stairs for wheelchair users). Access control information is subject to very few changes over time, and thus it can be processed statically.

When there are no clearly defined privileges for certain areas inferred from a given user model, which may be due to unavailable information or lack of attention when deploying a context-aware system, the closest upper-level category of user profiles (from the ones discussed above) is considered. This upper-level category is less restrictive. Therefore, there might be some inaccessible areas whose corresponding data could be considered for query processing. Consequently, the system may generate some answers which are not well adapted to a specific user, such as a route that passes through a restricted area.

4.2.2.3 Real-time event management

The occurrence of real-time events may have a significant impact on the nodes accessibility. For example, when considering a fire that is spreading in the Computer Science Department, fire alarms are expected to detect this event and communicate it to the system. With periodical updates performed automatically, the system is capable of representing the growing spatial extent of the fire, thus marking nodes of that physical space as inaccessible to users. This subset of nodes will be temporarily blocked when computing the operational space of the users. Instead, other subsets of nodes which correspond to emergency exit routes will be favoured to build the new operational space.

The distinction between attractive and repellent events is embodied by associating negative or positive impedance values to edge weights, respectively. Therefore, unscheduled or unexpected events are characterized by a triple: *event* ⟨`info_source`, `event_ps`, ±`value`⟩[7]. Common sources of information about events (`info_source`) include the system supervisor (if any), users and the social entities situated in the environment, and the communicating sensors. Their main task is to gather information about changes in the environment and to communicate that information to the system. The physical space of the event (`event_ps`) should also be determined in real-time in order to change edge weights as well node states accordingly. Finally, depending on the nature of each event, a positive or negative value (±`value`) is assigned to edge weights, so that adapted paths

[7] Temporal events are, on the other hand, regularly evaluated by means of the time-dependent functions previously described, and so they do not belong to this category of evens.

can be recomputed. Therefore, algorithms for handling continuous location-dependent queries are adapted in order to deal with these dynamic factors and with the information uncertainty.

4.2.3 Action component

The action component $\mathcal{A} = \bigcup(\mathcal{FA}, \mathcal{SA})$ models the set of actions that are either triggered automatically by the system ($\mathcal{SA}$) or performed by a given feature acting in the environment ($\mathcal{FA}$). System actions ($\mathcal{SA}$) denote context-aware notifications that are mainly triggered based on users' locations and implement a publish/subscribe approach; this means that events are published by service providers to address their subscribers. This also includes *geocast* messaging [Basagni et al., 1999; Navas and Imielinski, 1997], which can be described as a location-based multicast where messages are delivered to users located in a specific area instead of those subscribed to a given group.

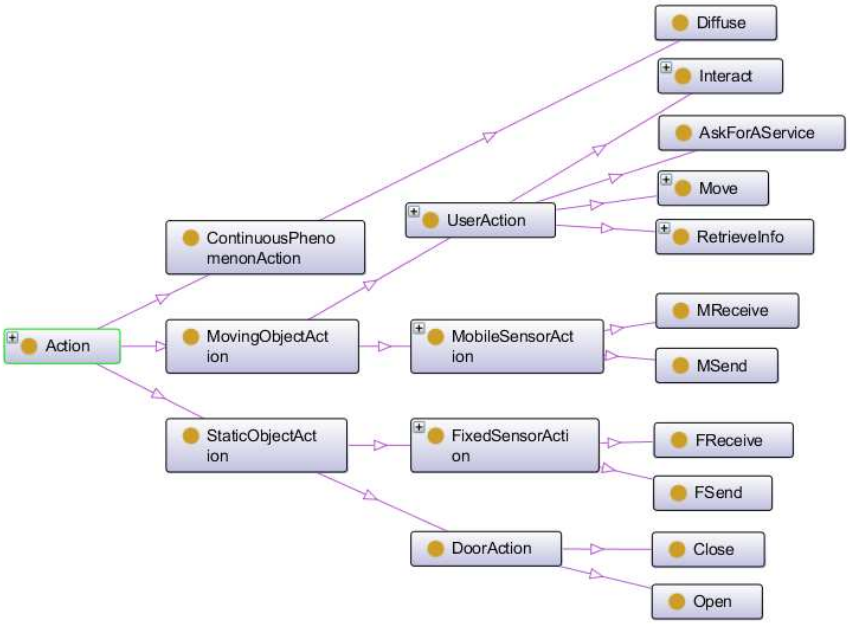


Figure 4.10: Actions performed by different types of features

Feature actions ($\mathcal{FA}$) encompass static and moving object actions, as well as continuous phenomenon actions (see Figure 4.10). When considering objects, actions specify whether and how objects of a given type change their states in order to behave in a certain way. For instance, objects can adapt their behaviour and properties according to some contextual changes in the environment. This model implicitly builds semantic and topological relations among the features situated in space, by establishing relations between interaction spaces of different features.

Actions are context-dependent; this means that, at a given time instant and for a certain

feature, only a specific list of possible actions is valid, which can then be performed according to some execution constraints. For a mobile user, actions comprise a sequence of movements, interactions with other neighbouring entities and artefacts, and requests for some services in order to achieve a predefined goal (Figure 4.10). This approach allows to represent artefacts of interest located in the environment, so that users who are engaged in a certain activity can gather knowledge and understand their physical surroundings, as well as reconfigure and manipulate physical artefacts (e.g., a chair, a door, a heating, etc.) or virtual artefacts (e.g., a 2D/3D image of a physical artefact, a digital user interface, some recommendation/information, etc.) in order to produce changes in the environment. Moreover, a user can communicate with any fixed or mobile sensor located in the range space of a (mobile) sensor integrated in his/her device or attached to him/her (e.g., a MEMS sensor, an RFID tag, etc.). When considering continuous phenomena, their actions can materialize the way a given phenomenon diffuses in space.

It is worth noting that the set of actions presented in Figure 4.10 is not exhaustive and could be extended depending the application requirements. One can, for example, add the set of basic activities performed by a given user, such as *running, walking, going upstairs, going downstairs, sitting, standing*, and which can be detected by the sensors embedded in his/her mobile phone (e.g., acceleration sensors) [Yan et al., 2012]. This further favours incorporating the user activity as an important contextual dimension. However, this dimension is, for the moment, not completely taken into account in the model and in the query processing engine.

4.3 Conclusions

A modelling approach for representing an indoor mobile environment was introduced in this chapter. *Spatial*, *Feature*, and *Action* components were discussed which take static and dynamic aspects of this environment and the features located or moving within it. The spatial component considers a hierarchical data model that encompasses different contextual dimensions and is sufficiently flexible for supporting a large range of services and queries at different levels of abstraction. Indoor static and moving objects are represented with different properties associated to them depending on their type. Actions that might emerge from those objects are classified, so that they can be further used in query processing. The following chapters will more closely consider typical user requests and services by integrating this modelling approach in a general system architecture for continuous query processing. It should be emphasized this approach is generic and rich enough to support other kinds of services and queries than those discussed afterwards.

CHAPTER 5

A Language for Continuous Location-Dependent Queries in Indoor Environments

Contents

The aim of this chapter is to provide an indoor-based system that favours navigation of users in such environments based on the hierarchical and context-dependent data model presented in Chapter 4. A continuous query processing architecture and query semantics to represent location-dependent queries in indoor environments are introduced in this chapter. The architecture is meant to be as generic as possible, thus allowing different kinds of queries to be performed and others to be further added. The key role of this architecture is to maintain candidate objects or paths to target objects up-to-date. Moreover, a query grammar is presented which introduces different semantics to support location-dependent queries and the hierarchical layout of the environment.

The remainder of this chapter is organized as follows. Section 5.1 describes the principles of the architecture and discusses the continuous execution flow of a given navigation-related query. A detailed description of the *route manager* and *query execution engine* components is provided, along with a discussion on *distribution management* issues. The principles of a query grammar for expressing location-dependent queries in indoor environments are presented in Section 5.2. Supported database

operations and additional clauses for enriching query semantics and user preferences are introduced. A specific operator for representing and managing *location granules* is provided, which reflects the hierarchical structure of the environment. Moreover, motivating examples of location-dependent queries are illustrated to show the potential behind those query semantics.

5.1 Continuous query processing architecture

This section presents the general architecture proposed for the continuous processing of several types of location-dependent queries in indoor environments. First, the main principles and assumptions on how moving objects are handled and to what extent they are expected to cooperate. The principal components considered in the architecture are then introduced, with especial focus on *route* and *distribution management* issues.

5.1.1 Principles

Our approach assumes that moving objects cooperate with a given system by providing up-to-date location data when needed. Thus, a minimum intervention of a user device is required for query processing by communicating the location of the user to the system according to a certain *location update policy* [Wolfson et al., 1999a]. The mobile devices of the users are therefore not overloaded with additional query processing tasks. As soon as a location update is received from a moving object involved in a given query, the server starts the reevaluation process by considering the impact of such updates on the active queries. Accurate location data are assumed to be received in real-time from an indoor positioning system, based on recent technologies such as MEMS sensors, Wireless fingerprinting, and magnetic fields [Liu et al., 2007; Gu et al., 2009; Ray et al., 2010; IndoorAtlas, 2012].

For each location-dependent query, the following terms are used (as suggested in [Ilarri et al., 2006a]). A *reference object* denotes an object that represents the reference for a given location-dependent query (e.g., for a range query, the object that indicates the centre of the range). A *target object* represents an object of interest to a given outstanding query, and which belongs to a specific *target class*. It is worth noting that no constraints are imposed on the movements and directions of the reference and target objects. Accordingly, a reference object is assumed to be either in a static location or moving freely in a spatial network with dynamically changing edge weights. Similarly, a location-dependent query can request information about static or dynamic data, depending on whether the target objects are moving or not. For instance, a reference/target object could be a person or a point of interest -POI- (e.g., a room). Therefore, a unique combination of

challenges arises, as the proposed architecture must be able to continuously process different kinds of location-dependent queries, and to take into account additional contextual information, such as the time-dependency and the user profiles (e.g., some areas may be restricted to specific kinds of users, such as the security personnel), as well as the hierarchical layout of the indoor environment.

The general query processing flow is illustrated in Figure 5.1. Navigation-related queries are processed in accordance with this flow, and are executed continuously while the request is not explicitly cancelled by the user. Unlike many query processing approaches that focus on specific types of queries and on specific scenarios, this architecture has the advantage of supporting many different types of queries without making any restrictive assumption. The features that are managed in the environment are: (i) *mobile persons*, each of them carries a mobile device that allows computing his/her current location and communicating it with other entities, and (ii) *objects of interest*, which contribute to enrich the context of the query and are used by the user to provide his/her preferences and constraints (e.g., by using a digital user interface). These features are managed by a set of fixed servers, each of them in charge of: (1) maintaining a part of the hierarchical spatial graph that represents the environment (i.e., a part of the graph covering a certain spatial area); (2) managing data and communicating with objects located within its area; and (3) executing queries or parts of queries whose data are locally available.

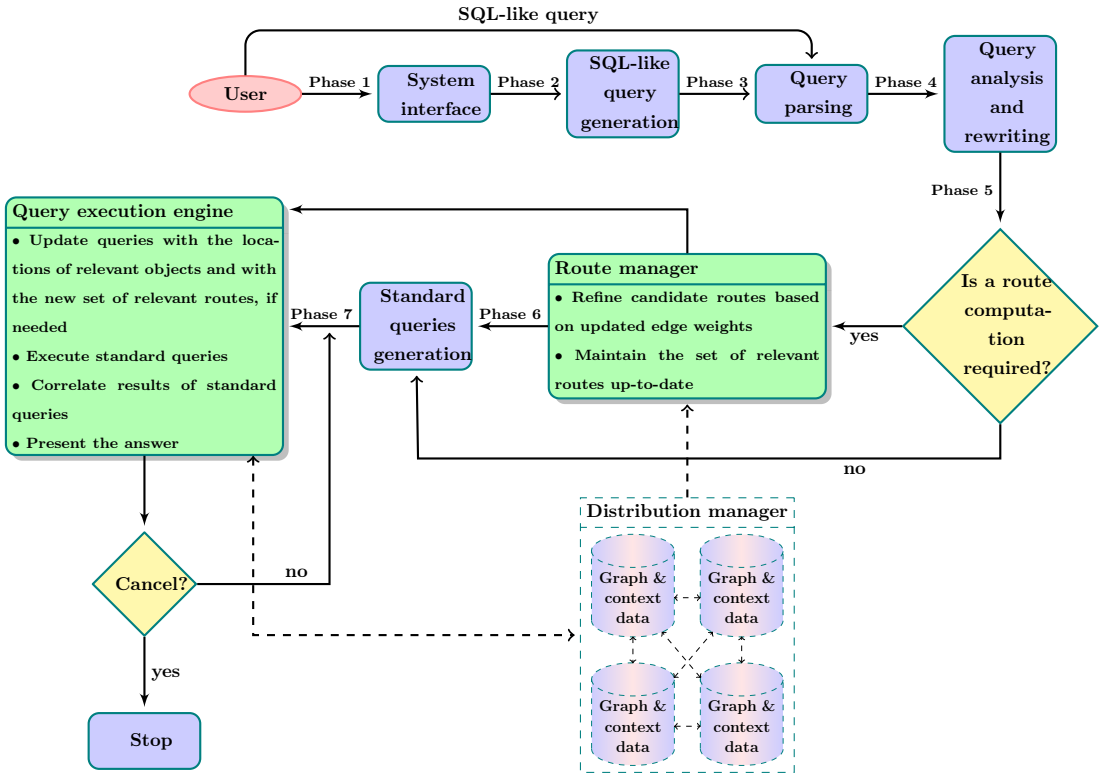


Figure 5.1: Execution of location-dependent queries in indoor environments

5.1.2 Architecture overview

The main phases of the query processing architecture are illustrated in Figure 5.1 and can be described as follows:

Phase 1 and 2. A user interacts with the system interface to issue a query. The system transforms the query expressed in a natural or high-level language into an SQL-like format, as proposed in [Ilarri et al., 2006a]. We assume that an expert user can also directly issue an SQL-like query based on the syntax described in Section 5.2.

Phase 3. Parsing a query implies lexical, syntactic, and semantic analysis of the query expressed in an SQL-like format in order to derive a valid internal representation (e.g., a query graph [Kossmann, 2000]).

Phase 4. A query plan is prepared that is composed of all the operations that are required to appropriately answer the user request. This not only includes typical relational operations (e.g., selection, projection, join, etc.), but also external calls to specific functions that implement new query operators that are defined and discussed in the next section. For optimization purposes, some typical transformations can take place, such as the removal of redundant predicates, the simplification of complex expressions, etc. In addition, for each constraint in the query, the reference object and its target classes are obtained. Furthermore, information regarding the *location granules* (discussed in the next section) of the reference and target objects is retrieved, if the use of location granules is specified in the query.

Phase 5. All navigation-related queries that need to expand routes either towards a specified target object (e.g., an optimal path search towards a destination) or in all directions with a maximum threshold (in the case of range queries), are directed to the route manager in charge of determining the candidate routes based on user-defined preferences and contextual data (e.g., information about user profiles and descriptions of objects of interest). The main tasks performed by the route manager are explained in Section 5.1.2.1.

Phase 6. Obtaining *standard SQL queries* from an SQL-like query is required, since data elements are assumed to be stored in relational databases, which only accept standard SQL. A location-dependent query is broken up into standard queries and operations that are organised in an execution plan to optimise system resources. Not all the operators are necessarily translated to equivalent

standard queries; for instance, operators related to route computation are directly handled at the algorithmic level. The candidate routes obtained by applying such operators could, however, be used as the input data to complete the construction of some standard queries.

Phase 7. In this phase, candidate routes along with an execution plan including standard queries and operations arrive at the *query execution engine*. Timestamped data about locations of relevant objects as well as other contextual data are associated with operations, so that the query engine can execute these queries appropriately. The continuous processing of a query means that the execution of simple queries and operations is kept alive until receiving an explicit request from the user to cancel that query. Therefore, the engine must repeatedly perform the following tasks: (1) update simple queries with the locations of relevant objects and with the new set of relevant routes, if needed; (2) execute standard queries; (3) correlate the results of the different subqueries; and finally (4) present the answer to the user.

5.1.2.1 Route management

Two main tasks are performed by the *route manager* in order to execute navigation-related queries:

Task 1: *Obtaining an initial answer.* Depending on whether a target object is specified or not, different strategies are applied. A specified target implies expanding a directed tree routed at the node where the reference object is located, and oriented towards the target object. For a static shortest path problem, this can typically be solved using Dijkstra's or A*'s algorithm [Dijkstra, 1959; Hart et al., 1968]. A more complex and challenging scenario for estimating the route cost and computing the optimal path arises when considering parameters such as dynamic edge weights, a hierarchical graph structure and, most importantly, the need for an incremental approach for continuous path search with moving reference and target objects.

In a range query, a maximum threshold or a radius is specified instead of a target object. Therefore, all the qualifying objects located within this radius are retrieved. A slightly different strategy consists of expanding all the routes whose network distance from the source node is less than or equal to the specified radius. Once again, this typical problem becomes significantly more complex when incorporating the aforementioned elements. New path and range search algorithms that can appropriately deal with all these challenges are introduced in Chapter 6.

Task 2: *Maintaining the answer up-to-date.* Incremental search algorithms are required in order to execute continuous location-dependent queries, without having to solve each search problem independently from scratch [Sun et al., 2009; Yuan and Schneider, 2010]. Incremental search implies

reusing information from previous searches in order to obtain the current result adaptively. In the case of navigation and range queries, a route planner needs to maintain the set of relevant routes up-to-date, especially when dealing with moving objects. For a navigation query, this means transforming the search tree to an updated tree depending on the movements of objects and other changes in the environment. In a range query, this implies either expanding new sub-trees from boundary nodes (i.e., leaves) or eliminating some of them if the new network distance exceeds the specified threshold.

5.1.2.2 Distribution management

Another component that contributes to the process of refining candidate routes and to the execution of queries is the *distribution manager*. The architecture for processing continuous location-dependent queries over a large space should consider scalability and performance requirements. When considering a large indoor space, a decentralised approach should therefore be proposed to alleviate performance problems when answering continuous queries and managing the corresponding data about moving objects [Gedik and Liu, 2006; Ilarri et al., 2006a]. This approach makes no assumptions about either the number (one or many) of computers deployed in the environment or the spatial zone that should be managed by each computer. Therefore, decisions about data distribution management can be taken at the application level.

Let us consider the sample scenario described in Chapter 4. A three-storey building could be managed by deploying three servers, one for each floor, so that a hierarchical graph is created to represent each floor and stored in the corresponding server[1]. Consequently, (sub)queries and operations whose data are locally available are computed independently and results are communicated to other servers, if needed, or to the user if no other dependencies exist. Therefore, two challenging tasks must be performed by the distribution manager in order to support a distributed query processing:

1. *Keeping track of the relevant servers required to execute a given continuous query.* One can observe that the set of relevant servers changes depending on the locations of the reference and target objects. In the scenario mentioned above, a user from the Linguistics Department located on the first floor may want to reach his/her colleague currently located on the second floor. This scenario needs the first and second servers to be involved in the path planning query. If the target colleague moves down to the ground floor, the distribution manager must detect that the target leaves the area that is currently being watched and then, based on the new location of the target,

[1] Nodes corresponding to a staircase between two floors could be assigned to the server of either of such floors.

the third server should be added to the set of relevant servers.

Therefore, each server is considered to be in charge of keeping (and providing) information about moving objects located within a fragment of the hierarchical data model. Then, for a given query in an indoor environment, a path or a set of paths, that can change dynamically along time, is computed. In this particular scenario, a path between the reference object (i.e., the query issuer in this example) and the target colleague is computed. On the contrary, in the case of a range query (which requests the objects that satisfy the specified constraints and are located within a certain radius from the reference object), the set of all the potential paths within the radius specified is computed. From this set of paths, the set of relevant servers for the query is derived. Therefore, queries that refer to a specific spatial area only require the service of a small subset of servers.

This optimisation phase has more important effects in large spaces (e.g., a campus with several buildings or in scenarios with a high number of moving objects), where using only the relevant servers can significantly reduce the query processing overhead. It should be noted that the information about the hierarchical graph itself is also distributed[2]. In that case, some nodes in the subgraph stored in a server actually represent entry nodes in another subgraph cached in a different server. Such nodes store the information needed to contact the other server.

2. *Computing the answer in a distributed environment.* Once the relevant servers have been identified, each server is queried in parallel to retrieve the objects located within the relevant fragments of the graph. This query has to be executed as a continuous query, as the relevant objects may move continuously. Moreover, objects' movements and changes in the environment can lead to changes in the set of relevant paths, which may in turn modify the set of relevant servers. Therefore, the query processing is assumed to be executed according to a certain *refreshment period* (as in other works, such as [Ilarri et al., 2006a]), since the answer can change continuously.

Thanks to the distribution management, this architecture is generic and can be easily adapted to meet the requirements of a specific scenario. It works in small scenarios where a single server is enough as well as in scenarios that require a higher number of servers.

5.2 A language for continuous location-dependent queries

Several types of queries, such as navigation, range, and nearest neighbour queries, are of interest in an indoor environment. In order to improve query expressiveness, a query grammar is introduced to

[2]Nevertheless, it could also be stored in a centralised manner, as the proposal is general enough to support any scheme.

represent those queries. The detailed syntax of the types of queries considered in this grammar is illustrated in Figure 5.2. This grammar supports navigation queries (of key importance in context-aware indoor navigation systems), and incorporates some other preferences and semantics in the query model. For example, this grammar includes operators (e.g., *All-routes*) and constraints (e.g., *Stop-vertices*) used for navigation queries and inspired by [Booth et al., 2009], which define spatio-temporal restrictions and preferences on the resulting path. In the grammar presented in Figure 5.2, non-terminals start in upper-case and literals are in italics (reserved words) or in lower-case. The following description of the query language highlights the main elements involved in the definition of the queries.

5.2.1 Principles

The query semantics defined in this grammar provide the user with the ability of expressing navigation-related queries in a straightforward manner. Moreover, user preferences are supported at different levels. For instance, a user can specify the *optimization criterion* (i.e., *length* or *time*) on which query computation will be based. Moreover, intermediate stops can be incorporated into query processing which represent some points of interest to the user. Furthermore, users can specify the class of objects that have certain properties or people that share a common interest with them (e.g., friends). Furthermore, queries can take place at different levels of granularities according to user's preferences. A main concern of this approach is to be able to appropriately represent location information at different levels of abstraction, thus supporting the hierarchical indoor data previously described.

On the modelling side, the concept of *location granule*, first introduced in [Ilarri et al., 2011], represents a location at a given level of granularity (i.e., a node at the base level of the hierarchical data model, a room, a floor or a building) . The idea is that it should be possible to express the queries and retrieve the results according to a given location granularity specified by the user. Location granules have an impact on: 1) the presentation of results; 2) the semantics of the queries; and 3) the performance of the query processing (see Section 5.2.3). Additional parameters are implicitly taken into account with the aim of providing more appropriate results depending on the user's context. Indeed, all path computations (i.e., *network length* and *travel time*) are time-dependent. In addition, the answer to a given query depends on the user profile as described in Chapter 4. Even though those parameters are not explicitly represented at the query level, the result to a given query will always depend on them.

General query structure

Query	→	(Standard-query | Navigation-query)
Standard-query	→	*select* (Attr-Projections | '*') *from* Class-names (*where* Conds)?
Navigation-query	→	*select* (Attr-Projections | '*') *from* All-routes-expression (',' Class-names)* (*with* Stop-vertices)? (*where* Conds)? (*optimization-criteria*)?
Attr-Projections	→	Attr-Loc-Select (',' Attr-Loc-Select)*
Attr-Loc-Select	→	attribute | Loc-Select
attribute	→	Qualified-attr | Unqualified-attr
Qualified-attr	→	Class-name '.' Unqualified-attr
Loc-Select	→	Object-id '.' 'loc' | *gr* '('Map-id ',' Class-name ')' | *gr* '('Map-id ',' Route-id ')'
Class-names	→	Class-name (',' Class-name)*
All-routes-expression	→	*All-routes* '(' Loc-Ref ',' Loc-Target ')'
Loc-Ref	→	Object-id ('.' 'coord')? | *gr* '('Map-id ',' Object-id ')' | *gr-map* '('Map-id ',' Gr-id ')' | Vertex-id
Loc-Target	→	Class-name | Object-id | Vertex-id '.' 'coord' | *gr* '(' Map-id ',' Class-name ')'
Stop-vertices	→	Stop-vertex (',' Stop-vertex)*
Stop-vertex	→	Vertex-id
optimization-criteria	→	(*minimize* | *maximize*) Measure
Measure	→	*time* | *distance*

Conditions can be standard conditions on attributes or location-dependent conditions

Conds	→	Cond ((*and* | *or*) Cond)*
Cond	→	(Bool-Cond | LDQ-Cond)
Bool-Cond	→	attribute Comp Value | *intersect* '(' Vertex-set ',' Vertex-set ')' | Value IN Vertex-id '.' POI

Location-dependent conditions

LDQ-Cond	→	*inside* '(' Args-Inside ')' | *nearest* '(' Args-Nearest ')' | ...
Args-Inside	→	Radius ',' Loc-Ref ',' Loc-Target
Args-Nearest	→	K ',' Loc-Ref ',' Loc-Target
Radius	→	Real Units

Basic grammar productions

String	→	([a-z] | [A-Z] | [0-9])+
Real	→	([0-9]+) ('.' [0-9]+)?
K	→	[1-9] [0-9]*
Class-name	→	" String " /* Name of a class of objects */
Unqualified-attr	→	" String " /* Name of a class attribute */
Object-id	→	" String " /* Identifier of an object */
Map-id	→	" String " /* Identifier of a granule map */
Gr-id	→	" String " /* Identifier of a granule */
Vertex-id	→	" String " /* Identifier of a vertex */
coord	→	'(' Real ',' Real ')' /* Two dimensions are assumed */
Units	→	*meters* | *kilometres* | ...
Comp	→	'=' | '>' | '<' | '>=' | '<=' | '<>'
Value	→	([0-9]+) | " String "
POI	→	" String " /* A point of interest */

Figure 5.2: Query grammar for location-dependent queries in an indoor environment

5.2.2 Query semantics

In the general structure of the query language, two kinds of queries are identified: the former typically represents an SQL standard structure (*Standard-query*) along with specific kinds of location-dependent constraints, which are mainly used to express range and nearest neighbour queries. The latter (*Navigation-query*) represents navigation queries that incorporate route computation into the query processing while optimising distance/time criteria.

For a navigation query, the ⟨ FROM clause ⟩ contains an external call to the *All-routes* operator, which has a general syntax as follows: *All-routes(Loc-Ref, Loc-Target)*. This operator returns a non-materialised set of tuples representing valid routes between the current locations of the reference and target objects. A route is a sequence of nodes and edges that can belong to different levels of granularity, which is determined by taking into account the context-dependent data integrated into the hierarchical data model. As a non-materialised table, the set of tuples (routes) obtained as a result of this operator are generated at runtime, and used only to execute the corresponding query. Each generated route is defined as: *route (route-id, source-vertex-id, target-vertex-id, ⟨v1-id, e1-id, v2-id, e2-id, ..., vk-id⟩, length, time)*, where route-id is a route identifier automatically assigned by the system when the route is computed. The *Loc-Ref* and *Loc-Target* arguments may correspond to either a "Vertex-id" or to the locations of the reference and target objects, respectively, but they can also be interpreted as the location granules that contain the corresponding objects, as discussed below. Moreover, *Loc-Target* can refer to the class name of objects of interest (target objects); this is used, for instance, in an inside constraint to retrieve all the objects of a given type.

The ⟨ WITH Stop-vertices ⟩ clause is an optional statement that expresses a user preference implying that the route must go through some place(s) that is(are) of interest to the user. Several *Stop-vertices* can be specified within a single query, and it is assumed that vertices are processed in the order they appear in the query. Furthermore, two different optimisation criteria are applied: *time* and *distance*, which can be considered based on the time-dependent functions defined in Chapter 4. In addition to classical predicates presented in the standard structure of the query, two kinds of location-dependent conditions can be expressed in the ⟨ WHERE clause ⟩: *inside(Radius, Loc-Ref, Loc-Target)* and *nearest(K, Loc-Ref, Loc-Target)*.

A constraint *inside* is applied when performing a continuous range query processing, which takes into account the radius as a maximum threshold to consider, and is used to build the set of routes around the reference object, whose network distances are less than or equal to the radius. The *nearest* constraint is expressed to process continuous k nearest neighbour queries, by specifying the class name of objects of interest in Loc-Target, so that the k objects of interest that are the closest to the current location of the reference object are retrieved.

5.2.3 Managing and representing location granules

The concept of *location granule*, first introduced in [Ilarri et al., 2011], is used and adapted to the context of indoor environments. A location granule identifies a set of fine-grained locations (i.e., geometric coordinates of vertices in the base graph) under a common name. This is completely consistent with the hierarchical spatial graph proposed in Chapter 4. The use of location granules allows to formulate queries with a location resolution which is appropriate for the intended application. With them, it is possible to formulate queries using the location terminology required by the user (e.g., vertices at the fine-grained level, rooms, floors, and buildings). For example, a user may be interested in persons that are near the room where another (moving) object is currently located (see Example 1 in Section 5.2.4). In such a case, the location granule is set to the room level. The operator *gr* is a shorthand for *granule* and returns the location granule associated with a certain object according to a specified *granule map* (i.e., a named set of granules). The use of location granules can have an impact on:

- The presentation of results. The user may want, for example, to retrieve the precise location of the objects in the answer to a given range query. Alternatively, he/she may prefer a different location granularity (e.g., the rooms that contain the objects) as it is more appropriate for his/her context. The answer can be presented using different mechanisms (e.g., different types of graphical, sound-based, or textual representations), which are independent of the required location granularity.

- The semantics of the queries. It is possible to define the queries using a specific location granule. In this case, the answer to the query depends on the interpretation of the location granules. For example, the user may be interested in a specific class of objects within a certain threshold that are near the room where another object is currently located. In this example, the user does not need to know the exact locations of these objects. Therefore, the management of the granules must be implicitly performed, and the search should be based on expanding the location hierarchy (i.e., at the room level) instead of the fine-grained graph.

- The performance of query processing. Some tasks in the continuous query processing demand less resources when coarse location granules are considered. Thus, the objects would move less frequently between granules, and keeping track of their current locations is easier than if precise locations should be considered.

As depicted in Figure 5.2, the location granule operator can be referenced in the SELECT clause, the FROM clause and/or the WHERE clause of a query, depending on whether the granules are used for the presentation of the results and/or for the processing of constraints or routes. For

visualisation purpose, a location granule operator can be used in a *Loc-Select* projection in the SELECT clause, according to the request submitted by the user, to show the result at the desired level of granularity; for example, SELECT gr('room-level', Person) can be used to project the rooms where the persons retrieved by the query are located. In addition, the *gr* operator can be applied on a route, which is the result of a navigation query, to show the sequence of nodes and edges obtained in the route at a chosen abstraction level. For instance, SELECT gr('room-level', Routes.id) could be used to illustrate the sequence of rooms of the valid route, which is made of nodes and edges of different levels (e.g., fine-grained and exit hierarchy levels). In this case, nodes and edges of the resulting route are abstracted to the room level, and the corresponding nodes of this chosen level are shown.

On the other hand, the same *gr* operator can be specified for processing-oriented uses as a *Loc-Ref* and/or *Loc-Target* argument within the FROM clause (i.e., in an *All-routes-expression*), and/or within the location-dependent query constraints (i.e., inside and nearest constraints), in reference to the locations of the reference and target objects, so that they can be interpreted as granules according to a given granule map (i.e., a given level of granularity). For instance, *inside(100 meters, gr('room-level', 'o1'), Person)* is a constraint satisfied by the persons within 100 meters around the room where object o1 is located (Example 1 in Section 5.2.4); similarly, *inside(100 meters, gr('room-level', 'room12'), Person)* is satisfied by the persons within 100 meters around room 12 (note that, in contrast to the previous example, the reference object here is not moving). On the contrary, *inside(100 meters, 'o1', Person)* would be used when the desired range is determined around object *o*1 itself; it should be clarified that *gr('micro-level', 'o1')* is equivalent to o1, that is, a fine-grained granule corresponding to the current fine-grained location of the object is considered by default when the *gr* operator is not explicitly expressed. Consequently, *gr(Map-id, Object-id)* indicates that the location of the object named *Object-id* must be interpreted as a granule in the location granule map identified by *Map-id*. A *location granule map* is a set of granules at a given level of abstraction (e.g., room or floor level). Similarly, *gr(Map-id, Class-name)* generalizes this concept to all the objects of a given class.

Different operations are defined based on location granules and are used for location-dependent query processing. For instance, *contains(Granule-id, Object-id)* returns whether a given object is currently located within a specified granule. *getGranules(Map-id, Class-name)* returns all those granules that contain objects of interest of a given class name. *getDirectGranule(Map-id, Object-id)* returns the direct granule (at the upper level of granularity) that contains the current location of that object.

5.2.4 Motivating examples of location-dependent queries

This section presents typical examples of location-dependent queries. These examples follow the scenario introduced in Chapter 4. In particular, we consider navigation queries, range queries, nearest neighbour queries, and also other specific types of queries.

Navigation queries

The continuous processing of navigation queries is based on a hierarchical path search that relies on a bottom-up technique with different levels of abstraction (i.e., fine-grained, room, floor, and building). The hierarchical path search starts from a user-specified level of granularity, depending on the location granule specified in the request and which contains the initial query point. Below are some typical examples of navigation queries:

1. A user identified by 'userID' wants to find the *fastest* path from his/her current location to the meeting room 'MR01' of the Computer Science Department[3] that goes through a break-room, showing the result at the *room* level:

```
SELECT gr('room-level', RO)
FROM Room AS R, Person AS P,
All-routes(gr('micro-level',P.id),R) AS RO
WITH Stop-vertices v1
WHERE R.id = 'MR01' AND P.id ='userID'
AND 'break-room' IN v1.POI
MINIMIZE time(RO)
```

where time(RO) = $time_{start \rightarrow goal}(t_{start})$ is the estimated time to traverse the path *RO* from 'userID' located at v_{start} to 'MR01'. As previously mentioned, the *gr* operator used in the SELECT statement returns an ordered set of nodes of the optimal route at the room level.

2. Find the *shortest* route from person 'userID1' to person 'userID2', showing the results at the *room* level:

```
SELECT gr('room-level', RO)
FROM Person AS P1, Person AS P2
All-routes(gr('micro-level', P1.id),
    gr('micro-level', P2.id)) AS RO
WHERE P1.id = 'userID1'
```

[3]MR01 is a unique identifier of the structural unit specified by the user and which belongs to the Computer Science Department.

```
AND P2.id = 'userID2'
MINIMIZE length(RO)
```

where length(RO) = $length_{start \to goal}(t_{start})$ is the time-dependent network distance from 'p1' located at v_{start} to 'p2' located at v_{goal}.

3. Retrieve the time needed by all my colleagues of the Computer Science Department to reach the room where I am located:

```
SELECT MAX(t)
FROM Person As P,
(SELECT RO.time
FROM All-routes(gr('micro-level',P.id)
     , gr('room-level', 'myID')) AS RO
MINIMIZE time(RO)) AS t
WHERE 'C.S. Department member' IN P.FD
```

A similar query could be "Retrieve the time needed to evacuate the building", which could be computed as the estimated time needed for the evacuation of the slowest person in the building.

Range queries

Range queries are used to retrieve information about objects or places within a specified range or area. Some range queries have a static reference object and others have a moving reference object. Similarly, the target objects of the queries can be static or moving. The continuous processing of range queries consists in hierarchically expanding all routes whose *network distance* from the source node is less than or equal to the specified radius. Examples of such queries are:

1. Retrieve the identifiers of persons accessible at a network distance smaller 100 meters of the room where object $o1$ is located:

```
SELECT Person.id
FROM Person
WHERE inside(100 meters,
      gr('room-level', 'o1'), Person)
```

2. Retrieve all the communicating entities accessible at a network distance smaller than 100 meters of the user identified by 'userID' and with a communication range of at least 200 meters:

```
SELECT CO.id
FROM Object AS CO
```

```
WHERE inside(100 meters,
      gr('micro-level','userID'), CO)
AND CO.communicate = true
AND CO.commRange >= 200
```

3. Retrieve all the persons who belong to the Computer Science Department and that are accessible at a network distance smaller than 100 meters of the user identified by 'userID':

```
SELECT P.id
FROM Person AS P
WHERE inside(100 meters,
      gr('micro-level','userID'), P)
AND 'C.S. Department member' IN P.FD
```

In the previous query examples, the query issuer does not play the role of the reference object of the query, which shows the generality of the types of queries supported. On the other hand, the reference object can certainly be the query issuer himself/herself.

k Nearest Neighbour queries

A (K) nearest neighbour query retrieves the (K) objects that meet certain specifications and which are the closest to a certain object or location. As in the case of other location-dependent queries, this kind of queries can also be issued by either a static or a dynamic reference object, and applied to either static or dynamic data. Let us show some examples:

1. Find the nearest available bathroom to the user identified by 'userID':

```
SELECT BR.id
FROM Bathroom AS BR
WHERE nearest(1, gr('micro-level', 'userID'), BR)
AND BR.state = 'free'
```

2. Find the two nearest colour printers to each member of the C.S. department:

```
SELECT Pr.id, P.id
FROM Printer AS Pr, Person AS P
WHERE nearest(2, gr('micro-level',P.id), Pr)
AND 'C.S. Department member' IN P.FD
AND Pr.type = 'ColourPrinter'
```

3. Retrieve the nearest extinguisher to each person located in buildings where a fire alarm is beeping:

```
SELECT P.id, Ex.id
FROM Sensor AS Ex,
(SELECT Person.id
FROM Person, Sensor AS FS
WHERE inside(0,gr('building-level',FS.id),P)
AND FS.state = 'active'
AND FS.type = 'Fire-Sensor'     ) AS P
WHERE nearest(1,gr('micro-level',EX.id),P)
AND Ex.type = 'Extinguisher'
```

Reachability queries and reverse range queries

Finally, examples that show the relevance of the proposed query language and implicitly embed interaction spaces associated with each object are hereafter illustrated. Reachability queries check for places and/or objects that are reachable from the current position of the reference object. Indeed, those queries are implicitly processed as range queries by assigning a default threshold value θ to the inside constraint, which is large enough to determine whether the target object/place is reachable or not.

1. *Reachability queries:*

   •Is the room where the object 'objID' is currently located accessible to the user identified by 'userID'? This request should check whether the room that contains 'objID' is within the range of 'userID' having a maximum threshold value set to θ:

   ```
   SELECT Room.id
   FROM Room
   WHERE inside(θ meters,
   gr('micro-level','userID'),
   gr('room-level', Room))
   AND inside(0 meters,Room.id,'ObjID')
   ```

   •Retrieve all the rooms that are accessible to the user 'userID':

   ```
   SELECT Room.id
   FROM Room
   WHERE inside(θ meters,
   gr('micro-level', 'userID'),
   gr('room-level', Room))
   ```

   •Retrieve all the floors of the building that have at least one room accessible to the user 'userID':

```
SELECT DISTINCT gr('floor-level', Room)
FROM Room
WHERE inside(θ meters,
gr('micro-level', 'userID'),
gr('room-level', Room))
```

2. *Continuous reverse range queries.* Retrieve all the entities of type 'Sensor' that are covering the user 'userID' can be considered as an example of a *continuous reverse range query*, according to the definition provided in [Xu and Jacobsen, 2007], since it continuously checks whether a moving object is inside the range of some sensor:

```
SELECT S.id
FROM Sensor AS S, Person AS P
WHERE inside(S.radius, S.id, P)
AND P.id = 'userID'
```

The different examples shown in this section which correspond to several kinds of queries emphasize the potential behind this query grammar in supporting location-dependent queries. This list of examples is definitely not exhaustive since many other scenarios can be imagined, and the language is sufficiently flexible to express other queries in other contexts.

5.3 Discussion

The architecture presented in this chapter provides a continuous query processing approach that can be applied on both static and moving objects, and proposes a generic execution flow for different kinds of location-dependent queries in indoor environments. The query language grammar supports navigation-related queries and incorporates other preferences and semantics into the query model. This language handles the granularity of moving objects' locations, thus favouring the hierarchical indoor data model previously presented.

Although the management of location granules during query processing introduces a certain overhead due to some extra computations, this cost is limited and affordable. Indeed, the use of location granules together with incremental processing help reducing the communication overhead. Moreover, dealing with coarse location granules reduces the number of location updates that must be communicated to the mobile device. Similarly, efforts needed to keep track of the current positions of the reference and target objects are also smaller when coarse location granules are specified in the query constraints. Query processing issues behind the operators developed in this chapter are

presented in detail in Chapter 6. Particularly, the *All-routes* operator and the *inside* constraint are executed by applying incremental and hierarchical approaches to design path and range search algorithms, respectively.

CHAPTER 6

Algorithms for Continuous Path and Range Queries in Indoor Mobile Environments

Contents

This chapter presents several solutions to address the algorithmic issues behind processing continuous location-dependent queries. The algorithms introduced in this chapter constitute the core functions behind the operators and constraints defined in the query grammar introduced in Chapter 5. Algorithms for continuous path and range searches are developed on top of the hierarchical data model described in Chapter 4. These algorithms take advantage of the different levels of abstraction of the data model, and develop an incremental processing paradigm in order to provide an efficient and scalable solution for indoor navigation systems.

This chapter contains the following sections: Section 6.1 highlights the need for an incremental processing paradigm as well as for a hierarchical organization of the underlying data model, in order to improve continuous query processing performance. Section 6.2 presents two complementary algorithms to process indoor path queries, by first introducing the *hierarchical path search* and then by discussing the *incremental processing* of those queries in cases where both the reference and target objects are freely moving in space. Section 6.3 introduces several algorithms for continuous processing of indoor range queries. It first presents a mechanism for *Hierarchical Range Network Expansion*

(HRNE) that extends the RNE approach discussed in Chapter 3. The incremental algorithm for continuous range search is then described. Finally, Section 6.4 emphasizes several points on query processing.

6.1 Hierarchical and incremental processing of continuous LDQs

In contrast to conventional location-dependent queries that consider static reference and/or target objects, the algorithms presented in this chapter seek maximum generality by assuming that target objects can be either in a static location or moving freely in space. Therefore, continuous reevaluation of the query is required while keeping track of the relevant information of objects involved in the processing. An incremental processing paradigm is used for continuously reevaluating such queries, so that previous searches help finding the new answer without recomputing everything from scratch. This requires to build a specific search tree adapted to each kind of queries addressed in this chapter. This search tree is created and dynamically maintained for each query, so that elements are added and/or removed according to a certain policy.

On the other hand, hierarchical search algorithms became important tools for computing shortest paths and other services in time-critical applications like Intelligent Transportation Systems (ITS), due to the growing size of spatial networks [Shekhar et al., 1997]. The algorithms introduced in this chapter rely on the hierarchical data model described in Chapter 4. Particularly, path planning and network expansion are performed hierarchically by using the different hierarchies of the model, thus decreasing computational complexity and favouring a promising solution to alleviate performance and scalability issues in location-dependent query processing. The following sections provide more details on how both the hierarchical and incremental aspects are incorporated in query processing.

6.2 Continuous processing of indoor path queries

For a matter of simplicity, the query processing approach assumes that the reference object will follow its optimal path towards the target. Therefore, incremental search algorithms are required in order to efficiently execute continuous location-dependent queries, thus avoiding solving each search problem independently from scratch [Sun et al., 2009]. Incremental search implies reusing information from previous searches for each query to obtain the current result without having to recompute everything each time. This section first presents an overview on the approach for the continuous processing of path queries is given. Secondly, a detailed description of the hierarchical search and the continuous

processing of indoor path queries is provided, along with pseudocodes of the corresponding algorithms.

6.2.1 Algorithm principles

Path queries encompass all the queries that directly help the users to find and reach points/objects of interest, by providing them with navigational information while optimizing some criteria such as the traversed distance or travel time. Let us introduce an approach for the continuous processing of path queries that relies on a bottom-up technique, and which uses two levels of abstraction, that is, a fine-grained layer at the first level and the exit hierarchy at the second level of abstraction[1]. The search starts from a user-defined level of granularity (depending on the location granule specified in the request and which contains the initial query point) to the highest level of abstraction to find the optimal route at an abstract level. Refinement processes are executed, when needed, to find the exact location of the target object. The main steps of the process can be summarized as follows:

- **Step 1:** Find the optimal path within the initial granule until reaching the optimal exit.
- **Step 2:** Search at the abstract level (exit hierarchy) for the optimal path from the exit of the initial granule to the granule containing the target object.
- **Step 3:** Find the optimal path within the last granule to the target object, starting from the corresponding entrance of the granule.
- **Step 4:** Start a continuous path search by taking into account updated locations of both the reference and the target objects (considering the general case of moving targets). This implies transforming an initial search tree rooted at the previous location of the reference object to an updated tree rooted at its current location. The process continues by either expanding new subtrees from the leaves towards the target and/or by removing subtrees that are no longer needed.

The approach retained for processing continuous path queries is refined by two complementary algorithms. Steps 1 to 3 represent the first iteration, which performs the *hierarchical path search algorithm*, presented in detail in Algorithm 1. Step 4 addresses the *continuous processing* of the query, which is presented in detail in Algorithm 2. Those are generic steps that may or may not be completely executed depending on whether the reference and the target objects are moving or not,

[1]The location hierarchy presented in Chapter 4 could also be used if exact positions of objects of interest and accurate distances are not critically important for the user.

and on the location granule specified in the query. Two algorithms are described in the following section, which represent the implementation behind the *All-routes* operator defined in Chapter 5.

6.2.2 Hierarchical and incremental path search algorithm

Algorithms 1 and 2 introduced in this section perform continuous path queries in two phases by taking advantage of the previously mentioned techniques. This approach is considered as a hierarchical and incremental version of A* applied to indoor moving objects. It is based on the hierarchical data model previously described in Chapter 4. Without loss of generality, a complex path query that requires performing all the steps described at the beginning of Section 6.2.1 is considered, which in the example given implies finding an optimal route from person p_1 to person p_2, assuming that p_2 is moving freely in space. Other cases, for instance where the target object is in a static location, can be easily tackled by performing a first iteration of the whole process and then skipping other unnecessary processing tasks (i.e., only Algorithm 1 would be executed). For the sake of clarity, handling granules is also not detailed in the pseudocodes. Scenarios where, for instance, granules at the room level are considered for the reference and/or the target objects, are easier to process and can be directly derived from this general scenario since no fine-grained network search is required.

As a variant of A*, our approach keeps two main data structures: 1) the CLOSED list contains exactly all the nodes that have been expanded (i.e., generated and added to the search tree); and 2) the OPEN list comprises all the nodes of the outer perimeter of the CLOSED list (i.e., outgoing neighbours of the leaves in the CLOSED list) that are not yet expanded. For each node v in the CLOSED list, the following properties are associated:

- The network distance from v to current location of the reference object v_{start} is computed, and referred to as the g-value $g(v)$; it holds that: $g(v) = g(parent(v)) + length_{parent(v),v}$, and $g(v_{start}) = 0$.
- An estimated heuristic value to the target node v_{goal} is applied, and referred to as the h-value, $h(v)$, which helps propagating a *"wavefront"* expansion towards the target node. $h(v)$ is computed as follows: $h(v) = Ed(v_{start}, v_{midway1}) + pathLength_{EX_s,Ex_g} + Ed(v_{midway2,v_{goal}})$, where $Ed(v, v')$ is the Euclidean distance between two nodes at the fine-grained level, $v_{midway1}$ and $v_{midway2}$ represent two midway nodes that belong to the reference and the target exits, respectively, and $pathLength_{EX_s,Ex_g}$ is the precomputed optimal network distance between the optimal exit at the start granule and the corresponding target exit. All-pairs optimal network paths between exits are precomputed and stored by using the $pathLength()$ function. $pathLength()$ is equal to zero if the reference room is equal to the target room or if the dimension

of the *optimal exit path* is equal to one.

- In addition, each generated node is stored along with its path to the start node, a pointer to the predecessor node, the $f(v) = g(v) + h(v)$ value, and whether it has been expanded or not.

Notice that $g(v)$ and $h(v)$ are also time-dependent functions since they are computed by invoking the other time-dependent methods previously defined. The pointer to the parent node, $parent(v)$, is assigned in order to identify a reverse optimal path from the current node to the start node by following v's ancestors. The algorithm expands the node v with the smallest $f(v) = g(v) + h(v)$ from the OPEN list, and terminates when the OPEN list is empty or when the target node has been expanded.

Two main methods are frequently invoked during the execution of Algorithms 1 and 2. They are explained as follows:

- The *adaptedAstar*(*source*, *target*, *inPath*, *out outPath*, *out outLength*) method is used by the hierarchical path search for computing the fine-grained paths at the reference and the target granules. This method can also manage the different layers of the hierarchical data model. It can perform searches either on the fine level separately or on the exit hierarchy to find optimal exit paths. The main feature of this method is that it uses a *priority-queue-like* data structure which is indexed based on the value of $f(v)$ and represents the CLOSED list. A node with the minimum $f(v)$ is indexed on the top of the queue and thus retrieved first. Different values are dynamically inserted in the priority queue and used afterwards for continuous processing. The *inPath* parameter of the *adaptedAstar*(...) method is used by the hierarchical path search to concatenate searches at different layers. Otherwise, *inPath* is considered to be NULL. An expansion procedure *expand*(v) is performed throughout this method, and consists of checking for each neighbour v' of v whether it belongs or not to the OPEN list. If $v' \notin$ OPEN, the method generates v' by setting $g(v')$ to $g(v)+length_{v,v'}(t)$, setting its parent to v, and inserting it into the OPEN list. If v' is already in the OPEN list, then it checks whether $g(v') > g(v) + length_{v,v'}(t)$; if so, then the algorithm sets $g(v')$ to $g(v) + length_{v,v'}(t)$, and $parent(v')$ to v. The way the heuristic function is computed and other optimizations developed to improve the performance of the *adaptedAstar*(...) method are presented in Chapter 7.

- The *computeRefTarExits*($vstart, vgoal$) method is used for computing the *optimal exit path* that minimizes the path between the current locations of the reference and target objects. The result contains an optimal path at the exit hierarchy starting from the granule containing the reference object and ending with the corresponding entrance at the target granule. Notice that an optimal exit to a given object's location should not be necessarily the nearest one in term of

Algorithm 1: *HierarchicalPathSearch(locRef, locTarg,* ***out*** *outPath,* ***out*** *outLength)*

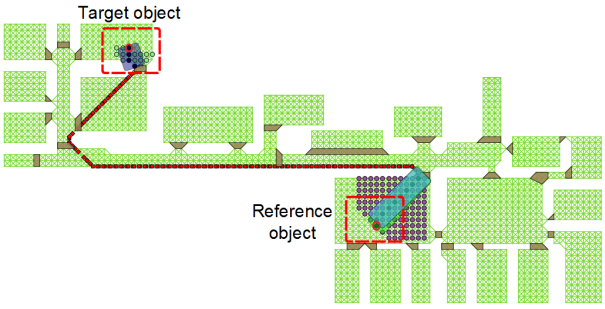
Data: $\mathcal{S} : \bigcup_{i=1,2} \mathcal{S}_i : G_i = (V_i, E_i)$: hierarchical graph data; q: path query.
Result: A sequence of nodes of the optimal path $outPath = \{v_{start}, v_2, \ldots, v_{goal}\}$ to the target object where $v_i \in \mathcal{S}_1 \cup \mathcal{S}_2$, and the resulting network distance $outLength$
// $locRef/locTarg$: Reference/target object location; $g(v) = length_{v_{start},v}(t)$, $h(v)$, $parent(v)$: a predecessor is associated with each node and for each query; $CLOSED$: set of expanded nodes; $OPEN$: set of boundary nodes;

begin
  $CLOSED \longleftarrow \emptyset$
  $v_{start} = getNode(locRef)$; $SGranuleId = getDirectGranule(v_{start})$;
  $OPEN = \{v_{start}\}$; // A new position implies a new root of the tree
  $v_{goal} = getNode(LocTarg)$; $TGranuleId = getDirectGranule(v_{goal})$;
  if $SGranuleId = TGranuleId$ **then**
    $outRecord = adaptedAstar(v_{start}, v_{goal})$;
    $outPath = outRecord.outPath$;
    $outLength = outRecord.outLength$;
  else
    // Retrieve the best pair of exits of the source and target granules, and the corresponding optimal path
    $optimalExitPath = computeRefTarExits(vstart, vgoal)$;
    $sourceExit = optimalExitPath[1]$;
    // Step 1: Directed A* in G_{micro} from v_{start} to the source exit
    select a node $v_{midway1}$ such that $\{v_{midway1} \in sourceExit.nodeListIds$ *and* $v_{midway1} \in SGranuleId\}$;
    $outPath = adaptedAstar(v_{start}, v_{midway1})$;
    // *adaptedAstar* implies removing v with $f(v) = g(v) + h(v) = \min_{v' \in neighbours(v)} f(v')$ from $OPEN$
    // And then *inserting* v into $CLOSED$
    // Step 2: Insert all exits of *optimalExitPath* into $OPEN$
    $generate(sourceExit)$; $insert(sourceExit)$ into $OPEN$; $parent(sourceExit) = v_{midway1}$;
    foreach *exit* $e \in optimalExitPath$ **do**
      // All-pairs optimal network paths between exits are already precomputed
      $generate(e)$; $insert(e)$ into $OPEN$; $parent(e) = e'$; // e' is the predecessor of e
    end
    // Step 3: Directed A* in G_{micro} until reaching v_{goal}
    $targetExit = optimalExitPath[length(optimalExitPath)]$;
    select a node where $\{v_{midway2} \in targetExit.nodeListIds$ *And* $v_{midway2} \in TGranuleId\}$;
    $currentPath = append(outPath, optimalExitPath)$;
    // The final $outPath$ is obtained by applying a $reversePath$ procedure from v_{goal} to v_{start} following v_{goal}'s ancestors
    $outRecord = adaptedAstar(v_{midway2}, v_{goal}, currentPath)$;
    $outPath = reversePath(v_{goal}, v_{start})$;
    $outLength = outRecord.outLength$;
  end
end

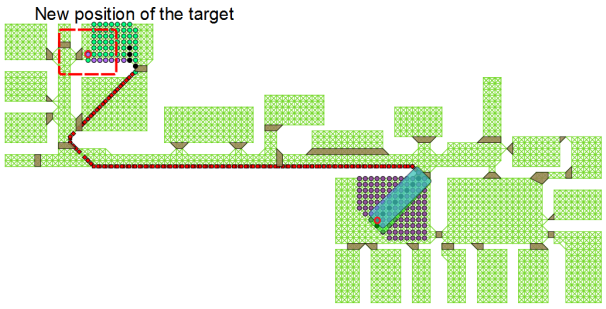
distance, but rather the one that optimizes the whole network distance between the reference and the target objects. This function is invoked when applying the continuous processing of hierarchical path and range searches.

6.2.2.1 Hierarchical path search

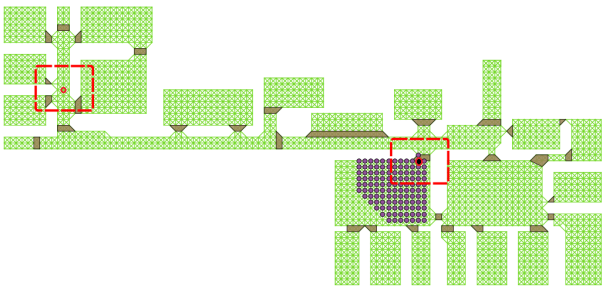
The pseudocode of the hierarchical path search is illustrated in Algorithm 1. Figure 6.1(a) shows an example of a hierarchical path returned as a first path result between two moving objects. The main steps performed in this algorithm are explained as follows:



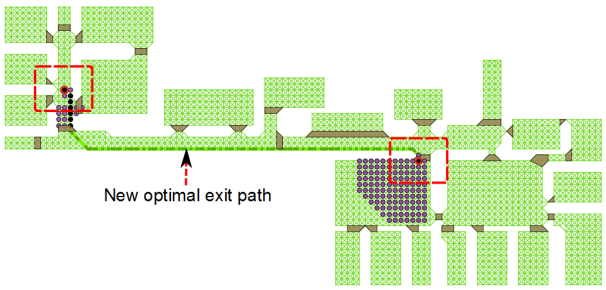
(a) Hierarchical path search



(b) 1st scenario: target object is moving



(c) 2nd scenario: target object is changing its current granule



(d) 2nd scenario: adapting the pair of exits

Figure 6.1: Hierarchical and incremental path search algorithm

- The first part (*lines* 2 $\rightarrow$ 4) depicts the initialization of variables by involving the *queryLocation*() and *getNode*() methods, which return the current location of the object and the corresponding node in G_{micro}, respectively. *getDirectGranule*(v) returns the identifier of the granule containing v. The *getNode* method computes the nearest node to the current location of the moving object.

- The *lines* $5 \rightarrow 8$ check whether the current granules of the reference and target objects match. In that case, no hierarchical search is needed, but instead an invocation the *adaptedAstar* method is performed, and a first result is returned. Otherwise, the $computeRefTarExits(vstart, vgoal)$ method is invoked to retrieve the best pair of exit/entrance that correspond to the source and target granules, along with the corresponding optimal exit path (*lines* $10 \rightarrow 11$).

- **Step 1:** Once the optimal exit path is computed, the algorithm starts the hierarchical path search by firstly performing a fine-grained search at the reference granule until reaching a midway node that belongs to the source exit previously identified (*lines* $12 \rightarrow 13$).

- **Step 2:** Upon identifying and reaching the first exit of the computed optimal exit path, the algorithm moves up to the upper level at the exit hierarchy, and inserts all exits of the optimal exit path into the priority queue (i.e., CLOSED list), so that concatenated paths of two levels of granularity will be associated to each of those generated exits (*lines* $14 \rightarrow 17$). The $generate(e)$ methods sets $g(e)$ to $g(v) + length_{v,e}(t)$, and $parent(e)$ to v. This step continues until reaching the optimal entrance at the target granule.

- **Step 3:** The hierarchical path search algorithm ends by performing a fine-grained search starting from an identified midway node at the target granule towards the exact location of the target, and then by identifying a shortest path in reverse by following v_{goal}'s ancestors until reaching the source node (*lines* $18 \rightarrow 23$).

6.2.2.2 Continuous query processing

A continuous processing of path queries starts at this phase by taking into account the updated locations of the reference and the target objects. A pseudocode of this algorithm is illustrated in Algorithm 2. A description of this algorithm is given as follows:

- A fundamental step consists of invoking the hierarchical path search method previously described in order to build for the first time the search tree, which will be stored in the priority queue structure (*lines* $2 \rightarrow 4$). After this step, a first path result is returned to the user, and all generated nodes in the search tree rooted at v_{start} are associated with the above mentioned properties.

- A continuous path search starts with the aim of keeping the initial tree up-to-date. At each iteration, the algorithm looks for up-to-date locations of the reference and target objects, and then matches those locations to nodes at the fine-grained network (*lines* $6 \rightarrow 8$). As long as the search tree is rooted at the same v_{start} (i.e., the reference object is not moving) and the target

Algorithm 2: *ContinuousHPath(refObjId, tarObjId)*

Data: $\mathcal{S} : \bigcup_{i=1,2} \mathcal{S}_i : G_i = (V_i, E_i)$: hierarchical graph data; q: path query; up-to-date location data of ref/target objects.

Result: A continuous set of optimal paths *outPath*, and the resulting network distance *outLength*

// $refObjId/tarObjId$: Reference/target object identifier; $CLOSED$: set of expanded nodes; $OPEN$: set of boundary nodes.

1 **begin**
2 $locRef = q.queryLocation(refObjId)$; $v_{start} = getNode(locRef)$;
3 $locTarg = q.queryLocation(tarObjId)$;
4 $[outPath,\ outLength] = hierarchicalPathSearch(locRef, locTarg)$;
// At this stage, a complete search tree has been built and stored
// Continuous path search (keeping the initial answer up-to-date)
5 **while** $v_{start} \neq v_{goal}$ **do**
6 $previous\text{-}v_{start} = v_{start}$;
7 $v_{start} = getNode(q.queryLocation(refObjId))$;
8 $v_{goal} = getNode(q.queryLocation(tarObjId))$;
9 **if** $previous\text{-}v_{start} == v_{start}$ and $v_{goal} \in CLOSED$ **then**
// The answer is returned without extra computation
10 $[outPath, outLength] = \{reversePath(v_{goal}, v_{start}),\ g(v_{goal})\}$;
11 **else**
12 **if** $previous\text{-}v_{start} \neq v_{start}$ **then**
// An updated search tree is being created with a new root v_{start}
13 $updateTreeRootedAt(v_{start})$; // *g-values* are not affected
14 $deleteUnnecessaryNodes()$; // Unnecessary nodes from previous $CLOSED$ are deleted
15 $completeOPEN()$; // Nodes of the outer perimeter are added
16 **end**
17 **if** $v_{goal} \notin CLOSED$ **then**
// Tracking of v_{goal}; check the new optimal pair <exit/entrance>
18 $newOptimalExitPath = computeRefTarExits(vstart, vgoal)$;
19 **if** $(newOptimalExitPath == optimalExitPath)$ **then**
20 continue A* in G_{micro} with the same $OPEN$ and $CLOSED$ lists until reaching v_{goal};
21 **else**
// v_{goal} is either nearer to another exit within the same granule or has left the last granule
22 *delete* subtree rooted at $LastExit$ from $CLOSED$;
23 *insert* not generated exits from $newOptimalExitPath$ into $OPEN$;
24 *repeat Step 3* of Algorithm 1 starting from $parent(newOptimalExitPath[n])$ until reaching v_{goal};
25 **end**
26 **end**
27 $[outPath, outLength] = \{reversePath(v_{goal}, v_{start}),\ g(v_{goal})\}$;
28 **end**
29 $sleepUntilNextPositionUpdate(minWaitingTime)$; // The thread remains asleep while no location update is performed or the minimum waiting time between iterations (if specified) has not been consumed
30 **end**
31 **end**

object is located on a node in the CLOSED list, a shortest path can be easily determined in reverse from v_{goal} towards v_{start} ($lines$ $9 \rightarrow 10$).

- When the reference object moves ($lines$ $12 \rightarrow 15$), additional steps to transform an initial search tree rooted at the previous v_{start} to an updated tree rooted at the current v_{start} are needed. Three main functions are invoked to perform this transformation: 1) The $updateTreeRootedAt$ method firstly updates pointers to parent nodes at the reference granule so that nodes of the reference granule are rooted at the new v_{start}; 2) secondly, the $deleteUnnecessaryNodes()$ method removes unnecessary nodes from the previous CLOSED list; and finally 3) $completeOPEN$ is called to add nodes of the outer perimeter of the new CLOSED list to the new OPEN list.

- In case v_{goal} is not located in the CLOSED list, a new invocation to the $computeRefTarExits()$ method is performed to determine the new optimal exit path towards the target ($lines$ $17 \rightarrow 21$). If the new optimal exit path matches the previous one (i.e., this means that the same target exit is still the nearest one), the algorithm performs a directed search in G_{micro} with the same OPEN and CLOSED list until reaching v_{goal} (Figure 6.1(b)).

- Otherwise, the target is either nearer to another exit within the same granule or has left the last granule ($lines$ $22 \rightarrow 25$). In that case, additional checks are performed to detect the *last common exit* between the new and previous exit paths. Once determined, the subtree rooted at that *Last exit* is no longer needed and will be removed from the CLOSED list, along with the nodes at the fine-grained level. Instead, a new subtree is created starting from the *Last exit* and by inserting exits of the new optimal exit paths, if any, until reaching the new optimal target exit. Finally, a similar search similar to the one performed in step 3 of Algorithm 1 is afterwards completed to reach the target (see Figures 6.1(c) and 6.1(d)).

- An optimal path is returned for each iteration from the current location of the reference object towards the current location of the target object. The $sleepUntilNextPositionUpdate()$ method is then invoked so that the thread remains asleep until the reference and/or target objects update their locations. Additionally, to keep the query processing overhead low in the presence of high location update rates, we may require a minimum time interval between iterations, by passing an optional argument $minWaitingTime$.

Notice that we are refreshing the answer periodically, as advocated in other works such as [Ilarri et al., 2006a]. This is necessary because the answer will change all the time (even if slightly) due to the movements of the reference object and the target objects.

6.3 Continuous processing of indoor range queries

Range queries find and retrieve objects or places of interest within a user-specified range or area. Those queries support navigation by continuously updating relevant details according to the users' movements. In our scenarios, ranges are characterized by a circular in which objects of interest must be located. This section introduces an approach for the continuous processing of range queries which considers the mobility of both the reference and the target objects. This approach is based on a hierarchical range network expansion mechanism. The principle behind that approach is to continuously update the set of visited nodes that compose the range around the reference object. Furthermore, an indexed data structure referred to as *range queue* is built as a result of the hierarchical network expansion. Similarly to the priority queue structure described in Section 6.2.1, this structure maintains several properties associated to the generated nodes, such as the optimal path from the current node towards the source node. This particular property offers a significant advantage since it allows the system to provide not only information about whether an object is inside a specified range, but also to return a complete optimal path to that target object. Consequently, the result of a continuous range query includes the set of qualifying object identifiers, their optimal path towards the reference object, and the corresponding network distance. A detailed description along with pseudocodes for the corresponding algorithms are hereafter provided.

6.3.1 Hierarchical range network expansion

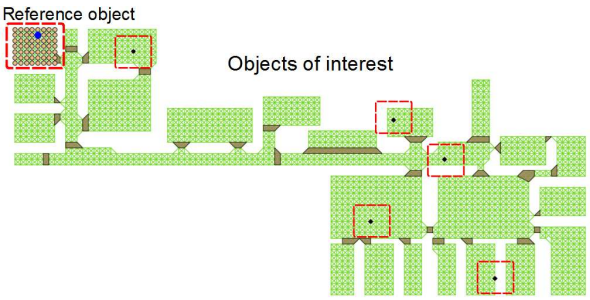
A *hierarchical range network expansion* (HRNE) is first computed starting from the location of the reference object in a similar way to the Dijkstra algorithm with multiple destinations and the Range Network Expansion algorithm [Papadias et al., 2003], see Figure 6.2(a). It consists of a "*wavefront*" expansion of the hierarchical network starting from the initial query point in all directions to find all nodes whose network distances to the source are less than the maximum specified threshold (i.e., the radius of the range query). The original idea here is that the valid routes are expanded hierarchically (cf. Figure 6.2(b)). The hierarchical network expansion mechanism is introduced in Algorithm 3 and takes into account the bottom-up approach explained in Section 6.2 to efficiently expand the valid routes within the specified radius (and considering network-based distances, not Euclidean distances). The main steps of the $hierarchicalNetworkExpansion(refObjId, radius, objectIds[])$ algorithm are described as follows:

- *Step 1:* Gradually expand the valid routes in all directions within the initial granule of the reference object v_{start}, located at $LocRef$, while $length_{v,v'}(t_v) < radius$ ($lines\ 3 \rightarrow 4$). Nodes that are temporarily inaccessible or occupied by physical objects are automatically ignored.

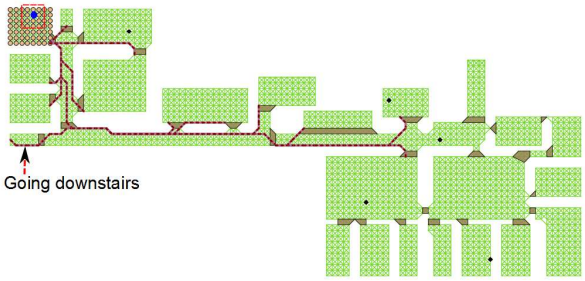
Algorithm 3: *HierarchicalNetworkExpansion(refObjId, radius, objectIds[])*

Data: $\mathcal{S} : \bigcup_{i=1,2} G_i = (V_i, E_i)$: hierarchical graph data; q: range query; up-to-date location data.
Result: *ResultSet*: Returns a SETOF [targObjID, outPath, outLength] for the qualifying target objects
// $C \subseteq objectIds[]$: candidate set; $LocRef$; $g(v)$; $parent(v)$; $RANGE$: set of nodes around the reference object; $coveredRooms$: set of totally/partially covered rooms.

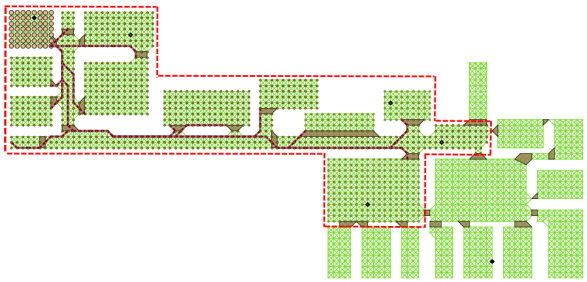
```
1  begin
2      C ⟵ ∅; coveredRooms ⟵ ∅;
       // Step 1: At this stage, only nodes of the reference object's granule are expanded
3      v_start = getNode(q.queryLocation(refObjId)); SGranuleId = getDirectGranule(v_start);
4      RANGE = networkExpansion(v_start, radius);// Network expansion only at the reference granule
       // A new search tree RANGE is built after Step 1 and stored in the range queue
       // Step 2: Network expansion at the Exit Hierarchy
5      foreach accessible exit e ∈Exits of the reference granule do
6          select an expanded node v_midway1 in e;
7          RANGE = append(RANGE, networkExpansion(e, radius - g(v_midway1), v_midway1.path));
8      end
       // Step 3: Search for the qualifying objects
9      foreach potentialQualifyingObject ∈ objectIds do
10         v_goal = getNode(q.queryLocation(objectIds[i])); TGranuleId = getDirectGranule(v_goal);
           // Totally/partially covered rooms have at least one accessible exit
11         if objectIds[i] ∈ coveredRooms then
               // Apply Euclidean restriction at the target granule
12             retrieve exit e ∈ TGranuleId that minimizes g(e) + Ed(e, v_goal);
13             if g(e) + Ed(e, v_goal) > radius − g(e) then
14                 display objectIds[i] is out of range;
15             else
                   // A fine-grained search at the target granule is required
16                 select an expanded node v_midway2 in e;
17                 RANGE = append(RANGE, networkExpansion(v_midway2, radius − g(e), e.path));
18                 if v_goal is expanded then
19                     [targObjID, outPath, outLength] =
                       {objectIds[i], reversePath(v_goal, v_start), g(v_goal)};
20                 else
21                     display objectIds[i] is finally out of range;
22                 end
23             end
24         else
25             display objectIds[i] is out of covered rooms;
26         end
27     end
28 end
```



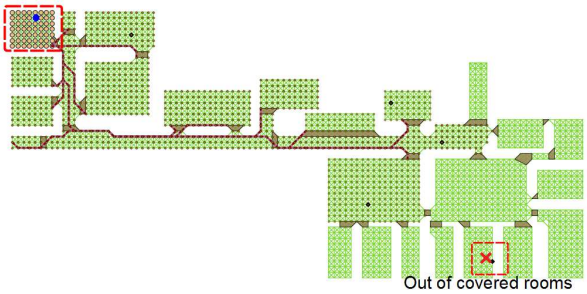
(a) Fine-grained search at the reference granule



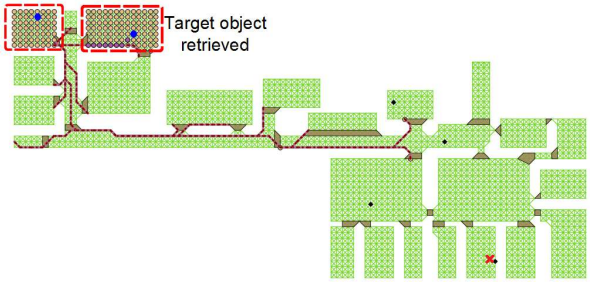
(b) Network expansion at the exit hierarchy



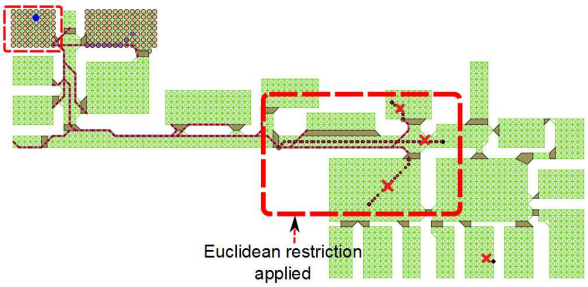
(c) Totally/partially covered rooms are retrieved



(d) Objects outside covered rooms are directly eliminated



(e) First object in range is returned



(f) Second object is eliminated by Euclidean restriction

Figure 6.2: Incremental algorithm for continuous range search: A range of 50 meters is applied in this example

An internal invocation to the $networkExpansion(v_{start}, radius, inPath[])$ method is performed to execute this step. The range queue data structure is created for storing nodes accessible within the specified range. The $networkExpansion(...)$ method also handles expansion at different layers of granularity depending on the input values. When the network is expanded at the fine-grained level, only nodes that belong to the same granule as the reference object are expanded. Moreover, an $inPath$ parameter is also used to smoothly generate routes that are concatenated to previously expanded paths from earlier range searches ($inPath$ takes NULL as a default value). In a similar way to Algorithm 1, an expanded node is stored along with its path to the reference object, a pointer to the predecessor node, and its g value.

- *Step 2:* On the exit hierarchy, start an expansion in all directions from the detected exits of the initial granule by taking into account the set of precomputed network distances ($exitPaths$) between directly reachable exits ($lines\ 5 \rightarrow 8$). The expansion stops when no more exits can be added (i.e., when $g(e) \geq radius$). The resulting search tree includes all valid routes that consists of sets of vertices at two different levels of granularity. The rooms that are reachable from at least one entrance are considered as *covered rooms* (Figure 6.2(c)). Those covered rooms are determined to limit the search scope, so that target objects located outside this area are directly discarded.

- *Step3:* Search for the qualifying target objects by taking into account their up-to-date locations ($lines\ 10 \rightarrow 11$). Different filtering processes are applied in order to avoid extra-computations resulting from searches at the fine-grained level ($lines\ 11 \rightarrow 15$). The algorithm first discards an object if its current location is out the covered rooms (Figure 6.2(d)). For an object located within the covered rooms, it is checked whether the Euclidean distance between the optimal exit of the target granule and its current location is greater than the radius. If the check is successful, the object will also be discarded for that iteration (Figure 6.2(f)). Otherwise, the algorithm proceeds by performing a network range expansion at the fine-grained level within the target granule until reaching all valid nodes that satisfy the specified threshold ($lines\ 16 \rightarrow 22$). if v_{goal} has been discovered, a composite result that consists of a triple $\langle targObjID, outPath, outLength \rangle$ is returned (Figure 6.2(e)).

At the end of this process, all the generated nodes that constitute the valid routes within the radius are stored, along with their associated properties. The leaves, also referred to as *boundary nodes*, resulting from the range search are also returned. Such a hierarchical expansion provides a light way of exploring the network around the reference object and is performed just for the first time. It should be clarified that only exits and the paths between those exits are examined, but knowledge of nodes of the corresponding granules, which are not necessarily reachable with the same specified threshold, is not available. Therefore, all the corresponding granules of the valid exits are assumed to be accessible, but extra computations are required to determine, for each candidate object located at one of those nodes, whether that object really satisfies the distance constraint (i.e., to avoid false positives). This is done by computing the optimal path at the fine-grained level starting from the entrance of the target granule until reaching the target object.

6.3.2 Incremental algorithm for continuous range search

The algorithm introduced for the continuous processing of range queries in indoor environments applies an Euclidean restriction mechanism to retrieve candidate target objects that might be relevant

to the final answer, as well as the hierarchical network expansion mechanism previously described. The continuous processing of range queries consists of:

1. Hierarchically expanding all the routes whose network distance from the source node is less than or equal to the specified radius. A hierarchical network expansion is performed once for the first iteration so that all the visited nodes within the range around the reference object are stored.

2. Continuously maintaining the set of parent nodes up-to-date when changing the root of the search tree (i.e., when the reference object moves). Boundary nodes are checked to decide, for each of them, whether to further expand that node or to perform a reverse search towards the source in order to remove nodes that are not relevant any more.

Algorithm 4 illustrates the implementation of the *inside* constraint used in this kind of query. The algorithm is described as follows:

- Two functions are first invoked in *lines* $3 \rightarrow 5$. The first one applies the Euclidean restriction principle to retrieve candidate target objects, and the second one performs the hierarchical network expansion mechanism previously described in Algorithm 3.

- The first round of the algorithm returns a set of triples for the qualifying target objects. For the continuous processing, the main point is to update the set of parent nodes when changing the root of the search tree (i.e., when the reference object moves). There is no need to update all the distances to the new root. Instead, only distances and the parent pointers of nodes that belong to the granule of the reference object need to be rechecked, so that the tree rooted at the new position of the reference object is rebuilt (*lines* $10 \rightarrow 12$). This update operation performs checks and modifies properties associated to the leaves as explained in the next step.

- Only boundary nodes (i.e., leaves) of the RANGE list are checked to decide, for each of them, whether to further expand that node or to perform a reverse search towards the ancestors to remove nodes that are not relevant any more (i.e., the network distance to the new source is greater than the radius). For each boundary node, the algorithm first updates the network distance to the new source node, and checks whether that new distance is still less than the specified threshold (*lines* $13 \rightarrow 17$). If the check is successful, it completes the RANGE list by starting a new network expansion, and adds the valid nodes to the RANGE list. Otherwise (*lines* $18 \rightarrow 25$), it starts searches in the reverse direction from each boundary node, and removes nodes that are no longer needed from the RANGE list. This reverse search continues while the network distance from the current node towards the source node is exceeding the

Algorithm 4: *ContinuousRangeSearch(refObjId, radius, objectIds[])*

Data: $\mathcal{S} : \bigcup_{i=1,2} G_i = (V_i, E_i)$: hierarchical graph data; q: range query; r: network distance; up-to-date location data; $NetDistanceSet$.

Result: $ResultSet$: Returns a SETOF [targObjID, outPath, outLength] for the qualifying target objects

// $C \subseteq objectIds$: candidate set; $locRef$; $g(v)$; $parent(v)$; $RANGE$: set of accessible nodes around the reference object; $N \subseteq RANGE$: set of boundary nodes, $tempSet$: temporary set of nodes.

```
1  begin
2      C ⟵ ∅;
3      locRef = q.getRefObj.queryLocation(refObjId);
4      C = getObjectsInEuclideanRange(locRef, objectIds, radius);
5      RANGE = hierarchicalNetworkExpansion(refObjId, radius, C);
6      while NotCancel do
7          if locRef != q.getRefObj.queryLocation(refObjId) then
               // A new position implies a new root of the tree
8              locRef = q.getRefObj.queryLocation(refObjId);
9              C = getObjectsInEuclideanRange(locRef, objectIds, radius);
10             foreach v ∈ RANGE and v ∈ getDirectGranule(getNode(locRef)) do
11                 UpdateParent(v);
                   // After this step, all the nodes in RANGE are rooted at the new locRef
12             end
13             foreach v ∈ N do
14                 length_{locRef,v}(t_v) = updateLength(v);        // reverse path search to locRef
15                 if length_{locRef,v}(t_v) ≤ radius then
16                     tempSet = networkExpansion (v, radius − length_{v,locRef}(t_v), v.path);
17                     append(RANGE, tempSet);
18                 else
19                     v_current = parent(v);
20                     delete(RANGE, v);
21                     length_{LocRef,v_current}(t_{v_current}) = updateLength(v_current);
22                     while length_{LocRef,v_current}(t_{v_current}) > radius do
23                         v_current = parent(v_current);
24                         delete(RANGE, v_current);
25                     end
26                 end
27             end
28         end
29         foreach o ∈ C do
               // computePartOfPath will repeat steps similar to Step 3 in Algorithm 3
30             v_goal = getNode(q.queryLocation(o));
31             if intersect(RANGE, getNode(o.LocTarg)) and
               computePartOfPath(e.path, getNode(o.LocTarg)) < radius then
32                 [targObjID, outPath, outLength] = {o, reversePath(v_goal, v_start), g(v_goal)}; // e is the optimal target exit to o
33             else
34                 display objectIds[i] is finally out of range;
35             end
36         end
37     end
38 end
```

specified threshold. After this step, a new set of valid routes around the current position of the reference object is rebuilt.

- *Lines* 29 $\rightarrow$ 35 determine whether the target object is located on a node of the RANGE list. If so, the algorithm completes the partial path computation, as explained in Step 3 of Algorithm 3, starting from the optimal entrance of the target granule, and then checks whether that distance satisfies the specified threshold (i.e., $computePartOfPath(e.path, getNode(o.LocTarg)) < radius$). Target objects whose network distances to their current positions satisfy the maximum distance constraint are returned in the result.

6.4 Discussion

Algorithms for continuous path and range searches over indoor moving objects were proposed in this chapter. Two complementary algorithms (Algorithms 1 and 2) represent a hierarchical and incremental path search mechanism, which can be executed at different levels of granularity, and applied on static and/or mobile data. Algorithms 3, and 4 perform continuous range searches by applying a hierarchical network expansion mechanism and an incremental Euclidean restriction approach.

The algorithms take advantage of the underlying hierarchical data model to provide more scalable and more efficient solutions to address these kinds of queries. It should be emphasized that the underlying properties defined in the data model (e.g., time-dependent functions, user profiles, and event management) remain valid, and are implicitly taken into account in query processing. Management of location granules can be easily applied to those algorithms, and can be considered as safe areas around reference and/or target objects when a coarser granule is specified. The solutions proposed are promising in terms of scalability and performance as will be shown in Chapter 7.

CHAPTER 7

A PostgreSQL Extension for Continuous Location-Dependent Query Processing

Contents

This chapter presents the design and implementation of a database extension based on the open source DBMS PostgreSQL. The prototype developed handles continuous path searches and range queries on top of the hierarchical network-based data model of an indoor environment. The main parts of the prototype developed include: (i) a hierarchical network-based data model of indoor environments (cf., Chapter 4); (ii) operators and location-dependent constraints introduced in the query grammar (cf., Chapter 5); and (iii) algorithms to process continuous location-dependent queries over moving objects (cf., Chapter 6). Results of the experiments that have been conducted to investigate the scalability and performance of the whole approach are also reported.

The remainder of this chapter is organised as follows. A technical assessment of existing prototypes and systems that support location-dependent queries is provided in Section 7.1, along with some implementation requirements. Section 7.2 presents the system architecture and summarizes the main parts of the developed prototype. Algorithmic optimizations are introduced in Section

7.2.2 in order to improve the efficiency of the proposed solutions. An experimental evaluation of the proposed solutions is provided in Section 7.3. Finally, Section 7.4 draws some conclusions

7.1 Comparative study of existing platforms for handling LDQs

This technical summary first describes some implementation requirements and then briefly presents a comparative study of existing platforms for handling location-dependent queries or that can serve as a basis to support such kinds of query processing. The goal of this study is to assess existing systems with respect to certain implementation requirements that are established in order to design a prototype which encompasses the different contributions described earlier.

The implementation requirements to design a prototype for handling continuous location-dependent queries in indoor environments are as follows:

1. An extensible DBMS that supports developing network-based data models to represent the hierarchical indoor data model described in Chapter 4. One can think of designing the spatial network either in *relational* or in *graph-oriented* database management systems [Angles and Gutierrez, 2008]. Although graph databases are well suited for designing such network models, other issues regarding the query language support and continuous query processing are still not completely addressed and might be more difficult to implement.

2. The ability to develop a new algebra (i.e., specific data types and spatio-temporal operators) that represents the different queries handled in this context. This supports the implementation of the clauses and operators defined in Chapter 5.

3. Implementation of several algorithms for location-dependent query processing over moving objects. Particularly, the algorithms presented in Chapter 6 on top of the hierarchical data model should be implemented with an incremental processing paradigm.

Existing platforms that can handle location-dependent queries are briefly presented in this section. The goal is to derive the best suited system that supports the previously mentioned requirements. It can then be decided whether one of these platforms can lend itself to support further developments and to be extended by other functionalities, such as the solutions introduced in previous chapters. It should also be emphasized that this comparison is provided from a system design and performance points of view. The methodologies behind some of these systems have been discussed in Chapter 3. The criteria selected to evaluate those systems depend mainly on whether they support spatio-temporal databases as well as network-based data models, and whether they are extensible or not.

In addition, efficient management of moving objects is an essential requirement, by either supporting moving object database concepts or incorporating a data stream management technique. Such a system should also provide an ease of learning and acceptable programming time, so that new features can be integrated. Open source platforms and those that have a large community support are thus preferred. Furthermore, whether a decentralised or centralised approach is supported is also of interest to certain kinds of applications. Table 7.1 summarizes different properties of the relevant systems and platforms. The systems considered in this evaluation are *DOMINO*, *LOQOMOTION*, *PLACE*, *SECONDO*, and *PostgreSQL*.

- The *DOMINO* prototype handles current and future movements with the notion of dynamic attributes [Wolfson et al., 1999b]. It is built on top of spatio-temporal databases and has been implemented on top of a network-based data model, thus favouring MOD applications in road networks. The *Future Temporal Logic* language supports spatial and temporal operators, and handles uncertainty issues derived from moving object data. However, this prototype has not been maintained and no functionality has been provided to support continuous location-dependent queries.

- *LOQOMOTION* is a platform that supports distributed processing of continuous location-dependent queries in mobile environments using mobile agents [Ilarri et al., 2006a]. The platform uses MySQL for data storage and simulates moving objects in a free Euclidean space (geometric coordinates). It also handles locations at different levels of granularity, and location-dependent constraints. This platform could be adapted to support network-based models but substantial adaptations should be made regarding the underlying query processing algorithms, as well as on the movements of simulated objects.

- *PLACE* presents an approach that is based on spatio-temporal databases and data stream management systems to handle continuous queries [Mokbel et al., 2005]. This platform integrates and handles real-time data streams for continuous processing of location-dependent queries. It proposes a progressive evaluation paradigm with a predicate-based expiration approach to handle data streams. However, there is no support for a query language associated to this system. In addition, this platform has not been maintained and the source code is not available for further developments.

- *SECONDO* is an Open Source and extensible database system that uses the Berkeley DB for data storage. It is written in C++. It is designed to support MOD concepts and spatio-temporal queries over objects' trajectories. This systems does not have a fixed data model but rather allows one to implement DBMS data models as a set of algebra modules; each providing specific data types and operators. For instance, algebras to represent standard data types, spatial data

	System requirements					Other requirements			
	Spatio-temporal DB	**Network-based Modelling**	**Support Decentralized Architecture**	**Extensible Query Language**	**Moving Object Stream Management**	**Ease of Learning**	**Programming Time Estimation**	**Community Support**	**Is Open Source**
PLACE [Mokbel et al., 2005]	⊕ yes	⊖ no	⊖ no	⊖ no language support	⊕ yes	⊖ no support	⊖ cannot be estimated	⊖ very limited	⊖ no
DOMINO [Wolfson et al., 1999b]	⊕ yes	⊕ yes	⊖ no	⊕ yes	⊖ no	⊖ no support	⊖ cannot be estimated	⊖ very limited	⊖ no
LOQOMOTION [Ilarri et al., 2006a]	⊕ yes	⊖ no	⊕ yes	⊕ yes	⊖ no	⊖ cannot be estimated	⊖ cannot be estimated	⊖ limited	⊖ no
SECONDO [De Almeida et al., 2006]	⊕ yes	⊕ yes	⊕ yes (Parallel Secondo)	⊕ yes	⊕ could accept streams from an external application	⊖ difficult	⊖ could be long	⊖ limited	⊕ yes
PostgreSQL (+ extensions)	⊕ yes (PostGIS, pgRouting & Hermes [Pelekis et al., 2006] extensions)	⊖ graphs are represented as tables	⊕ yes	⊕ yes	⊕ can be supported (TelegraphCQ extension [Chandrasekaran et al., 2003])	⊕ medium	⊕ reasonable	⊕ very large	⊕ yes

Table 7.1: Comparative study of existing systems for handling location dependent queries

types, relations, B-tree indexes, R-tree indexes, and network-based data types are supported. Some work on indoor data types is also under development. So far, SECONDO handles only histories of moving objects, but it may be extended to fit further needs. One challenge is the design of a hierarchical data model on top of the proposed network algebra. *Parallel SECONDO* is a new extension that combines the MapReduce paradigm [Dean and Ghemawat, 2008] for parallel and distributed processing with a set of SECONDO databases, providing almost all existing SECONDO data types and operators [Lu and Guting, 2012]. Therefore, parallel queries are also supported and can be executed on a cluster of computers. Although SECONDO provides a comprehensive set of features with an efficient query processing paradigm, the development and technical adaptations required in this system are not straightforward, especially, given that there is no large community support behind.

- *PostgreSQL*[1] (with different extensions: PostGIS [Obe and Hsu, 2011], pgRouting[2], TelegraphCQ [Chandrasekaran et al., 2003], and Hermes [Pelekis et al., 2006]) is an Open Source Object-Relational Database System (ORDBMS). It is a multi-user, multi-threaded database management system that runs on multiple OS platforms. PostgreSQL is developed by the PostgreSQL Global Development Group. This DBMS system supports the design of relational data models, new data types, functions, operators, triggers, etc. Some indexing methods are also provided as built-in features such as B-Tree indexes, R-Tree indexes, and GiST indexes used to speed up searching in large data sets. PostGIS and pgRouting could be used to add spatial properties and navigation functions based on geometric data types and operators. pgRouting extends PostGIS to further support routing algorithms (shortest paths, etc.) on spatial networks. Several procedural languages are supported by PostgreSQL, such as PL/pgSQL, in order to develop functions and algorithms directly as stored procedures within the server. A similar problem to the one presented in SECONDO is the lack of a built-in functionality that provides us with a continuous processing engine. Nevertheless, there exists a PostgreSQL extension called *TelegraphCQ* that manages data streams such as location data of moving objects. A potential use of this extension can be expected for expressing continuous queries over spatial streams. The *Hermes* extension has also been developed on top of PostgreSQL to further provide additional spatio-temporal operators for querying objects' trajectories. Examples of such queries are *nearest neighbour trajectories* and *trajectory similarities*.

Concluding remarks: Several system and implementation requirements are defined in this section in order to evaluate existing systems for handling location-dependent queries. The assessment shows that PLACE and DOMINO have not been maintained, so they cannot be considered for

[1] http://www.postgresql.org/
[2] http://pgrouting.org/

further development. LOQOMOTION can be extended but requires substantial adaptations at the algorithmic level, as well as on the movements of simulated objects. Consequently, two systems are considered suitable for the development required in our context: Secondo and PostgreSQL. On the one hand, Secondo is an Open Source platform that appears to be sufficiently powerful and extensible, and can be adapted to integrate our proposed solutions. However, there is still no direct way to process current movements, even though adjustments are believed to be possible in order to achieve that purpose. Moreover, the development and adaptations required in this system are not straightforward, especially, given that there is no large community support behind. On the other hand, PostgreSQL is supported by a large community of developers and is highly extensible. Network-based models are represented in PostgreSQL with a relational schema. Even though PostgreSQL performance appears to be less efficient than Secondo, it presents several relevant characteristics and advantages. Consequently, it has been selected for the development of our solutions as will be shown in the following sections.

7.2 System implementation

This section underlines the implementation experience by describing the PostgreSQL system architecture as well as several optimizations performed at the algorithmic level.

7.2.1 Overview

A database extension based on the open source DBMS PostgreSQL [Matthew and Stones, 2005] for handling continuous path searches and range queries has been implemented on top of a hierarchical network-based indoor data model. Several procedural languages are supported in PostgreSQL for developing functions and algorithms at the server side, so that the connection overhead and interprocess communication can be avoided. As a result, queries that are written as internal functions have the same access privileges and speed as native database functions and statements. The main parts of the prototype developed are:

1. The hierarchical network-based data model of an indoor environment. Automatic methods to build the multi-storey fine-grained network and the time-dependent functions described in Chapter 4 have been developed. Methods to derive the exit and location hierarchies are also included.

2. The operators and location-dependent constraints introduced in Chapter 5. Those are implemented in the backend executor (i.e., query processor) of the PostgreSQL as PL/pgSQL

functions applied on user-defined types. For instance, the *gr* operator for managing location granules in indoor environments, the *All-routes(Loc-Ref, Loc-Target)* operator for representing continuous path queries, and the *inside(Radius, Loc-Ref, Loc-Target)* constraint for computing continuous range queries were developed. Other operators, data types, and keywords were also incorporated in the system design.

3. The algorithms to process continuous location-dependent queries over moving objects (cf. Sections 6.2 and 6.3). The following sections discuss in more detail the technical optimizations and the performance evaluation of these solutions.

The main advantage behind this prototype implementation as a core database solution is that user-defined PL/pgSQL functions are used afterwards with native SQL statements to write location-dependent queries. However, the major problem encountered in implementing those functions was the lack of data structures naturally supported in the database system. This required the use of temporary tables in the implementation of the priority and range queues. For system evaluation, synthetic moving object datasets have been generated by using the Brinkhoff's network-based generator of moving objects [Brinkhoff, 2002], and then adapted to fit our needs.

7.2.2 Optimization

A series of optimization techniques have been employed to improve the efficiency of the proposed solutions. Those are explained with respect to their use in the path and range query processing as follows:

- A specific heuristic function has been developed and applied overall to the continuous path query processing, which tries to optimize the network distance based on the hierarchical data model previously described. As mentioned earlier, a heuristic value is computed as follows: $h(v) = Ed(v_{start}, v_{midway1}) + pathLength_{EX_s,Ex_g}(t) + Ed(v_{midway2,v_{goal}})$. This heuristic function has been specifically designed to fit the hierarchical structure of the indoor environment. Consequently, a best estimation of the network distance towards the destination is taken into account during the expansion process, so that the node that minimizes the $gval + fval$ value is expanded first. The hierarchical-based heuristic function is used in the $adaptedAstar(...)$ method for directed path search, and in the $computeRefTarExits(...)$ method for computing the optimal exit path.

- An indexed *priority-queue-like* data structure for implementing the CLOSED list. A priority queue is characterized by a tuple $\langle vertexID, gval, fval, path, predecessor, expanded \rangle$, where

vertexID depicts the node identifier, *gval* represents the network distance, $fval = gval + hval$ is an indexed parameter that is used as the priority measure to allow optimal network expansion, *predecessor* contains the parent node, and the *expanded* field depicts whether the node has been expanded or not. On the other hand, the range queue is indexed based on the *gval*, since no heuristic function is used in the network expansion mechanism. Those two data structures constitute the foundations on which the continuous processing for both algorithms is performed.

- Directional bounding boxes that help propagating a *"wavefront"* path search either towards the optimal exit or the target (cf. Figures 6.1(a) and 6.1(b)). Directional bounding boxes are considered as an important optimization of the *adaptedAstar*(...) method. Those directional boxes limit the search for the neighbour nodes to those that are in the route direction towards the next goal. As a result, only five neighbours are generated and stored in the priority queue each time, instead of eight (the maximum number of neighbours). This reduces the execution time by 40%. A directional box is either oriented towards the next nearest exit or towards the target node if the reference and target objects are in the same room.

- The *computeRefTarExits*(...) method computes, for path searches, the best pair of exit/entrance when the reference and/or the target objects move. For example, in Figure 4.3 let us consider $MR12$ and $HW14$ as a reference room (i.e., a room where the reference object is located) with five exits and a target room (i.e., a room where a target object is located) with seven exits, respectively. Next, a basic approach is to check all combinations of pairs of exits to determine the best pair. To optimize this process, an additional filtering process is developed in order to prune exits that do not have direct links to the target room and where no other open paths through them are available.

- For the continuous processing of range queries, two filtering techniques are employed, thus reducing the number of fine-grained network expansions at the target granules. First, totally/-partially covered rooms are determined to limit the search scope, so that objects out of that search scope are directly discarded. Secondly, an Euclidean restriction at the target granule is applied to detect candidate objects that are far enough away from the reference object.

7.3 Experimental evaluation

To the best of the authors' knowledge, no other work in the field of location-dependent query processing deals with hierarchical and continuous path searches and/or range queries on both moving reference and target objects in indoor environments. As mentioned in Chapter 3 (Sections 3.3.2 and 3.3.3), other approaches do not consider a multi-storey network, and in the case of range queries,

either the reference or the target objects are assumed as static. Therefore, experimental results to evaluate the intrinsic properties of the proposed solutions are presented in this section, since it is not possible to experimentally compare our approach with other proposals. In particular, a specific comparison between hierarchical and non-hierarchical processing approach is shown throughout these experiments. This criterion shows to what extent the continuous processing of a query is affected with respect to the mean execution time (in milliseconds), as well as the total number of expanded nodes in the search tree. The mean execution time shows the average CPU time of a continuous query answer for each location update. In contrast, the criterion about the total number of nodes shows the usefulness of the incremental processing approach by giving an indication of the global size of the search tree for a complete query evaluation.

In order to test the non-hierarchical configuration of both algorithms, two main methods have been developed. The $nonHierarchicalCPS(refObjId, tarObjId)$ method uses the $adaptedAstar$ function to build a complete search tree at the fine-grained level, instead of making use of the hierarchical path search described in Chapter 6. This $nonHierarchicalCPS$ method performs techniques for the continuous path search similar to those applied in Algorithm 6.1, in order to update the search tree. For the continuous range search, the $nonHierarchicalCRS$ method uses a $nonHierarchicalNE$ method for the network expansion at the fine-grained level. The performance of these methods is evaluated in the following sections with respect to the solutions proposed in Chapter 6 (Sections 6.2 and 6.3).

7.3.1 Experimental settings

Two different system architectures can be applied for query processing. The former considers a server-based query processing architecture (either centralised or decentralised as discussed in [Afyouni et al., 2013]), where moving objects cooperate with the system by providing up-to-date location data (and possibly other information) when needed. Thus, a minimum intervention of a user device is required for query processing by communicating the location of the user to the system according to a certain location update policy [Wolfson et al., 1999a; Ilarri et al., 2010]. The latter applies a client-based mobile architecture in which query processing is fully performed at the mobile device and locations of objects of interest are retrieved from the server. The first scenario implies more communication overhead, while the second scenario requires mobile devices with advanced processing capabilities. The first approach is adopted in our experimental settings, but nothing prevents testing those algorithms on a client-based mobile architecture. In the following section, scalability as well as performance testing are evaluated.

Experiments have been carried out on a MacBook Pro machine with a 2.3 GHz Intel Core i7 CPU and 4GB of RAM DDR3, and which runs Mac OS X 10.8.3. All tests were run 10 times in a completely independent way, and we verified that the individual results were consistent. The fine-grained two-storey network used for prototype evaluation consists of 4146 nodes and 13963 edges (see Figure 4.2 in Chapter 4, Section 4.2.1.1). The scenario considered for the performance evaluation retains a fine resolution of the fine-grained network by using a 50 cm distance between horizontal and vertical neighbour nodes. A coarser resolution would provide better performance results, but with less accurate representation of space and objects movements. Thus, the size of the network as well as the experimental results heavily depends on this parameter.

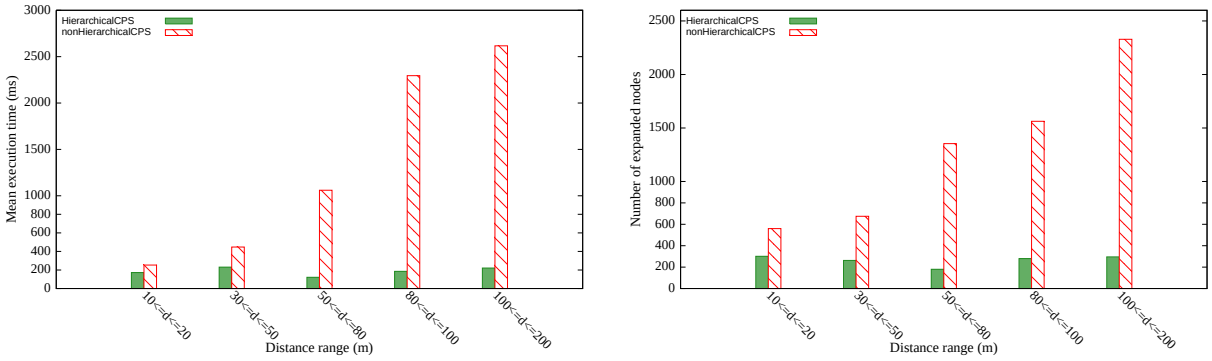
Due to the lack of real indoor moving object data, a synthetic dataset of around 1000 moving objects have been generated to evaluate how the prototype would behave in realistic scenarios. We use the Brinkhoff's network-based generator [Brinkhoff, 2002], which is suitable for all kinds of spatial networks. It is a generic framework that can be adapted to specific scenarios. Indeed, the original purpose of this generator was to deal with moving objects on road networks. Hence, this generator does not directly deal with 3D network models, as the third dimension is not taken into account. Consequently, moving object data have been generated for each floor separately. The fine-grained graph of our model has then been integrated within the generator and some parameters have been tuned to generate indoor moving objects with realistic movements. Moreover, *a-posteriori* adaptations to the data set have been performed in order to take the multi-storey settings into account, and to transform the location data to the relative coordinate system considered in our scenario. In particular, we adapted some trajectories to simulate moving objects that move from one floor to another. Two additional methods have also been developed: (i) the $inverseTransformation()$ function computes an inverse transformation to obtain coordinates in our referential system; and (ii) the $computeNearestNode(xCoordinate,\ yCoordinate)$ determines the nearest node to a given moving object position.

The generator takes two input files (i.e., *.node* and *.edge* network data files) which correspond to the nodes and edges of the fine-grained network of the spatial data model. Different configurations have been adopted to define the mobility patterns of the generated moving objects. The duration of the evaluation period was set to 1000 timestamps. The waiting period between two successive timestamps was set to 1 second. The entire evaluation period is estimated to be around 15 minutes. Every moving object reports its motion parameters (i.e., location update, current speed) with a probability of 80% for each timestamp within the evaluation period. Objects move on the network at different speeds, with a maximum speed limit equals to 4 km/h. We choose some objects randomly and consider them as reference and target objects for both path and range queries.

7.3.2 Experimental results

The following experiments first evaluate the continuous path search while varying the distance parameter between the reference and target objects. Secondly, a performance evaluation of the continuous range query processing is performed with respect to the number of objects and the radius parameters.

Continuous path searches: The first set of experiments shows how the continuous path search can be affected by applying a hierarchical or non-hierarchical-based query processing. The estimated distance between the reference and target objects is varied to demonstrate its impact with respect to both the average CPU time and the number of nodes expanded (see Figures 7.1(a) and 7.1(b), respectively).



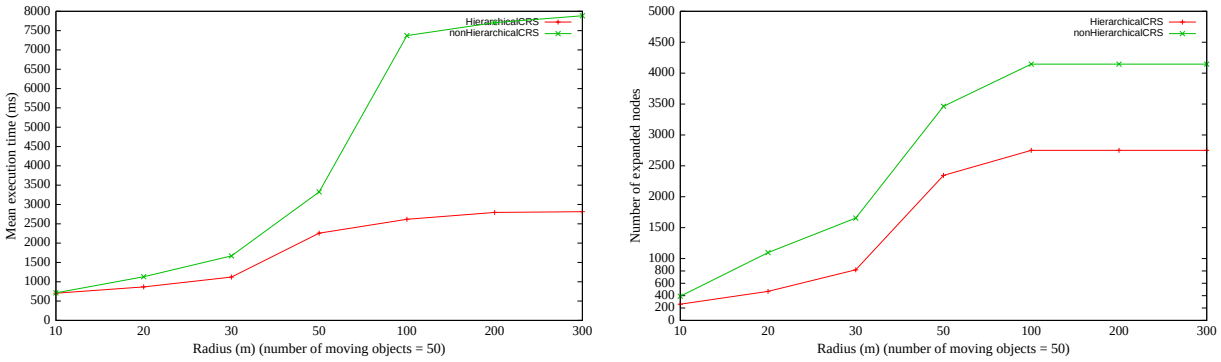
(a) (b)

Figure 7.1: Varying the distance parameter: Hierarchical vs. non-Hierarchical Continuous Path Search

Figure 7.1(a) illustrates the mean execution time of a continuous path search with a distance range that varies from 10 to 200 meters. The same continuous processing techniques were applied for both hierarchical and non-hierarchical configurations. The results show that the hierarchical approach keeps constant time responses when the distance between the reference and the target object increases. On the contrary, a non-hierarchical configuration appears to grow with the distance between the reference object and the target objects. This is due to the fact that the hierarchical method processes fine-grained searches only at the reference and target granules, and thus whatever the distance between the two moving objects the time processing remains constant. We also performed several tests with other distance values (not shown for the sake of clarity) and observed a similar behaviour. On the other hand, a relatively large distance between the two moving objects implies exploring a big part of the fine-grained network until reaching the target. This is clearly reported in Figure 7.1(b). As illustrated, a path search between two moving objects whose distance

is between 100 and 200 meters, requires expanding around 200 nodes when applying the hierarchical approach, and around 2400 nodes with a non-hierarchical configuration. This demonstrates that the hierarchical and incremental path search algorithm is scalable to large indoor spaces (e.g., several multi-storey buildings in a campus) with constant time responses.

Continuous range queries: Regarding the continuous range query algorithm, some parameters such as the range and the number of moving objects can be varied. This aims to show the scalability of the system and the algorithm behaviour over time. A performance comparison between two different scenarios based on either a hierarchical or a non-hierarchical network expansion mechanism is also considered. The next set of experiments studies the impact of varying the radius of a range query while setting the number of moving objects to 50 (see Figures 7.2(a) and 7.2(b)).



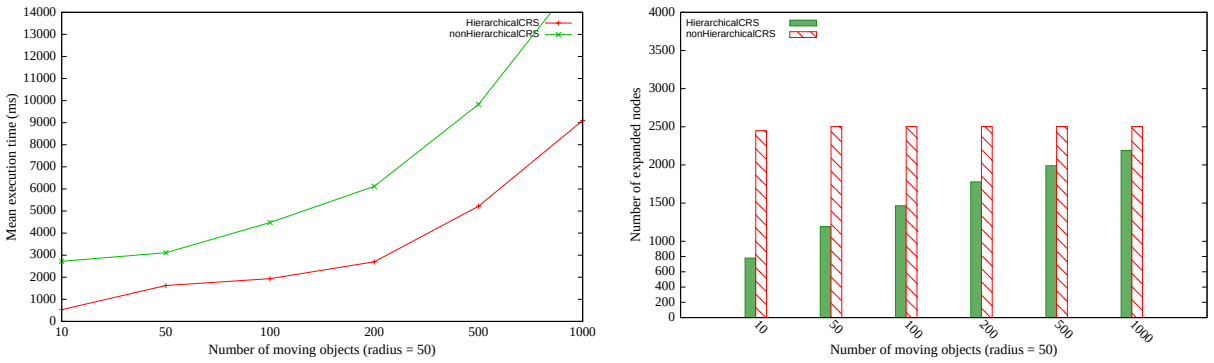
(a) (b)

Figure 7.2: Varying the radius: Hierarchical vs. non-Hierarchical Continuous Range Search

The hierarchical network mechanism takes advantage of the exit hierarchy to explore the search scope without having to expand all the nodes in range at the fine-grained level. This means that this mechanism is dependent on the number and the locations of the target objects in the search scope. Wherever a candidate object requires a fine-grained search, the algorithm will explore the granule containing that object to decide whether it is is really within the range. Other granules in the search scope are not going to be expanded while no candidate objects enter those granules. On the other hand, a non-hierarchical network expansion is completely dependent on the search space. It expands all nodes within the range without taking into account the number and locations of the candidate objects. Notice that for the hierarchical approach a constant time is obtained after reaching a certain radius. This means that once the corresponding granules have been explored no extra-computation is required. Moreover, the evaluation of both configurations shows gains with respect to the number of nodes expanded during the whole process. Again, the total number of expanded nodes reaches a maximum for the hierarchical approach once the corresponding granules

have been explored. On the contrary, the non-hierarchical approach reaches the maximum number of nodes of the network being considered in those experiments (i.e., it explores all the network).

The next set of experiments presents the performance evaluation and scalability of the two configurations with respect to the number of moving objects (see Figures 7.3(a) and 7.3(b)). In these experiments the radius parameter is set to 50 meters, while the number of moving objects varies between 1 and 1000. The results show significant improvement when applying the hierarchical processing, and an acceptable execution time even with 1000 moving objects. We should remind that this query returns, on each timestamp and for each moving object in range, the optimal path to the reference object. Consistently with the first result, Figure 7.3(b) shows that, whatever the number of moving objects specified by the user, the non-hierarchical configuration explores all the nodes within the specified radius. On the contrary, even with 1000 moving objects, the hierarchical approach is able to answer the query with a much smaller number of expanded nodes.



(a) (b)

Figure 7.3: Varying the number of moving objects: Hierarchical vs. non-Hierarchical Continuous Range Search

7.3.3 System Scalability

PostgreSQL supports full parallelism at the client-side, so that applications can open multiple database connections and manage them asynchronously, or via threads. Multi-thread Java programs with a connection pooling mechanism have been developed in order to simulate a multi-user environment, and to show the effect of concurrent continuous queries on the performance of the system (in this scenario, a single multi-core PostgreSQL server). We investigate the average response time of a continuous query per user. The response time considered in these tests is the average time interval between issuing a continuous query and getting the response from the system at a given timestamp when the search is successfully completed. The number of threads varies from 1 to 200 in

the first set of experiments that consider concurrent path queries, and from 1 to 50 for concurrent range queries.

Figure 7.4 illustrates the average response time for a continuous path query at a given timestamp for a given user. The number of concurrent users querying the system in real-time varies from 1 to 200. Simulation results suggest that, with 30 to 50 concurrent accesses, the average response time varies between 1 and 1.5 seconds. Even with 100 to 200 concurrent users, the time for a query answer remains acceptable, and the number of concurrent queries has a linear impact on the performance (it should be noted that the X-axis in the figure is not linear).

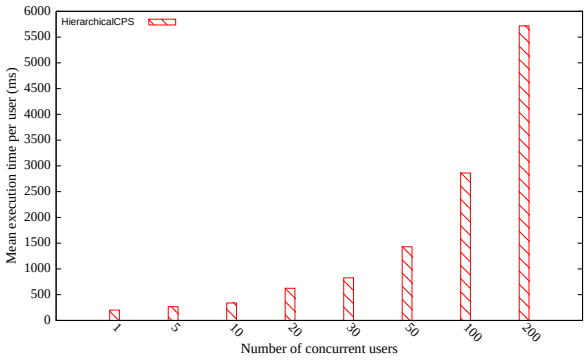
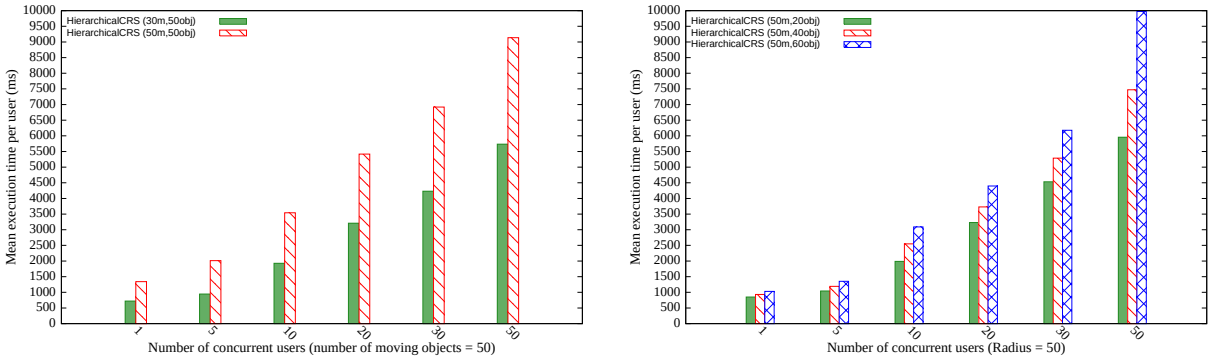


Figure 7.4: Varying the number of concurrent access: Hierarchical Continuous Path Search



(a) (b)

Figure 7.5: Varying the number of concurrent access: Hierarchical Continuous Range Search

The results of the experiments that consider concurrent range queries are illustrated in Figures 7.5(a) and 7.5(b). As may be expected, a range query results in heavy processing costs, which have been noted by previous tests (Figures 7.3(a) and 7.3(b)). Therefore, the simulation shows a reduced number of concurrent accesses that varies from 1 to 50. Two types of experiments have been performed. On the one hand, Figure 7.5(a) illustrates the average response time while varying the number of concurrent users, and with two different thresholds: 30 and 50 meters. Simulation

results show a good performance of the average query answer with up to 30 concurrent users, and an acceptable time response with up to 50 users. On the other hand, Figure 7.5(b) shows the performance evaluation of a concurrent range query with different numbers of moving objects: 20, 40 and 60. The system shows a good scalability with up to 50 concurrent users when the number of objects specified is low. With a bigger number of moving objects, the system can report good response times with up to 30 users.

7.3.4 Summary of the experiments

According to the above experimental results, the execution of the algorithm developed for continuous path search appears as satisfactory regarding execution time and scales well with the number of expanded nodes. It has been shown to be scalable enough to large indoor spaces thanks to the hierarchical-based query processing. Moreover, the continuous range query processing approach provides satisfactory scalability with respect to the radius parameter, and acceptable performance in processing range queries when the number of moving objects increases. Regarding the experimental results for the continuous range search, all moving objects involved are assumed to be of interest to the corresponding query. Indeed, only objects of a certain type (the ones involved in the query) have a direct impact on the performance of the query processing, so this generates a worst-case situation. A pre-filtering of objects based on static properties (e.g., people in my friend list) has a similar effect, as this reduces the number of objects to consider as a potential candidate (i.e., moving objects not in the friend list are immediately discarded). Consequently, and for example, the largest mean execution times shown in Figure 7.5(b), when applied to a friend-finder application, would imply a range of 50 meters and 60 persons in the list of friends that should be at the same time in the same indoor environment. Nevertheless, the total number of moving objects, independently of their type, has also a slight impact on the performance of the server due to the need to manage their location updates.

Furthermore, the whole system has been tested for scalability with respect to the number of concurrent continuous queries. A multi-user environment has been simulated in order to show the effect of concurrent continuous queries on the performance of the system. The system shows satisfactory scalability for path queries with up to 200 concurrent users, and acceptable response times for range queries with up to 50 concurrent users. Consequently, a general analysis and assessment of the algorithms suggest that our approach can be used for real-time services. Moreover, in some scenarios where the number of concurrent users becomes high, the performance can be increased by adopting the distributed data management approach described in Chapter 5.

7.4 Conclusions

This chapter introduces a database extension that has been fully implemented on top of the open source DBMS PostgreSQL for handling continuous path searches and range queries has been implemented on top of the hierarchical network-based indoor data model. This system architecture has been developed, and optimizations that improve query processing performance have been discussed. Experiments that investigate the scalability and performance with respect to the intrinsic properties of the proposed solutions have been presented. Results show that our proposal achieves a satisfactory performance, and it is efficient enough to be used in a real scenario. Experimental results show a mean execution time of around 0.2 second for continuous path searches, even in cases where the distance is quite large for an indoor scenario, and reasonable response times for continuous range searches. Furthermore, the whole system was tested for scalability with respect to the number of concurrent users issuing a continuous query. The results show that the system is fairly scalable and adapted to a multi-user environment.

A technical observation can be highlighted regarding the implementation on top of PostgreSQL. In PostgreSQL, internal functions can be thoroughly written in PL/pgSQL or dynamically linked to functions in C libraries and which use the Server Programming Interface (SPI) provided by PostgreSQL to access data. Internal PL/pgSQL functions are well suited for running code that consists of sets of inter-referenced database queries. However, some of the core algorithmic functions can be exported and written as C/SPI procedures, because these procedures are able to access things deeply internal to the database engine inaccessible by any other means, so that they can perform faster. Therefore, by exporting the basic A* star and network expansion modules, the performance of the continuous path and range searches increases.

CONCLUSIONS AND FUTURE WORK

A successful integration of indoor knowledge representation and mobile information systems requires several developments at the conceptual and design levels. At the conceptual level, our study shows the interest and advantage of a hierarchical spatial data model to represent an indoor environment. This model should support continuous processing of location-dependent queries applied to moving objects acting in indoor spaces. This is exactly the scope of the approach that has been developed by this research, and which can be considered at the intersection of *Mobile and ubiquitous computing* and *location-based services*. The research developed provides an approach for knowledge representation and management in indoor mobile environments. The design of a platform for context-aware indoor navigation that encompasses an indoor data model and a query language. Several algorithms for continuous location-dependent query processing have been developed and implemented. This chapter first presents a summary of the contributions, and then discusses some of the perspectives opened by this work, as well as possible directions to explore.

Summary of the contributions

The integration of an indoor spatial data model into a context-aware system is considered as a key building block for designing advanced navigation services. Our study shows that a hierarchically organized context-dependent indoor data model can support a wide range of location-based services. An indoor data model has been introduced that can be viewed as a tree structure in which location information is represented at different levels of abstraction (Chapter 4): (1) a fine-grained graph embedded within an occupancy grid; (2) an exit hierarchy; and (3) a location hierarchy. Such a model represents: (i) static/moving features of interest, (ii) their spatial properties, and (iii) the behaviours or actions that emerge from them. The hierarchical design provides better performance and scalability when processing location-dependent queries. Time-dependent functions that compute navigational path distances and travel times are introduced. A classification of user profiles supports

an offline filtering of the hierarchical data model, thus reducing the amount of data that needs to be processed in real-time.

The semantics of a query grammar tied to the indoor data model has also been developed (Chapter 5). This grammar supports location-dependent queries. It represents navigation queries and incorporates user preferences and semantics in the query model (e.g., *gr* operator, *MINIMIZE* and *Stop-Vertices* keywords). It also supports the hierarchical data model thanks to the concept of location granules that integrate different levels of abstraction. Location granules favour the specification of queries using the location terminology preferred by the user (e.g., nodes at the fine-grained level, rooms, floors, buildings, etc.).

Several algorithms and a generic architecture for the continuous processing of location-dependent queries have been designed and implemented (Chapter 6). Navigation-related queries can be processed according to this architecture, and query results are continually kept up-to-date over a certain period of time. Two algorithms for hierarchical path searches and range queries applied to both static and moving objects have been developed. The former represents an incremental and hierarchical path search that can be executed at different levels of granularity, and applied on static and/or mobile data. The latter performs continuous range searches by applying a hierarchical network expansion mechanism and an incremental Euclidean restriction approach. Those algorithms take advantage of the hierarchical data model of the indoor environment, and employ an incremental approach in order to efficiently execute continuous location-dependent queries, thus avoiding the reevaluation of each search problem independently from scratch.

The whole approach has been implemented as a database solution based on the PostgreSQL DBMS (Chapter 7). The main parts of the prototype developed are: (i) a hierarchical network-based data model of an indoor environment; (ii) operators and location-dependent constraints introduced in the query grammar; and (iii) algorithms to process continuous location-dependent queries over moving objects. Several experimental validations have been conducted to investigate scalability and performance with respect to the intrinsic properties of the proposed solutions. Results show that our proposal achieves satisfactory performance, and it is sufficiently efficient to be used in a real-time scenario.

Research perspectives

This dissertation raises a number of research challenges related to context-aware and continuous query processing, as well as to semantic location-based services. This section gives an overview of meaningful directions for future research. It outlines three categories of perspectives: *conceptual*,

technical, as well as *application perspectives*. On the one hand, *conceptual perspectives* discuss the concept of an *extended context model*, and present several open challenges for *context-aware queries* related to navigation services. The concept of *semantically annotating users' trajectories* is also presented. On the other hand, *technical perspectives* include optimization at the algorithmic and system levels so that a more efficient and extended system can be designed. In addition, implementation issues such as a close integration of the solutions developed in this dissertation with an indoor localization technique, a graphical user interface, and other extensions are discussed. Finally, *application perspectives* that discuss the future of indoor-based systems from academic and industrial points of view are highlighted.

Conceptual perspectives

Different extensions to the modelling approach can be explored in order to enrich the context dimension of the data model. For instance, a better integration of context-awareness in query processing as well as the development of other kinds of navigation-related services are still directions to investigate. These issues are discussed as follows.

Extended context model: The indoor data model introduced in Chapter 4 represents several contextual dimensions such as spatio-temporal information, user profiles, and real-time event management. The data model developed still requires the integration of an extended context model in order to incorporate users' activities as well as content generated by other social entities when executing location-dependent queries. Such contextual dimensions can be gathered via different ways: using *physical* as well as *virtual sensors*. Physical sensors embedded in mobile devices provide diverse kinds of raw data, and other virtual sensors (agenda, profile, social network activity, etc.) can also help the system to track the user context. A filtering step should be performed in order to only keep relevant data for analysis and real-time processing. Such a model classifies contextual dimensions as follows.

- *Spatio-temporal information* includes the location and time dimensions as represented in the indoor spatial data model previously developed. Besides, a user trajectory might also be of interest to enrich the user context. Based on such spatio-temporal patterns, such a system can infer some user interests and provide him/her with relevant recommendations. Moreover, the user orientation can be associated to the user device, and is represented by either a quantitative value (128° to the north) or using a qualitative representation [Patroumpas and Sellis, 2009]. The orientation parameter can be used along with other sensor data in order to detect some basic user activities as shown afterwards.

- The *user activity* is an important context dimension that depicts the activity performed by the user when navigating in an indoor space. By focusing on such activities, the system will gain a better understanding of the context, thus providing a step forward to develop truly ambient intelligent systems. For instance, a mechanism for user activity recognition based on accelerometer data can be used for that purpose [Yan et al., 2012]. Such a system can identify different activities such as *running, walking, going upstairs, going downstairs, sitting, standing, in an elevator, etc.*, and consider them while reevaluating the answer to a given query.

- The *user profile* dimension includes gender, age, a set of interests, physical capabilities, access privileges, and status. The user profile contains static (gender, age, profession) as well as dynamic properties (physical capabilities, access privileges, status: busy, available, etc.). An overview of the sensors embedded in mobile phones and their potential uses as a computing platform to acquire those parameters is reviewed in [Lane et al., 2010]. Those properties are very important for providing personalized answers to a given user's request. Dynamic properties can be extended to incorporate information shared by the corresponding user to the system, which in turn can broadcast it to other users when necessary. This concept of user-generated content might be of interest to recommender systems that can relay useful information to users based on shared interests.

- An effective model of *collaborative usage* can also be useful. A collaborative usage model classifies users into groups or communities, within which they share a common characteristic or interest. A user can belong to a community either explicitly by a subscription-based mechanism or implicitly if he/she has a common characteristic or property [Ben Nejma et al., 2013]. On the one hand, *subscription-based communities* encompass users that share a given interest or involved in a similar practice (i.e., group of people working together and constantly conducting solutions in a given shared field). On the other hand, *implicitly generated communities* depend either on the current location of users or on their trajectories (i.e., if they have similar spatio-temporal patterns). They might also depend on their behaviours in a similar context. This collaborative usage model derives similar behaviours and interests, so that the system can act proactively by notifying users of certain events or adapting certain query answers. Push-based services can thereafter be designed based on such communities.

Based on data gathered from fixed and/or mobile sensors and on user-generated content (i.e., information shared by users to the system or within communities to notify certain events), the system should be able to manage events in real-time and filter relevant ones for corresponding users. Real-time event management means that people who share a common interest or common context should be notified by such events. Depending on the relevance of such events, the system might either let the user decide whether to integrate this event into his/her relevant queries, or simply ignore it.

Some types of events should be directly handled by the system itself (in case of emergency or other important events). These contextual dimensions represent the user-centric context, and should be taken into account by the system in real-time in order to completely incorporate context-awareness in intelligent navigation services.

Context-aware Query Processing: This dissertation introduced a continuous query processing framework that takes several contextual dimensions into account. An extension of our proposal should be designed to support the extended context model previously described. Such a system should deal with a highly dynamic environment in which objects are moving continuously and their context is dynamically changing. One important direction to consider in future work is the complete integration of the context model into the query processing engine. This implies the development of a *"cost model for evaluating context-aware queries"* that provides not only a *quantitative value* as a result of a cost function, but also a semantic indicator that evaluates the quality, usefulness, and uncertainty of query results. The main challenge in developing cost models is to take into account the different parameters of the context while associating different quantitative scores and semantic indications to each parameter according to the corresponding user. This means that, for instance, a user with special needs has different constraints with different impedances than a user with complete physical capabilities. Taking all these factors when designing a cost-model for context-aware queries is challenging. The main goal is to provide a context-aware query processor that avoids modifying the database engine with the addition of each new context parameter. Instead, an extensible query engine that is general enough to support any kind of context should be designed.

Regarding the representation of context-aware queries, this dissertation introduced several spatio-temporal operators that can be combined with conventional query operators in order to execute standard as well as navigation-related queries in a straightforward manner. To fully integrate the proposed context model, additional *"preference- and context-aware query operators"* are still to be developed in order to take into account explicit preferences and other users' constraints and wishes. This helps to compute cost functions for the continuous processing of location-dependent queries.

Semantic annotation of heterogeneous trajectories: As another possible direction to further research, semantic location-based services based on users' trajectories might be of interest to pave the way for the development and usage of advanced intelligent services [Yan et al., 2011b,a]. Not only current movements will be taken into account, but also the trajectories of the corresponding users within a given period of time can be stored and annotated by semantic information, so that different kinds of services may be provided. Objects of interest, which can be classified as *repellent,*

attractive or neutral, as well as events should be considered. Users are given the opportunity to annotate points of interest (POI) with several meanings to describe those landmarks [Mata et al., 2011]. The system can then classify those POIs as either attractive, repellent or neutral to other users depending on their context. Relevant users might share a common interest or be following a similar pattern (e.g., shops recommendations, etc.). Therefore, semantic annotations of positions, trajectories, and typical movement patterns of moving objects open new research directions and allow for additional user-oriented services.

Technical perspectives

From a technical point of view, there are a number of possibilities for extending the overall system and enhancing its individual components. This section first discusses optimization of different components, and then presents the main directions and ideas about potential extensions.

Continuous query optimization: The algorithms developed minimize the CPU cost at the central processing server by introducing an incremental approach for continuous query processing. Other techniques can reduce the communication overhead caused by frequent location updates. Those assume that objects have some computational capabilities and knowledge of the queries so that they can perform location updates only when they influence some query results. A valuable direction for future work is to combine these approaches with ours and to design a comprehensive system that minimizes both CPU and communication costs. This implies to introduce a policy based on lazy updates to reduce extra computations. A more general approach should be designed, such that only relevant users (i.e., those involved in active queries) update their locations, and only queries that might be affected by those updates are reevaluated.

Another important goal is to investigate further component integration such as supporting a Data Stream Management System (DSMS) for the processing of continuous queries over spatial data streams. Continuous queries are naturally processed over streams of data as they arrive in real-time. So far, our algorithms do not handle data streams from an external system, but we simulate this scenario by creating a named pipe and a program that writes the stream elements as a single line into that pipe. Coupling our solutions developed in PostgreSQL with the TelegraphCQ extension [Chandrasekaran et al., 2003] would provide a more consistent system for continuous query processing over moving objects.

Other extensions: The prototype provided in this work focusses on back-end development (i.e., at the server side) of the proposed solutions. However, a mobile real-time information system

requires cooperation of different components at the client side, and other extensions such as an indoor positioning system that provides moving object data.

Client-side personalisation of mobile context-aware services has important benefits when building professional and user-oriented systems. Consequently, different client-side extensions are expected to be integrated in such mobile applications. Particularly, a high-level query interface is expected to embed a voice recognition system that allows verbal requests and a graphical user interface with predefined operators. The system transforms the query expressed in a natural or high-level language into an SQL-like format, as proposed in [Ilarri et al., 2006a]. Moreover, a user modelling framework that can support client-side personalisation on different mobile platforms should be designed. This implies taking into account the different preferences and constraints (e.g., physical capabilities) of the mobile user.

Our prototype should also be coupled with an indoor localization technique, such as the one developed in [Ray et al., 2010], so that an evaluation over real moving object data can be performed. In addition, An automatic generation of the fine-grained graph of the indoor data model should be introduced, along with a map-matching mechanism adapted to the corresponding localization technique should be developed.

Application perspectives

The modelling approach developed in this research is sufficiently flexible to support the development of different levels of data manipulation and interactions. This approach can be applied for the monitoring of different types of built environments (e.g., airports, museums) and ships in terrestrial and maritime contexts, respectively. For instance, an indoor navigation system is currently being developed at the French Naval Academy Research Institute for the real-time monitoring of moving objects within ships. Additional properties to the indoor data model might be considered in such applications in order to take into account the movement of ships as well as an approximation of seafloor slope at each location within a ship. Regarding the design of the hierarchical graph-based data model, impedances on edges at the fine-grained graph should be added by considering the slope value at this location and other parameters such as the current speed and direction of the ship. Such adaptations on the data model provide more realistic navigation services (e.g., get-together services or prepare the crew and passengers for an evacuation process) and adapted query answers.

Application perspectives are very large, from the development of interactive systems for built environments, to additional professional- and user-oriented services. The numerous application types range from mobile location-aware services to context-aware recommender systems. New classes of

applications, which can take advantage of both low-level sensor data and high-level events, context, and activities inferred from mobile phone sensor data, are being explored not only in academic and industrial research laboratories, but also within large corporations such as Google and Apple as well as start-up companies.

Not only are the application perspectives promising, but also in terms of business as the range of possibilities will surely open many opportunities. In a recent report published by ABI Research[3] on *"Indoor Location Smartphone Applications"*, it has been noted that the indoor location market will break 1 Billion application downloads by 2016. It has been also emphasized that the future adoption of a variety of indoor location technologies will be considered across a range of different application categories, such as retail, navigation, environmental monitoring, location-aware social networking, health and well being, personal tracking, while also enhancing services such as advertising, ambient intelligence, augmented reality, and local search.

[3]http://www.abiresearch.com

List of Publications

Peer-reviewed Journal Articles

Imad Afyouni, Cyril Ray, Sergio Ilarri, Christophe Claramunt, A PostgreSQL Extension for Continuous Path and Range Queries in Indoor Mobile Environments, *Journal of Pervasive and Mobile Computing (PMC), Special Issue on Information Management in Mobile Applications*, 25 pp., 2013, Elsevier, accepted.

Imad Afyouni, Sergio Ilarri, Cyril Ray, Christophe Claramunt, Context-Aware Processing of Continuous Location- Dependent Queries in Indoor Environments, *Journal of Ambient Intelligence and Smart Environments (JAISE), Special Issue on Context-Aware Systems*, 5(1):65-88, IOS Press, 2013, doi:10.3233/ AIS-120186.

Imad Afyouni, Cyril Ray, Christophe Claramunt, Spatial Models for Indoor and Context-Aware Navigation Systems: A Survey. *Journal of Spatial Information Science, (JOSIS)*, 4(1), pages 85-123, 2012, ISSN 1948-660X (online).

International Conferences

Imad Afyouni, Cyril Ray, Sergio Ilarri, Christophe Claramunt, Algorithms for Continuous Location-Dependent and Context-Aware Queries in Indoor Environments, *Proceedings of the 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* pages 329-338, Redondo Beach, CA, USA, November 2012.

Imad Afyouni, Cyril Ray, Christophe Claramunt, A Fine-grained Context-dependent Model for Indoor Spaces, In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness (ISA 2010)*, pages 33- 38, San Jose, CA, USA, November 2010.

National Conferences

Imad Afyouni, Cyril Ray, Sergio Ilarri, Christophe Claramunt, Un Modèle de Données pour les Requêtes Sensibles au Contexte dans les Environnements Indoor, *Actes des Huitièmes Journées Francophones Mobilité et Ubiquité (UBIMOB)*, Cépaduès, pages 23-33, 2012.

Communications

Imad Afyouni, Cyril Ray, Christophe Claramunt, Knowledge Representation and Management in Indoor Mobile Environments, *ERCIM News No. 93, Special Theme: Mobile Computing*, 2013.

Imad Afyouni, Context-Aware Processing of Continuous Location-Dependent Queries in Indoor Environments, *COST MOVE Workshop*, Delft - Netherlands - STSM Session, 2 pages, March 2012.

Research Reports

Imad Afyouni, Cyril Ray, Christophe Claramunt, Spatial Models for Indoor Context-Aware Navigation Systems, IRENav research report, 37 pages, June, 2011.

Bibliography

Aalto, L., Gothlin, N., Korhonen, J., and Ojala, T. (2004). Bluetooth and WAP push based location-aware mobile advertising system. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, pages 49–58. ACM.

Aarts, E. and Wichert, R. (2009). Ambient intelligence. *Technology Guide*, pages 244–249.

Abowd, G. and Mynatt, E. (2000). Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction*, 7(1):29–58.

Afyouni, I., Ilarri, S., Ray, C., and Claramunt, C. (2013). Context-aware modelling of continuous location-dependent queries in indoor environments. *Journal of Ambient Intelligence and Smart Environments*, 5(1):65–88.

Afyouni, I., Ray, C., and Claramunt, C. (2012). Spatial models for indoor and context-aware navigation systems: A survey. *Journal of Spatial Information Science*, 4(1):85–123.

Aitenbichler, E. (2008). Event-based and publish/subscribe communication. In *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, pages 152–171. Information Science Publishing.

AL-Khalidi, H., Abbas, Z., and Safar, M. (2013). Approximate range query processing in spatial network databases. *Multimedia Systems*, 19(2):151–161.

Alamri, S., Taniar, D., Safar, M., and Al-Khalidi, H. (2013). Spatiotemporal indexing for moving objects in an indoor cellular space. *Neurocomputing*, 122:70–78.

Albinali, F. (2008). *Activity-aware computing: Modeling of human activity and behavior*. PhD thesis, The University of Arizona - Department of Computer Science.

Albinali, F. and Gniady, C. (2006). CPM: Context-aware power management in WLANs. In *Proceedings of the 18th Conference on Innovative Applications of Artificial Intelligence*, volume 2, pages 1745–1750. AAAI Press.

Angles, R. and Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys*, 40(1):1–39.

Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., and Widom, J. (2004a). Stream: The stanford data stream management system. Technical Report 2004-20, Stanford InfoLab.

Arasu, A., Babu, S., and Widom, J. (2004b). CQL: A language for continuous queries over streams and relations. In *Database Programming Languages*, pages 1–19. Springer.

Arasu, A., Babu, S., and Widom, J. (2006). The CQL continuous query language: Semantic foundations and query execution. *The International Journal on Very Large Data Bases*, 15(2):121–142.

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

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–16. ACM.

Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277.

Baldoni, R. and Virgillito, A. (2005). Distributed event routing in publish/subscribe communication systems: A survey. Technical report, Dipartimento di Informatica e Sistemistica, Universita di Roma "La Sapienza", TR-1/06.

Bao, J., Chow, C., Mokbel, M., and Ku, W. (2010). Efficient evaluation of k-range nearest neighbor queries in road networks. In *Proceedings of the 11th International Conference on Mobile Data Management (MDM)*, pages 115–124. IEEE.

Bardram, J. (2005). Activity-based computing: Support for mobility and collaboration in ubiquitous computing. *Personal and Ubiquitous Computing*, 9(5):312–322.

Basagni, S., Chlamtac, I., and Syrotiuk, V. (1999). Geographic messaging in wireless ad hoc networks. In *Proceedings of the 49th Vehicular Technology Conference (VTC)*, volume 3, pages 1957–1961. IEEE.

Batty, M., Xie, Y., and Sun, Z. (1999). Modeling urban dynamics through GIS-based cellular automata. *Computers, Environment and Urban Systems (CEUS)*, 23(3):205–233.

Becker, C. and Durr, F. (2005). On location models for ubiquitous computing. *Personal and Ubiquitous Computing*, 9(1):20–31.

Becker, T., Nagel, C., and Kolbe, T. (2009). Supporting contexts for indoor navigation using a multilayered space model. In *Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 680–685. IEEE.

Becker, T., Nagel, C., and Kolbe, T. (2010). Discussion of euclidean space and cellular space and proposal of an integrated indoor spatial data model. Technical report, Institute of Geodesy and Geoinformation Science.

Beckmann, N. (1990). The R*-tree: An efficient and robust access method for points and rectangles. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 322–331.

Beeson, P., Jong, N., and Kuipers, B. (2005). Towards autonomous topological place detection using the extended voronoi graph. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4373–4379. IEEE.

Beigl, M., Zimmer, T., and Decker, C. (2002). A location model for communicating and processing of context. *Personal and Ubiquitous Computing*, 6(5-6):341–357.

Ben Nejma, G., Roose, P., Dalmau, M., and Gensel, J. (2013). TALDEA: Un outil d'aide à la création de communautés spontanées avec géolocalisation. *Ingénierie des Systèmes d'Information*, 18(1):39–58.

Benyelloul, A., Jouanot, F., and Rousset, M. (2010). Conquer: An rdfs-based model for context querying. 6èmes Journées Ateliers UbiMob 2010.

Berchtold, S., Keim, D. A., and Kriegel, H. (1996). The X-tree: An index structure for high-dimensional data. In *Proceedings of the 22th International Conference on Very Large Data Bases*, pages 28–39. Morgan Kaufmann Publishers Inc.

Berger, A., Grimmer, M., and Müller-Hannemann, M. (2010). Fully dynamic speed-up techniques for multi-criteria shortest path searches in time-dependent networks. In *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA)*, pages 35–46. Springer.

Bhatt, M., Dylla, F., and Hois, J. (2009). Spatio-terminological inference for the design of ambient environments. In *Proceedings of the 9th International Conference on Spatial Information Theory (COSIT)*, pages 371–391. Springer.

Bielli, M., Boulmakoul, A., and Mouncif, H. (2006). Object modeling and path computation for multimodal travel systems. *European Journal of Operational Research*, 175(3):1705–1730.

Bonacich, P. (1987). Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182.

Booth, J., Sistla, P., Wolfson, O., and Cruz, I. (2009). A data model for trip planning in multimodal transportation systems. In *Proceedings of the 12th International Conference on Extending Database Technology (EDBT)*, pages 994–1005. ACM.

Booth, J. A. (2011). *Modeling and Querying Multimodal Urban Transportation Networks*. PhD thesis, University of Illinois, Chicago, USA.

Bosc, P., Pivert, O., and Mokhtari, A. (2009). On fuzzy queries with contextual predicates. In *Proceedings of the 18th International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 484–489.

Botea, A., Muller, M., and Schaeffer, J. (2004). Near optimal hierarchical path-finding. *Journal of Game Development*, 1(1):7–28.

Brinkhoff, T. (2002). A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180.

Burak, A. and Sharon, T. (2004). Usage patterns of friendzone: Mobile location-based community services. In *Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia*, pages 93–100. ACM.

Buschka, P. (2005). *An investigation of hybrid maps for mobile robots*. PhD thesis, Örebro Universitetsbibliotek, Sweden.

Cagigas, D. (2005). Hierarchical D* algorithm with materialization of costs for robot path planning. *Robotics and Autonomous Systems*, 52(2-3):190–208.

Cagigas, D. and Abascal, J. (2004). Hierarchical path search with partial materialization of costs for a smart wheelchair. *Journal of Intelligent and Robotic Systems*, 39(4):409–431.

Cai, Y., Hua, K., and Cao, G. (2004). Processing range-monitoring queries on heterogeneous mobile objects. In *Proceedings of the 5th IEEE International Conference on Mobile Data Management (MDM 2004)*, pages 27–38. IEEE.

Cai, Y., Hua, K., Cao, G., and Xu, T. (2006). Real-time processing of range-monitoring queries in heterogeneous mobile databases. *Mobile Computing, IEEE Transactions on*, 5(7):931–942.

Cao, H., Wang, S., and Li, L. (2003). Location dependent query in a mobile environment. *Information Sciences*, 154(1Ũ2):71–83.

Champandard, A. (2003). *AI game development: Synthetic creatures with learning and reactive behaviors*. New Riders Pub.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M., Hellerstein, J., Hong, W., Krishnamurthy, S., Madden, S., Reiss, F., and Shah, M. (2003). TelegraphCQ: Continuous dataflow processing. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 668–668. ACM.

Chatila, R. and Laumond, J. (1985). Position referencing and consistent world modeling for mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 138–145. IEEE.

Cheema, M., Brankovic, L., Lin, X., Zhang, W., and Wang, W. (2011). Continuous monitoring of distance-based range queries. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1182–1199.

Cheema, M., Zhang, W., Lin, X., Zhang, Y., and Li, X. (2012). Continuous reverse k nearest neighbors queries in euclidean space and in spatial networks. *The International Journal on Very Large Data Bases*, 21(1):69–95.

Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College Hanover, NH, USA.

Chen, J., DeWitt, D., Tian, F., and Wang, Y. (2000). NiagaraCQ: A scalable continuous query system for internet databases. In *ACM SIGMOD Record*, volume 29, pages 379–390. ACM.

Chen, S., Jensen, C. S., and Lin, D. (2008). A benchmark for evaluating moving object indexes. *Proceedings of the VLDB Endowment*, 1(2):1574–1585.

Cheng, R., Xia, Y., Prabhakar, S., and Shah, R. (2005). Change tolerant indexing for constantly evolving data. In *Proceedings. 21st International Conference on Data Engineering*, pages 391–402. IEEE.

Chon, H., Agrawal, D., and Abbadi, A. (2003). Fates: Finding a time dependent shortest path. In *Proceedings of the 4th International Conference on Mobile Data Management*, pages 165–180. Springer-Verlag.

Choset, H. (1997). Incremental construction of the Generalized Voronoi Diagram, the Generalized Voronoi Graph, and the Hierarchical Generalized Voronoi Graph. In *Proceedings of the 1st CGC Workshop on Computational Geometry*.

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

Chow, C., Mokbel, M., and Aref, W. (2009). Casper*: Query processing for location services without compromising privacy. *ACM Transactions on Database Systems (TODS)*, 34(4):24.

Christensen, H. and Bardram, J. (2002). Supporting human activities exploring activity-centered computing. In *Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp)*, pages 107–116. Springer-Verlag.

Cook, D., Augusto, J., and Jakkula, V. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298.

Crowley, J. (1989). World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 674–680. Computer Society Press.

De Almeida, V. and Güting, R. (2005). Indexing the trajectories of moving objects in networks. *GeoInformatica*, 9(1):33–60.

De Almeida, V. and Güting, R. (2006). Using dijkstra's algorithm to incrementally find the k-nearest neighbors in spatial network databases. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 58–62. ACM.

De Almeida, V., Güting, R., and Behr, T. (2006). Querying moving objects in secondo. In *Proceedings of the 7th International Conference on Mobile Data Management (MDM)*, pages 47–51. IEEE.

De Berg, M., Cheong, O., Van Kreveld, M., and Overmars, M. (2008). *Computational geometry: Algorithms and applications*. Springer-Verlag.

Dean, J. and Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Delling, D., Sanders, P., Schultes, D., and Wagner, D. (2009). Engineering route planning algorithms. In *Algorithmics of Large and Complex Networks*, Lecture Notes in Computer Science, pages 117–139. Springer.

Delot, T., Ilarri, S., Cenerario, N., and Hien, T. (2011). Event sharing in vehicular networks using geographic vectors and maps. *Mobile Information Systems*, 7(1):21–44.

Demiryurek, U., Banaei-Kashani, F., and Shahabi, C. (2009). Efficient continuous nearest neighbor query in spatial networks using Euclidean restriction. In *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*, pages 25–43. Springer-Verlag.

Demiryurek, U., Banaei-Kashani, F., and Shahabi, C. (2010). Efficient k-nearest neighbor search in time-dependent spatial networks. In *Proceedings of the 21st International Conference on Database and Expert Systems Applications (DEXA)*, pages 432–449. Springer.

Demyen, D. and Buro, M. (2006). Efficient triangulation-based pathfinding. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 942–947.

Deng, K., Zhou, X., Shen, H., Sadiq, S., and Li, X. (2009). Instance optimal query processing in spatial networks. *The VLDB Journal*, 18(3):675–693.

Deshpande, A., Ives, Z., and Raman, V. (2007). Adaptive query processing. *Foundations and Trends in Databases*, 1(1):1–140.

Dey, A. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7.

Dey, A. and Abowd, G. (1999). Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC)*. Springer-Verlag.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1:269–271.

Ding, B., Yu, J., and Qin, L. (2008). Finding time-dependent shortest paths over large graphs. In *Proceedings of the 11th International Conference on Extending Database Technology (EDBT)*, pages 205–216. ACM.

Disser, Y., Müller-Hannemann, M., and Schnee, M. (2008). Multi-criteria shortest paths in time-dependent train networks. In *Experimental Algorithms*, Lecture Notes in Computer Science, pages 347–361. Springer.

Dourish, P. (2004). *Where the action is: The foundations of embodied interaction*. The MIT Press.

Dudas, P., Ghafourian, M., and Karimi, H. (2009). ONALIN: Ontology and algorithm for indoor routing. In *Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services, and Middleware*, pages 720–725. IEEE Computer Society.

D'Ulizia, A., Ferri, F., and Grifoni, P. (2012). Moving GeoPQL: A pictorial language towards spatio-temporal queries. *GeoInformatica*, 16:357–389.

Dunham, M. and Kumar, V. (1998). Location dependent data and its management in mobile databases. In *Proceedings of the 9th International Workshop on Database and Expert Systems Applications*, pages 414–419. IEEE.

Durr, F. and Rothermel, K. (2003). On a location model for fine-grained geocast. In *Proceedings of the 5th International Conference on Ubiquitous Computing, UbiComp*, pages 18–35. Springer.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57.

Elmongui, H., Mokbel, M., and Aref, W. (2013). Continuous aggregate nearest neighbor queries. *GeoInformatica*, 17:63–95.

Eppstein, D., Goodrich, M., and Sun, J. (2005). The skip quadtree: A simple dynamic data structure for multidimensional data. In *Proceedings of the 21st Annual Symposium on Computational Geometry*, pages 296–305. ACM.

Erwig, M., Güting, R., Schneider, M., Vazirgiannis, M., et al. (1999). Spatio-temporal data types: An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3):269–296.

Erwig, M. and Schneider, M. (1999). Developments in spatio-temporal query languages. In *Proceedings of the 10th International Workshop on Database and Expert Systems Applications (DEXA)*, pages 441–449. IEEE.

Evennou, F. and Marx, F. (2006). Advanced integration of WiFi and inertial navigation systems for indoor mobile positioning. *Eurasip Journal on Applied Signal Processing*, (17):1–11.

Ferhatosmanoglu, H., Stanoi, I., Agrawal, D., and El Abbadi, A. (2001). Constrained nearest neighbor queries. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, pages 257–278. Springer.

Fernandez, J. and Gonzalez, J. (1998). Hierarchical graph search for mobile robot path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 656–661. IEEE.

Fernandez, J. and Gonzalez, J. (2002). *Multi-hierarchical representation of large-scale space: Applications to mobile robots*. Kluwer Academic Publishers Norwell, MA, USA.

Fernández-Madrigal, J., Galindo, C., and González, J. (2004). Assistive navigation of a robotic wheelchair using a multihierarchical model of the environment. *Integrated Computer-Aided Engineering*, 11(4):309–322.

Ferri, F. and Rafanelli, M. (2005). Geopql: A geographical pictorial query language that resolves ambiguities in query interpretation. In *Journal on Data Semantics III*, pages 50–80. Springer.

Filho, J., Miron, A., Satoh, I., Gensel, J., Martin, H., et al. (2010). Modeling and measuring quality of context information in pervasive environments. In *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 690–697. IEEE.

Franz, G., Mallot, H., Wiener, J., and Neurowissenschaft, K. (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, InterSymp*, pages 30–38.

Frentzos, E., Gratsias, K., Pelekis, N., and Theodoridis, Y. (2007). Algorithms for nearest neighbor search on moving object trajectories. *Geoinformatica*, 11(2):159–193.

Friedman, J. (2004). System, apparatus and method for location-based instant messaging. US Patent 6,714,791.

Froese, T., Fischer, M., Grobler, F., Ritzenthaler, J., Yu, K., Sutherland, S., Staub, S., Akinci, B., Akbas, R., Koo, B., et al. (1999). Industry foundation classes for project management - A trial implementation. *Electronic Journal of Information Technology in Construction (ITCon)*, 4:17–36.

Fu, Q. and Retscher, G. (2009). Using RFID and INS for indoor positioning. In *Location Based Services and TeleCartography II*, pages 421–438. Springer.

Galindo, C., Saffiotti, A., Coradeschi, S., Buschka, P., Fernandez-Madrigal, J., and Gonzalez, J. (2005). Multi-hierarchical semantic maps for mobile robotics. In *Proceedings of the International Conference on Intelligent Robots and Systems, IROS*, pages 2278–2283. IEEE.

Gao, Y., Zheng, B., Chen, G., Li, Q., and Guo, X. (2011). Continuous visible nearest neighbor query processing in spatial databases. *The VLDB Journal*, 20(3):371–396.

Gedik, B. and Liu, L. (2004). Mobieyes: Distributed processing of continuously moving queries on moving objects in a mobile system. In Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., and Ferrari, E., editors, *Advances in Database Technology - EDBT 2004*, Lecture Notes in Computer Science, pages 67–87. Springer.

Gedik, B. and Liu, L. (2006). MobiEyes: A distributed location monitoring service using moving location queries. *IEEE Transactions on Mobile Computing*, 5(10):1384–1402.

Geisberger, R., Sanders, P., Schultes, D., and Delling, D. (2008). Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Proceedings of the 7th international conference on Experimental algorithms*, pages 319–333. Springer-Verlag.

Ghanem, T., Hammad, M., Mokbel, M., Aref, W., and Elmagarmid, A. (2007). Incremental evaluation of sliding-window queries over data streams. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):57–72.

Golab, L. and Özsu, M. (2003). Issues in data stream management. *ACM Sigmod Record*, 32(2):5–14.

Gratsias, K., Frentzos, E., Delis, V., and Theodoridis, Y. (2005). Towards a taxonomy of location based services. In *Proceedings of the 5th International Workshop on Web and Wireless Geographical Information Systems (W2GIS)*, pages 19–30. Springer.

Grütter, R., Scharrenbach, T., and Bauer-Messmer, B. (2008). Improving an RCC-derived geospatial approximation by OWL axioms. In *Proceedings of the 7th International Conference on The Semantic Web (ISW)*, pages 293–306. Springer.

Gu, Y., Lo, A., and Niemegeers, I. (2009). A survey of indoor positioning systems for wireless personal networks. *IEEE Communications Surveys & Tutorials*, 11(1):13–32.

Guerra-Filho, G. and Samet, H. (2006). An optimal iterative algorithm for shortest path query. Technical report, Department of Computer Science, University of Maryland.

Güting, R. (1994). Graphdb: Modeling and querying graphs in databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 297–308. Morgan Kaufmann.

Güting, R., Almeida, V., Ansorge, D., Behr, T., Ding, Z., Hose, T., Hoffmann, F., Spiekermann, M., and Telle, U. (2005). Secondo: An extensible dbms platform for research prototyping and teaching. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, pages 1115–1116. IEEE.

Güting, R., Behr, T., and Xu, J. (2010). Efficient k-nearest neighbor search on moving object trajectories. *The International Journal on Very Large Data Bases (VLDB)*, 19(5):687–714.

Guting, R., Bohlen, M., Erwig, M., Jensen, C., Lorentzos, N., Schneider, M., and Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. *ACM Transactions on Database Systems (TODS)*, 25(1):1–42.

Güting, R., Braese, A., Behr, T., and Xu, J. (2009). Nearest neighbor search on moving object trajectories in Secondo. In *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, pages 427–431. Springer.

Güting, R., De Almeida, V., and Ding, Z. (2006). Modeling and querying moving objects in networks. *The VLDB Journal*, 15(2):165–190.

Güting, R. and Schneider, M. (2005). *Moving objects databases*. Morgan Kaufmann Pub.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 18, pages 47–57.

Hadjali, A., Mokhtari, A., and Pivert, O. (2010). A fuzzy-rule-based approach to contextual preference queries. In *Proceedings of the Computational Intelligence for Knowledge-based Systems Design, and 13th International Conference on Information Processing and Management of Uncertainty*, pages 532–541. Springer-Verlag.

Hadjali, A., Mokhtari, A., and Pivert, O. (2012). Expressing and processing complex preferences in route planning queries: Towards a fuzzy-set-based approach. *Fuzzy Sets and Systems*, 196:82–104.

Haghighi, P., Zaslavsky, A., and Krishnaswamy, S. (2006). An evaluation of query languages for context-aware computing. In *Proceedings of 17th International Workshop on Database and Expert Systems Applications (DEXA)*, pages 455–462. IEEE.

Hakkila, J. and Hexel, R. (2003). Interaction in location-aware messaging in a city environment. In *Proceedings of the Australasian Computer Human Interaction Conference (OZCHI)*, pages 84–93.

Hammad, M., Franklin, M., Aref, W., and Elmagarmid, A. (2003). Scheduling for shared window joins over data streams. In *Proceedings of the 29th International Conference on Very Large Databases - Volume 29*, pages 297–308. VLDB Endowment.

Hammad, M., Mokbel, M., Ali, M., Aref, W., Catlin, A., Elmagarmid, A., Eltabakh, M., Elfeky, M., Ghanem, T., Gwadera, R., et al. (2004). Nile: A query processing engine for data streams. In *Proceedings of the 20th International Conference on Data Engineering*, page 851. IEEE.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Hu, H. and Lee, D. (2004). Semantic location modeling for location navigation in mobile environment. In *Proceeding of the IEEE International Conference on Mobile Data Management (MDM)*, pages 52–61. IEEE.

Hu, H., Xu, J., and Lee, D. L. (2005). A generic framework for monitoring continuous spatial queries over moving objects. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 479–490. ACM.

Huang, H. and Gartner, G. (2010). A survey of mobile indoor navigation systems. In *Proceedings of the 1st ICA Symposium on Cartography in Central and Eastern Europe*, pages 305–319. Springer.

Ilarri, S., Bobed, C., and Mena, E. (2011). An approach to process continuous location-dependent queries on moving objects with support for location granules. *Journal of Systems and Software*, 84(8):1327–1350.

Ilarri, S., Mena, E., and Illarramendi, A. (2006a). Location-dependent queries in mobile contexts: Distributed processing using mobile agents. *IEEE Transactions on Mobile Computing*, 5(8):1029–1043.

Ilarri, S., Mena, E., and Illarramendi, A. (2010). Location-dependent query processing: Where we are and where we are heading. *ACM Computing Surveys*, 42(3):1–73.

Ilarri, S., Trillo, R., and Mena, E. (2006b). Springs: A scalable platform for highly mobile agents in distributed computing environments. In *In Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM*, pages 633–637. IEEE.

IndoorAtlas, L. (2012). Ambient magnetic field-based indoor location technology: Bringing the compass to the next level. IndoorAtlas Ltd.

Iwerks, G. (2004). *Maintenance of spatial queries on continuously moving points.* PhD thesis, University of Maryland at College Park College Park, MD, USA.

Jain, N., Mishra, S., Srinivasan, A., Gehrke, J., Widom, J., Balakrishnan, H., Çetintemel, U., Cherniack, M., Tibbetts, R., and Zdonik, S. (2008). Towards a streaming sql standard. *Proceedings of the VLDB Endowment*, 1(2):1379–1390.

Jayaputera, J. and Taniar, D. (2005a). Data retrieval for location-dependent queries in a multi-cell wireless environment. *Mobile Information Systems*, 1(2):91–108.

Jayaputera, J. and Taniar, D. (2005b). Query processing strategies for location-dependent information systems. *International Journal of Business Data Communications and Networking (IJBDCN)*, 1(2):17–40.

Jensen, C., Lin, D., and Ooi, B. (2004). Query and update efficient B+-tree based indexing of moving objects. In *Proceedings of the 30th International Conference on Very Large Data Bases*, pages 768–779. VLDB Endowment.

Jensen, C., Lu, H., and Yang, B. (2009a). Graph model based indoor tracking. In *Proceedings of the 10th IEEE International Conference on Mobile Data Management (MDM)*, pages 122–131. IEEE.

Jensen, C., Lu, H., and Yang, B. (2009b). Indexing the trajectories of moving objects in symbolic indoor space. In *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*, pages 208–227. Springer-Verlag.

Jensen, C., Lu, H., and Yang, B. (2010). Indoor - A new data management frontier. *IEEE Data Engineering Bulletin*, 33(2):12–17.

Jiang, B. and Claramunt, C. (2000). Extending space syntax towards an alternative model of space within gis. In *Proceedings of the 3rd European Agile Conference on Geographic Information Science*. University of Utrecht, Faculty of Geographical Sciences.

Jiang, B. and Claramunt, C. (2004). A structural approach to the model generalization of an urban street network. *GeoInformatica*, 8(2):157–171.

Jiang, C. and Steenkiste, P. (2002). A hybrid location model with a computable location identifier for ubiquitous computing. In *Proceedings of the 4th International Conference on Ubiquitous Computing, (UbiComp)*, volume 2498, pages 246–263. Springer-Verlag.

Jung, S. and Pramanik, S. (2002). An efficient path computation model for hierarchically structured topographical road maps. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1029–1046.

Kainz, W., Egenhofer, M., and Greasley, I. (1993). Modelling spatial relations and operations with partially ordered sets. *International Journal of Geographical Information Science (IJGIS)*, 7(3):215–229.

Kalashnikov, D., Prabhakar, S., and Hambrusch, S. (2004). Main memory evaluation of monitoring queries over moving objects. *Distributed and Parallel Databases*, 15(2):117–135.

Kalashnikov, D., Prabhakar, S., Hambrusch, S., and Aref, W. (2002). Efficient evaluation of continuous range queries on moving objects. In *Database and Expert Systems Applications*, pages 25–64. Springer.

Kang, H., Kim, J., and Li, K. (2010). strack: Tracking in indoor symbolic space with rfid sensors. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 502–505. ACM.

Kang, H., Li, K., and Kim, M. (2007). A framework for dynamic updates of map data in mobile devices. *International Journal of Web Engineering and Technology*, 3(2):176–195.

Kaptelinin, V. and Nardi, B. (2009). *Acting with technology: Activity theory and interaction design*. The MIT Press.

Kawsar, F., Fujinami, K., and Nakajima, T. (2007). Experiences with building intelligent environment through sentient artefacts. In *Proceedings of the 3rd IET International Conference on Intelligent Environments*, pages 260–267. IET.

Kießling, W., Endres, M., and Wenzel, F. (2011). The preference sql system-an overview. *IEEE Data Engineering Bulletin*, 34(2):11–18.

Kießling, W. and Köstler, G. (2002). Preference SQL: Design, implementation, experiences. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 990–1001. VLDB Endowment.

Kim, H. and Jun, C. (2008). Indoor spatial analysis using space syntax. In *Proceedings of the XXIth Congress of the ISPRS*, pages 1065–1070.

Kofod-petersen, A. and Cassens, J. (2006). Using activity theory to model context awareness. In *Proceedings of the 2nd International Workshop on Modeling and Retrieval of Context*, pages 1–17. Springer.

Kolodziej, K. and Hjelm, J. (2006). *Local positioning systems: LBS applications and services*. CRC Press.

Kossmann, D. (2000). The state of the art in distributed query processing. *ACM Computing Surveys*, 32(4):422–469.

Kostakos, V. (2009). *Space Syntax and Pervasive Systems*, chapter 3, pages 31–52. Springer.

Kritsotakis, M., Michou, M., Nikoloudakis, E., Bikakis, A., Patkos, T., Antoniou, G., and Plexousakis, D. (2009). Design and implementation of a semantics-based contextual navigation guide for indoor environments. *Journal of Ambient Intelligence and Smart Environments*, 1(3):261–285.

Kuipers, B. (2000). The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119(1-2):191–233.

Kuipers, B. and Byun, Y. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8(1-2):47–63.

Kuipers, B., Modayil, J., Beeson, P., MacMahon, M., and Savelli, F. (2004). Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, volume 5, pages 4845–4851. IEEE.

Kwan, M. and Lee, J. (2005). Emergency response after 9/11: The potential of real-time 3d gis for quick emergency response in micro-spatial environments. *Computers, Environment and Urban Systems (CEUS)*, 29(2):93–113.

Kwon, D., Lee, S., and Lee, S. (2002). Indexing the current positions of moving objects using the lazy update r-tree. In *Proceedings of the 3rd International Conference on Mobile Data Management*, pages 113–120. IEEE.

Lam, K., Ulnsoy, O., Lee, T., Chan, E., and Li, G. (2001). An efficient method for generating location updates for processing of location-dependent continuous queries. In *Proceedingsof the 7th International Conference on Database Systems for Advanced Applications*, pages 218–225. IEEE.

Lane, N., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., and Campbell, A. (2010). A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150.

Lazaridis, I., Porkaew, K., and Mehrotra, S. (2002). Dynamic queries over mobile objects. pages 269–286.

Ledoux, H. (2006). *Modelling three-dimensional fields in geoscience with the Voronoi diagram and its dual*. PhD thesis, GIS Research Centre, School of Computing.

Lee, C., Wu, Y., and Chen, A. (2007). Continuous evaluation of fastest path queries on road networks. In *Proceedings of the 10th International Conference on Advances in Spatial and Temporal Databases*, pages 20–37. Springer.

Lee, D., Zhu, M., and Hu, H. (2005). When location-based services meet databases. *Mobile Information Systems*, 1(2):81–90.

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 Zlatanova, S. (2008). *A 3D data model and topological analyses for emergency response in urban areas*, volume 6, chapter Geospatial Information Technology for Emergency Response (ISPRS book series), pages 143–168. Taylor & Francis.

Leonhardt, U. (1998). *Supporting location-awareness in open distributed systems*. PhD thesis, Imperial College.

Levandoski, J., Mokbel, M., and Khalefa, M. (2010a). FlexPref: a framework for extensible preference evaluation in database systems. In *Proceedings of the IEEE 26th International Conference on Data Engineering (ICDE)*, pages 828–839. IEEE.

Levandoski, J. J., Mokbel, M. F., and Khalefa, M. E. (2010b). Caredb: A context and preference-aware location-based database system. *Proceedings of the VLDB Endowment*, 3(1-2):1529–1532.

Li, D. and Lee, D. (2008a). A lattice-based semantic location model for indoor navigation. In *Proceedings of the 9th International Conference on Mobile Data Management*, pages 17–24. IEEE.

Li, D. and Lee, D. (2008b). A topology-based semantic location model for indoor applications. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–10. ACM.

Li, K.-J. (2008). Indoor space: A new notion of space. In *Proceedings of the 8th International Symposium on Web and Wireless Geographical Information Systems, W2GIS*, pages 1–3.

Li, X., Claramunt, C., and Ray, C. (2010). A grid graph-based model for the analysis of 2D indoor spaces. *Computers, Environment and Urban Systems*, 34(6):532–540.

Liebich, T., Adachi, Y., Forester, J., Hyvarinen, J., Karstila, K., and Wix, J. (2006). Industry Foundation Classes: IFC2x Edition 3 TC1, technical corrigendum 1. *International Alliance for Interoperability (Model Support Group)*.

Liu, H., Darabi, H., Banerjee, P., and Liu, J. (2007). Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews*, 37(6):1067–1080.

Lu, H., Cao, X., and Jensen, C. (2012). A foundation for efficient indoor distance-aware query processing. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 438–449. IEEE.

Lu, J. and Guting, R. (2012). Parallel Secondo: Boosting database engines with Hadoop. In *Proceedings of 18th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 738–743. IEEE.

Lyardet, F., Szeto, D., and Aitenbichler, E. (2008). Context-aware indoor navigation. In *Proceedings of the 2nd European Conference in Ambient Intelligence (AmI)*, pages 290–307. Springer.

Malviya, N., Madden, S., and Bhattacharya, A. (2011). A continuous query system for dynamic route planning. In *Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE)*, pages 792–803. IEEE.

Mata, F., Claramunt, C., and Juarez, A. (2011). An experimental virtual museum based on augmented reality and navigation. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 497–500. ACM.

Matthew, N. and Stones, R. (2005). *Beginning databases with postgreSQL: From novice to professional*. Apress.

Mekni, M. (2010). *Automated generation of geometrically-precise and semantically-informed virtual geographic environments populated with spatially-reasoning agents*. Universal-Publishers.

Mokbel, M. (2005). *Scalable continuous query processing in location-aware database servers*. PhD thesis, Purdue University.

Mokbel, M. and Aref, W. (2008). Sole: Scalable on-line execution of continuous queries on spatio-temporal data streams. *The International Journal on Very Large Data Bases (VLDB)*, 17(5):971–995.

Mokbel, M., Ghanem, T., and Aref, W. (2003). Spatio-temporal access methods. *IEEE Data Engineering Bulletin*, 26(2):40–49.

Mokbel, M. and Levandoski, J. (2009). Toward context and preference-aware location-based services. In *Proceedings of the 8th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 25–32. ACM.

Mokbel, M., Xiong, X., and Aref, W. (2004). SINA: Scalable incremental processing of continuous queries in spatio-temporal databases. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 623–634. ACM.

Mokbel, M., Xiong, X., Hammad, M., and Aref, W. (2005). Continuous query processing of spatio-temporal data streams in place. *GeoInformatica*, 9(4):343–365.

Mokhtari, A. (2011). *Système personnalisé de planification d'itinéraire : Une approche basée sur la théorie des ensembles flous*. PhD thesis, Université de Rennes 1 - IRISA - France.

Montello, D. (2005). Navigation. *The Cambridge handbook of visuospatial thinking*, 18:257–294.

Moravec, H. and Elfes, A. (1985). High Resolution Maps from Wide Angle Sonar. In *Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 116–121. IEEE.

Mouratidis, K., Yiu, M., Papadias, D., and Mamoulis, N. (2006). Continuous nearest neighbor monitoring in road networks. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*, pages 43–54. VLDB Endowment.

Nagel, C., Becker, T., Kaden, R., Li, K., Lee, J., and Kolbe, T. (2010). Requirements and space-event modeling for indoor navigation. Technical Report OGC 10-191r1, Open Geospatial Consortium Discussion Paper.

Nardi, B. (1996). *Context and consciousness: activity theory and human-computer interaction*. The MIT Press.

Navas, J. and Imielinski, T. (1997). GeoCast - Geographic addressing and routing. In *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 66–76. ACM.

Nehme, R. and Rundensteiner, E. (2006). SCUBA: scalable cluster-based algorithm for evaluating continuous spatio-temporal queries on moving objects. In *Proceedings of the 10th International Conference on Advances in Database Technology (EDBT)*, pages 1001–1019. Springer-Verlag.

Nguyen-Dinh, L., Aref, W., and Mokbel, M. (2010). Spatio-temporal access methods: Part 2 (2003 - 2010). *IEEE Data Eng. Bull.*, 33(2):46–55.

Obe, R. and Hsu, L. (2011). *PostGIS in Action*. Manning Publications Co.

Oliveira, M., Hairon, C., Andrade, O., Moura, R., Sicotte, C., Denis, J., Fernandes, S., Gensel, J., Bringel, J., and Martin, H. (2010). A context-aware framework for health care governance decision-making systems: A model based on the brazilian digital tv. In *Proceedings of the IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, pages 1–6. IEEE.

Orda, A. and Rom, R. (1990). Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 37:607–625.

Papadias, D., Zhang, J., Mamoulis, N., and Tao, Y. (2003). Query processing in spatial network databases. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, pages 802–813. VLDB Endowment.

Park, I., Jang, G., Park, S., and Lee, J. (2009). Time-dependent optimal routing in micro-scale emergency situation. In *Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 714–719. IEEE.

Patroumpas, K. and Sellis, T. (2004). Managing trajectories of moving objects as data streams. In *Proceedings of the 2nd International Workshop on Spatio-Temporal Database Management (STDBM)*, volume 30.

Patroumpas, K. and Sellis, T. (2009). Monitoring orientation of moving objects around focal points. In *Advances in Spatial and Temporal Databases*, pages 228–246. Springer.

Patroumpas, K. and Sellis, T. (2012). Event processing and real-time monitoring over streaming traffic data. In *Proceedings of the 11th International Conference on Web and Wireless Geographical Information Systems (W2GIS)*, pages 116–133. Springer-Verlag.

Pederson, T. (1999). Physical-virtual instead of physical or virtual-designing artefacts for future knowledge work environments. In *Proceedings of the 8th International Conference on Human-Computer Interaction: Communication, Cooperation, and Application Design*, pages 1070–1074. L. Erlbaum Associates Inc.

Pelekis, N., Frentzos, E., Giatrakos, N., and Theodoridis, Y. (2008). HERMES: Aggregative LBS via a trajectory DB engine. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1255–1258. ACM.

Pelekis, N., Frentzos, E., Giatrakos, N., and Theodoridis, Y. (2010). Supporting movement in ORDBMS.

Pelekis, N., Kopanakis, I., Marketos, G., Ntoutsi, I., Andrienko, G., and Theodoridis, Y. (2007). Similarity search in trajectory databases. In *Proceedings of the 14th International Symposium on Temporal Representation and Reasoning*, pages 129–140. IEEE.

Pelekis, N., Theodoridis, Y., Vosinakis, S., and Panayiotopoulos, T. (2006). Hermes–a framework for location-based data management. In *Proceedings of the 10th International Conference on Extending Database Technology (EDBT)*, pages 1130–1134. Springer.

Pelekis, N., Theodoulidis, B., Kopanakis, I., and Theodoridis, Y. (2004). Literature review of spatio-temporal database models. *The Knowledge Engineering Review*, 19(03):235–274.

Petit, M. (2010). *Approche spatiale pour la caractérisation du contexte d'exécution d'un système d'information ubiquitaire*. PhD thesis, Arts et Métiers ParisTech - Institut de Recherche de l'Ecole Navale.

Peytchev, E. and Claramunt, C. (2001). Experiences in building decision support systems for traffic and transportation gis. In *Proceedings of the 9th ACM International Symposium on Advances in Geographic Information Systems*, pages 154–159. ACM.

Porros Pérez, A., Pérez Álvarez, D., Melià Seguí, J., et al. (2011). Context-aware routing system in an indoor scenario. Technical report, Institute of Electronic Systems - Division of Telecommunications. Universitat Politècnica de Catalunya.

Prabhakar, S., Xia, Y., Kalashnikov, D., Aref, W., and Hambrusch, S. (2002). Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. *Computers, IEEE Transactions on*, 51(10):1124–1140.

Priyantha, N., Chakraborty, A., and Balakrishnan, H. (2000). The cricket location-support system. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 32–43. ACM.

Raubal, M. (2001). Human wayfinding in unfamiliar buildings: A simulation with a cognizing agent. *Cognitive Processing*, 2(3):363–388.

Ray, C., Comblet, F., Bonnin, J.-M., and Le Roux, Y.-M. (2010). *Wireless and information technologies supporting intelligent location-based services. In Book: Wireless Technologies in Intelligent Transportation Systems, Chapter 9, M.-T. Zhou, Y. Zhang, L.T. Yang (eds.)*, chapter Wireless Technologies in Intelligent Transportation Systems, Chapter 9, M.-T. Zhou, Y. Zhang, L.T. Yang (eds.), pages 225–265. Nova Science Publishers.

Reichle, R., Wagner, M., Khan, M., Geihs, K., Valla, M., Fra, C., Paspallis, N., and Papadopoulos, G. (2008). A context query language for pervasive computing environments. In *Proceedings of the*

6th Annual IEEE International Conference on Pervasive Computing and Communications, pages 434–440. IEEE.

Remolina, E., Fernandez, J., Kuipers, B., and Gonzalez, J. (1999). Formalizing regions in the spatial semantic hierarchy: An AH-graphs implementation. In *Proceedings of the International Conference on Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science, COSIT*, pages 109–124. Springer Verlag.

Remolina, E. and Kuipers, B. (2004). Towards a general theory of topological maps. *Artificial Intelligence*, 152(1):47–104.

Ren, Q. and Dunham, M. (2000). Using semantic caching to manage location dependent data in mobile computing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 210–221. ACM.

Retscher, G. and Fu, Q. (2007). Integration of RFID, GNSS and DR for ubiquitous positioning in pedestrian navigation. *Journal of Global Positioning Systems*, 6(1):56–64.

Rice, M. and Tsotras, V. (2010). Graph indexing of road networks for shortest path queries with label restrictions. *Proceedings of the 36th International Conference on Very Large Data Bases (VLDB)*, 4(2):69–80.

Rivera, M., Claramunt, C., and Juarez, A. (2011). Analyzing semantic locations cloaking techniques in a probabilistic grid-based map. In *Proceedings of the 19th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 497–500. ACM.

Roussopoulos, N., Kelley, S., and Vincent, F. (1995). Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*.

Rundensteiner, E., Ding, L., Zhu, Y., Sutherland, T., and Pielech, B. (2005). Cape: A constraint-aware adaptive stream processing engine. In Chaudhry, N., Shaw, K., and Abdelguerfi, M., editors, *Stream Data Management*, volume 30 of *Advances in Database Systems*, pages 83–111. Springer.

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260.

Satoh, I. (2007). A location model for smart environments. *Pervasive and Mobile Computing*, 3(2):158–179.

Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8(4):10–17.

Schiller, J. and Voisard, A. (2004). *Location-Based Services*. Morgan Kaufmann, San Francisco, CA, USA.

Schmidt, A., Aidoo, K., Takaluoma, A., Tuomelam, U., Van Laerhoven, K., and Van de Velde, W. (1999). Advanced interaction in context. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC)*, pages 89–101. Springer.

Seder, M., Mostarac, P., and Petrović, I. (2011). Hierarchical path planning of mobile robots in complex indoor environments. *Transactions of the Institute of Measurement and Control*, 33(3-4):332–359.

Sellis, T., Roussopoulos, N., and Faloutsos, C. (1987). The R+-Tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB)*, pages 507–518. Morgan Kaufmann Publishers Inc.

Shang, S., Yuan, B., Deng, K., Xie, K., and Zhou, X. (2011). Finding the most accessible locations–reverse path nearest neighbor query in road networks. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 181–190. ACM.

Shekhar, S., Fetterer, A., and Goyal, B. (1997). Materialization trade-offs in hierarchical shortest path algorithms. In *Proceedings of the 5th International Symposium on Advances in Spatial Databases, SSD'97*, pages 94–111. Springer-Verlag.

Shirani-Mehr, H., Banaei-Kashani, F., and Shahabi, C. (2012). Efficient reachability query evaluation in large spatiotemporal contact datasets. *Proceedings of the VLDB Endowment*, 5(9):848–859.

Siegwart, R. and Nourbakhsh, I. (2004). *Introduction to autonomous mobile robots*. The MIT Press.

Sistla, A., Wolfson, O., Xu, B., and Rishe, N. (2011). Answer-pairs and processing of continuous nearest-neighbor queries. In *Proceedings of the 7th ACM SIGACT/SIGMOBILE International Workshop on Foundations of Mobile Computing*, pages 16–24. ACM.

Sistla, P., Wolfson, O., Chamberlain, S., and Dao, S. (1997). Modeling and querying moving objects. In *Proceedings of the 13th International Conference on Data Engineering (ICDE)*, pages 422–432. IEEE.

Sommer, C. (2010). *Approximate shortest path and distance queries in networks*. PhD thesis, The University of Tokyo.

Song, X., Yu, L., and Sun, H. (2008). An incremental query algorithm for optimal path queries under traffic jams. In *Proceedings of the International Symposium on Computer Science and Computational Technology, ISCSCT*, pages 472–475.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3310–3317. IEEE.

Stentz, A. (1995). The focussed D* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1652–1659. LAWRENCE ERLBAUM ASSOCIATES LTD.

Stirbu, V. (2009). NavTag: An inter-working framework based on tags for symbolic location coordinates for smart spaces. In *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing (UIC)*, pages 266–280. Springer.

Stoffel, E., Schoder, K., and Ohlbach, H. (2008). Applying hierarchical graphs to pedestrian indoor navigation. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM.

Stoffel, E.-P. (2009). *Hierarchical graphs as organisational principle and spatial model applied to pedestrian indoor navigation*. PhD thesis, LMU München: Faculty of Mathematics, Computer Science and Statistics.

Stojanovic, D., Papadopoulos, A., Predic, B., Djordjevic-Kajan, S., and Nanopoulos, A. (2008). Continuous range monitoring of mobile objects in road networks. *Jounrnal of Data & Knowledge Engineering*, 64(1):77–100.

Sun, X., Yeoh, W., and Koenig, S. (2009). Efficient incremental search for moving target search. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence (IJCA)*, pages 615–620. Morgan Kaufmann.

Sun, X., Yeoh, W., and Koenig, S. (2010a). Generalized Fringe-Retrieving A*: Faster moving target search on state lattices. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '10, pages 1081–1088. International Foundation for Autonomous Agents and Multiagent Systems.

Sun, X., Yeoh, W., and Koenig, S. (2010b). Moving target D* lite. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 67–74. International Foundation for Autonomous Agents and Multiagent Systems.

Tao, Y., Papadias, D., and Shen, Q. (2002). Continuous nearest neighbor search. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, pages 287–298. VLDB Endowment.

Terry, D., Goldberg, D., Nichols, D., and Oki, B. (1992). Continuous queries over append-only databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 321–330. ACM.

Thrun, S. (1997). To know or not to know: On the utility of models in mobile robotics. *AI Magazine*, 18(1):47–55.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

Thrun, S. (2003). Robotic mapping: A survey. In *Exploring artificial intelligence in the new millennium*, pages 1–35. Morgan Kaufmann Publishers Inc.

Torres-Solis, J., Falk, T., and Chau, T. (2010). *A review of indoor localization technologies: towards navigational assistance for topographical disorientation*, chapter Ambient Intelligence. INTECH.

Trajcevski, G., Ding, H., and Scheuermann, P. (2005a). Context-aware optimization of continuous range queries maintenance for trajectories. In *Proceedings of the 4th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 1–8. ACM.

Trajcevski, G., Scheuermann, P., Brönnimann, H., and Voisard, A. (2005b). Dynamic topological predicates and notifications in moving objects databases. In *Proceedings of the 6th International Conference on Mobile Data Management*, pages 77–85. ACM.

Trajcevski, G., Scheuermann, P., Ghica, O., Hinze, A., and Voisard, A. (2006). Evolving triggers for dynamic environments. In *Proceedings of the 10th International Conference on Advances in Database Technology*, pages 1039–1048. Springer-Verlag.

Trajcevski, G., Scheuermann, P., Wolfson, O., and Nedungadi, N. (2004a). CAT: correct answers of continuous queries using triggers. In Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., and Ferrari, E., editors, *Advances in Database Technology (EDBT)*, volume 2992 of *Lecture Notes in Computer Science*, pages 837–840. Springer.

Trajcevski, G., Wolfson, O., Hinrichs, K., and Chamberlain, S. (2004b). Managing uncertainty in moving objects databases. *ACM Transactions on Database Systems (TODS)*, 29(3):463–507.

Tsetsos, V., Anagnostopoulos, C., Kikiras, P., and Hadjiefthymiades, S. (2006). Semantically enriched navigation for indoor environments. *International Journal of Web and Grid Services*, 2(4):453–478.

Tsetsos, V., Kikiras, P., and Stathes, P. (2005). OntoNav: A Semantic Indoor Navigation System. In *Proceedings of the 1st International Workshop on Managing Context Information in Mobile and Pervasive Environments*. Citeseer.

Turner, A. (2001). Depthmap: A program to perform visibility graph analysis. In *Proceedings of the 3rd International Symposium on Space Syntax*, pages 31.1–31.12. Georgia Institute of Technology.

Turner, A. and Penn, A. (2002). Encoding natural movement as an agent-based system: An investigation into human pedestrian behaviour in the built environment. *Environment and Planning B: Planning and Design*, 29(4):473–490.

van Bunningen, A., Feng, L., and Apers, P. (2006). A context-aware preference model for database querying in an ambient intelligent environment. In Bressan, S., Küng, J., and Wagner, R., editors, *Database and Expert Systems Applications (DEXA)*, Lecture Notes in Computer Science, pages 33–43. Springer.

van Treeck, C. and Rank, E. (2004). Analysis of building structure and topology based on graph theory. In *Proceeding of the 10th International Conference on Computing in Civil and Building Engineering (ICCCBE)*.

Vargas-Solar, G., Ibrahim, N., Collet, C., Adiba, M., Petit, J., and Delot, T. (2010). *Pervasive Computing and Communications Design and Deployment: Technologies, Trends, and Applications*, chapter Querying issues in pervasive environments, pages 1–20.

Vazirgiannis, M. and Wolfson, O. (2001). A spatiotemporal model and language for moving objects on road networks. In *Advances in Spatial and Temporal Databases*, pages 20–35. Springer.

Wallgrun, J. (2005). Autonomous construction of hierarchical Voronoi-based route graph representations. In *Proceedings of the International Conference on Spatial Cognition IV. Reasoning, Action, and Interaction*, pages 413–433. Springer.

Wallgrun, J. (2010). *Hierarchical Voronoi graphs: Spatial representation and reasoning for mobile robots*. Springer.

Walton, L. and Worboys, M. (2012). A qualitative bigraph model for indoor space. In Xiao, N., Kwan, M.-P., Goodchild, M., and Shekhar, S., editors, *Geographic Information Science*, volume 7478 of *Lecture Notes in Computer Science*, pages 226–240. Springer.

Wang, H. and Zimmermann, R. (2011). Processing of continuous location-based range queries on moving objects in road networks. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1065–1078.

Want, R., Hopper, A., Falcao, V., and Gibbons, J. (1992). The Active Badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102.

Ward, A., Jones, A., and Hopper, A. (1997). A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94–104.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–84.

Wenzel, F., Soutschek, M., and Kießling, W. (2011). A preference sql approach to improve context-adaptive location-based services for outdoor activities. In *Advances in Location-Based Services: 8th International Symposium on Location-Based Services, Vienna 2011*, pages 191–207. Springer.

Werner, S., Krieg-Bruckner, B., and Herrmann, T. (2000). Modelling navigational knowledge by route graphs. In *Spatial Cognition II, Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications*, pages 295–316. Springer.

Wolfson, O., Sistla, A., Chamberlain, S., and Yesha, Y. (1999a). Updating and querying databases that track mobile units. *Distributed and parallel databases*, 7(3):257–387.

Wolfson, O., Sistla, P., Xu, B., Zhou, J., and Chamberlain, S. (1999b). DOMINO: Databases for moving objects tracking. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 28, pages 547–549. ACM.

Wolfson, O., Xu, B., Chamberlain, S., and Jiang, L. (1998). Moving objects databases: Issues and solutions. In *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pages 111–122. IEEE.

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

Wu, K., Chen, S., and Yu, P. (2006). Incremental processing of continual range queries over moving objects. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1560–1575.

Wu, L., Xiao, X., Deng, D., Cong, G., Zhu, A., and Zhou, S. (2012). Shortest path and distance queries on road networks: an experimental evaluation. *Proc. VLDB Endow.*, 5(5):406–417.

Wu, S. and Fan, H. (2009). Activity-based proactive data management in mobile environments. *IEEE Transactions on Mobile Computing*, 9(3):390–404.

Xie, X., Lu, H., and Pedersen, T. (2013). Efficient distance-aware query evaluation on indoor moving objects. Proceedings of the 29TH IEEE International Conference on Data Engineering (ICDE).

Xiong, X. and Aref, W. (2006). R-trees with update memos. In *Proceedings of the 22nd International Conference on Data Engineering*, pages 22–32. IEEE.

Xiong, X., Mokbel, M., and Aref, W. (2005). SEA-CNN: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, pages 643–654. IEEE.

Xu, J. and Güting, R. (2011). A generic data model for moving objects. FernUniversität Hagen, Germany.

Xu, M., Pan, Z., Lu, H., Ye, Y., Lv, P., and El Rhalibi, A. (2010). Moving-target pursuit algorithm using improved tracking strategy. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):27–39.

Xu, Z. and Jacobsen, A. (2007). Adaptive location constraint processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 581–592. ACM.

Xuan, K., Taniar, D., Safar, M., and Srinivasan, B. (2010). Time constrained range search queries over moving objects in road networks. In *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*, pages 329–336. ACM.

Xuan, K., Zhao, G., Taniar, D., Rahayu, W., Safar, M., and Srinivasan, B. (2011). Voronoi-based range and continuous range query processing in mobile databases. *Journal of Computer and System Sciences*, 77(4):637–651.

Yan, Z., Chakraborty, D., Misra, A., Jeung, H., and Aberer, K. (2012). SAMMPLE: detecting semantic indoor activities in practical settings using locomotive signatures. In *Proceedings of 16th International Symposium on Wearable Computers (ISWC)*, pages 37–40. IEEE.

Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., and Aberer, K. (2011a). SeMiTri: a framework for semantic annotation of heterogeneous trajectories. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 259–270. ACM.

Yan, Z., Giatrakos, N., Katsikaros, V., Pelekis, N., and Theodoridis, Y. (2011b). SeTraStream: semantic-aware trajectory construction over streaming movement data. In *Proceedings of the 12th International Conference on Advances in Spatial and Temporal Databases*, pages 367–385. Springer.

Yang, B., Lu, H., and Jensen, C. (2009). Scalable continuous range monitoring of moving objects in symbolic indoor space. In *Proceeding of the 18th Conference on Information and Knowledge Management (ICIKM)*, pages 671–680. ACM.

Yang, B., Lu, H., and Jensen, C. (2010). Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT)*, pages 335–346. ACM.

Yin, X., Ding, Z., and Li, J. (2008). A shortest path algorithm for moving objects in spatial network databases. *Progress in Natural Science*, 18(7):893–899.

Yu, S. and Spaccapietra, S. (2010). A knowledge infrastructure for intelligent query answering in location-based services. *Geoinformatica*, 14(3):379–404.

Yuan, W. and Schneider, M. (2010). Supporting continuous range queries in indoor space. In *Proceedings of the 11th International Conference on Mobile Data Management (MDM)*, pages 209–214. IEEE.

Zhang, J., Zhu, M., Papadias, D., Tao, Y., and Lee, D. (2003). Location-based spatial queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 443–454. ACM.

Zhang, S., McCullagh, P., Nugent, C., and Zheng, H. (2011). *An ontology-based context-aware approach for behaviour analysis*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, pages 127–148. Atlantis Press, Activity Recognition in Pervasive Intelligent Environment edition.

Zhao, L., Jing, N., Chen, L., and Zhong, Z. (2010). A novel framework for processing continuous queries on moving objects. In Chen, L., Tang, C., Yang, J., and Gao, Y., editors, *Web-Age Information Management*, Lecture Notes in Computer Science, pages 321–332. Springer-Verlag.

Knowledge Representation and Management in Indoor Mobile Environments

Abstract

The range of applications in ambient information systems progressively evolves from large to small scale environments. This is particularly the case for applications that assist humans in navigation-related activities in indoor spaces (e.g., airports, museums, office buildings). The research presented by this Ph.D. dissertation develops a data and knowledge representation of an indoor environment that takes into account user-centred contextual dimensions and mobile data management issues. We introduce a hierarchical, context-dependent, and feature-based indoor spatial data model. This model takes into account additional contextual dimensions such as time, user profiles, and real-time events. The model is based on a tree structure in which location information is represented at different levels of abstraction. The hierarchical design favours performance and scalability of location-dependent query processing. A query grammar is developed and implemented on top of that model. This query language supports continuous location-dependent queries and takes into account user preferences at execution time. The concept of location granules is introduced at the query execution and presentation levels.

This modelling approach is complemented by the development of a generic architecture for continuous query processing. Several algorithms for location-dependent query processing over indoor moving objects have been designed and implemented. These algorithms include path searches and range queries applied to both static and moving objects. They are based on an incremental approach in order to execute continuous location-dependent queries. The operators and constraints introduced in the query language and the algorithms for location-dependent query processing have been implemented as a database extension of the open source DBMS PostgreSQL, and where the hierarchical network-based indoor data model has been developed at the logical level. Several experiments have been conducted to evaluate the scalability and performance of the whole framework.

Keywords: Indoor data models, context-aware systems, mobile data management, continuous location-dependent queries, moving objects

Représentation et Gestion des Connaissances dans les Environnements Intérieurs Mobiles

Résumé

Les systèmes d'information mobiles et ambiants liés à la localisation et à la navigation évoluent progressivement vers des environnements à petite échelle. La nouvelle frontière scientifique et technologique concernent les applications qui assistent les utilisateurs dans leurs déplacements et activités au sein d'espaces bâtis dits «indoor» (e.g., aéroports, musées, bâtiments). La recherche présentée par cette thèse développe une représentation de données spatiales d'un environnement «indoor» qui tient compte des dimensions contextuelles centrées sur l'utilisateur et aborde les enjeux de gestion de données mobiles. Un modèle de données «indoor» hiérarchique et sensible au contexte est proposé. Ce modèle intègre différentes dimensions du contexte en plus de la localisation des entités concernées, telles que le temps et les profils des utilisateurs. Ce modèle est basé sur une structure arborescente dans laquelle l'information spatiale est représentée à différents niveaux d'abstraction. Cette conception hiérarchique favorise un traitement adaptatif et efficace des Requêtes Dépendantes de la Localisation (RDL) qui sont considérées comme des éléments clés pour le développement des différentes catégories de services de géolocalisation sensibles au contexte. Un langage de requêtes continues est développé et illustré par des exemples de requêtes RDL. Ce langage exploite le concept des granules spatiales, et permet de représenter les requêtes continues et dépendantes de la localisation en prenant compte des préférences de l'utilisateur au moment de l'exécution.

Cette approche de modélisation est complétée par le développement d'une architecture générique pour le traitement continu des requêtes RDL et par la conception et la mise en œuvre de plusieurs algorithmes qui permettent un traitement efficace des requêtes dépendantes de la localisation sur des objets mobiles en «indoor». Plusieurs algorithmes de traitement continu des requêtes de recherche de chemin hiérarchique et des requêtes de zone appliquées à des objets statiques et/ou en mouvement sont présentés. Ces algorithmes utilisent une approche hiérarchique et incrémentale afin d'exécuter efficacement les requêtes RDL continues. Un prototype encapsulant le modèle de données hiérarchique, les opérateurs et les contraintes introduits dans le langage de requête ainsi que les différents algorithmes et méthodes pour la manipulation de requêtes RDL a été développé comme une extension du SGBD Open Source PostgreSQL. Une étude expérimentale des solutions développées a été menée pour étudier la performance et le passage à l'échelle à l'égard des propriétés intrinsèques des solutions proposées.

Mots-clés : Modèles de données indoor, systèmes sensibles au contexte, gestion de données mobiles, requêtes continues dépendantes de la localisation, objets mobiles.

Discipline : Informatique/Computer Science