Location-Dependent Query Processing: Where We Are and Where We Are Heading

SERGIO ILARRI, EDUARDO MENA
University of Zaragoza
and
ARANTZA ILLARRAMENDI
University of the Basque Country

The continuous development of wireless networks and mobile devices has motivated an intense research in mobile data services. Some of these services provide the user with context-aware information. Specifically, location-based services and location-dependent queries have attracted a lot of interest.

In this article, the existing literature in the field of location-dependent query processing is reviewed. The technological context (mobile computing) and support middleware (such as moving object databases and data stream technology) are described, location-based services and location-dependent queries are defined and classified, and different query processing approaches are reviewed and compared.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; H.2.4 [**Database Management**]: Systems—*Query processing*; H.4.0 [**Information Systems**]: Information Systems Applications—*General*

General Terms: Algorithms, Design, Management

Additional Key Words and Phrases: continuous queries, location-based services, location-dependent data, location-dependent queries, mobile computing, query processing

1. INTRODUCTION

Nowadays, there is a great interest in mobile computing, motivated by an ever-increasing use of mobile devices, that aims at providing data access *anywhere and at anytime*. These devices are used not only to make voice connections (e.g., mobile phones) or to work locally (e.g., laptops, palmtops, etc.) but also to transmit data. Indeed, several market forecasts predict a growth in the use of mobile data services. For example, iGR (`http://www.igillottresearch.com`) estimates that 2.3 billion people will use wireless data services by 2011, and Informa (`http://www.`

Address of Sergio Ilarri and Eduardo Mena: IIS Department, University of Zaragoza, María de Luna 1, 50018 Zaragoza (Spain).
Address of Arantza Illarramendi: LSI Department, University of the Basque Country, Apdo. 649, 20080 San Sebastián (Spain).
Emails: silarri@unizar.es, emena@unizar.es, jipileca@si.ehu.es.

informatm.com) predicts that mobile data revenues will grow by 70 %, from $82.5 billion in 2006 to $124.5 billion in 2010 (see http://www.cdg.org/resources/white_papers/files/Data_Applications_Oct_07.pdf, by CDMA Development Group, October 2007).

Along with all its benefits, mobile computing also introduces many challenges for data management. Thus, despite the continuous development and improvement of wireless networks and mobile computing devices, they still present important limitations. For example, mobile devices are subject to an intermittent connectivity, a short battery lifetime, limited communications (slow, unreliable, and/or expensive), and limited capabilities in terms of memory, processing, storage, and display size. Moreover, the resources that a mobile device can access usually depend on its location. Due to these factors, early works in the field of mobile computing emphasize that classical computing techniques are insufficient and must be adapted to the new environment [Imielinski and Nath 1993; Imielinski 1996a; Pitoura and Samaras 1998; Jing et al. 1999; Barbará 1999].

Whereas many mobile computing applications are the counterpart of those available in desktop computers, there exist others that exploit the dynamic features of the mobile environment to provide the user with context-aware information. One of the most relevant context factors is the location of the user, which has motivated an intensive research effort in *location-based services* (*LBS*) [Schiller and Voisard 2004]. These services provide value-added by considering the location of the mobile user in order to offer more customized information. LBS are very enticing from an economic point of view, which has motivated the development of many interesting applications, such as emergency services (e.g., for roadside assistance), navigation and information services (e.g., digital travel assistants, or location-dependent yellow pages), location-dependent advertising (e.g., sending interesting offers to users near a supermarket), or tracking services (e.g., to keep track of fleets of vehicles, elderly people, or suspected criminals).

Mobile computing is the global framework of the survey presented in this article. More specifically, it focuses on *location-dependent queries*, which are probably one of the most challenging issues in location-based services. The answer to a location-dependent query depends on the location of the objects involved. For example, a user with a PDA may want to locate available taxi cabs that are near him/her while he/she is walking home in a rainy day [Veijalainen and Weske 2003]. Location-dependent queries are usually considered as *continuous queries*. Thus, the answer to the previous sample query must be continuously refreshed because it can change immediately due to the movements of people and taxi cabs. Moreover, even if the set of taxis satisfying the query condition does not change, their locations and distances to the user do change continuously, and therefore the answer to the query must be updated with the new location data if the current locations of these objects must be presented to the user (e.g., locating the taxis on a map). A taxi cab could also want to track nearby people looking for a taxi, in order to offer a better service than making the users wait at taxi stops or wander the streets in search of a cab, which would lead to similar difficulties. The challenges of processing queries with location constraints were probably considered for the first time in [Imielinski and Nath 1992]. After more than a decade of continuous work, the processing

of continuous location-dependent queries remains an important focus of research. The reason is twofold. On the one hand, the commercial applicability and the ever increasing use of mobile devices motivate work in the area. On the other hand, the challenges that the processing of location-dependent queries imply make it very interesting for research.

Regarding the topic of this survey, two main related elements can be considered: architectural aspects and query processing techniques.

— *Context and architecture.* In mobile computing, a set of devices with limited features communicate with each other, and with fixed computers, through a wireless infrastructure. Along with its challenges, this new environment also offers interesting opportunities. In particular, as mentioned previously, location-based services [Schiller and Voisard 2004] have attracted much attention, both commercially and in research. Processing location-dependent queries, a major building block of location-based services, requires some approach to manage and model the moving objects. Two major trends can be identified: moving object databases (e.g., see [Güting and Schneider 2005]) and data stream technology (e.g., [Huang and Jensen 2004]). The first approach implies extending traditional database techniques with models and index structures suitable to track the locations of the moving objects efficiently. The second approach focuses on the processing of continuous location updates as they arrive. The boundary between these two approaches is not always clear in relation to the topic of this survey: Both propose alternatives to classical database techniques, which are not considered appropriate to manage the continuously changing locations of the moving objects. The title of the work presented in [Chen et al. 2003] and its use of index structures, and the fact that relations can be mixed with streams [Arasu et al. 2003] are two examples of how these two approaches are not always clearly separated.

— *Query processing techniques.* Different approaches have been proposed to process location-dependent queries. These approaches depend on the way the moving objects are modeled and managed but also on the types of location-dependent queries considered, on the assumptions made about the mobility patterns of the moving objects, and on the amount of cooperation required from the moving objects, among others. For example, some works require that the moving objects monitor the queries that they may affect (e.g., [Gedik and Liu 2006; Cai et al. 2006]), others require that the objects follow a specific update policy that depends on the set of active queries (e.g., [Prabhakar et al. 2002; Hu et al. 2005a]), some of them assume that the objects follow known trajectories [Wolfson et al. 1999b; Trajcevski et al. 2005a; Lam and Ulusoy 2006; Stojanovic et al. 2008], and other works do not make these assumptions (e.g., [Mokbel et al. 2004; Ilarri et al. 2006a]). Similarly, some works focus on moving range queries (e.g., [Gedik and Liu 2006; Ilarri et al. 2006a]), static range queries (e.g., [Prabhakar et al. 2002; Cai et al. 2006]), nearest neighbor queries (e.g., [Frentzos et al. 2007; Mouratidis and Papadias 2007]), or queries on location-dependent data [Dunham and Kumar 1998; Lee 2007], for example.

The structure of the rest of this article is as follows. The context of mobile computing is introduced in Section 2. Location-based services and location-dependent queries are defined and classified in Section 3. Data management issues for moving

objects are reviewed in Section 4, considering two main alternatives: moving object databases and data streams. Then, different query processing approaches are described, with an emphasis on works that consider the general problem of processing location-dependent queries in mobile environments instead of, for example, just appropriate index structures. First, location-dependent query processing approaches that rely on some specific cooperation from the moving objects are reviewed in Section 5. Second, approaches that assume that moving objects follow known trajectories are described in Section 6. Third, other works where neither of these two assumptions is made are reviewed in Section 7. Fourth, some works focused specifically on nearest neighbor queries are presented in Section 8. And fifth, aggregate queries are considered in Section 9. Different research issues related to the problem of managing and querying location-dependent data, which are data whose values depend on the location of the user, are reported in Section 10. Finally, some conclusions and ideas for prospective research are summarized in Section 11.

2. TECHNOLOGICAL CONTEXT: MOBILE COMPUTING

According to [Milojicic et al. 2000], in general two types of mobility can be distinguished: *software mobility* and *hardware mobility*. Software mobility implies the transfer of passive data (e.g., files) or active data (mobile code, process migration, or mobile agents) [Carzaniga et al. 1997; Lange and Oshima 1999; Milojicic et al. 2000] among computers. However, the survey presented in this article relates to hardware mobility, which implies the physical mobility of mobile devices while providing computing services *anywhere and at anytime*. This type of mobility is located within the mobile computing framework, which "[...] no longer requires users to maintain a fixed and universally known position in the network and enables unrestricted mobility of the users" [Barbará 1999].

Recent advances in wireless communication technologies and portable devices (e.g., mobile phones, *Personal Digital Assistants –PDAs–*, *handheld* and portable computers, laptops, smartphones, etc.) have motivated an intense research effort in the field of mobile computing. Within mobile computing, nomadic computing and ad hoc computing are distinguished. In *nomadic computing* [Porta et al. 1996] the network infrastructure is fixed. On the contrary, in *ad hoc computing* [Sesay et al. 2004] a network is established temporarily and on-demand among the mobile devices, without relying on fixed computers connected to a wired network. In most works on location-dependent query processing, nomadic computing is assumed. In the rest of this section, the main aspects of the classical underlying wireless infrastructure for mobile (nomadic) computing and the limitations of mobile devices are described. Then, the focus is on some software models proposed for wireless environments. Finally, some challenges and related fields are reviewed.

2.1 Wireless Infrastructure

Wireless infrastructures have been evolving over time, from analog cellular architectures such as *AMPS* (*1G*) [Ehrlich 1979], to *GSM* (*2G*) [Mouly and Pautet 1992], GPRS (*2.5G*) [Lindemann and Thümmler 2003] and, more recently, *UMTS* (*3G*) [Kaaranen et al. 2001], *Bluetooth* [Morrow 2002], *Wi-Fi* [Ohrtman and Roeder 2003], and *WiMax* [Ohrtman 2005], as alternatives for wireless data communication. Some of these technologies are briefly described in Section 3.1.1 as facilitating

technologies for location-based services.

Looking beyond the details of the specific wireless technology, the generally accepted mobile environment infrastructure [Dunham and Helal 1995; Pitoura and Samaras 1998; Barbará 1999; Pitoura and Samaras 2001] is composed of *mobile hosts* (*MHs*) and *base stations* (*BSs*). Each BS serves all the mobile hosts within its *coverage area* or *cell*. The communication between a mobile host and the BS that provides it with coverage is wireless (using cellular network technology, satellites, or a wireless LAN) and the communication among BSs is wired. Thus, BSs allow the communication between mobile hosts and *fixed hosts* (*FHs*) connected to the Internet.

In the context of this article, the term *moving object* will be used to refer to any entity whose location is interesting. Some mechanism is needed to obtain the location of a moving object, which usually requires the object to be equipped with some kind of wireless connection. A car equipped with a portable computer connected to the Internet or a person carrying a PDA connected to a wireless hotspot are examples of moving objects[1]. Moving objects can be attached to devices such as *GPS* [Kumar and Stokkeland 2003] (*Global Positioning System*, a free-to-use global network of 24 satellites run by the US Department of Defense, which allows a person to obtain his/her current location) receivers or sensors that feed them with context information. Similarly, they can be capable of executing different software, such as a word processor, a web browser, or a route guiding system that shows on a map the GPS location of the moving object. Thus, depending on their features, moving objects can have different functionalities.

A moving object is usually attached to a mobile device (mobile host). As mobile devices move, they can change from one coverage area to another, which is called *handoff/handover* [Seshan 1995; Markopoulos et al. 2004]. The mobility of mobile devices introduces an important problem that does not exist in the traditional wired network: In order to communicate with a mobile device, the cell where it is located must be first obtained. The problem of *location management* in a mobile network [Pitoura and Samaras 2001; Mukherjee et al. 2003] implies the need of a location infrastructure and a location update policy, as explained in the following.

On the one hand, an appropriate *architecture* must be defined to store and provide the locations of the mobile devices. For example, the location strategies proposed in the GSM standard use a two-tier system [Pitoura and Samaras 2001]. Each mobile device is associated with a fixed *Home Location Register* (*HLR*) and a *Visitor Location Register* (*VLR*) that changes with the location of the device. Calls to a device are first directed to the VLR of the caller; if the callee is not found there, then the callee's HLR is contacted.

On the other hand, mobile devices must communicate changes in their locations following a certain *update policy*. There is a trade-off between the cost of location updates and the search cost. On the one hand, no search is needed if the mobile device updates its location whenever it hands-off to another cell, but in this case the update cost is high even if no call is intended for that device. On the other hand, the device could avoid updating its location when it moves to another cell, but then

[1]For the sake of generality, static objects (i.e., objects that do not move) can also be considered moving objects if their locations are interesting.

the device must be located by broadcasting the network when another device wants to communicate with it. Strategies such as the *time-based* strategy (e.g., periodic updates every 2 minutes), the *number of movements-based* strategy (based on the number of boundary crossings between cells), and the *distance-based* strategy (in terms of cells) have been studied in the literature [Bar-Noy et al. 1995]; a work that improves the distance-based strategy is [Ng and Chan 2005], and another recent work can be found in [Chang et al. 2008]. Location update policies are also usually needed for location-dependent query processing (see Section 4.1.2), but in that case the granularity of the location updates is much finer (e.g., geographical coordinates instead of cell identifiers).

2.2 Mobile Devices

Mobile devices are subject to a number of limitations that have an impact on the processing they can perform:

— *Frequent Disconnections.* A mobile device can disconnect anytime, because the device is powered off, it runs out of battery, it enters an area without coverage, or the wireless connection goes down.

— *Varying resources.* As the mobile device moves, the amount and quality of available resources, such as accessible services or bandwidth, change constantly.

— *Power limitations.* Mobile devices operate on batteries, which have a very short life. Thus, energy management has become an important requirement in mobile computing. Moreover, there is no expectation that this technical limitation will be overcome in the near future: The rate of improvement in the capacity of rechargeable batteries has leveled off and other alternatives, such as fuel cells, are being investigated [Viredaz et al. 2003].

— *Communication constraints.* In general, wireless communications are slow (e.g., 9.6 kbps in GSM and up to 170 kbps in GPRS)[2], expensive (economically and in terms of energy consumption), and unreliable (e.g., due to interferences). Due to the power limitations mentioned before, wireless communications should be minimized so as to limit the energy consumption. In a cellular network, more power is consumed in the transmit mode than in the receive mode [Jones et al. 2001]. Although in a WLAN the network interface requires the same cost to operate whether in the receive, transmit, or idle state [Anastasi et al. 2003], it is still important to minimize transmissions due to the asymmetric communication bandwidth [Barbará 1999]: There is more bandwidth available from the server to the mobile clients than from the clients to the server.

— *Limited capabilities of the mobile devices.* In comparison with fixed computers, mobile devices exhibit memory, processing, and storage constraints. Furthermore, the size of mobile devices also imposes important limitations on the display.

Due to these disadvantages, it is important to minimize the usage of wireless communications and the processing performed on the mobile devices [Nath et al. 1993a; Peng and Chen 2005]. The fixed network and wired computers should be used instead whenever possible.

[2]Some WLAN technologies offer higher speeds (e.g., 54 Mbps with Wi-Fi 802.11g) but they do not provide wide area coverage.

2.3 Wireless Computational Models

In a wireless environment, most of the assumptions that guide the definition of the traditional client/server architecture are not valid: 1) fast, reliable and cheap communications; 2) robust and powerful devices; and 3) fixed locations of the participating devices. Thus, the client/server architecture is not adequate anymore for wireless environments. In [Jing et al. 1999], an *extended client-server model* is proposed, where the functions of clients and servers can be mixed: Some typical client operations may be performed on the servers, due to resource limitations of the clients, and the clients can carry out some typical server functionalities so as to reduce the impact of limited connectivity with the servers. Several other software models have been proposed [Spyrou et al. 1999; Spyrou et al. 2004]:

— *Client/agent/server* (c/a/s) [Nath et al. 1993b]. It is a three-tier architecture that introduces an agent on the server side, which is in the wired network. The agent becomes an intermediate for the interactions between the client and the server.

— *Client/agent/agent/server* (c/a/a/s), called *client/intercept/server* in [Samaras and Pitsillides 1997]. The previous model is used as the basis of this one, which proposes the addition of a client-side agent. The pair of agents shields the mobile device from the limitations of the wireless environment, so legacy applications can be used more easily in the new environment. For example, these agents interact to reduce the wireless communications. Moreover, this model also facilitates adaptivity because both agents can divide the work between them according to the current conditions. The client-side agent can also include optimizations such as *view materialization* [Wolfson et al. 1995] and an asynchronous-disconnected mode by which queries that cannot be satisfied by the view are queued when connectivity is lost and resumed later when it is available again [Barbará and Imielinski 1995; Housel and Lindquist 1996].

— *Peer-to-peer* (*P2P*), which considers devices that may act simultaneously as clients and servers of a data service. For example, in [Reiher et al. 1996] a peer-to-peer model for data replication in a mobile computing environment is advocated. It is claimed that the P2P model can help improve the availability, reliability, and performance.

— *Mobile agents* [Spyrou et al. 2004; Ilarri et al. 2006b]. They have the ability to move autonomously from computer to computer to perform their tasks.

[Spyrou et al. 1999; Spyrou et al. 2004] show the flexibility of the mobile agent model in materializing the client/server, c/a/s, and the c/a/a/s models.

2.4 Challenges and Related Fields

In this section, we first present some challenges posed by mobile computing. The focus is on data management issues. Therefore, other works concerning how to support mobile computing on top of existing mobile environments, such as [Ioannidis et al. 1991; Teraoka et al. 1991; Perkins and Bhagwat 1994; Perkins 1998], are not considered here.

While mobile computing is expected to provide great economical benefits, it also opens up new research issues for data management, due to the need for adapting existing techniques to the mobile environment [Imielinski and Nath 1993; Alonso

and Korth 1993; Forman and Zahorjan 1994; Imielinski 1996a; Pitoura and Samaras 1998; Barbará 1999]. Particularly, the concept of *mobile databases*, where the users and/or data may be mobile, appears to highlight the differences with traditional distributed database systems [Özsu and Valduriez 1999]. An early characterization of the new problems appeared in [Alonso and Korth 1993]. The new environment has a great impact on:

—Data management [Imielinski and Nath 1993; Pitoura and Samaras 1998; Pitoura and Bhargava 1999]: data fragmentation, replication among mobile units, consistency maintenance, etc.

—Query processing [Imielinski and Nath 1992]: approximate answers, query fragmentation and routing, location-dependent queries, optimization of query plans (e.g., based on battery consumption, cost of wireless communications, bandwidth available, throughput, ...), etc.

—Transaction management [Kayan and Ulusoy 1999; Lee et al. 2004]: The ACID properties must be relaxed in a mobile environment because frequent disconnections may lead to long or blocked transactions.

—User interfaces and query languages [Chang 2003]: They must be adapted to the small displays of mobile devices.

Finally, some fields related to mobile computing, where a huge effort in research and development will continue to be carried out in the future, are the following:

— *Ubiquitous computing* [Weiser 1999b; Román et al. 2002; Milner 2004] (also called *pervasive computing* [Satyanarayanan 2001; Ray and Kurkovsky 2003]). The term was adopted by Mark Weiser in the mid 90's [Weiser 1999a] to describe reactive environments [Cooperstock et al. 1995] where *smart* objects [Bohn et al. 2004] (e.g., household appliances) interact among themselves to provide the user with the needed computational abilities, wherever they are needed and without too much user involvement. Ubiquitous computing is linked to two important technologies: 1) *wearable computing* (e.g., see [Rhodes et al. 1999; Feldman et al. 2005]), which implies body-worn small computers (e.g., embedded in glasses, wristwatches, etc.) that facilitate human-computer interaction; and 2) *augmented reality* [Vallino 1998], which are computer-generated scenes that augment the real scene with additional information, creating *natural interfaces* [Alisi et al. 2005] for the user.

— *Ambient intelligence* [Ducatel et al. 2001]. This term is closely related to that of ubiquitous/pervasive computing, but it can be considered a step further that implies a seamless environment of computing which is at the same time useful and unobtrusive. Thus, it considers *smart environments* [Cook and Das 2004], such as smart homes, that recognize people and adapt to the current situation or context. Software agents are expected to play a major role in this area [Hagras et al. 2004].

— *Context-aware computing* [Schilit et al. 1994; Chen and Kotz 2000; Smailagic et al. 2001; Baldauf et al. 2006] refers to computing that adapts to the environment. Thus, context information (e.g., the user's location, preferences, physiological status, activity, temperature, humidity, date and time, the device's capabilities and available battery, accessible networks, network access cost, etc.), can help to support better ways to provide data relevant to the user, to enable improved interoperability with the environment and with other mobile users, and to decide when and how

to process data. One of the most relevant factors in context-aware computing is the location of the user, which is exploited by location-based services to provide him/her with information relevant to his/her location.

3. LOCATION-BASED SERVICES AND LOCATION-DEPENDENT QUERIES

In this section, the importance of location-based services is highlighted and several sample applications are first presented. Then, different types of location-dependent queries are described and classified.

3.1 Location-Based Services

The advances in wireless technologies and mobile devices, together with the continuous improvement of positioning systems, has given rise to *location-based services* (*LBS*) [Schiller and Voisard 2004], also called *mobile location services* (*MLS*) in [Giaglis et al. 2003]. LBS are services which provide value-added by considering the location of the mobile users in order to give them customized information. ABI Research (`http://www.abiresearch.com`) estimates that the global location-based market will increase from $981 million in 2006 to $8 billion in 2010 (see `http://www.ibm.com/news/nl/nl/2006/06/nl_nl_news_20060629_2.html`). One of the reasons for this growth will be the widespread availability of GPS devices. Thus, Berg Insight predicts that the global number of GPS-enabled handsets will grow from 175 million units in 2007 to 560 million units in 2012 (see `http://www.berginsight.com/News.aspx?m_m=6&s_m=1`).

3.1.1 *Technologies for LBS.* Two types of technologies relevant to LBS are considered in [Giaglis et al. 2003]:

(1) *Enabling technologies*, which make LBS possible. They are positioning techniques used to determine the location of a mobile device. Several positioning methods with different precision can be used [Dye and Buckingham 1999; Deitel et al. 2001; Hjelm 2002]. For example, the *cell-id* [Trevisani and Vitaletti 2004] technique, also called *Cell of Origin* (*COO*) or *Cell Global Identity* (*CGI*), allows to identify the cell in which a certain wireless device is located. This method takes advantage of the location data that a wireless infrastructure stores about the mobile devices it serves (see Section 2.1).
However, the precision of cell-id is usually not enough for the purposes of existing LBS, as cells in urban environments have a typical size between 500 meters and 2 kilometers [Coombs and Steele 1999]. Thus, there is a myriad of other methods to obtain the location of a mobile device, such as *CGI+TA* (*Cell Global Identity with Timing Advance*) [Rizos 2005], *UL-TOA* (*Uplink Time of Arrival*) [Rizos 2005] or *TOA* [Giaglis et al. 2003; Rizos 2005], *AOA* (*Angle of Arrival*) [Pagès-Zamora et al. 2002], *OTD* (*Observed Time Difference*) [Giaglis et al. 2003] or *EOTD* (*Enhanced Observed Time Difference*) [Rizos 2005], *GPS* (*Global Positioning System*) [Kumar and Stokkeland 2003], *AGPS* (*network-Assisted GPS*) [Djuknic and Richton 2001], the upcoming European *Galileo* [Swann et al. 2003], *timing advance* [Mouly and Pautet 1992], etc. There are also methods based on infrared/RFID [Want et al. 1992; Bahl and Padmanabhan 2000; Ni et al. 2004], Ultra-WideBand (e.g., the *Ubisense Smart Space Platform*, see `http://www.ubisense.net`), Wi-Fi (e.g., see `http://www.herecast.com/`), Bluetooth [Aalto et al. 2004], sensor networks [Ray

et al. 2003], card swipe readers, login information on desktop computers, fingerprint recognition, and even network addresses (e.g., see [Roth 2003b]). Some surveys can be found in [Sun et al. 2005; Rizos 2005].

(2) *Facilitating technologies*, which are complementary technologies that provide the contextual environment where LBS can be implemented in a value-added fashion. In [Giaglis et al. 2003], four facilitating technologies for LBS are identified:

—The *Wireless Access Protocol* (*WAP*) [Tull 2002]. It enables the access to information from the Internet taking into account limitations of mobile devices such as processing power and display size. WAP is a facilitating technology as it allows the specification of semantic links between locations and information.

—The *General Packet Radio Service* (*GPRS*) [Andersson 2001; Lindemann and Thümmler 2003], or 2.5-generation mobile technology (*2.5G*). It is the successor to GSM [Mouly and Pautet 1992], which uses circuit-based connections, and supports *packet-based* connections instead. This minimizes the connection time and allows charging based on data volume instead of air-time.

—The *Universal Mobile Telecommunication System* (*UMTS*) [Kaaranen et al. 2001], or third generation mobile technology (*3G*). It also uses packet-based connections, but it provides more bandwidth than GPRS. This could allow the delivery of enhanced content to the user, such as videos of facilities in nearby hotels.

—The *Geographic Information Systems* (*GIS*) [Burrough and McDonnell 1998]. GIS manipulate, store and present geographically-related information. They allow to associate geographic coordinates with their context in the physical environment.

Technologies such as Bluetooth [Morrow 2002], Wi-Fi [Ohrtman and Roeder 2003] and WiMax [Ohrtman 2005] can also be considered as facilitating technologies for LBS.

3.1.2 *Privacy Protection and Security.* Privacy protection and security are important issues in location-based services. Thus, for example, the alteration of location information should be avoided, and appropriate access control policies must be in place so that location information is only disclosed to trusted parties. Moreover, the delivered location should not be more precise than necessary for the intended service. Two privacy protection requirements considered in the literature are: *location anonymity* and *identifier anonymity* [Xiao et al. 2007]. Spatial and temporal cloaking are two approaches that are applied in this context [Gruteser and Grunwald 2003].

Several works deal with the problem of location privacy in location-based services. For example, a personalized k-anonymity model is proposed in [Gedik and Liu 2005] (see Section 5.2). In such a model, the location information sent by each mobile node is perturbed by replacing it with a spatial range, such that there are k other mobile nodes within that range. [Liu 2007] indicates that k-anonymity is not enough and describes other complementary strategies, such as location l-diversity and location m-invariant. In [Mokbel et al. 2006], a privacy-aware query processor is presented. Moreover, there is an IETF (Internet Engineering Task Force) working group on Geographic Location/Privacy (*geopriv*, see `http://www.ietf.org/html.charters/geopriv-charter.html`). For more information about privacy preserving for location-based services, see [Gruteser and Grunwald 2003; Atallah and Frikken 2004; Görlach et al. 2005; Gedik and Liu 2005; Mokbel et al. 2006;

Xiao et al. 2007]. Reports on ethical concerns related to security and privacy in LBS can be found in [Michael et al. 2006; Perusco and Michael 2007].

3.1.3 *Standardization Issues.* LBS must be provided in a location-independent and user-transparent way. Thus, the user should be able to use the same service independently of his/her location and without any extra intervention. Moreover, it is not convenient for the developer of LBS to have to handle the details of the specific mechanism used to determine the locations of the mobile devices. This requirement stresses the importance of standardization.

An interesting initiative is the *Mobile Location Protocol* (*MLP*, see `http://www.openmobilealliance.org/tech/affiliates/lif/lifindex.html`), supported by the *Open Mobile Alliance* (*OMA*) (`http://www.openmobilealliance.org`), which defines a set of rules for querying and representing location information managed by *Location Servers*[3]. Moreover, some researchers also propose mechanisms for creating an infrastructure of location servers providing a common location model; e.g., [Myllymaki and Edlund 2002; Jiang and Steenkiste 2002; Roth 2003a; Graumann et al. 2003; Chen et al. 2004; Lehmann et al. 2004; Ranganathan et al. 2004].

3.1.4 *Examples and Scenarios.* LBS are not only interesting from an economic point of view, but they are also expected to provide other benefits. Thus, for example, according to [Junglas 2005], they increase the productivity and enjoyment when performing location-dependent tasks. In the following, some examples of location-based services are shown:

— *Emergency services.* The *911* and *E911* rules by the *Federal Communications Commission* (*FCC*) in the United States, which require the localization of a mobile subscriber in emergencies (with different coverage and precision depending on the implantation phase), have been major boosters of LBS. A similar role has played the *E112* requirements by the *Coordination Group on Access to Location Information for Emergency Services* (*CGALIES*) in the European Union. There also exist several commercial products (e.g., *Networkcar*, see `http://www.networkcar.com`) that provide enhance roadside assistance by tracking the GPS location of a vehicle. These examples highlight the importance of the ability to locate and individual who is unaware of his/her location in an emergency situation. Similarly, *geographic-based messaging* [Basagni et al. 1999; Imielinski and Navas 1999] can be very useful, for example, to warn users within a certain area about a danger.

— *Navigation and information services.* There are many navigation services that could be referenced here. For example, *digital travel assistants* (e.g., TomTom Navigator, see `http://www.tomtom.com/`) that provide driving directions optimizing some criteria such as the total distance or travel time, *tourist guides* (e.g., in a museum [Raptis et al. 2005] or a city [Cheverst et al. 2000; Pashtan et al. 2003]), and group management.
Other examples are *location-dependent web pages* [Acharya et al. 1995a; Haghighat et al. 2004] and *mobile yellow pages* that provide information about nearby points of interest (e.g., see [Chon et al. 2002b], `http://www.vindigo.com`, and `http:`

[3]Initially, this protocol was being developed by the *Location Interoperability Forum* (*LIF*), which has now consolidated into OMA and no longer exist as an independent organization.

//www.portableinternet.com), local events [Göker et al. 2004], etc.

— *Advertising services.* In the field of mobile commerce, the concept of *location-dependent advertising*, or *proximity-triggered advertisements*, arises as an effective way of attracting nearby users to stores [Aalto et al. 2004]. Thus, for example, people in the outside of a supermarket could receive e-discounts that encourage them to shop inside. As a specific example, the *Shopping Jacket infrastructure* [Randell and Muller 2000] uses GPS and local *pingers* in stores for positioning, and alerts wearers when they pass near a relevant shop.

— *Tracking services.* They encompass all those services whose goal is tracking the current locations of moving objects (e.g., vehicles, pets, children, elderly people, people suffering from Alzheimer's disease, products, or personnel). Thus, for example, *friend finder applications* let you know when your friends are nearby (e.g., see [Amir et al. 2007] and http://www.locatrix.com/products/finder.php). As another example, the *Ride Finder* (http://labs.google.com/ridefinder) monitors the locations of taxis in certain cities in the United States. Controversy has arisen recently regarding the existence of applications that allow parents to track their children; see http://www.accutracking.com as an example of a company offering these services. Similarly, ethical concerns also face existing applications to track parolees, suspected criminals, and employees [Michael et al. 2006; Perusco and Michael 2007]. The availability and miniaturization of GPS devices (e.g., in wristwatches and bracelets [Michael et al. 2006], or in clothes such as the *Know Where Jacket* from Interactive Wear AG, see http://www.interactive-wear.de) is contributing to the development of these services.

— *Billing services. Location-sensitive billing* allows service providers to customize rate zones to fit the needs of individual subscribers. Thus, a different rate can be charged for each zone, such that a wireless provider can compete more effectively with wired service providers. In this category, applications where a person is automatically charged for products upon leaving a supermarket, without going to the cashier [Giaglis et al. 2002], can also be considered.

The list of available LBS is constantly growing, and there are many other services that have not been mentioned above. For example, *location-based games* are games where the players move in the real world and interact with each other and with the environment depending on their locations. As an example, *Gunslingers* (http://guns.mikoishi.com/gunsSingTel/index.html) is a multiplayer network game where the players track and engage enemies within their vicinity. Similarly, *Spacerace* [Drab and Binder 2005] is a treasure hunt game for Assisted GPS phones. Swordfish (http://www.blisterent.com/swordfish/) simulates a fishing experience within the city: Thanks to a fish-finder the player can see where the nearest shoal of virtual fish is located in relation to his/her current position. It should be noted that location-based games, where the users need to interact with geo-located virtual objects, may require a high location precision [Pelanis et al. 2006]. Therefore, appropriate positioning techniques (see Section 3.1.1) and location update policies (see Section 4.1.2) should be considered.

To conclude this section, two other sample services will be mentioned. *Heresay* (part of *Herecast*, see http://www.herecast.com/) features *location-based message boards*. This service lets the user post messages that can be seen by other people in

the same building. Along the same lines, *StreetHive* (*Friend Finder* application at `http://www.wavemarket.com/`) allows the user to leave notes and photos attached to the places he/she has been to.

3.2 Location-Dependent Queries

Location-dependent queries, called *location-based queries* in works such as [Huang and Jensen 2004], are a fundamental building block of LBS. The importance of queries with location constraints and the challenges that they pose were already highlighted over a decade ago by T. Imielinski and B. Nath in a few papers as part of their *DATAMAN* project [Imielinski and Nath 1992; Acharya et al. 1995a; Imielinski 1996b]. In general, two types of queries can be distinguished in a mobile environment:

— A *traditional query*, called *non location-related query* in [Seydim et al. 2001; Marsit et al. 2005], is a query whose answer does not depend on locations or retrieve location data; for example, "retrieve the names of the employees of a company".

— A *location-dependent query*, on the contrary, is a query whose answer depends on the locations of objects; that is, the location is an attribute that determines whether an object is part of the answer or not. For example, "find hotels within 5 miles" depends on the location of the user that poses the query, and "find taxi cabs near Peter" depends on both the location of the taxi cabs and the location of Peter; in works such as [Seydim et al. 2001], the term location-dependent query refers to queries that depend *only* on "the query issue location" (i.e., the location of the mobile user), so this last example is not admitted by those definitions. Some works, such as [Ilarri et al. 2006a], assume the implicit desire to retrieve the current locations of the objects in the answer (e.g., to show them in the right locations on a map); on the contrary, other works only retrieve location data for a so-called *location query* [Seydim et al. 2001; Xu et al. 2008] (called *position query* in [Becker and Dürr 2005]), such as "where is person A?". Finally, in a location-dependent query, the region containing the interesting objects can be implicit or explicit. In the example about hotels the implicit region is a five-mile circle around the user, and in the example about taxi cabs it depends on the locations of the taxis, on the location of Peter, and on how many taxis must be retrieved. On the contrary, "retrieve the taxi cabs in Chicago" is a query with an explicit region, called *location-aware query* in [Seydim et al. 2001]. In [Marsit et al. 2005] the term *moving object database queries* is used to mean location-dependent queries about moving objects.

In the following, location-dependent queries are classified according to their purpose. Then, other classification criteria are examined.

3.2.1 *Types of Location-Dependent Queries According to their Purpose.* In the specialized literature, different types of location-dependent queries have attracted attention (see Table I), among which the following could be highlighted (some examples will be presented in relation to the sample scenario presented in Figure 1 on page 15):

— *Range queries* (e.g., see [Xu and Wolfson 2003; Trajcevski et al. 2004b; Yu et al. 2006]). They retrieve the objects located within a certain range/region. Range queries can be *static range queries* or *moving/mobile range queries*, depending on

Table I. Types of location-dependent queries according to their purpose

Type of query	Some variants	Sample references
Range queries	Static/moving range queries Window queries Within-distance queries: DD, DS, SD, SS Inverse range queries Range aggregate queries	[Xu and Wolfson 2003; Tao et al. 2003d; Trajcevski and Scheuermann 2003; Trajcevski et al. 2004b; Yu et al. 2006]
(k)NN queries	Nearest neighbor (NN) queries (k=1) Constrained NN queries Aggregate NN (ANN) queries, group NN queries Distance [semi-]joins, iceberg dist. semi-joins, etc. $(1+\epsilon)$/e-approximate (vs. exact) NN queries, etc. [k] closest pairs Nearest *surrounder* (NS) queries In-route NN queries Reverse [k]NN (R[k]NN) queries Visible [k]NN (V[k]NN) queries	[Roussopoulos et al. 1995; Corral et al. 2000; Ferhatosmanoglu et al. 2001; Tao et al. 2002; Chon et al. 2003; Mouratidis et al. 2005b; Iwerks et al. 2006; Benetis et al. 2006; Wu et al. 2008]
Navigation	Trip-planning queries	[Hung et al. 2003; Li et al. 2005]
Point queries	Where_at(t, oid), When_at(x, y, oid), When_closest_to(x, y)	[Trajcevski and Scheuermann 2003; Trajcevski et al. 2002b]
Others	[Static] n-body constraints, [Effective] density queries, etc.	[Xu and Jacobsen 2007]

whether the interesting region is fixed or moves. Thus, row 1 in Table II corresponds to a moving range query that retrieves coffee shops within an semicircular area S defined by the location of person $p2$; this query retrieves initially only the coffee shop $s1$; however, query region S moves with $p2$, so eventually $r5$ and $r6$ will be retrieved if $p2$ continues moving in the same direction (indicated by an arrow).

Range queries are also called *window queries* when the range is a rectangular window (e.g., in [Tao et al. 2003d]). Thus, row 2 in Table II represents a static window query that retrieves objects of class *person* within the rectangular area $R2$. Similarly, *within-distance queries* [Trajcevski and Scheuermann 2003] can be considered as a variant of range queries where the range is a circle. A within-distance query returns the set of objects that are within a certain distance of a specified object. In other words, it is a range query with a circular range centered on a certain object; row 3 in Table II shows an example. Depending on whether this object and/or the objects that are the *target* of the query move or not, four variants are considered: *DD* (dynamic query, dynamic data), *DS* (dynamic query, static data), *SD* (static query, dynamic data), and *SS* (static query, static data); it should be noted that this classification could also be applied to other types of location-dependent queries.

— *Nearest neighbor* (*NN*) *queries* (e.g., see [Tao et al. 2002]). They retrieve the object of a certain class which is the closest to a certain object or location; for example, row 4 in Table II corresponds to a query that retrieves the car closest to car $c0$. When k objects must be retrieved instead of just the nearest one, they are called *kNN queries* (e.g., row 5 in Table II). In [Chon et al. 2003], static and dynamic nearest neighbor queries are distinguished: Static NN queries are executed against a static database whereas in dynamic NN queries the target objects can move. Related to kNN queries, a *reverse kNN query* retrieves objects that have a specified object/location among their k nearest neighbors (e.g., see [Benetis et al. 2006; Wu et al. 2008]); row 6 in Table II shows an example that retrieves the reverse 2-nearest coffee shops for coffee shop $s1$. In the context of spatial databases, *constrained NN queries* are defined in [Ferhatosmanoglu et al. 2001] as NN queries with a range constraint for the objects retrieved; as an example, in row 7 in Table II the nearest objects retrieved must be within the rectangular area $R1$. Finally, a *k closest pairs*

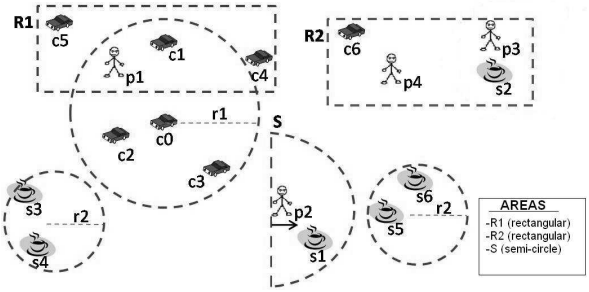

Fig. 1. Sample scenario for location-dependent queries

Table II. Sample results for several types of location-dependent queries

Type	Parameters	Mobility		Answer
		Query	Target	
Range	area S (person p2), class: coffeeShop	Dynamic	Static	{s1}
Static window	area R2, class: person	Static	Dynamic	{p3, p4}
Within-distance	car c0, radius r1, class: car	Dynamic	Dynamic	{c1, c2, c3}
NN	car c0, class: car	Dynamic	Dynamic	{c2}
kNN	k=2, car c0, class: car	Dynamic	Dynamic	{c2, c3}
Reverse kNN	k=2, coffeeShop s1, class: coffeeShop	Static	Static	{s3, s4, s5, s6}
Constrained kNN	k=2, area R1, car c0, class: car	Dynamic	Dynamic	{c1, c4}
k closest pair	k=2, class1, class2: coffeeShop	N/A	Static	{(s3, s4), (s5, s6)}
n-body constraint	(n=2) radius r2, shops: s3 and s4	N/A	Static	true
n-body constraint	(n=2) radius r2, shops: s5 and s6	N/A	Static	true
n-body constraint	(n=3) radius r2, shops: s1, s5 and s6	N/A	Static	false

query [Corral et al. 2000] retrieves the pairs of objects (from two datasets) with the k smallest distances; thus, row 8 in Table II shows an example that retrieves the 2 closest pairs of coffee shops.

— *n-body constraints* [Xu and Jacobsen 2007]. These are location constraints satisfied by sets of n objects that are closer/further than a certain *alerting distance* d from each other, depending on whether a $<$ or $>$ comparator is involved. Thus, with $<$, the constraint is satisfied if the objects can be enclosed all within a circle of radius d. Rows 9-11 in Table II correspond to several examples: Both the pair <s3, s4> and the pair <s5, s6> satisfy a 2-body constraint with alerting distance $r2$; however, the triple <s1, s5, s6> does not satisfy the corresponding 3-body constraint. If the constraint is specified in relation to a given point of demarcation in the environment, it is called *n-body static constraint* in [Xu and Jacobsen 2007]: The n objects returned by a within-distance query with radius r and a static query point (explained before) satisfy the corresponding n-body static constraint, with alerting distance r, regarding such query point.

— *Navigation queries* retrieve the best path for a mobile client to get to his/her destination, based on the underlying road network and the current traffic conditions [Hung et al. 2003; Li et al. 2005].

Related to the above classification, location-dependent queries are distinguished in [Huang and Jensen 2004] based on different types of predicates: *one-to-many* queries (a predicate is applied to many objects; e.g., nearest neighbor queries) vs. *many-to-many* queries (a predicate is checked for every object in relation to every other object; e.g., –constrained– spatial or spatio-temporal joins [Tao et al. 2003c;

Table III. Criteria to classify location-dependent queries

Criteria	Classification	Sample references
Evaluation time and past states	Instantaneous	[Sistla et al. 1997a; 1997b]
	Snapshot	[Porkaew 2000]
	One-time	[Babu and Widom 2001; Huang and Jensen 2004]
	Continuous	[Terry et al. 1992; Babu and Widom 2001] [Lazaridis et al. 2002; Marsit et al. 2005]
	Persistent	[Sistla et al. 1997a; 1997b]
Time reference	Present	[Chen et al. 2003]
	Past	[Frentzos et al. 2007]
	Future/predictive/prediction	[Sistla et al. 1997a]/[Seydim and Dunham 2002]
Semantics for uncertainty	May/must	[Sistla et al. 1997b]
	Possibly/definitely	[Wolfson et al. 2001; Trajcevski et al. 2004b]
	Possibly/surely/probably	[Moreira et al. 2000]
Semantics for time interval	Sometimes/eventually	[Sistla et al. 1997b; Trajcevski et al. 2005a]
	Always	
	"Hybrid"	
Purpose/Goal	See Table I	
Dynamics	DD, DS, SD, SS (see Section 3.2.1)	
Others	Predefined, ad hoc	[Huang and Jensen 2004]
	One-to-many, many-to-many	
	Approximate (vs. exact) queries	[Xu et al. 2008]

Sun et al. 2006] or closest pair queries [Corral et al. 2000]), queries that involve topological/directional/metric predicates, etc. Other types of queries can also be useful in a mobile environment. For example, several location-related operators, such as *straight ahead*, are considered in [Seydim and Dunham 2002]. In the specific context of spatial databases, various operators have been defined that could also be useful in mobile environments; for example, some spatial operators specified by the *OpenGIS Consortium* are presented in [Clementini and Felice 2000].

While there are many variants of location-dependent queries, two types have been a major focus of attention in research: 1) static/moving range queries on static/moving objects, and 2) k nearest neighbor queries. In this survey, some works related to the first type of queries appear in Sections 5, 6, and 7, and Section 8 is devoted to nearest neighbor queries.

3.2.2 *Other Criteria to Classify Location-Dependent Queries.* Besides the purpose of the query, location-dependent queries can be classified according to a variety of criteria (see Table III), such as:

— *According to the evaluation time and past states*, the following types of queries can be distinguished [Sistla et al. 1997a; 1997b]: 1) *snapshot queries*, also called *instantaneous queries*, for which the answer is computed only once and transmitted immediately to the user; 2) *continuous queries*, which are reevaluated continuously until they are canceled by the user; and 3) *persistent queries*, which can be defined as continuous queries that need to consider not only the current state but also the past states of the moving objects since a specific starting time instant. For example, a user in a car can launch "retrieve the motels within 5 miles of my position" as a continuous query, so that the answer to this query is automatically refreshed until he/she finds a satisfactory motel. An example of persistent query is "retrieve the objects whose speed duplicates within 10 seconds". These are generally accepted definitions, although some works use a different terminology. For example, in [Benetis et al. 2006] persistent queries are defined as "queries with answer sets that are maintained under updates", that is, continuous queries that take into account the objects' location updates, and not only the estimated objects' locations, for reevaluation.

— *According to whether they refer to past, present, or future states*, the following types of queries can be distinguished: 1) *present queries*, which concern the current locations of objects; 2) *past queries*, which refer to a time instant or temporal interval in the past; and 3) *predictive queries*, which encompass some future time instant (e.g., "display the motels that I will reach within 3 minutes"). Some works also distinguish between *timestamp queries* (called *time-slice queries* in [Saltenis et al. 2000]) and *interval queries*, depending on whether they refer to a single time instant or a time interval (e.g., [Tao and Papadias 2005]), and *current/now* queries, which are interval queries where the starting time of the interval is given by the current time instant (e.g., in [Choi and Chung 2002; Tao et al. 2003d]). For predictive queries, works focusing on prediction issues for moving objects are of particular relevance; for example, see [Xu and Wolfson 2003; Karimi and Liu 2003; Tao et al. 2004a; Yavas et al. 2005; Civilis et al. 2005].

— *According to the way that uncertain locations must be handled*, the following types of queries can be distinguished: 1) queries with *may* semantics, where all the objects that may be in the answer are returned; and 2) queries with *must* semantics, where only the objects that are in the answer for sure (even considering the greatest possible error regarding their known locations) are returned. In systems where the queries are expressed using a query language, the same query can be specified as a *may* or *must* query by using an appropriate query modifier (e.g., see [Sistla et al. 1997b]). In [Wolfson et al. 2001; Trajcevski et al. 2004b], the modifiers proposed are called *possibly* and *definitely*. In [Moreira et al. 2000], three semantics are distinguished: *possibly* (equivalent to *may*), *surely* (equivalent to *must/definitely*), and *probably* (where locations are estimated and so the results can contain false hits and also be incomplete). Section 4.1.4 indicates several works where the uncertainty of locations and *probabilistic* versions of location-dependent queries are considered.

— *According to their semantics regarding a certain time interval*, the following types of queries can be distinguished: 1) queries with *sometimes* semantics, where all the objects that satisfy the query conditions during some amount of time are retrieved as part of the answer; 2) queries with *always* semantics, where the retrieved objects must satisfy the conditions during the whole time interval; and even 3) queries with hybrid semantics, where the retrieved objects must satisfy the constraints at least during a certain percentage of the time interval. As in the case of *may* and *must* queries, a modifier can be used to express the different semantics.

— *According to the dynamics of the query*, that is, depending on whether the query and/or the target objects are static or mobile. This was already discussed in Section 3.2.1 in relation to *within-distance queries*.

Location-dependent queries can also be classified according to other criteria. For example, in [Huang and Jensen 2004] the time instant at which a location-dependent query is registered is considered to distinguish between *predefined* queries (if the query is present before the data streams it relies on start) and *ad hoc* queries (if the query is registered after at least one of its streams has started). Some authors tackle the problem of approximate location-dependent query processing, and so they consider approximate location-dependent queries (e.g., approximate location queries in [Xu et al. 2008]). Similarly, aggregate location-dependent queries (see Section 9) and probabilistic queries (see Section 4.1.4) are also important in this

context. It should be noted that the criteria presented above, and summarized in Table III, are in principle orthogonal, but only to a limited extent; thus, for example, a navigation query with *may* or *must* semantics makes no sense, and the same can be said about considering past queries as continuous queries.

Before concluding this section, it should be noted that the category under which a query is classified according to the previous criteria will obviously have an impact on the query processing. For example, different access methods are required for past and future queries (see Section 4.1.3), considering query semantics for uncertainty usually implies the computation of satisfaction probabilities (see Section 4.1.4), and the semantics regarding the query time interval are considered in works such as [Trajcevski et al. 2004b] to determine the trajectory-based computational geometry techniques that must be applied to process the query. As another example, [Trajcevski and Scheuermann 2004] includes a brief discussion about how the dynamics of the query could be exploited for optimization.

4. DATA TECHNOLOGIES FOR MOVING OBJECTS

In this section, two technologies that are used in the context of moving objects are described: moving object databases and data streams. As it was mentioned in the introduction of this article, the boundary between these two technologies, considering the existing approaches to handle moving objects, is not always clear. It should be also emphasized that some works do not explicitly develop or use a full-fledge moving object database, but they consider the features of moving objects to propose data management mechanisms (e.g., index structures for moving objects) appropriate for their query processing needs. Some other aspects of data management related to this context are also briefly indicated at the end of this section.

4.1 Moving Object Databases

Moving object databases (*MODs*) are the extension of database technology to support the representation of moving objects in databases (e.g., see [Theodoridis 2003; Wolfson and Mena 2004; Güting and Schneider 2005]). In contrast to earlier work on spatio-temporal databases [Koubarakis et al. 2003], that supported only discrete changes, objects in a MOD may change their locations continuously. Research on MODs has attracted a lot of attention in the recent years, focusing on several aspects: access methods (e.g., [Saltenis and Jensen 2002; Jensen 2002]), probabilistic queries and uncertainty (e.g., [Pfoser and Jensen 1999; Trajcevski et al. 2004b; Cheng et al. 2004a; Yu and Kim 2006; Prager 2007]), query languages and models (e.g., [Sistla et al. 1997a]), etc. Some of these issues are briefly commented on in the following.

4.1.1 *Models for Moving Objects in Databases.* Modeling moving objects implies the representation of their continuous movements. Traditional *Database Management Systems* (*DBMSs*) can naturally use an approach where the movements of the objects are represented by sampled locations [Nascimento et al. 1999]. However, this technique is not considered appropriate because there would be a high update rate needed to maintain the stored locations of moving objects up-to-date. So, motion functions are proposed instead, which significantly reduce the number of

required updates [Sistla et al. 1997a; Civilis et al. 2004; Jensen and Pakalnis 2007].

One of the most popular and first models proposed is the *MOST model* [Sistla et al. 1997a], which is part of the DOMINO project (see Section 6.1). MOST represents a moving object as a function of its location and velocity. Thus, one of its main contributions is the introduction of the concept of *dynamic attributes*, which are attributes whose values change (according to a function) even in the absence of explicit database updates. Thus, the location of a moving object is considered as a dynamic attribute, allowing the modeling of the current and near-future positions of moving objects.

There are other alternative models. For example, [Güting et al. 2000] defines an algebra with data types such as moving points, moving lines, and moving regions. There are also models based on constraint databases [Su et al. 2001; Mokhtar and Su 2005], models that consider objects moving in road networks [Vazirgiannis and Wolfson 2001; Ding and Güting 2004; Güting et al. 2006] (issues regarding the modeling of road networks are dealt with in works such as [Hage et al. 2003]), or the *SV* model [Chon et al. 2001a].

Finally, it should be mentioned that the different query languages proposed in the literature are tightly related to the corresponding data model. Thus, new spatio-temporal operators and functions are proposed for querying in [Güting et al. 2000], a constrained-based query language (*TQ*) for [Mokhtar and Su 2005], FTL for the MOST model [Sistla et al. 1997a], etc.

4.1.2 *Location Update Policies.* In the context of moving object databases, most works consider that the database updates are generated by the moving objects [Wolfson et al. 1999a]. Thus, a location update policy must be defined for the moving objects in order to keep their locations up-to-date in the database. A popular approach is an update policy based on the relative costs of communication, deviation, and uncertainty [Wolfson et al. 1999a]. The *deviation* of a moving object at a particular time instant is the distance between its actual location and its location in the database at that time instant. On the other hand, the *uncertainty* is the size of the area in which the object can possibly be. Most update policies set a maximum uncertainty for each moving object, such that the object will communicate its updated location whenever the deviation exceeds the allowed threshold. Several policies have been proposed in the literature, that try to strike the balance between imprecision and cost. As mentioned previously in Section 2.1, this is similar to the location management problem existing in the context of mobile networks, except for the finer location granularity required. In the following, some location updates policies that have been proposed in the literature are indicated:

—*Periodic location updates* [Trajcevski et al. 2005b]. A moving object simply updates its location periodically.

—*Periodic location updates with velocity information* [Trajcevski et al. 2005b]. A moving object communicates periodically its location and speed vector. The speed vector allows the estimation of the location of the moving object at time instants different from the update time.

—*Speed dead-reckoning* (*sdr*) [Wolfson et al. 1999b], called also *plain dead-reckoning* (*pdr*) in works such as [Lam et al. 2001]. An uncertainty value is defined at the

beginning of the trip and stays constant. When the threshold is violated, the moving object reports its current speed and location.

—*Hybrid dead-reckoning* [Gowrisankar and Nittel 2002]. It not only considers a distance threshold but also an angular threshold. Therefore, an object will update its location and speed whenever either its predicted location or direction deviates significantly.

—*Adaptive dead-reckoning* (*adr*) [Wolfson et al. 1999b]. It extends *sdr* by considering a dynamic threshold, which is provided with each update based on the uncertainty, deviation, and update costs.

—*Disconnection detecting dead-reckoning* (*dtdr*) [Wolfson et al. 1999b]. With this policy, the update threshold decreases continuously as the time elapsed since the last location update increases, to detect objects that may have disconnected.

—*Trajectory updates* [Trajcevski et al. 2005b], also called *segment-based tracking* in [Civilis et al. 2005]. Given the object's initial location and the set of points that it intends to visit, *map matching* [Yin and Wolfson 2004] is used to determine the road where the object is moving. With this policy, an object would just communicate changes in its travel plan.

As opposed to the previous approaches, works such as [Lam et al. 2001; Prabhakar et al. 2002; Hu et al. 2005a; Hu and Lee 2005; Tao et al. 2005b; Xu and Jacobsen 2007; Cheng et al. 2007] propose a *query-driven* update policy. With this type of policy, the objects update their locations depending on the existing queries: A location update is issued only if it may affect the results of an active query. Some of these works are reviewed in Section 5. Consistency issues may arise during the query processing in the presence of updates. However, this has hardly been considered in the literature, except for the *position locking* mechanism proposed in [Budiarto et al. 1997].

4.1.3 *Indexing Moving Objects in Databases.* One of the important issues studied in the context of spatio-temporal databases, and more specifically moving object databases, is that of indexing moving objects[4]. In a context where the locations of the objects are constantly changing, it is of paramount importance to devise an indexing mechanism (or *spatial access method* [Gaede and Günther 1998]) that incurs a low update overhead and, at the same time, allow an efficient search of the objects that satisfy certain properties regarding their locations (e.g., range searching).

R-trees [Guttman 1984] are among the most popular spatial indexing mechanisms and many variants have been suggested, such as the *R*-tree* [Beckmann et al. 1990] and the *X-tree* [Berchtold et al. 1996]. Specifically, many efforts have been devoted towards reducing the burden of updates. Thus, the *Lazy Update R-tree* (*LUR-tree*) [Kwon et al. 2002] is based on a continuous model of moving objects (i.e., it considers predefined moving patterns like the moving objects' speed vectors) to minimize the number of updates on the R-tree. In [Lee et al. 2003], the R-tree is enhanced with a bottom-up update method to support frequent updates (the resulting index structure is called the *FUR-tree* in [Xiong et al. 2006]). The *RUM-*

[4]Some works, such as [Chon et al. 2002a], distinguish between indexing (bottom-up approach, which clusters together nearby objects) and partitioning (which *a priori* partitions the data space).

tree [Xiong and Aref 2006] proposes the use of an *update memo* to perform lazy deletions on the R-tree, which means that several versions of the same object can coexist. Finally, [Pfoser et al. 2000] presents two extensions of the R-tree called the *STR-tree* and the *TB-tree*. The former considers the trade-off between *spatial closeness* (tight MBRs) and *trajectory preservation* (storing segments of the same trajectory together), whereas the latter aims at strict trajectory preservation.

Some moving object databases store past and future information of moving objects, such as their past locations and expected trajectories. Therefore, the term *spatio-temporal access methods* [Theodoridis et al. 1998] is used to refer to indexing mechanisms that take into account both dimensions, space and time, to retrieve the objects of interest. In other words, it refers to auxiliary structures to support spatio-temporal queries efficiently. Works such as [Tayeb et al. 1998; Kollios et al. 1999b; Agarwal et al. 2003] present design specifications for spatio-temporal access methods. In general, index structures for moving objects can be classified according to a variety of criteria, such as:

—*The temporal considerations* [Mokbel et al. 2003; Pelanis et al. 2006]. Thus, there are indexing methods for historical spatio-temporal data, such as the *H-R-tree* [Nascimento and Silva 1998], the *HR+-tree* [Tao and Papadias 2001a], the *MV3R-tree* [Tao and Papadias 2001b], the *STR-tree* [Pfoser et al. 2000], the *TB-tree* [Pfoser et al. 2000], or the approach presented in [Kollios et al. 2001] for linearly-moving objects and extended in [Hadjieleftheriou et al. 2006] to consider general moving functions. Some structures focus on indexing the current locations of moving objects; for example, hashing [Song and Roussopoulos 2001a], the *LUR-tree* [Kwon et al. 2002], the *LUGrid* [Xiong et al. 2006], the *CT-R-tree* [Cheng et al. 2005], the R^R*-tree* [Biveinis et al. 2007], and the *RP-tree* [Lin et al. 2006]. For indexing the current and future positions of moving objects, some methods are the *TPR-tree* [Saltenis et al. 2000], the *TPR*-tree* [Tao et al. 2003b], the *VCI* (see Section 5.3), the *STP-tree* [Tao et al. 2004a], *STRIPES* [Patel et al. 2004], the B^x*-tree* [Jensen et al. 2004], the B^{dual}*-tree* [Yiu et al. 2008], the space-time grid [Chon et al. 2001b], the *STAR-tree* [Procopiuc et al. 2002], or the proposal in [Aggarwal and Agrawal 2003] for non-linear trajectories. Finally, in [Lin et al. 2005] and [Pelanis et al. 2006], the BB^x*-index* and the R^{PPF}*-tree*, respectively, are proposed as integrated approaches to index the past, present and future locations of moving objects.

—*The type of model*. [Chen et al. 2003] distinguishes two different models:

(1) In the *continuous model*, moving objects are modeled as moving points that start from a specific location with a constant speed vector. The *Time-Parametrized R-tree* (*TPR-tree*) [Saltenis et al. 2000] and, in general, all those methods devised for predictive queries belong in this category.

(2) In the *discrete model*, the locations of the moving objects are stored in a database as tuples with a location and timestamp. For example, the hashing-based approach [Song and Roussopoulos 2001a] and the LUR-tree [Kwon et al. 2002] previously mentioned fall in this category.

—*The space representation used*. Some approaches, such as [Kollios et al. 1999b; Agarwal et al. 2003; Patel et al. 2004; Kollios et al. 2005; Yiu et al. 2008], advocate indexing in the dual space instead of using a primal representation. An

Table IV. Main index structures for moving objects

Index	Basic idea
Index current locations	
LUR-tree	Update the structure only when an object leaves its MBR
FUR-tree	Bottom-up update strategy
LUGrid	Lazy insertion/deletion with a grid index
RUM-tree	Delete outdated entries lazily using an *update memo*
CT-R-tree	Minimize updates from one MBR to another, by identifying *quasi-static regions* (qs-regions)
R^R-tree	Buffer and group operations in main memory to reduce disk I/O
Index past locations	
H-R-tree	"Temporalized" R-tree (several logical R-trees) without duplicating unmodified nodes
HR+-tree	Store entries of different versions (times) in the same node, to reduce *version redundancy*
MV3R-tree	Use a multi-version R-tree (MVR-tree) and a small 3D R-tree (timestamp/interval queries)
STR-tree	Try to keep segments of the same trajectory close
TB-tree	Strict trajectory preserving (leaves store "trajectory bundles")
Index present and future locations	
TPR-tree	Consider, besides MBRs, velocity-bounded rectangles (VBRs)
TPR*-tree	TPR-tree with new insertion/deletion algorithms that consider the mobility of the objects
VCI	Consider, besides MBRs, the maximum speed of objects inside
STP-tree	Extend the TPR-tree to handle polynomial motion functions
STRIPES	Index trajectories in dual-space using a regular hierarchical grid
B^x-tree	B^+-tree on space filling curve's values of objects, index partitioned based on update time
B^{dual}-tree	B^+-tree on Hilbert values, *MORs* (moving rectangles) to consider objects' velocities
Index past, present and future locations	
BB^x-index	Extend the B^x-tree by storing times of deletions and updates, to support past positions
R^{PPF}-tree	Combine ideas of the TPR-tree with *partial persistency*

interesting recent study comparing primal and dual methodologies for predictive queries appears in [Tao and Xiao 2008], indicating that the superiority of the primal or dual techniques depends on the update rate: Primal techniques achieve better query performance when the data *agility* is sufficiently high, and vice versa. This study also emphasizes that it should not be concluded that primal techniques exhibit worse update performance only because existing methods do; moreover, it is important to keep in mind that dual indexing methods require at least one update from every object within a certain time period.

—*The method used*. For example, the survey on multidimensional access methods presented in [Gaede and Günther 1998] distinguishes methods based on transformations, overlapping regions, clipping, multiple layers, etc.

Specific indexing mechanisms have been proposed for objects moving in road networks (e.g., [Frentzos 2003; Pfoser and Jensen 2003; Almeida and Güting 2005; Chang et al. 2006]), for objects with predefined trajectories (e.g., [Chon et al. 2003]), or for broadcast environments (e.g., see [Zheng et al. 2003b; Lee 2007]). Similarly, some works study index structures that are appropriate for specific types of queries (e.g., continuous range queries in [Yu et al. 2006] and kNN queries in [Lin et al. 2006]) or for computing aggregations (see Section 9). Finally, it is interesting to mention that some index structures, such as the *R^R-tree* (mentioned before) and the *U-tree* (see Section 4.1.4), take the inaccuracy of the objects' locations into account.

It is not a goal of this article to provide an exhaustive review of existing indexing methods for moving objects, on which there is a huge amount of literature available. Instead, the interested reader is referred to works such as [Gaede and Günther 1998; Nascimento et al. 1999; Saltenis and Jensen 2002; Jensen 2002; Hadjieleftheriou et al. 2002; Mokbel et al. 2003; Almeida and Güting 2005] and the other references included in this section. According to [Tao and Xiao 2008], more studies are needed to evaluate index structures in scenarios with concurrent updates

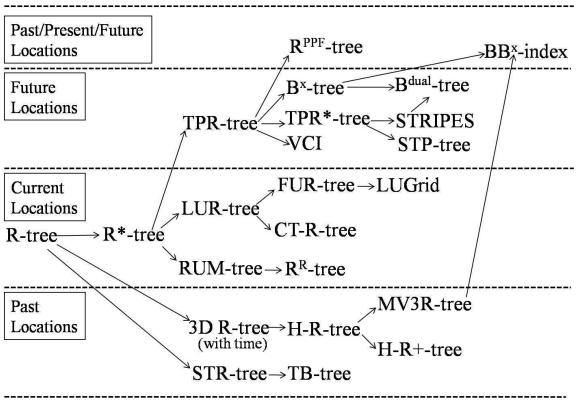

Fig. 2. Evolution of the main index structures for moving objects

and queries; [Gaede and Günther 1998; Jensen et al. 2004] describe some access methods that propose mechanisms to support concurrent access. A brief summary of some of the index structures mentioned along this section appears in Table IV. Figure 2 shows the flow of improvement among the different index structures in the table; for example, the R*-tree improves the R-tree, the TPR-tree is an improvement over the R*-tree to index future locations, and both the STP-tree and STRIPES improve the TPR*-tree according to the experimental results presented in [Tao et al. 2004a] and [Patel et al. 2004]. For more details, the reader should look at the corresponding papers.

4.1.4 *Dealing with Uncertainty.* The location of a moving object is inherently uncertain for several reasons, such as: 1) delayed location updates, due to the use of different location update policies (see Section 4.1.2) and the existence of network communication delays; 2) measurement imprecision (several positioning mechanisms are enumerated in Section 3.1.1); or 3) privacy issues (e.g., spatial cloaking, see Section 3.1.2). Because of this uncertainty, for example, the answer to a query may be based on out-of-date locations. Although the answer to the query will be updated automatically if a continuous query is considered, the uncertainty of the answer (due to delayed updates or other factors) is unavoidable. Therefore, different works have proposed techniques to deal with this uncertainty.

Several works considered modeling this uncertainty on databases; for example, *may* and *must* semantics (see Section 3.2.2) are introduced in [Sistla et al. 1997b], [Wolfson et al. 1999b; Wolfson et al. 1999a] evaluate the uncertainty cost to propose several location update policies (see Section 4.1.2 and Section 6.1), [Pfoser and Jensen 1999] quantifies the measurement and sampling errors and advocates a classical filter-and-refinement approach to process queries on uncertain data, [Trajcevski et al. 2002b; Trajcevski et al. 2004b] propose modeling uncertain trajectories as three-dimensional cylinders (each line segment in the trajectory is enlarged according to an uncertainty threshold) and also introduce several spatio-temporal operators and algorithms to query these trajectories (modifiers *definitely* vs. *possibly* and *sometimes* vs. *always* can be used to determine different variants of spatio-temporal range queries defined with a query region and a time interval),

and [Almeida and Güting 2005] considers uncertain trajectories for objects moving within road networks. There are also works that focus on query processing approaches that take uncertainty into account, as described in the rest of this section.

General *probabilistic queries* (not in the context of moving objects) are introduced in [Cheng et al. 2003a], which proposes a taxonomy of queries depending on whether they retrieve entities or values and on whether they require aggregation. Works that focus on probabilistic queries on moving objects advocate estimating the locations of the objects by modeling the uncertainty through a probability density function (e.g., uniform, Gaussian, Zipf, Poisson, or represented with a histogram), such that the probability that an object is within a certain region (zero outside the *uncertainty region*) can be computed. The computation of these probabilities is expensive, as it requires the evaluation of an integral that in many cases must be solved numerically [Cheng et al. 2004a; Tao et al. 2005a; Tao et al. 2007]. Therefore, approaches with a filter step and a refinement step are proposed, and pruning the search space during filtering is a key issue to reduce the query processing overhead.

Following the terminology presented in [Dai et al. 2005], which focuses on spatial queries, *thresholding probabilistic queries* (called *Probabilistic Threshold Queries*, *PTQ*, in [Cheng et al. 2004b]) obtain the objects that satisfy the query conditions with a probability higher than a given *probability threshold* (e.g., see [Tao et al. 2005a; Tao et al. 2007]), and *ranking probabilistic queries* order the results of a query according to the probability that each object satisfies the query predicate (e.g., see [Wolfson et al. 1999b; Cheng et al. 2004a]); the computed probabilities can be returned as part of the answer. Some important proposals are highlighted:

— Pioneering works for *static probabilistic range queries* are presented in [Wolfson et al. 1999b; Pfoser and Jensen 1999]. [Wolfson et al. 1999b] describes an approach for timestamp predictive queries and considering objects moving in straight line routes and under the assumption of normal probability density functions. [Pfoser and Jensen 1999] shows that the location of an object between two location samplings lies within an error ellipse and proposes a filter-and-refinement approach to process predictive interval queries.

— [Tao et al. 2005a; Tao et al. 2007] propose different approaches for *static probabilistic range queries*, called *probabilistic range search* (*prob-range*) in [Tao et al. 2005a] and *probabilistic thresholding range retrieval* in [Tao et al. 2007]. Thus, [Tao et al. 2005a] presents a filter-and-refinement approach with the use of the *U-tree*, an R*-tree-like index structure that stores information that helps pruning the search of objects that qualify the query. In particular, *conservative functional boxes* (*CFBs*) are stored, which are compact representations of the so-called *probabilistically constrained regions* (*PCRs*), termed *probabilistically constrained rectangles* in [Tao et al. 2007]. The work in [Tao et al. 2007] extends [Tao et al. 2005a] by refining the pruning and validating heuristics and considering also the so-called *probabilistic thresholding fuzzy range queries*. These are queries whose query region may be uncertain, which occurs for example when the region is defined based on the location of a moving object, like in the case of *within-distance queries* (see Section 3.2.1). Thus, this last work allows the processing of moving range queries.

— In [Cheng et al. 2004a], which extends a short preliminary version presented in [Cheng et al. 2003b], both *static probabilistic range queries* (*PRQs*) and *probabilis-*

tic nearest neighbor queries regarding a static query point (*PNNQ*), are considered. The algorithm for evaluating PQRs basically integrates the time-dependent probability density function of the objects over the overlapping area of the uncertainty region and the query region. PNQs are more challenging, as the probability of each object cannot be computed independently of the others. Therefore, PNNQs is the main focus of this work, proposing an algorithm with four phases (projection, pruning, bounding, and evaluation) and the use of a VCI index (see Section 5.3) to prune the search space. A generic model of uncertainty is considered, which is also applied to the cases of line-segment uncertainty (the uncertainty is given by a line segment) and free-moving uncertainty (the uncertainty area is a circle).

— [Kriegel et al. 2007] considers *probabilistic nearest neighbor queries*. This work advocates the use of a discrete probability density function obtained through Monte-Carlo sampling, approximating these samples by clustering, and storing the MBRs of the clustered representations in an R-tree like index, instead of integrating directly a continuous probability density function.

— *Probabilistic nearest neighbor queries* are also considered in [Cheng et al. 2008]. For efficiency, the authors propose deciding whether an object satisfies a query or not without obtaining the exact probability value. *Probabilistic verifiers* are used, which compute probability bounds instead. A query processed according to this approach is called *Constrained Probabilistic Nearest Neighbor Query* (*C-PNN*) and has two parameters: a probability threshold and a tolerance value.

The works described above focus on probabilistic range queries and/or nearest neighbor queries. Other proposals consider other types of queries. For example, [Ni et al. 2003] considers joins in the context of spatial databases. To conclude this section, it is also interesting to mention the work presented in [Dai et al. 2005], which deals with range and nearest neighbor spatial queries. As opposed to the spirit of the works described in this section, it considers *existential uncertainty* (instead of location uncertainty). Thus, this work belongs to the field of *probabilistic databases* [Dalvi and Suciu 2004], where data records are associated with an existential probability. This type of uncertainty can be useful, for example, when data are obtained automatically from images, as in that case some non-existing elements could be wrongly identified in the data extraction.

4.2 Data Stream Technology

This section reviews some key aspects of data streams. First, the applicability and challenges of data streams are explained in general. Then, data streams are considered specifically in the context of moving objects.

4.2.1 *Data Streams: Applicability and Challenges.* One of the first works on data streams [Hartzman and Watters 1990] defines a data stream as "a sequence of characters or bits that is too large to be viewed in its entirety". Along the same lines, a data stream is defined in [Guha et al. 2000] as "an ordered sequence of points that can be read only once or a small number of times". Finally, in [Patroumpas and Sellis 2004] a data stream is defined as "an ordered sequences of tuple values evolving in time".

As opposed to traditional databases, that store static data sets, data stream

technology focuses on continuously changing data that arrive in large amounts; for example, a GPS receiver and a temperature sensor are continuously measuring the location and temperature, respectively. Thus, data streams are potentially infinite, dynamically changing, and must be processed in real-time [Babcock et al. 2002; Golab and Özsu 2003; Cai et al. 2004b]. The following challenges can be highlighted regarding the management of data streams:

— *Impossibility of processing an entire data stream.* As processing the whole history of the stream is unfeasible, it is necessary to limit the scope of interest through the concept of *sliding window* [Babcock et al. 2002; Ghanem et al. 2006]. This is particularly important for blocking operators, such as joins or aggregates, as it is not possible to wait for the entire input data stream to generate an answer; instead, the answer must be generated incrementally.

— *Impossibility of storing an entire data stream.* Traditional DBMSs are not designed for rapid and continuous loading of individual data items, so it is not feasible to accumulate all the past history of a data stream on a classical DBMS for later processing. On the contrary, it is highly desirable to make use of approximation techniques [Barbará et al. 1997] for representing the data stream in a small amount of space as accurately as possible [Guha et al. 2001; Gilbert et al. 2001; Dobra et al. 2002].

— *Need for support for continuous queries.* Traditional DBMSs do not directly support continuous queries [Terry et al. 1992; Babu and Widom 2001], which are typical of data stream applications, and they consider stable query plans instead of adapting dynamically to the current features of the incoming data streams [Avnur and Hellerstein 2000].

Many applications could benefit from handling incoming data as data streams, such as applications that need to process sensor data, network traffic, location data, time-series data, stock exchange data, telephone records, web clicking streams, weather data, environmental data, and so on [Cai et al. 2004b]. Several *Data Stream Management Systems* (*DSMSs*) have been proposed in the literature, such as *Cougar*, *Telegraph*, *STREAM*, *Aurora/Borealis*, *NiagaraCQ*, and *Nile*; for more information, we refer the interesting reader to papers such as [Babcock et al. 2002; Golab and Özsu 2003; Mokbel et al. 2005].

4.2.2 *Data Streams and Moving Objects.* Some works have highlighted that the research fields of data streams and management of moving objects can naturally come together. Thus, the output of a positioning mechanism used to obtain the location of a moving object (e.g., a GPS receiver) can be considered as a data stream, where new tuples with an updated location are released continually. In certain scenarios, a large number of moving objects send their locations *continually* (following a certain update policy, as explained in Section 4.1.2) to a centralized server, that deals with them "in real time" so as to update the answers to active queries. In these scenarios, data streams are a promising technology which could allow an efficient processing of the incoming location data. In the following, some works that share this vision are enumerated:

— *Data streams for defining the semantics of location-based queries.* [Huang and Jensen 2004] (which builds on [Arasu et al. 2003]) proposes a framework, based on

concepts from data streams and temporal databases, for defining the semantics of continuous queries in mobile environments. Data streams are used to model the locations of moving objects, and the answer to a location-based query is a data stream obtained by processing the input data streams. For example, a user in a vehicle can ask about near hotels at a "good" price, which involves relational data (information about hotels) as well as continuously-evolving data (the location of the vehicle), and the answer is presented to the user as a data stream.

The aforementioned work also shows that data stream query languages can be used to express location-based queries. In the queries, a *partitioned window* is first applied in order to obtain the most recent locations of objects from the data streams. Continuous queries are treated as a sequence of snapshot queries which are evaluated with a certain frequency. Unfortunately, the advantages/disadvantages of using data stream technology instead of moving object databases are not discussed.

— *Trajectory streams, experiences with STREAM and Telegraph.* [Patroumpas and Sellis 2004] proposes modeling objects by representing their continuous movements as *trajectory streams* (sequences of tuples evolving in space and time), and presents a basic framework for managing them.

Experiments with some queries using the prototypes of the systems STREAM and Telegraph are presented, and it is concluded that the data stream paradigm is powerful enough to manage data about moving objects. Nevertheless, certain limitations to express certain types of queries are identified. Particularly, the adoption of four new special-purpose window operators is proposed: 1) *binary windows*, which are windows of size two time units, to examine changes in motion such as an object crossing the boundary of a stationary area; 2) *landmark windows*, encompassing tuples with timestamps smaller or greater than a certain time instant; 3) *area-based windows*, to extract tuples that fall within a specific area; and 4) *trajectory-based windows*, for example to split a trajectory stream into separate substreams based on the objects' identities. Furthermore, three improvements that could be useful for processing continuous queries on moving objects are examined: 1) the use of punctuations [Tucker et al. 2003; Tucker et al. 2007] (indications in a stream that describe a subset of the domain of that stream) for query evaluation (e.g., to indicate that a moving object has just crossed the boundary of an area); 2) multiple query optimizations, such as grouping similar predicates or operators together; and 3) approximated trajectories using sampling, wavelets or other summary structures.

— In [Mokbel et al. 2005], the *PLACE* (*Pervasive Location-Aware Computing Environments*) project is presented, wherein a scalable location-aware data stream server is implemented on top of the DSMS *Nile* (`http://www.cs.purdue.edu/Nile/`). A *shared execution paradigm* is adopted for optimizing multiple continuous queries, so that the evaluation of a set of continuous spatio-temporal queries is performed as a spatial join between the moving objects and the moving queries. An *incremental execution paradigm* is also advocated, where only the updates to the previous answer are sent to the user; thus, positive/negative tuples indicate that a certain object must be added/removed to/from the previously reported answer. An SQL-like query language along with a set of spatio-temporal operators is offered. *Spatial windows*, *temporal windows*, and *predicate-based windows* (e.g., to consider only the latest reported data about each moving object) are considered.

In Section 7.1, we describe how PLACE is used as a basis for location-dependent query processing.

— [Chen et al. 2003] presents a prototype of *CAMEL* (*Continuous Active Monitor Engine for Location-based Services*), a location stream engine whose goal is to support a large number of moving objects. CAMEL supports queries such as *GetLocation*, *window queries*, *kNN queries*, *triggers* (notifications of events), and *historical queries*. It uses a grid to index the locations of moving objects, and it allows for two kind of triggers: *Single Moving Object Trigger* (*SiMOT*) and *Binary Moving Object Trigger* (*BiMOT*). [Yu et al. 2004] considers several pluggable filter algorithms to cope with a high location insertion rate and, at the same time, guarantee a good query accuracy.

— The specification of the *Linear Road Benchmark* (see Section 4.3) in the stream query language CQL [Arasu et al. 2003] is available at `http://www-db.stanford.edu/stream/cql-benchmark.html` (as part of the STREAM project web site), showing the possibility of using CQL to express queries related to moving objects.

— The *SCUBA* query processing approach and its extension *ClusterSheddy* (see Section 7.2) are implemented within the stream processing system *CAPE* [Rundensteiner et al. 2004].

— Finally, works such as [Xu et al. 2003c; Lee et al. 2006; Xu et al. 2006] focus on *geosensor networks* (also called *location-aware wireless sensor networks*), where sensor nodes can move. In this type of networks, the location of the nodes is an important parameter for query processing, and window queries retrieve values of sensors located within a certain area using spatial query routing. Works like this could bridge the gap between research on sensor networks and moving objects.

As a conclusion, data streams can be an interesting technology for managing moving objects and location-dependent queries, as the locations of moving objects are continuously changing.

4.3 Other Data Management Issues

Works such as [Jensen et al. 2001; Jensen et al. 2002] study location-based services from a database perspective, focusing on defining adequate *multidimensional data models* and a query language that supports partial containment relationships. In the context of these works, a user issues a request that is characterized by a combination of values in different dimensions (including the time and the user's location and profile), which are used to obtain the relevant content.

Some works define *benchmarks* to evaluate the performance of different approaches to manage and query moving objects' data. [Seydim and Dunham 2002] emphasizes the need for benchmarks targeting the location-dependent query processing domain and presents some preliminary guidelines concerning the query, data, and mobility behavior of the user. [Theodoridis 2003] discusses the requirements of databases for location-based services and provides a set of ten benchmark database queries. The Linear Road Benchmark (see [Arasu et al. 2004] and `http://www.cs.brandeis.edu/~linearroad/`) describes a traffic scenario for benchmarking data stream management systems.

Table V. Main features of approaches that require cooperation from the moving objects

Work	Proposal	Advantages	Disadvantages
MQM	-Static range queries on mov.objs. -Rectangular range -Subdomains -Resident domain -Monitoring region -Relevant query -D-tree (SQM) → BP-tree (MQM)	-Semi-distrib. approach -Saving location updates -Heterogeneous devices	-Centralized architecture -Overloading devices -Capabilities of devices -No moving query -No interest in locations -Reevaluation on update -Rely on objects' policy
MobiEyes	-Moving range queries on mov.objs. -kNN queries on static objects	-Semi-distrib. approach	-Centralized mediator -Overloading devices -Capabilities of devices -No kNN on mov.objs -No interest in locations -Comm. result on update -Rely on objects' policy
	-Grid division -Monitoring region -Safe period		
	-Query grouping -Lazy query propagation		
	-k-anonymity model -Identity removal -Cloaking		
	-Motion Adaptive Indexing		
Q-index	-Static range queries on mov.objs.	-Saving location updates -Incremental evaluation -Periodic reevaluation	-Centralized architecture -No moving query -Maximum speed -No interest in locations -Rely on objects' policy
	Q-index -Query indexing -R-tree → quadtree -Safe region		
	-*VCI* -Object indexing (R-tree) -Velocity-constraint		
	-Combined schema		
SRB	-Static range queries on mov.objs. -kNN queries on mov.objs.	-Saving location updates -Incremental evaluation -Generic framework	-Centralized architecture -Overloading devices -No moving query -No interest in locations -Reevaluation on update -Rely on objects' policy
	-Safe region -Quarantine area -Reachability circle		
	-Object indexing (R-tree) -Query indexing (grid)		

Finally, an overview of location models can be found in [Becker and Dürr 2005], distinguishing geometric and symbolic location models and classifying them according to the application requirements.

5. QUERY PROCESSING APPROACHES THAT REQUIRE COOPERATION FROM THE MOVING OBJECTS

This section describes works where the moving objects must communicate updates to the server according to a certain policy, in order to perform efficient query processing. Therefore, these approaches cannot be used when it is not possible to set an update policy on the moving objects or when the objects do not explicitly communicate with the server (e.g., enemy objects tracked by radar). Firstly, in Sections 5.1 and 5.2, two approaches where the moving objects must monitor the queries they can affect, communicating changes to the server if needed, are considered: MQM for static range queries and MobiEyes for (more general) moving range queries; in these approaches, the moving objects could be overloaded with query processing tasks in some situations. Secondly, two works that use the concept of a safe region, computed for every object based on the continuous queries in execution, such that an object must update its location when it exits its safe region, are described: Q-index and SRB (see Sections 5.3 and 5.4, respectively). Table V summarizes the main features of all these works.

5.1 MQM (*Monitoring Query Management*)

In *MQM* [Cai et al. 2004a; Cai et al. 2006] the focus is on *continuous static range queries on moving objects*. In this approach, the query processing is distributed

among the moving objects: Every moving object monitors its nearby queries, participating directly in the query processing. Thus, the workspace is partitioned into rectangular *subdomains*, and every moving object is assigned a *resident domain* (set of neighbor subdomains), such that each object will process the queries involving its resident domain. When a query, represented by a rectangular region, overlaps with a subdomain, the overlapping area is called a *monitoring region*, and the query is a *relevant query* to such a monitoring region. A moving object reports its location to the server whenever it moves outside its resident domain or crosses any query boundary. In the first case, the server needs to assign a new resident domain to the object; in the second, the affected query results are updated accordingly.

In the initial work [Cai and Hua 2002], the technique is called *SQM* (*Spatial Query Management*), and the index structure *D-tree* (*Domain Tree*) is proposed to efficiently determine the resident domain of an object. In [Cai et al. 2004a; Cai et al. 2006], the work in [Cai and Hua 2002] is considerably extended. For example, these works consider the *heterogeneous storage and processing capabilities* of the objects so as to assign every object a resident domain of an appropriate size: More powerful moving objects are assigned a larger resident domain (i.e., they will monitor more queries) in order to reduce the number of times they need to request a new one. For the assignment of suitable resident domains to objects, a new spatial access method called *BP-tree* (*Binary Partitioning tree*) is used, which is a modified *quadtree* [Tayeb et al. 1998] that stores information about the queries intersecting each subdomain. In this way, the resident domain of an object is as large as possible according to the moving object's capabilities, that is, a resident domain that does not contain more than a certain number of monitoring regions.

Summing up, in MQM the moving objects play a major role in the query processing. The main disadvantages of this work are:

—The approach is not completely distributed, since all the moving objects must communicate with a single server. Besides being a single point of failure, this server could become a bottleneck.

—Moving objects must have certain processing, memory, and communication capabilities in order to perform their work, as they must: 1) be aware of their resident domain and of the relevant queries (there could be many), and 2) be able to compute if they belong in the answer to such queries. The user devices could be overloaded with such tasks. Moreover, the processing and communication load on a moving object depend on the queries issued into the system, as the object must monitor its spatial relationships with such queries.

—It is assumed that the user is not interested in the precise locations of the target objects (which would be useful, for example, to continuously show the objects on their current locations on a map), but only in their identifiers.

—Only static range queries are processed, that is, the query point must be static. Thus, for example, a mobile user asking about objects in its vicinity is a scenario not directly supported.

—The queries are reevaluated on every location update, as an update from a moving object implies that the answer to some query may have changed.

MobiEyes, described in the following section, is a work strongly related to MQM.

It additionally allows moving queries.

5.2 MobiEyes

MobiEyes [Gedik and Liu 2006] focuses on the processing of *continuous moving range queries on moving objects*. It partitions the space using a *grid*, and associates every query to a *monitoring region*, which is the set of grid cells that the range query can intersect while its center moves within its current grid cell. An object within the monitoring region of a query receives information about the query's location and speed, and must notify the server when it enters or leaves the query region (predicted based on the query's speed) or its current cell. In this way, the objects monitor periodically (according to a computed *safe period*) their spatial relationships with the queries they can *influence*, playing an active role in the query processing (like in MQM, see Section 5.1). A *focal object* of a query (i.e., an object that indicates the center of the range of a moving range query, called reference object in [Ilarri et al. 2006a]) updates its data (position and speed) whenever it moves to another grid cell or changes its current speed vector above a certain threshold (a dead-reckoning technique is used to predict the position changes of moving objects of interest); in order to deal with reliability issues (disconnections), some periodic updates by focal objects, called *forced velocity updates*, are required. Some optimizations are also analyzed: 1) *Query grouping* allows a reduction in the amount of computation to be performed by the moving objects; and 2) *lazy query propagation* removes the need for non-focal moving objects to contact the server when they change to a different cell, at the expense of some inaccuracy. In MobiEyes, the query processing approach is performed on the moving objects themselves.

Similarly to MQM (see Section 5.1), the main disadvantages of MobiEyes are: 1) It considers a centralized architecture, as a single centralized computer acts as a *mediator* between the moving objects; 2) the moving objects must be aware of existing queries and detect if they cross any query boundary; and 3) it does not retrieve the locations of the objects in an answer.

There are other query processing approaches related to the MobiEyes project, such as:

— A mechanism to perform *exact kNN search* on R-trees [Gedik et al. 2004], based on the use of histograms to guide the search. Furthermore, the effects of different *broadcast* organizations on the search performance is discussed. Unfortunately, kNN queries only about *static objects* are considered.

— A scheme, called *Motion Adaptive Indexing* (*MAI*) [Gedik et al. 2006], that optimizes the evaluation of continuous queries according to the dynamic motion behavior of the objects, by using the concept of *motion-sensitive bounding boxes* (*MSBs*) to model both moving objects and moving queries. Instead of indexing locations, which change frequently, MSBs are indexed (using R*-trees), that only need to be updated when objects and queries (focal objects) move across the boundaries of their boxes; the linear motion function of the object/query is stored together with its MSB in order not to introduce inaccuracy in the query results. Two techniques to reduce the cost of query reevaluation and index searching are provided. First, query results are optimistically precalculated and incrementally maintained

when objects' motions change. Second, adaptive indexing is supported: The size of an MSB changes according to the changing moving behaviors of the corresponding individual objects. This work is an *alternative* to the approach described in [Gedik and Liu 2006] and focuses on server-side optimization issues.

Moreover, the authors of MobiEyes have also studied privacy issues (see Section 3.1.2). Thus, [Gedik and Liu 2005] defines a *personalized k-anonymity model*. The proposed framework allows each mobile node to specify the minimum *level of anonymity* it desires as well as the *maximum temporal and spatial resolutions* allowed when requesting location-based services. A message perturbation engine performs location anonymization on location messages, such as *identity removal* and *spatio-temporal cloaking* of location information. This mechanism is defined outside the approach described in [Gedik and Liu 2006], as it is not needed if the moving objects themselves process the queries.

5.3 Q-index (Query Indexing)

In [Prabhakar et al. 2002], the goal is to develop techniques for the efficient and scalable evaluation of multiple *continuous static range queries on moving objects*. The focus is on analyzing alternatives to the traditional indexing of moving objects, which has two main disadvantages: 1) the high cost of keeping the index updated as objects move, and 2) the need to reevaluate all the queries whenever an object moves. Two complementary techniques are considered:

— *Query indexing*. Queries are indexed, instead of objects. This reduces the cost of maintaining the index by minimizing the number of index updates, as queries are not added/removed as frequently as objects move. [Prabhakar et al. 2002] uses an R-tree to index the queries and [Kalashnikov et al. 2004] compares several indexing approaches and shows how a *grid* implementation of the Q-index is more efficient for in-memory evaluation. The proposed query processing approach performs an *incremental evaluation*: At each evaluation step, and for every moving object that has changed its location, the index is probed in order to determine the set of queries that can be affected by the movement of that object.
According to this proposal, objects should update their locations only when they can influence some query. To develop this idea, the concept of a *safe region* is introduced, which is an area around every object that does not intersect any query boundary. Therefore, an object does not need to issue an update while it moves inside its safe region. Safe regions are computed based on the Q-index. Both *circular and rectangular* safe regions are considered. Moreover, it is proved experimentally that rectangular safe regions minimize the number of location updates needed.

— *Velocity-constraint indexing* (*VCI*). Objects are indexed, but their maximum speeds are considered in order to delay the expensive operation of updating the index to reflect the movements of the objects. A VCI is a regular R-tree where every *MBR* (*Minimum Bounding Rectangle*) is tagged with the maximum speed of the objects within, such that an *expanded MBR* is considered at evaluation time. The expanded MBR covers the area that contains all its objects at that time instant. There is no need to know the precise speed of each object, it is enough if they do not exceed a certain maximum speed [Mokbel et al. 2003].

The experimental results presented show that query indexing outperforms traditional indexing techniques and also VCI. However, the arrival of new queries is expensive with query indexing, as every new query must be initially compared with every object and may potentially invalidate the safe regions. Therefore, a *combined scheme* is advocated, such that an index of each type is maintained: New queries are evaluated using VCI and, at an appropriate time (e.g., when the number of queries being handled by VCI becomes large), all the queries being processed through VCI are transferred to the Q-index.

The main disadvantages of this approach are: 1) It considers a centralized architecture; 2) it processes only static range queries; 3) every moving object must monitor its spatial relationship with its safe region; and 4) it assumes that it is not interesting to retrieve the locations of the objects in an answer, as an object does not update its location unless it can change the set of objects in an answer.

5.4 The SRB (*Safe-Region-Based*) Framework

In [Hu et al. 2005a], the *SRB* framework, for monitoring spatial queries over moving objects, is presented. The focus is on *continuous static range queries and kNN queries on moving objects*. The disadvantages of a *query-blind* location update policy (i.e., objects that update their locations independently of the requirements of the active queries) are highlighted and, as in the Q-index approach (see Section 5.3), every moving object is associated with a safe region. For simplicity and efficiency in the query processing, *rectangular safe regions* are chosen. *Two types of location updates* are considered:

— *Source-initiated updates*. In principle, an object can move freely within its safe region without issuing source-initiated updates. A *quarantine area* is assigned to each query, which is used to identify if such a query is affected by a source-initiated location update: This occurs only if, given the new and the last updated locations, one is within the quarantine area of the query and the other is not. For a range query, the quarantine area is the range of the query; for a kNN query, it is a circle centered at the query point and covering the k^{th} nearest object.

— *Server-initiated updates*. For a query that involves the spatial relationships of several objects, such as a kNN query, a location update from one object may invalidate the safe regions of other objects. In this case, the server will need to request an immediate location update from some objects in order to determine the answer to the query.

As in the Q-index approach, indexes are defined on both objects and queries; *an R-tree is used to index the moving objects*, and *a grid to index the quarantine areas* of the queries. Some optimizations are considered that can further reduce the number of location updates, based on: 1) the *maximum speed* assumption, which allows to compute the *reachability circle* of an object (which shows how far the object can reach at any moment since the last location update), used to reduce the number of server-initiated updates; and 2) the *steady movement* assumption (with a certain steadiness parameter), for efficient safe region computation.

The main disadvantages of SRB are: 1) It assumes a centralized architecture; 2) it does not consider moving range queries; 3) every object must monitor its spatial relationship with its safe region; 4) it assumes that the queries are not interested

in the precise locations of the objects; and 5) the query reevaluations are triggered by the location updates, so the wireless communications of results are unbounded.

6. QUERY PROCESSING APPROACHES THAT ASSUME KNOWN MOVING OBJECTS' TRAJECTORIES

This section reviews works that assume that objects follow known trajectories in order to process location-dependent queries efficiently. This main assumption could be an important limitation because: 1) Some objects move freely (e.g., people in a mall, airplanes, etc.); and 2) even if they move following certain trajectories, they can be unknown to the query processor, and requiring the moving objects to communicate them would have similar disadvantages to those explained in Section 5. First, two works that do not focus explicitly in query processing strategies but propose data management approaches for moving objects, appropriate location update policies, and strategies to minimize the cost of wireless communications are described. Firstly, Section 6.1 is devoted to the DOMINO project, which defines a model to represent moving objects by using dynamic attributes and proposes several location update policies and approaches to transmit the predicted results to queries. Secondly, Section 6.2 describes some works by Ulusoy et al., who propose more approaches for the transmissions of query results along with some new location update policies. Then, two works that propose specific query processing strategies are reviewed: CAT/CAT++ and ARGONAUT. In Section 6.3, the CAT/CAT++ framework is described, which advocates the use of triggers to efficiently recompute changes in the answers to continuous location-dependent queries. Finally, Section 6.4 describes ARGONAUT, a model to represent and query moving objects, which introduces in the query processing a filter step based on the expected trajectories of the moving objects. CAT/CAT++ focuses on continuous static range queries and persistent queries, whereas both continuous static range queries and continuous moving range queries are considered in ARGONAUT. Table VI summarizes the main features of these works.

6.1 DOMINO (*Databases fOr MovINg Objects*)

DOMINO [Wolfson et al. 1999a; Wolfson et al. 1999b; Wolfson et al. 2001] proposes the *MOST model* to represent moving objects in a database. The concept of *dynamic attribute* is introduced, which evolves according to a certain function even in the absence of database updates (as already mentioned in Section 4.1.1). In this way, the location of a moving object is a dynamic attribute, which allows representing the current and near-future positions of moving objects without incurring excessive update overhead. A dynamic attribute is represented by three sub-attributes: 1) the time instant when the moving object updated the attribute, 2) the location at update-time, and 3) a function of time that allows to estimate the location of the object at a future time instant. The time function can be as simple as a linear function that depends on the current speed vector of the moving object, or can be represented as a trajectory in a road network [Vazirgiannis and Wolfson 2001]. As the accuracy of a dynamic attribute decreases with time, moving objects need to update their locations according to a certain policy. Several *update policies* are studied in DOMINO, such as *sdr*, *adr*, and *dtdr*, explained in Section 4.1.2.

Table VI. Main features of approaches that assume known moving objects' trajectories

Work	Proposal	Advantages	Disadvantages
DOMINO	-Range queries -MOD, MOST -FTL language -Predictive queries	-Temporal operators -Spatial operators -Dynamic attributes -Rely on trajectories -Minim. location updates	-Centralized architecture -Little query processing -No interest in locations
	-Tuple comm. (IT, DT)	-Minim. answer updates	
O. Ulusoy et al.	-Range queries	-Minim. answer updates	-Centralized architecture
	-*Tuple comm.:* -Periodic (PT) -Adaptive periodic (APT) -Mixed (MT)	-Minim. location updates -Data availability -Predictive queries	-Little query processing -No interest in locations -Rely on trajectories
	-*Monitoring* -Adaptive (AMM) -Categorized (CMM) -Deadline-driven (DDM)		
CAT/CAT++	-Static range, mov.objs. -Trajectory model (roads) -ECA^2 (CAT++), OMCAT -Spill-over -Z-curve	-Real-time traffic -Context-aware triggers -Predictive queries -Persistent queries	-Centralized architecture -No interest in locations -Answer when update -No mov. query -Rely on trajectories
ARGONAUT	-Moving range -2D indexing (roads)	-Incremental evaluation: 1) Filter step 2) Periodic refinement	-Centralized architecture -Rely on trajectories -No interest in locations

FTL (*Future Temporal Logic*) [Sistla et al. 1997a] is the query language used with the MOST model. FTL augments SQL with temporal operators (such as *until*, *nexttime*, *eventually*, *sometimes*, *within*, and *always*) and spatial operators (such as *inside* and *dist*). Three categories of queries are identified, depending on the history on which the query is evaluated and the evaluation time (*instantaneous queries*, *continuous queries*, and *persistent queries*), and both *present and future queries* are considered (see Section 3.2). In [Vazirgiannis and Wolfson 2001], the modifiers *ALONG EXISTING PATH* and *ALONG SHORTEST PATH* are added for a *within* operator, besides operands *DISTANCE s* and *TRAVELTIME t*. Finally, uncertainty issues are considered in [Trajcevski et al. 2004b] and new operators (*definitely* and *possibly*, similar to the distinction between *may* and *must* queries in [Sistla et al. 1997b]) are added to determine whether the objects retrieved as an answer are required to satisfy the query for sure or only possibly.

Although some work is devoted to the processing of FTL operators, DOMINO deals with some query processing issues only marginally. Thus, *predictive query results* are computed based on trajectories. The answer to a continuous query is a set of tuples $<S, begin, end>$, indicating that S is an answer from time *begin* to time *end*. Two approaches are proposed to transmit the answers to the mobile user [Sistla et al. 1997a]:

—*Immediate transmission* (*IT*), where the tuples are transmitted immediately, once they have been computed.

—*Delayed transmission* (*DT*), where every tuple is transmitted at the time it starts to be part of the answer.

The choice between IT and DT depends on factors such as the frequency of updates to the answer, the probability that a location update invalidates a previous answer, and the cost of propagating those updates to the mobile host. The ultimate goal is to minimize the number of updates while providing an up-to-date answer.

Summing up, DOMINO focuses on the problem of how to store location information about moving objects in an database with the goal of processing spatio-

temporal queries in an efficient way. As stated before, query processing issues are secondary in DOMINO, and the proposed predictive query processing assumes that the approximate trajectories of the moving objects are known. DOMINO is not concerned either about retrieving the current locations of the objects in the answer. Another important disadvantage is that it assumes a centralized architecture. A distributed approach is also proposed in the DOMINO project [Sistla et al. 1997a], but the information is distributed among the moving objects themselves instead of among databases, which could lead to problems similar to those of MobiEyes (see Section 5.2); for example, information about moving objects is actually sent to a central location to process a given query.

6.2 O. Ulusoy et al.: Transmission of Results and Location Update Policies

[Gök and Ulusoy 2000] considers a centralized approach for continuous queries in mobile computing environments. Once the answer to a query is computed, the main concern is *when to transmit* the tuples to the mobile host in order to minimize communications while providing an up-to-date answer to the user[5]. Besides IT and DT (see Section 6.1), three tuple transmission approaches are proposed to determine the transmission times of the tuples <*S, begin, end*> in a predicted answer:

—*Periodic transmission* (*PT*): The tuples in the answer set are transmitted periodically. Every w time units, all the tuples satisfying $t \leq begin < t + w$, where t is the current time instant, are transmitted.

—*Adaptive periodic transmission* (*APT*): The time window w in PT is adapted dynamically according to the current communication overhead.

—*Mixed transmission* (*MT*): Data objects are partitioned into two groups. Tuples corresponding to *hot objects* (objects that move frequently) are transmitted according to APT, and tuples for *cold objects* (objects that move rarely) according to IT, to limit the control message overhead and the retransmission overhead mainly to the *hot objects*.

These alternatives are compared experimentally in terms of *communication overhead* and *data availability* in case of disconnection. The communication overhead is determined by two factors: 1) the control message overhead, due to the control bytes appended to every message; and 2) the tuple retransmission overhead, which may occur when a location update from an object overrides a previous tuple in an answer.

Besides transmission policies, different methods for the generation of location updates from moving objects are also proposed:

— In [Lam et al. 2001], an *adaptive monitoring method* (*AMM*) is proposed. In a similar way to the dead-reckoning approaches *adr* and *pdr* (see Section 4.1.2), this update policy is based on the deviations of the locations of the moving objects. However, it considers an upper and a lower threshold bound. The *upper threshold* is for objects whose location uncertainty can be acceptably large, while the *lower threshold* is for objects that need close monitoring. Every object is assigned a

[5]Another work that proposes techniques to filter communications, in the field of spatial continuous queries, is [Brinkhoff 2002].

threshold value between the lower and the upper bounds, according to a certain formula, such that objects that are now or are expected to be soon in the query answer will have smaller update threshold values. Furthermore, a minimum time interval between updates by the same object is required to detect disconnections.

— In [Ulusoy et al. 2003], a *categorized adaptive monitoring method* (*CAMM*) is proposed, based on AMM. This method allows the users to specify various criticality levels for queries: Higher levels of accuracy are provided for the results of more critical queries.

— Finally, in [Lam and Ulusoy 2006] a *deadline-driven method* (*DDM*) is also proposed, by which each moving object sends a location update only when it is about to satisfy the condition of a query. DDM can reduce the location update overhead if the movements of the objects follow the predicted trajectories most of the time, whereas AMM is suitable for systems where disconnections are frequent.

Thus, the works described in this section consider a centralized architecture and focus on strategies for updating query results [Gök and Ulusoy 2000] or defining location update policies for moving objects [Lam et al. 2001; Ulusoy et al. 2003; Lam and Ulusoy 2006], instead of on how to compute those results. Besides, since a time interval is associated to each tuple in a result, it is assumed that it is possible to obtain predictive results of queries, that is, that approximate trajectories about moving objects are known to the query processor. Finally, these proposals are not well-adapted to deal with continuous queries which ask for locations of moving objects, as the focus is only on retrieving the set of objects in the answer.

6.3 The CAT/CAT++ Framework

Based on a *trajectory model* [Trajcevski et al. 2004b] to represent moving objects, *CAT* (*Correct Answers of continuous queries using Triggers*) [Trajcevski et al. 2004a; Trajcevski and Scheuermann 2004] is a framework to deal with continuous queries in moving object databases. It focuses on query processing issues regarding *continuous static range queries on moving objects*.

The main idea is that a continuous query should not be reevaluated unless its result may have been affected by a recent change. Therefore, the authors of this work draw from research in the field of *active databases* [Paton 1999] and propose the use of triggers (active rules) to detect the relevant changes.

The CAT architecture consists of: 1) *data tables*, where information about moving objects' trajectories and queries is stored; 2) *scripts*, which are in charge of keeping the data structures and answers to queries up to date; and 3) *triggers*, that execute the appropriate scripts when an event, such as a traffic abnormality or a trajectory change by a moving object, occurs. A prototype has been implemented using Oracle 9i [Loney and Koch 2002] as a database server, which offers PL/SQL capabilities for *User-Defined Data Types* (*UDT*), *User-Defined Functions* (*UDF*), and *Oracle Spatial* predicates [Gietz and Dupree 2002].

In [Trajcevski et al. 2004a], the focus is on the impact of traffic changes (e.g., traffic jams, accidents in a road, etc.), whereas [Trajcevski and Scheuermann 2003] considers updates that may be initiated by both the moving objects (e.g., changes in their travel plans) and the server (based on detected real-time traffic conditions).

In [Trajcevski et al. 2005a], the main concern is how to minimize the time needed

to update the answers to static range queries since a trajectory is updated. The authors suggest, and experimentally prove the benefits of, triggers that are *context-aware* at two levels:

—Regarding individual triggers, they can be executed in a *BEFORE* and *FOR EACH STATEMENT* (set-oriented) manner to reduce the cost of context switching. Moreover, the query regions are indexed to facilitate the detection of the triggers that do not need to be fired upon a trajectory update.

—For intra-trigger relationships, the execution of simultaneously enabled triggers can be ordered according to a *Z-curve*, so as to minimize the swapping overhead by exploiting the spatio-temporal correlation among queries.

In [Trajcevski et al. 2002a], the focus is on modeling the *spill-over* effect caused by a traffic abnormality. For example, a traffic accident not only affects the road where it occurs but also the neighbor roads, due to traffic rerouting. Three stages are considered: 1) the disturbance interval, 2) the persistent interval, and 3) the recovery interval.

In [Trajcevski et al. 2005b], CAT is extended and called *CAT++* (*optimal management of Correct Answers to continuous queries and persistent requests using Triggers*). The focus is on *persistent queries* [Sistla et al. 1997a], and two topological predicates are examined: *moving along* and *moving toward*. The $(ECA)^2$ (*Evolving Context-Aware Event-Condition-Action*) paradigm is proposed, which extends the traditional *ECA* (*Event-Condition-Action*) paradigm of active databases by allowing the specification of modifications to a trigger upon the failure of the condition part. Composite events and several policies for the consumption of their constitutive primitive events are considered. The term *metatrigger* is introduced in [Trajcevski et al. 2006], which is a module in charge of coordinating the detection of events and the evaluation of the query conditions. The goal is to minimize the communication overhead and, at the same time, ensure correctness.

Finally, it should be noted that a prototype demonstration was carried out in [Ding et al. 2006], and the focus is on how a traffic abnormality may affect queries related to future portions of affected trajectories, which must be reevaluated efficiently. The extended prototype is called *OMCAT* (*Optimal Maintenance of Continuous Queries' Answers on Trajectories*).

The query processing approach proposed in CAT/CAT++ is reactive: Queries are reevaluated upon changes on the database and the new answers are in principle transmitted to the user device immediately. CAT/CAT++ assumes a centralized architecture with a single server managing all the information. Moreover, it focuses solely on the efficient processing of the set of objects that satisfy certain conditions, and the current locations of those objects are neglected. The proposal is limited by several assumptions: 1) that the final destination and other locations to be visited by the moving objects are known to the server; 2) that objects move within road networks, so their expected trajectories can be built in advance; and 3) that there is a specific location update policy, as moving objects must inform about route changes.

6.4 ARGONAUT

ARGONAUT is a prototype application developed on top of *mSTOMM* (*Spatio-*

Temporal Object Modeling and Management), a model to represent and query moving objects [Stojanovic and Djordjevic-Kajan 2003; Predic and Stojanovic 2005; Stojanovic et al. 2006; Stojanovic et al. 2008]. ARGONAUT is a framework that provides moving object data management and query processing functionalities via HTTP/SOAP. The focus is on *continuous range queries over objects moving on known network paths*, which are particularly useful in scenarios for tourist and business guiding.

In this approach, moving objects can follow a *threshold-based* location update policy (called speed dead-reckoning in Section 4.1.2). Upon receiving a new location, the server must update in-memory data structures that keep information about objects and queries. For this purpose, it relies upon the availability of a 2D indexing scheme (called *SR*-tree* or *Segment R*-tree* in [Stojanovic et al. 2008]) on the segments of a transportation network. An *incremental evaluation paradigm* with two steps is considered:

(1) Initially, and based on the expected trajectories of the moving objects, a *filter step* produces a candidate result set with the help of the index, by obtaining the routes which have an MBR that intersects the query range's MBR (if the query is mobile, then the query route is also considered). This is a predictive query result, where each object in the answer is associated to one or several temporal intervals indicating when the object verifies the query condition, similarly to some approaches on NN query processing presented in Section 8.1.1.

(2) Then, with a certain frequency, a *refinement step* produces the results for the current time instant.

The specifications of the previous algorithms are different depending on whether *mobile queries* or *stationary queries* are considered. Furthermore, at every location update the expected trajectory of a moving object is updated according to the amount of time that it is behind (or ahead of) schedule, and the continuous query result sets in which that object is included are updated accordingly too.

As a main disadvantage of this approach, a centralized architecture is considered. Another disadvantage of this work is that several assumptions are made: 1) The user moves through a road network where the average speed and travel time of the road segments are known in advance; and 2) the user specifies, along with the query, his/her travel destination. This second assumption is not strictly necessary, but at least the user must move within a road network, as the network segment where the user is located is needed. The refreshment step is performed periodically, so it would be possible to estimate the locations of the objects in the answers based on their trajectories; nevertheless, obtaining the current locations of the objects in the answers is not explicitly considered.

7. GENERAL QUERY PROCESSING APPROACHES

This section reviews some other works wherein issues concerning the processing of location-dependent queries are considered. As opposed to the works presented in the previous two sections, these works do not make assumptions regarding the capability of moving objects to cooperate with the query processing or regarding the objects' motion patterns. The review of these works starts with a description of approaches

Table VII. Main features of general query processing approaches

Work	Proposal	Advantages	Disadvantages
PLACE	-Mov. range queries (SINA) -kNN queries (SEA-CNN)	-Incremental evaluation -Shared execution -Periodic comm. results -Out-of-sync clients -Generic (GPAC) -Anticipated areas (only for static objects)	-Centralized architecture -No interest in locations -Servers with state
	-Query indexing -Object indexing		
	-Grid index -Data streams -Predicate windows		
	-No-Action regions		
	-Data streams (SOLE)	-In-memory -Pipeline operator	-Uncertainty
SCUBA	-Moving range queries -kNN queries -Aggregate queries	-Return current locations -Shared execution -Periodic evaluation	-Centralized architecture -Approximate locations -Cluster maintenance -Assume road networks -No good clusters if the objects move independently -No interest in locations
	-Moving clusters (objects, queries) -Distance threshold -Speed threshold		
	-Join-within/between		
	-Improved in ClusterSheddy	-Incremental evaluation -Load shedding	
Dynamic Queries	-Moving range queries -Indexing (e.g., R-tree) -*Discardable* MBBs	-Return current locations -Reduce disk accesses (overlapping results) -Periodic reevaluation	-Centralized architecture
		-Predictive queries	-Approx. trajectories
Monash University	-Range queries around client -Query scope (squared)	-Distributed architecture -Reuse previous results	-Only static target objs. -No continuous queries -Not around any object -Servers with state
		-Minimize effects of answer latency	-Known user's direction -Estimated latency
LOQOMOTION	-General queries -Continuous queries	-Distributed processing -No overload mov.obj. -Return current locations -Location granules -Extended areas -No assumptions	-Generality could make optimization difficult
	-Mobile agents -Agent synchronization		

that propose a centralized architecture: PLACE, SCUBA, and the work on dynamic queries by Lazaridis et. al. Then, two works that consider a distributed architecture are reviewed: the work by Jayaputera et. al at Monash University (limited to queries on static target objects) and LOQOMOTION. Table VII summarizes the features of the approaches considered in this section.

7.1 PLACE

This section describes how the location-aware server *PLACE* [Mokbel et al. 2005], introduced in Section 4.2.2, is used as a basis for location-dependent query processing.

SINA (*Scalable INcremental hash-based Algorithm*) [Mokbel et al. 2004] is an algorithm for processing, on a centralized server, *continuous moving range queries on moving objects*. *Grid indexes* on both the queries and the objects are used, and an in-memory hash-based join algorithm between queries and objects is continually applied as location updates arrive. Periodically, the query processing is completed taking into account other data about moving objects stored on disk (when the memory is full, some in-memory data is flushed into disk), and then the *positive and negative updates* are communicated to the user device.

In [Xiong et al. 2004], three policies are considered to perform a join between a set of moving objects and a set of range queries: the *Clock-triggered Join Policy*, the *Incremental Join Policy*, and the *Hot Join Policy*. With the Hot Join Policy, the notion of *No-Action Region* is introduced, which is the region where an object

or a query can move without affecting the latest reported result of any continuous query. The concept of No-Action region is similar to the idea of safe region proposed in the Q-index approach (see Section 5.3), also adopted in the SRB framework (see Section 5.4), but the purpose is different: reducing the number of location updates in Q-index and SRB vs. minimizing the cost of joins. Thus, the No-Action regions are used to minimize the required join operations in a shared execution query plan.

Predictive, kNN, and aggregate queries are also considered, and support for *out-of-sync clients* (clients that disconnect and reconnect later, missing some answer updates) is introduced. Specifically, continuous kNN queries are the focus of *SEA-CNN* [Xiong et al. 2005], which relies on some other existing algorithm in charge of getting the initial answer. Objects are indexed with a grid, and each query is associated with an *answer region*, which is a circle centered on the query location with radius its distance to the current k^{th} nearest neighbor. When updates arrive, the cells (and therefore the answer regions) affected are determined and the new sets of k nearest neighbors for the queries are conveniently recomputed.

[Mokbel and Aref 2005] presents *GPAC* (*Generic and Progressive* algorithms), a general skeleton that can be adjusted through a set of methods to behave as various spatio-temporal queries (instances for range and kNN queries are shown). It proposes to cache potential future query results through the concept of *anticipated area* (similar to the idea of extended area considered in LOQOMOTION, see Section 7.5), which allows to reduce the uncertainty derived from the use of *predicate-based windows*: An incoming data tuple is stored in memory only if it satisfies a query predicate, so extra objects need to be stored in order to detect objects that become part of the answer due to a movement of the query point.

Finally, the *scalable online execution* (*SOLE*) algorithm, for the evaluation of concurrent continuous spatio-temporal queries over data streams, is presented in [Mokbel and Aref 2008]. SOLE uses an incremental evaluation paradigm and a grid structure, similarly to SINA. However, there are some fundamental differences. Thus, SINA is disk-based and SOLE is memory-based. Consequently, due to memory limitations, some objects may not be stored in SOLE (a load-shedding approach is applied), causing some uncertainty. Moreover, SOLE is an online algorithm (so it can be implemented as a pipeline operator) whereas SINA processes all the pending updates periodically. The authors of [Mokbel and Aref 2008] claim that "SOLE bridges the areas of spatio-temporal databases and data stream management systems".

The main disadvantage of this interesting work is that it relies on a centralized architecture. Moreover, only the set of objects that satisfy the query conditions is computed, without concerning about obtaining the current locations of the objects.

7.2 SCUBA (*Scalable ClUster-Based Algorithm*)

SCUBA [Nehme and Rundensteiner 2006] is a proposal to evaluate *continuous queries on objects moving in road networks*. Although this work focuses on *moving range queries* on moving objects, it is claimed that the approach is also applicable to other types of queries, such as *kNN queries* and *aggregate queries*.

The key idea is to group moving objects and queries into *moving clusters*, based on common spatio-temporal properties. This is suitable in scenarios where groups of objects move together, such as traffic jams, animal and bird migrations, groups

of children on a trip, etc. An *incremental cluster formation technique* is proposed, that forms clusters at run-time based on the use of two thresholds to maintain the quality of the clusters: 1) A *distance threshold* guarantees that clustered entities are close to each other at the time of clustering; and 2) a *speed threshold* ensures that they will stay close in the near future. Moreover, the centroid of a moving cluster can serve as an approximation of its members, which can be used for *load-shedding*.

SCUBA exploits a *shared cluster-based execution paradigm* by evaluating a set of queries following a two-step process. First, it performs a spatial *join-between moving clusters* as a cheap preprocessing step, based on the idea that if the clusters do not overlap then their individual moving objects and queries cannot join either. Then, it processes a fine-grained *join-within clusters*, that joins individual moving objects and queries within the clusters not filtered out in the previous step. Query execution is performed with a certain frequency.

SCUBA is implemented within the stream processing system *CAPE* [Rundensteiner et al. 2004]. Besides, SCUBA is compared experimentally with a traditional grid-based spatio-temporal algorithm that hashes objects and queries into a grid index based on their locations.

Finally, it should be noted that SCUBA is extended in [Nehme and Rundensteiner 2007], where the framework is called *ClusterSheddy*. ClusterSheddy focuses on load-shedding. Moreover, an incremental query evaluation is implemented to avoid a continuous reevaluation of the queries, which leads to a reduction in the processing and memory consumption of SCUBA and enables the encapsulation into a physical non-blocking pipelined query operator. kNN queries are treated as range queries with a variable region size.

Unfortunately, SCUBA considers a centralized architecture. Its efficiency depends on whether the objects in the scenario move in groups or not. There is no explicit concern about retrieving the current locations of the objects in an answer, but just about obtaining the positive and negative updates (i.e., detect objects which enter/exit the answer). Finally, although it seems that the approach would work even in the absence of road networks, a road-network mobility model is assumed in [Nehme and Rundensteiner 2006], and the proposal in [Nehme and Rundensteiner 2007] is evaluated using a generator of network-based moving objects.

7.3 I. Lazaridis et al.: Dynamic Queries

In [Lazaridis et al. 2002], the focus is on moving range queries in moving objects environments, which are called *dynamic queries* to stress that not only their results but also the queries themselves change along time. The motivating scenario is to support the rendering of moving objects in virtual tour-like applications. In this work, the objects interesting to the query can move, and mechanisms are proposed for the *periodic processing* of the queries, based on the use of a multi-dimensional data structure (e.g., an R-tree) to index the bounding boxes representing the objects' motion. The goal of the authors is to reduce the number of disk accesses by avoiding the recomputation of overlapping results; on a query reevaluation, the server remembers the previously executed query and can avoid repeating some work. Thus, some bounding boxes of the index are marked as *discardable* if they do not need to be examined, that is, if they have not received any update since the last evaluation instant and the overlapping part of that bounding box and the range of

the current query is covered by the range of the previous query. *Predictive dynamic queries* (*PDQ*) regarding the location of the mobile user are also considered for cases where the trajectory of the observer is known in advance, such that the query processor can precompute answers for each refreshment of the dynamic query to optimize the cost.

This work proposes a periodic reevaluation of continuous queries, so the current locations of the objects can be returned as part of the query answer. The main disadvantage is that it considers a centralized architecture, which could be easily overloaded with a high number of moving objects and queries. Moreover, it only considers range queries and not, for example, nearest neighbor queries.

7.4 Monash University: Static Target Objects

In [Jayaputera and Taniar 2005a; 2005b], the authors focus on *range queries on static objects regarding the location of the mobile user*. They consider a distributed environment where the information about the static objects is served by the *base stations*. They propose to use a *square* [Jayaputera and Taniar 2004] (instead of a circle or a rectangle) to indicate the range of a query (the *query scope*), due to two reasons: 1) It is very efficient to compute if a given target object is within a square; and 2) if a circle is used, objects near the borders of the circle (but outside it) are not retrieved as an answer to the query.

This work assumes that the user is only interested in targets that he/she will find following in the same horizontal or vertical direction: In this way, the query scope is divided into four equal areas and checking the whole scope can be avoided. Moreover, an estimation of the latency to get an answer is assumed to be available, and this is used to predict the location of the user when he/she receives the answer, considering a linear movement. [Jayaputera and Taniar 2005b] assumes that the query scope remains within the range of the same base station, whereas [Jayaputera and Taniar 2005a] considers a multi-cell wireless environment.

A client can acknowledge a missed query result and, in that case, the server will communicate again the result to the client. If the distance traveled by the user between the communication of the two results of the query is smaller than the length of the query scope, there is an *overlapping area between the results* that does not need to be checked again, which allows to optimize this subsequent communication step. Some experimental results validate the savings in processing load at the server.

The main limitation of this work is that it considers only range queries over static objects and only regarding the location of the mobile user, along the lines of approaches on querying location-dependent data (see Section 10). Moreover, it does not consider continuous queries and the approach requires the server to keep the status of their clients.

7.5 LOQOMOTION (*LOcation-dependent Queries On Moving ObjecTs In mObile Networks*)

LOQOMOTION [Ilarri et al. 2006a] is a mobile agent-based architecture for the distributed processing of continuous location-dependent queries issued by mobile users, which allows to retrieve the objects' locations and other interesting data. LOQOMOTION is able to process queries that depend on the location of any

moving object, as opposed to other works that, for example, only focus on queries about static objects that are in the vicinity of a certain static or moving object, objects moving within a fixed region, etc.

The key feature of LOQOMOTION is that it considers a *completely distributed environment* to process the queries. Thus, all the processing is performed distributively on the fixed network with a minimum intervention of the user device and without requiring any specific cooperation from the moving objects, unlike the works presented in Section 5. The underlying infrastructure of LOQOMOTION is composed of computers, called *proxies*, that manage location data about objects moving within different geographical areas. Different types of location-dependent constraints are considered (such as *inside*, *outside*, *nearest*, and *furthest*), and location-dependent queries are translated into standard SQL-like queries by expressing first the constraints in terms of *inside* constraints. Communication failures are taken into account, for example, by using *extended areas* for the *inside* constraints, such that more objects than those strictly required are communicated to the user's mobile device. This allows, for example, to estimate that a new object becomes part of the answer in the next refreshment, in case the location of such an object cannot be obtained due to communication failures. The idea of extended area is similar to the concept of anticipated area described in Section 7.1; however, only the maximum speed of the *reference object* (following the terminology of LOQOMOTION, called focal object in MobiEyes, see Section 5.2) is considered to compute the anticipated area (used for a different purpose), whereas the extended area depends on more factors (e.g., the required data *refreshment frequency*).

LOQOMOTION, makes *no assumption* regarding: 1) the mechanism used for retrieving the location data, that can be obtained using different positioning methods with different precision; 2) the way that this location information is managed (e.g., stored in databases, estimated using predefined trajectories, or pulled on demand from the moving objects themselves); and 3) the number of proxies that manage and provide these data, and their coverage areas. Moreover, it is considered that the user can be interested not only in the set of objects that satisfy the constraints in the query, but also *in their current locations* (for instance, to locate such objects on a map). Finally, the user can specify the required *location granularity* [Ilarri et al. 2007] through the idea of *location granules* (similar to the idea of *places* in [Hoareau and Satoh 2007], e.g., GPS or cities on a map). This specification can change the semantics of the query or just be used for presenting the results of the query according to the user's needs.

LOQOMOTION is based on a layered hierarchy of mobile agents [Spyrou et al. 2004; Ilarri et al. 2006b] that move autonomously over the network with the goal of tracking efficiently the relevant moving objects, correlating partial results and, finally, presenting and continually updating the answer to the user's query. Within the context of LOQOMOTION, a general agent synchronization mechanism for monitoring systems is also proposed, which is fault-tolerant and adaptive to changing environmental situations [Ilarri et al. 2002; 2003]. Furthermore, an agent-based simulation framework of mobile environments, that is used to test the query processing system under any wanted situation has also been developed [Ilarri et al. 2004].

As a main disadvantage, the system can miss the opportunity to optimize different aspects that are specific to certain queries and situations. This is due to the fact that the query processing approach in LOQOMOTION is very general and different types of queries are processed in a very similar way. For example, different location-dependent constraints are all expressed in terms of *inside* constraints. Moreover, the query processing is independent of the scenario; e.g., the objects could move freely, but also be constrained to move within roads. Whereas this general architecture allows LOQOMOTION to process many different types of queries, other approaches that focus only on specific types of queries could perform better in the scenarios they have been designed for.

8. QUERY PROCESSING APPROACHES FOR NEAREST NEIGHBOR QUERIES

Regarding nearest neighbor queries, works in the field of spatial databases focused initially on retrieving a set of k static objects that are the closest to a static query point. The proposed algorithms (e.g., [Henrich 1994; Roussopoulos et al. 1995; Hjaltason and Samet 1999; Ferhatosmanoglu et al. 2001]) index the data with a spatial access method (e.g., an R-tree) and search the k nearest objects using branch-and-bound. Recently, the focus has shifted towards moving NN queries in client/server architectures, where the query point (usually, the mobile user launching the query) moves.

This section indicates some works that focus specifically on the processing of nearest neighbor queries in mobile environments (although some of them may also consider other types of queries). The description of these works is not as detailed as for works in previous sections (Sections 5, 6, and 7). The motivation to describe in more detail the previous works is that either they consider different types of queries (e.g., MobiEyes, SRB, PLACE, or LOQOMOTION) or they process range queries, which can be the basis for processing nearest neighbor queries (e.g., see [Ferhatosmanoglu et al. 2001; Iwerks et al. 2003; Ilarri et al. 2006a]). In some way, the works described in previous sections are seen as more general, as they do not specialize in the problem of processing nearest neighbor queries. Indeed, some approaches to process kNN queries have been already mentioned in previous sections of this article (such as Sections 5.2 and 7.1) because they can be considered part of a more general query processing approach.

In the following, works that process nearest neighbor queries on static objects are first examined. Then, proposals that consider that the target objects can move are described. It should be emphasized that the amount of literature on the processing of nearest neighbor queries is impressive, and so selecting only a few works was required. The main limitation of the works presented in this section is that they consider a centralized architecture and do not strive to provide the current locations of the objects in an answer. Moreover, some of them make assumptions about the patterns of movements of the moving objects.

8.1 Nearest Neighbor Queries on Static Objects

Some approaches deal with the problem of processing nearest neighbor queries on static target objects. First, we consider works that may use some information about the movements of the query point, but do not assume a restricted movement. Then, we describe approaches where, in addition, the query point is assumed to

move within roads.

8.1.1 *Nearest Neighbor Queries on Static Objects with an Unconstrained Moving Query Point.* In the following, some approaches to process nearest neighbor queries on static objects, that do not focus on road networks, are described (see Table VIII). In most of the works presented, some *validity information* (a temporal scope [Zheng and Lee 2001] or a spatial scope [Zhang et al. 2003]) is attached to an answer, which allows the user to determine when it may become obsolete, and so, when a new query is needed to retrieve the new answer:

—[Song and Roussopoulos 2001b] presents a progressive approach to process continuous kNN queries on static objects, based on the idea that a kNN query in a sequence can be processed more efficiently if the result of the previous kNN query is taken into account. Four interesting algorithms are proposed; for example, in the *prefetching search* the server sends to the client more than the k required nearest objects, in order to minimize the number of queries issued.

—In [Zheng and Lee 2001], Voronoi diagrams [Okabe et al. 2000] are used to retrieve the nearest facility to a mobile client. In this work, the nearest neighbor retrieved is guaranteed to be valid at least during a *validity period* indicated in the answer. This validity period is a conservative approximation computed based on the maximum speed of the mobile client.

—The previous method is improved in [Zhang et al. 2003] by generalizing it to kNN queries and returning, instead of the validity period of the answer, its *validity region*, which is the area around the client where the answer does not change. Therefore, a new query is not needed unless the user exits that area.

—[Tao et al. 2002] considers kNN queries on static objects for a client moving in a specific direction (straight line). A result is computed as a set of tuples $<R, PI>$, such that R are the k nearest objects during the point interval PI. If known, the user can also submit his/her expected trajectory and the result of the query will consider all its changes of direction. With this approach, the user only needs to issue a new query if his/her direction or submitted trajectory changes.

—Finally, in [Lee 2007], an algorithm for processing kNN queries in broadcast environments is proposed. The mobile user is assumed to move in a straight line and the kNN objects corresponding to every point in the segment considered are retrieved (as in [Tao et al. 2002]). This approach is based on the use of a *Hilbert Curve Index*.

8.1.2 *Nearest Neighbor Queries on Static Objects with a Query Point Moving in a Road Network.* Some authors emphasize that most objects in the real world move within roads (e.g., see [Kollios et al. 1999a]). The following are some works that consider that the query point moves within road networks (see Table IX):

—The importance of considering the network distance [Civilis et al. 2005] (i.e., the length of the shortest path to traverse), instead of the Euclidean distance (i.e., the geographical distance), to compute the nearest objects is shown in [Shahabi et al. 2002]. This work proposes a technique called *Road Network Embedding* (*RNE*), to transform a road network to a high dimensional space where computationally simple metrics can be applied to measure distances. RNE is appropriate for kNN

Table VIII. Nearest neighbor queries on static objects with unconstrained moving query points

Work	Reevaluation	Indexing	Main features
[Song and Roussopoulos 2001b]	Sampling	-R-tree ([Xu et al. 1990])	-Upper bound -Lazy search -Prefetching -Dual buffer
[Zheng and Lee 2001]	Validity period	-Voronoi diagram ([Okabe et al. 2000])	-Maximum speed
[Zhang et al. 2003]	Validity region	-R*-tree ([Beckmann et al. 1990])	-Voronoi cells -Influence objects -Window queries -Spatial joins
[Tao et al. 2002]	Direction change	-R-tree ([Xu et al. 1990])	-Line segments -Result: <R, PI> -Split points
[Lee 2007]	Direction change	-Hilbert Curve ([Zheng et al. 2003b])	-Broadcast -Minimize tuning

queries with static query points. For kNN queries with moving query points, an extension to RNE, called *D-RNE*, is proposed. The proposed transformation requires the precomputation of distances between all pair of vertices in the road network, which may be impractical [Huang et al. 2005].

—[Cao et al. 2003] also emphasizes the importance of considering the real driving distance (the *attainability*). In this work, a *reachability graph* is used to index the static target objects and the end points of segments in the road. For efficiency, a certain number of objects which are the nearest to every end point can be precomputed.

—[Papadias et al. 2003] considers a modeling graph of the road network and introduces an *on-line computation approach*, wherein kNN queries are computed by incrementally scanning the network from the query point until k neighbors are found (*Incremental Network Expansion*, *INE*). The performance of this approach depends on the density of the data points: A low density will lead the expansion process to search a large part of the road network, until enough data points are retrieved [Huang et al. 2005].

—In [Kolahdouzan and Shahabi 2004], the *Voronoi-based Network Nearest Neighbor* (VN^3) technique is presented, which proposes processing kNN queries on static objects (points of interest) using *Network Voronoi Diagrams* (*NVD*, Voronoi diagrams based on the network distance), such that different Voronoi regions (polygons) around the points of interest are defined. For performance reasons, some network distances are precomputed. Experimental results show that VN^3 outperforms the INE approach presented in [Papadias et al. 2003]. According to [Huang et al. 2005], this technique is not efficient in areas with a high data density.

—In [Huang et al. 2005], the use of *islands* is proposed, which are sub-networks that are no further than a certain distance (the radius of the island) away from each data point. Islands in different regions in the network can have different radius, in order to manage the tradeoff between the update cost and the query cost. Experiments show that this approach can achieve better performance than INE [Papadias et al. 2003] and VN^3 [Kolahdouzan and Shahabi 2004].

—Finally, in [Kolahdouzan and Shahabi 2005], the goal is to retrieve all the answers to a kNN query along a path, associating each answer with a certain interval. Thus, the goal is similar to that of [Tao et al. 2002; Lee 2007] (see Section 8.1.1) but considering the actual network distances.

Table IX. Nearest neighbor queries on static objects with query points moving in roads

Work	Reevaluation	Indexing	Main features
[Shahabi et al. 2002] (*RNE: Road Network Embedding*)	Not considered	-Optional -X-tree ([Berchtold et al. 1996])	-RNE -Truncated-RNE -D-RNE (dynamic) -Chessboard metric -SP-RNE (shortest paths)
[Cao et al. 2003]	Not considered	Reachability graph	-Attainability -Precomputation
[Papadias et al. 2003] (*INE: Incremental Network Expansion*)	Not considered	-R-tree ([Xu et al. 1990])	-On-line computation -Euclidean distance (IER) -Network distance (INE) -Range queries -Closest pairs -e-distance joins
[Kolahdouzan and Shahabi 2004] (VN^3)	Not considered	-R-tree ([Xu et al. 1990])	-Voronoi diagram ([Okabe et al. 2000]) -Precomputation
[Huang et al. 2005] (*Islands*)	Not considered	Not considered	-Islands ↔ subnets, radius -Precomputation
[Kolahdouzan and Shahabi 2005]	Segment change	Not considered	-Split points -IE, UBA algorithms

8.2 Nearest Neighbor Queries on Moving Objects

In this section, some approaches to process nearest neighbor queries (with a moving query point) on moving objects are reviewed. First, some proposals that obtain predictive query results are described. Then, other works focusing on the continuous monitoring of the results to the queries (without predictions) are examined.

8.2.1 *Predictive Nearest Neighbor Queries on Moving Objects.* There are works that make assumptions about the movement patterns of the target objects in order to precompute future results. For example (see Table X):

—In [Kollios et al. 1999a], the objects are assumed to move at a constant speed along a 1-D line segment. The approach presented allows to retrieve the moving object that will be the closest to the query point at a future time instant or sometime within a time interval specified by the user, but not the nearest neighbors at every time instant during that interval. A 2D space is also considered by restricting the objects to move in a given collection of line segments (e.g., road networks). For this, the route that passes closest to the query point is first identified and the nearest object in that route is found. Then, other points of routes closer than the distance to that object are examined similarly, so as to ensure that the actual nearest object is eventually retrieved.

—[Tao and Papadias 2003] considers kNN queries and introduces the concept of *time-parametrized queries* (*TP queries*). TP queries are based on the TPR-tree index structure [Saltenis et al. 2000], an extension of the R-tree that can be used to process predictive queries on moving objects. The answer to a spatio-temporal TP query has the form $<R, T, C>$, where R is the current result, T is the *expiry time* of R, and C is the set of objects that causes the change of the result at T; therefore, from R and C, the new result at T can be computed. Continuous queries can be answered by repetitive execution of TP queries; however, subsequent TP queries only need to retrieve the corresponding T and C components of the answer, as R can be computed by applying the changes C incrementally. It is assumed that no object changes its speed; otherwise, the query should be processed again.

Table X. Predictive nearest neighbor queries on moving objects

Work	Reevaluation	Indexing	Main features
[Kollios et al. 1999a]	Not considered	-Exact: hB^{π}-tree ([Evangelidis et al. 1995]) -Approximate: B+-tree ([Elmasri and Navathe 2003])	-1D segments -Road networks (1.5D) -Primal/dual space -Hough-X, Hough-Y -Linear motion function (1.5D)
[Tao and Papadias 2003] (*TPNN*)	Expiry time	-TPR-tree ([Tao et al. 2003b])	-Results: <R, T, C> -Influence time -Repetitive approach -Constant speed assumed
[Iwerks et al. 2003] (CW*: Continuous Windowing*) [Iwerks et al. 2006] (*Improved CW*: *iCW*)	On events	-Event B-tree ([Lu et al. 2000])	-w-event (within) -oc-event (order change) -nn-event (nearest) -Extended TP (ETP) -*kNN queries* -CW, iCW -Basis: range queries -*Spatial joins* -AE (All Events) -NE (Next Event)
[Raptopoulou et al. 2003]	Change of query's speed vector	-TPR-tree ([Tao et al. 2003b])	-Predictive queries -Result: <R,I> -Split points -Basis: speed vector
[Benetis et al. 2006]	Incremental maintenance	-TPR-tree ([Tao et al. 2003b])	-Predictive queries -Continuous queries (recomputation interval) -Also reverse kNN queries -Result: <NN_j/RNN_j,I_j> -Best/depth-first -Temporal *mindist*: *integral*, *min*

—In [Iwerks et al. 2003], the method proposed in [Tao and Papadias 2003] is extended (*extended TP*, *ETP*) to support updates such as speed variations. Additionally, the *Continuous Windowing* (*CW*) kNN algorithm is proposed, which is based on the observation that continuous range queries are easier to maintain than kNN queries. Thus, CW outperforms ETP. The work in [Iwerks et al. 2006] extends [Iwerks et al. 2003] by proposing the *iCW* (improved CW) algorithm and studying the maintenance of spatial join queries.

—A TPR-tree is used in [Raptopoulou et al. 2003] in order to determine the set of tuples <*R, I*>, where I is a time interval (within the time interval specified by the user) during which the result of the kNN query is the set of moving objects R. The repetitive approach proposed in [Tao and Papadias 2003] (which also uses a TPR-tree) is avoided and each tree node is accessed only once.

—Another approach for kNN queries that obtains the results for each time point within a time interval is presented in [Benetis et al. 2006]. As in [Raptopoulou et al. 2003], a TPR-tree is used. However, refined heuristics for searching and pruning are provided. Moreover, reverse kNN queries are also considered and persistent queries (queries whose answers are maintained incrementally under updates, as opposed to the definition in Section 3.2.2) and continuous queries (whose time interval advances as time progresses) are explicitly supported.

8.2.2 *Continuous Nearest Neighbor Queries on Moving Objects.* Some recent proposals focus on the processing of continuous kNN queries on moving objects. For example (see Table XI):

—In [Mouratidis et al. 2005a], every object in the answer is assigned a threshold which defines the distance range where it can move without making the result change. Based on this idea, the amount of location updates from the moving

Table XI. Continuous nearest neighbor queries on moving objects

Work	Targets	Reevaluation	Indexing	Main features
[Mouratidis et al. 2005a]	Moving	Location updates	Not considered	-Threshold-based -Focus: update policy -[Inverse] range quer.
[Yu et al. 2005]	Moving	Periodic	-Hierarchical grids ([Hinrichs 1985])	-Object indexing -Query indexing
[Mouratidis et al. 2005b] (*CPM*: *Conceptual Partitioning*)	Moving	Location updates (periodic handling)	-Grid ([Hinrichs 1985])	-Influence region

objects is minimized: Only objects that may influence some query result need to communicate their updated locations.

—[Yu et al. 2005] presents two grid-based algorithms: an incremental approach based on object-indexing and an alternative approach based on query-indexing (and motivated by [Kalashnikov et al. 2004], described in Section 5.3). The movement of a query is treated as the termination of the old query and the insertion of a new one, which could be inefficient.

—[Mouratidis et al. 2005b] proposes a *conceptual partitioning* (*CPM*), extended in [Mouratidis and Papadias 2007] to the sliding window model, where only the data points that belong to a sliding window are considered valid). A grid index, whose cells are organized in rectangles based on their proximity to the query point, is used. When a kNN query is reevaluated, its previous result is used to restrict the search space: Only a minimal set of cells (the *influence region* of the query) needs to be considered. As in the previous case, the movement of a query is treated as the termination of the old query an the insertion of a new one. According to the experimental evaluation presented, CPM outperforms the proposal in [Yu et al. 2005].

The methods described above require no knowledge of the movement patterns (e.g., speed vectors) of the objects or the queries.

9. QUERY PROCESSING APPROACHES FOR AGGREGATE SPATIO-TEMPORAL QUERIES

Aggregate location-dependent queries, which retrieve summarized information on moving objects, are also important within the context of this survey. These queries are useful in different application scenarios where obtaining detailed information about the moving objects that satisfy the query conditions is not required or even useless. Moreover, they are also very important when strong privacy must be ensured ("finding out how many instead of who" [Hadjieleftheriou et al. 2003]) or when the high mobility of the moving objects makes the retrieval of specific results irrelevant [Sun et al. 2006]. A classical aggregation predicate counts the number of objects that satisfy the query (called *spatio-temporal count* in [Tao et al. 2004b]), which may be useful, for example, in traffic control. A naive solution where the tuples satisfying the queries are first obtained and then processed to compute the aggregate result is not the best choice, as it is inefficient compared to an approach with some kind of pre-aggregation of results [Papadias et al. 2002]. Moreover, the underlying data may not be stored due to privacy reasons [Tao et al. 2004b].

It should be noted that if precise results are not required, the count query can be

solved using some approximation technique. The problem of approximate aggregate query processing can be assumed to be equivalent to the problem of *selectivity estimation* of the corresponding (non-aggregate) query[6], which is a key issue for plan optimization in query processing. Methods for spatial selectivity estimation, such as [Acharya et al. 1999], cannot be applied efficiently in moving object environments because those methods do not take the mobility of the objects into account [Choi and Chung 2002; Sun et al. 2006].

Some works focus on *historical spatio-temporal aggregation*. Works using multi-tree structures, and another one using a multi-tree index and a histogram, are described:

— *Multi-tree index structures*. In the context of spatio-temporal data warehouses, a couple of interesting works must be considered. [Papadias et al. 2002] proposes several multi-tree index structures that pre-aggregate values in different regions (e.g., road segments) and intermediate nodes to efficiently process spatio-temporal window aggregate queries: the *aggregate 3DR-tree* (*a3DR-tree*) and the *aggregate R-B-tree* (*aRB-tree*) for static regions, and the *aggregate historical R-B-tree* (*aHRB-tree*) and the *aggregate 3D R-B-tree* (*a3DRB-tree*) for dynamic regions (which may change their extent). A *B-File* (*BF*) is proposed to manage the B-trees of the aHRB-tree and a3DRB-tree. Several experiments compare these techniques and simple *column scanning* (using data cubes with time and space dimensions and a B-tree to index the blocks with information about each timestamp). For static regions, the aRB-tree is usually the best choice. For dynamic regions, the a3DRB-tree performs the best regarding both space consumption and query processing overhead, but it cannot be used for *online indexing*: If the lifespans of the entries are not known in advance, then the aHRB-tree should be used instead. [Papadias et al. 2002] is extended in [Tao and Papadias 2005], that considers the a3DR-tree, the aRB-tree, the a3DRB-tree, the *aggregate multi-version R-B-tree* (*aMVRB-tree*, similar to the aHRB-tree), and the B-Files, and presents a cost analysis.

— *Histograms with multi-tree index structures*. [Tao et al. 2004b] proposes a solution that integrates spatio-temporal aggregate indexes with sketches, based on the *FM algorithm* for approximate counting. The main motivation is that the approaches for historical spatio-temporal aggregation commented above (e.g., the proposal using aRB-trees) have the *distinct counting problem*: An object may be counted more than once if it qualifies the query during several timestamps in the query interval. This problem cannot be solved by duplicate elimination, as only aggregate information is managed. Therefore, the *sketch index* (inspired by the aRB-tree) is proposed, which can also be used to reduce the amount of spatio-temporal data stored (approximation) and to mine spatio-temporal association rules.

There are also some works that consider the problem of *selectivity estimation of predictive queries*:

— *An approach using spatio-temporal histograms partitioned based on locations*. Predictive static (interval and timestamp) window queries are considered in [Choi

[6]Although the selectivity of a query is usually defined relative to the cardinality of the dataset, this distinction is not emphasized in the context of spatio-temporal data; e.g., see [Sun et al. 2006].

and Chung 2002]. Selectivity estimation is performed independently on the X and Y dimensions of space, and the global selectivity is obtained by multiplying both estimates. This approach uses a spatio-temporal histogram, inspired by the idea of time-parameterized bounding rectangles in the TPR-tree [Saltenis et al. 2000] and built using the MinSkew algorithm [Acharya et al. 1999]. The histogram is updated periodically (in the experiments presented, every second) from sample data, which is possible thanks to the low I/O overhead of the update strategy.

— *Approaches using spatio-temporal histograms partitioned based on both locations and speeds.* As opposed to the previous work, [Tao et al. 2003d] addresses selectivity estimation directly in a two-dimensional space, as the proposal in [Choi and Chung 2002] is shown to be inaccurate: In the case of interval queries, that method can over-count because the temporal condition is ignored when projecting onto the two spatial dimensions. Another advantage of this new proposal is that it also supports objects with rectangular extents and moving queries. It uses a spatio-temporal histogram, also based on the idea of time-parameterized bounding rectangles. However, partitioning the data considering not only the location but also the velocity information is advocated. The histogram is modified incrementally, when an object update is received, and it is re-built when the number of modifications reaches a certain threshold in order to adapt it to the new data distribution. In [Tao et al. 2003c], the authors extend this work to consider also spatio-temporal join queries and kNN queries. In the case of kNN queries, whose selectivity is trivially determined by the value of the requested k, the goal is to estimate the nearest distance from the query point to the k^{th} nearest object (called *nearest distance queries* in [Yiu et al. 2008]), which has a strong influence on the cost of the kNN query and therefore is the important factor for query optimization. Finally, [Sun et al. 2006] focuses only on predictive spatio-temporal joins.

— *Dual-based approaches.* In [Hadjieleftheriou et al. 2003], two new approaches for estimating the selectivity of static spatio-temporal window queries are considered. One approach is based on building histograms using the duality transform (velocity-intercept representation). The other approach, applicable on the dual or the primal space, benefits from the availability of an R-tree-like index method and uses a hash table whose entries are associated to leaf nodes in the tree: Each entry contains the number of objects of the corresponding MBR. These two methods are compared experimentally with the proposals presented in [Choi and Chung 2002; Tao et al. 2003d] and also with random sampling. The experiments presented indicate that further research is needed to develop accurate selectivity estimation methods for skewed distributions. Moreover, they show that the index-based estimator is only the best choice for a one-dimensional space and that histograms (for the case of uniform datasets) and random sampling (for objects moving in road networks[7]) are preferred for a two-dimensional space. From the different histogram-based methods evaluated, the one proposed in this work performs similarly to the proposal in [Tao et al. 2003d] and better than [Choi and Chung 2002]. Aggregate spatio-temporal window queries, and the problem of obtaining the distance to the

[7]According to [Tao et al. 2003c], road networks are not really considered in this work, but objects moving "on a set of infinite lines defined by the original road segments".

k^{th} nearest object, are also considered in [Yiu et al. 2008], where the B^{dual}-tree (see Section 4.1.3) is proposed to index moving objects in the dual space.

Finally, [Sun et al. 2004] considers *both historical and predictive queries*. For present-time queries, an *adaptive multidimensional histogram* (*AMH*) is proposed, which does not need periodic re-construction. Instead, idle processing cycles are used for bucket reorganization (merging and splitting). Moreover, a binary-partition tree (BPT) is used to increase the efficiency of these queries and the maintenance of the histogram. For historical queries, a *historical synopsis* is used, consisting of an AMH along with a disk-based index (a packed B-tree for high update rates or a 3D R-tree for heavy query workloads) that stores the less recent buckets that do not fit in main memory. For future queries, *exponential smoothing* is proposed[8].

10. MANAGEMENT OF LOCATION-DEPENDENT DATA

Some works focus on *location-dependent data* (LDD), which can be defined as "data whose value is determined by the location to which it is related" [Ren and Dunham 2000]. Location-dependent data usually involves general information whose precise contents change according to the geographical area (depending on the *data region*, following the terminology introduced in [Dunham and Kumar 1998]). Thus, local yellow pages, traffic reports, and weather or entertainment information are examples of location-dependent data. Although the initial definition given above does not allow it, data about static (i.e., not moving) objects, such as hotels and vending machines, could also be considered location-dependent data, as similar techniques can be applied in such a case[9]. It should be noted that, as opposed to more general location-dependent queries, only the user is allowed to move (a queried data item may have a relevant location/scope, but it cannot change). However, it should be indicated that this distinction was not made in some pioneering papers in the field; for example, [Imielinski and Nath 1993] considers queries such as "where is the nearest doctor" and "what is the number of taxi cabs in the area" as queries on location-dependent data. In the rest of this section, the main research issues related to location-dependent data are reviewed.

10.1 Caching and Prefetching LDD

Some works focus on caching issues for queries on location-dependent data. The *semantic caching* model [Dar et al. 1996], for general client-server database systems, proposes storing the descriptions of previously issued queries together with their results. In this way, a new query could be totally or partially answered from the cache. For queries only partially answered, the query is trimmed from the cached ones and a *remainder query* is submitted to the server.

In [Lee et al. 1999], the semantic caching model is adapted to the mobile environment, with the goal of improving not only the data access performance but also the availability. For the queries, each cached item is saved along with its *valid*

[8]Other works, such as [Tao et al. 2003c; Sun et al. 2006], also emphasize the importance of exponential smoothing for future queries in situations where knowledge about the velocity vectors of the objects cannot be assumed (e.g., because they can change abruptly, as in road networks).
[9]Some relevant works about processing queries on static objects are described in Sections 7.4 and 8.1.

scope, which is the spatial area (spatial scope) within which the data is known to be valid [Zheng et al. 2002; Zheng et al. 2003a]. In [Zheng and Lee 2001; Zheng et al. 2004], a semantic caching scheme for nearest neighbor queries is proposed, based on the construction of a Voronoi diagram [Okabe et al. 2000] (used as an index on the objects) and the idea of validity region [Zhang et al. 2003; Tao et al. 2003a]; [Gao and Hurson 2005] claims that the computation of the validity region is expensive and proposes a scheme to estimate it instead. In [Ren and Dunham 2000], a *FAR* (*Farthest Away Replacement*) cache replacement policy is proposed, which chooses the data which are the farthest from the user for replacement. Several other location-dependent cache invalidation schemes have been proposed; for example, see [Xu et al. 1999; Zheng et al. 2002; Xu et al. 2003a].

Semantic caching poses, however, several problems [Hu et al. 2005b]. First, the granularity of cache reuse is at the query level, so only queries of the same type can reuse the cached data. Second, it only supports certain kinds of queries: NN queries and range queries have been studied, but it is difficult to support more complex queries such as kNN or spatial join queries. Third, the management of the cache is complex; for example, when a new query to be cached overlaps a cached query, a decision has to be made regarding whether to coalesce both queries or trim one of them. And finally, the organization is plain, requiring sequential scans for query processing and cache replacement. For those reasons, a *proactive caching* is proposed in [Hu et al. 2005b]. With proactive caching, the granularity of cache reuse is at object level.

[Ku et al. 2005] studies data caching for location-based spatial queries in *P2P* environments[10]. The focus is on kNN queries, and *mobile cooperative caching* [Chow et al. 2004] is proposed to share query results and reduce the communications with the spatial database server. The IER algorithm proposed in [Papadias et al. 2003] (see Section 8.1.2) is extended to use cached results in a P2P sharing infrastructure.

Finally, it should be stressed that caching can be combined with prefetching strategies in order to improve data availability (e.g., in the event of client disconnections or when bandwidth is scarce), as suggested in works such as [Kubach and Rothermel 2001a; 2001b; Kirchner et al. 2004; Burcea et al. 2004].

10.2 Broadcasting LDD

Wireless broadcasting has been considered as an effective and economic way of disseminating data to a large number of mobile users, with the additional benefit of privacy (mobile users do not need to issue queries or communicate their locations). The concept of *broadcast disks* was proposed for organizing the contents of a data *broadcast program* and for managing client resources in response to such a program [Acharya et al. 1995b; Acharya et al. 1996; 1997]. Initially, research focused on a *push-based* approach, where data are sent out on the broadcast channel according to a periodic schedule [Acharya et al. 1995b; Acharya et al. 1996]. Later, a *pull-based* model, that uses a backchannel to allow clients to send explicit requests for data to the server, was also considered [Acharya et al. 1997]. There are also other *hybrid* (push-and-pull) approaches [Aksoy and Franklin 1999; Stathatos et al.

[10]There are other works that consider the P2P paradigm for computing spatial queries. For example, [Ku et al. 2005] proposes the *peer-tree*, a P2P version of a centralized R-tree.

1997]. The *RxW* algorithm [Aksoy and Franklin 1999] schedules data items that are requested frequently or have long-standing requests. The adaptive hybrid delivery proposed in [Stathatos et al. 1997] is based on the idea of *air-caching*, where the broadcast schedule is adapted to the request rate of data items (*broadcast misses*).

Some works specifically focus on broadcasting location-dependent data, such as [Hambrusch et al. 2001; Jung et al. 2002; Xu et al. 2003b; Zheng et al. 2003b; Acharya and Kumar 2005; Lee and Zheng 2005; Waluyo et al. 2005; Lee 2007]. Along the same lines, there are some works, such as [Burcea and Jacobsen 2003; Eugster et al. 2003; Chen et al. 2004; Zeidler 2004], that propose the use of the publish/subscribe paradigm as an information dissemination model for location-based services. Finally, index structures that are appropriate for broadcasting are being investigated (e.g., [Waluyo et al. 2005; Zheng et al. 2006]). *Air indexing* increases access latency, as broadcast cycles are longer due to the need to transmit the index information. However, the benefit is that they decrease the tuning time, and so the mobile devices can save energy by entering the power saving mode.

10.3 Architectures for LDD

Some works focus on defining appropriate architectures for storage and management of location-dependent data. For example:

—In [Imielinski and Nath 2002], the term *dataspace* is used to denote a new kind of physical space that is connected to the network. In this space, data is stored where it is produced, and so it is completely distributed. Thus, for example, messages can be attached to locations (e.g., sensors) and users can query these locations to obtain different types of location-dependent data (e.g., pollution information, announcements, etc.). Thus, the users directly query the physical world [Bonnet et al. 2000]. In this context, research in the field of wireless sensor networks (e.g., see [Chang and Lee 2007]) is of major importance.

—In [Gupta et al. 2005], a hierarchical framework is proposed for handling continuous queries on location-dependent data from a single zone (*single-ZQ*) or a set of neighbor zones (*multiple-ZQ*), focusing on data distribution issues. Works with similar concerns are [Madria et al. 2000; Ryu et al. 2003].

—[Wu and Wu 2006] proposes a technique, called *neighboring replication*, to store replicas of data in different servers with the goal of increasing the efficiency of range queries which may span the spatial boundaries of several local servers. The key factors studied are the replication condition, the selection of the replication range, and also dynamic adjustment strategies by which the replication range is expanded or reduced periodically by using statistics collected to estimate the tradeoff between update cost and query cost.

An appropriate architecture to manage location-dependent data is a key element to enable an efficient query processing. This section concludes with the reminder that queries on location-dependent data could be seen as a particular case of location-dependent queries. For example, approaches to process queries on location-dependent data benefit from the fact that only the user issuing the query moves. Therefore, they cannot be used to process more general queries.

11. CONCLUSIONS AND PROSPECTIVE RESEARCH

This article presents a review of the state of the art in the processing of location-dependent queries. First, the technological context of mobile computing has been described and, particularly, different types of location-based services and location-dependent queries have been introduced. Then, several data management approaches for moving objects (moving object databases and data stream processing) are described. Finally, several query processing approaches in different categories have been considered: 1) approaches that require some cooperation from the moving objects in order to process the queries (MQM, MobiEyes, the Q-index approach, and the SRB framework), 2) approaches that assume some knowledge about the moving objects' trajectories (such as DOMINO, CAT/CAT++, and ARGONAUT), 3) approaches that do not rely on the cooperation of moving objects or on the knowledge of the objects' trajectories (such as PLACE, SCUBA, and LOQOMOTION), 4) approaches that focus only on nearest neighbor queries, 5) approaches for aggregate queries, and 6) works considering queries over location-dependent data. This article is concluded with a comparative summary of the main query processing approaches and some challenges that could encourage future research.

11.1 Comparative Summary of Main Query Processing Approaches

As a summary, key distinctive features of the main approaches considered for the processing of location-dependent queries are collected (in the order described) in Table XII, and Table XIII shows the most important differences among them[11]. In Table XIII, the following interesting features are considered: 1) whether the approach supports continuous queries (column "CQ"); 2) whether it supports range queries (column "RQ"); 3) whether it supports nearest neighbor queries (column "kNN"); 4) whether it supports moving queries (column "MQ"), i.e., queries whose relevant areas move because, in the terminology of MobiEyes/LOQOMOTION, the focal/reference objects move; 5) whether it supports queries about moving (i.e., not only static) objects (column "MO"); 6) whether it does not require a specific location update policy for the moving objects (column "Any pol."); 7) whether it does not assume any knowledge about the expected trajectories of the moving objects (column "No traj."); 8) whether it does not perform any query processing on the moving objects (column "Obj. free"); 9) whether the approach proposes a distributed solution (column "Distr."); 10) whether it concerns retrieving the current locations of the objects in the answer (column "Gets loc."); and 11) whether it works with different location granularities (column "Loc. gran."). The $\surd$ symbol is used to indicate a supported feature, whereas the $\approx$ symbol means that it is supported only partially. This table cannot be considered in isolation and the reader is referred to the description of the works for more details; for example, DOMINO proposes some ideas for a distributed query processing (see Section 6.1) although this is not indicated in the table, and some works need some trajectory knowledge to process predictive queries, but they do not need it for other types of queries (e.g., the work on dynamic queries described in Section 7.3). The table shows that LOQOMOTION supports all the interesting features indicated, but this

[11] Works on specific types of queries, such as nearest neighbor queries or aggregate queries, and studies on location-dependent data are not shown in these tables.

Table XII. Distinctive features of the main approaches for location-dependent query processing

Work	Brief description
MQM	*Static range queries* are processed with cooperation of the moving objects: each object monitors the queries related to its *resident domain*. An object updates its location on a centralized server when it crosses a query boundary or its resident domain.
MobiEyes	A *moving range query* is processed with the help of the moving objects located within its *monitoring region*. Each object monitors the queries it can affect and communicates location updates to a centralized server whenever it is needed.
Q-index	*Static range queries* are processed on a centralized server using two index structures: a *query index* and a *velocity-constraint index* for the moving objects. An object communicates updates when it crosses its *safe region* (computed based on the boundaries of the queries).
SRB	*Static range* and *kNN queries* are processed by indexing the moving objects and the *quarantine areas* of the queries. An object communicates an update when it crosses its *safe region*. For kNN queries, the server may need to request updates explicitly.
DOMINO	A model to represent moving objects in databases (*MOST*) is proposed, based on the concept of *dynamic attribute*. The query language *FTL* is defined. Some aspects of query processing, based on trajectories, are considered. Some location update policies are also proposed.
O. Ulusoy et al.	The focus is on defining suitable strategies to transmit the results of a continuous query to a mobile user. Different location update policies are also proposed.
CAT/CAT++	A *trajectory model* is used to represent the moving objects. Continuous static range queries are processed using *triggers*. Persistent queries are also considered.
ARGONAUT	An incremental evaluation (with a filter step and a refinement step) is proposed to process *continuous range queries* on objects (represented according to the *mSTOMM* model) moving in path networks. An *SR*-tree* indexes the segments of the network.
PLACE	The *SINA* algorithm processes *moving range queries* by joining queries and objects. Continuous kNN queries are considered in *SEA-CNN*. A general skeleton for query processing (*GPAC*) and an online data stream algorithm (*SOLE*) are also proposed.
SCUBA	*Moving clusters* of objects and queries are considered. Query evaluation implies: 1) a *join-between* clusters, and 2) a fine-grained *join-within* clusters. *ClusterSheddy* provides incremental evaluation and load-shedding. Implemented within the DSMS *CAPE*.
Dynamic queries	An index-based approach to process *moving range queries* is proposed, which aims at minimizing the number of disk accesses by avoiding the recomputation of overlapping results in each periodic reevaluation. Predictive queries are also considered.
Monash University	The focus is on *range queries on static objects regarding the location of the mobile user*. The data about the static objects are stored in a distributed environment.
LOQOMOTION	A general software architecture, based on *mobile agents*, processes *continuous queries* on a completely *distributed infrastructure* of location servers. The locations of the objects in the answers are retrieved. It supports different types of *location granules*.

Table XIII. Differences between the main approaches for location-dependent query processing

	Types of queries					Assumptions		Special features			
Work	CQ	RQ	kNN	MQ	MO	Any pol.	No traj.	Obj. free	Distr.	Gets loc.	Loc. gran.
MQM	√	√			√		√		≈		
MobiEyes	√	√	≈	√	√		≈		≈		
Q-index	√	√			√		√	≈			
SRB	√	√	√		√		≈	≈			
DOMINO	≈	≈		≈	≈	√		√			
O. Ulusoy et al.	≈	≈		≈	≈	√		√			
CAT/CAT++	√	√			√	√		√			
ARGONAUT	√	√		√	√	√		√			
PLACE	√	√	√	√	√	√	√	√			
SCUBA	√	√	√	√	√	√	√	√			
Dynamic Queries	√	√		√	√	√	√	√		√	
Monash University		√		≈		√	√	√	√		
LOQOMOTION	√	√	√	√	√	√	√	√	√	√	√

is precisely because its main goal is to propose a general architecture suitable to different requirements; however, this does not imply that it is the best solution in every scenario: Due to its generality, specific types of queries could be processed more efficiently using other approaches, as explained in Section 7.5.

11.2 Future Research Directions

Future trends in the field of location-dependent query processing are yet to be unveiled. Some interesting future directions are indicated in the following:

— Currently, most works consider a centralized architecture, with a central server processing all the queries and managing the location data of all the moving objects. Therefore, many interesting proposals are limited to the scalability supported by a single computer, which also represents a single point of failure. Moreover, the single-computer assumption is not realistic in a large-scale setting. Thus, for example, a centralized computer could probably manage the location information of a fleet of vehicles (e.g., taxis) in a city. However, distributed approaches would be required when there is a large number of moving objects (maybe of different types, such as taxis, buses, people, etc.) and mobile users issuing queries over a large geographical area. Indeed, the moving objects themselves are distributed through the geographical space by nature. Some proposals already exist to overcome this problem, and future works will probably focus on distributed environments. Moreover, works on interoperability and integration of heterogeneous systems for the processing of location-dependent queries in a large-scale setting should be expected. Standardization efforts like those described in Section 3.1.3 will play a major role.

— At present, most works assume the existence of repositories to store the location information of moving objects. Indeed, a lot of research has been devoted to analyze efficient ways of representing and querying moving object databases. An interesting alternative is the use of data stream technology. As it was mentioned in this article, sometimes the difference between these two approaches is not always clear in the context of this survey, as approaches based on moving object databases also use techniques to deal efficiently with continuous updates. In any case, there is an ongoing research effort on the development of general-purpose data stream management systems that could be very interesting in this field. Thus, as data stream technology matures and debugged implementations become available, an increase in proposals based on the use of this technology might be expected. Processing location data as data streams may lead to an enhanced scalability of the query processing approaches, although it may also introduce some challenges (e.g., how to efficiently process joins of location data streams in a distributed environment).

— Most of the proposals so far do not delve into the details of the underlying communication infrastructure: They just assume the existence of Internet-like connections that allow, for example, the communication of a moving object with a fixed computer. However, in a variety of settings this is just not possible. Much research is being performed now in the field of *Inter-Vehicle Communications* (*IVC*) [Tsugawa 2005], where a global communication infrastructure is not available and two cars can only communicate, using wireless communications, while they are within communication range of each other. In this scenario, processing location-dependent queries is a challenge. Some works have already considered the processing of very specific queries in these environments (e.g., searching available parking spots in [Xu et al. 2004]). The importance of proposing solutions for these intermittent and ad hoc networks will increase, as they are also of great interest in scenarios of emergency or disaster situations where existing global communication infrastructures

may be damaged.

— Testing an approach for the processing of location-dependent queries is challenging. Most works use simulation environments (whose analysis may deserve another survey). However, it is usually impossible to compare the efficiency of two different approaches, as they probably use different simulation environments and are also based on a different set of assumptions. This problem seems very difficult to solve, although some proposals similar to the Linear Road Benchmark, mentioned in Section 4.2.2 in the context of data streams, could be expected in the future. The availability of advanced devices at reduced prices could also encourage testing in real scenarios, although in that case controlling all the factors affecting the results (to ensure repeatable experiments) is very challenging indeed.

— There is still no consensus about how to deal with different data management issues. For example, the amount of indexing structures proposed for moving objects is overwhelming and new proposals appear every year. In this sense, a future winner is not expected but rather an integrated approach that adapts the indexing strategy depending on different factors.

— Network limitations will continue being a bottleneck in the future. Although better network connections will be available, the needs of the mobile users will also probably increase. Thus, location-dependent queries could be enhanced with multimedia information. For example, retrieving video feeds from the moving objects that satisfy the constraints of a query will offer extra information about the objects' surroundings.

— The design of appropriate user interfaces for small devices will also attract interest (e.g., market researches predict an important growth in the number of GPS-enabled mobile phones).

— Finally, privacy protection and security (see Section 3.1.2) are always major concerns and new proposals will continue appearing in the future, which will probably have an influence on the query processing strategies.

So, despite an intensive and fruitful research effort in the area, important challenges lie ahead. This ensures that the processing of location-dependent queries will remain an active research topic for at least some years to come. This survey provides a structured view of the state of the art and should encourage future steps.

ACKNOWLEDGMENTS

This work was supported by the CICYT project TIN2007-68091-C02-02. We would also like to thank the anonymous reviewers for their useful suggestions.

REFERENCES

Aalto, L., Göthlin, N., Korhonen, J., and Ojala, T. 2004. Bluetooth and WAP push based location-aware mobile advertising system. In *2nd International Conf. on Mobile Systems, Applications, and Services (MobiSys'04)*. ACM, New York, NY, USA, 49–58.

Acharya, A., Imielinski, T., and Nath, B. 1995a. DATAMAN project: Towards a mosaic-like location-dependant information service for mobile clients. *MOBIDATA: An Interactive Journal of Mobile Computing 1,* 2 (Apr.).

Acharya, D. and Kumar, V. 2005. Indexing location dependent data in broadcast environment. *Journal of Digital Information Management 3,* 2 (June), 114–118.

Acharya, S., Alonso, R., Franklin, M. J., and Zdonik, S. 1995b. Broadcast disks: Data management for asymmetric communications environments. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'95)*. ACM, New York, NY, USA, 199–210.

Acharya, S., Franklin, M., and Zdonik, S. 1997. Balancing push and pull for data broadcast. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'97)*. ACM, New York, NY, USA, 183–194.

Acharya, S., Franklin, M. J., and Zdonik, S. 1996. Prefetching from broadcast disks. In *12th International Conf. on Data Engineering (ICDE'96)*. IEEE Computer Society, Washington, DC, USA, 276–285.

Acharya, S., Poosala, V., and Ramaswamy, S. 1999. Selectivity estimation in spatial databases. *SIGMOD Record 28,* 2 (June), 13–24.

Agarwal, P. K., Arge, L., and Erickson, J. 2003. Indexing moving points. *Journal of Computer and System Sciences 66,* 1 (Feb.), 207–243.

Aggarwal, C. C. and Agrawal, D. 2003. On nearest neighbor indexing of nonlinear trajectories. In *22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'03)*. ACM, New York, NY, USA, 252–259.

Aksoy, D. and Franklin, M. 1999. R × w: a scheduling approach for large-scale on-demand data broadcast. *IEEE/ACM Trans. Netw. 7,* 6 (Dec.), 846–860.

Alisi, T. M., Bimbo, A. D., and Valli, A. 2005. Natural interfaces to enhance visitors' experiences. *IEEE Multimedia 12,* 3 (July), 80–85.

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

Alonso, R. and Korth, H. F. 1993. Database system issues in nomadic computing. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'93)*. ACM, New York, NY, USA, 388–392.

Amir, A., Efrat, A., Myllymaki, J., Palaniappan, L., and Wampler, K. 2007. Buddy tracking – efficient proximity detection among mobile friends. *Pervasive and Mobile Computing 3,* 5 (Oct.), 489–511.

Anastasi, G., Conti, M., Gregori, E., and Passarella, A. 2003. Balancing energy saving and QoS in the mobile internet: An application-independent approach. In *36th Hawaii International Conf. on System Sciences (HICSS'03)*. IEEE Computer Society, Washington, DC, USA, 305–314.

Andersson, C. 2001. *GPRS and 3G Wireless Applications.* Wiley, New York, NY, USA.

Arasu, A., Babu, S., and Widom, J. 2003. CQL: A language for continuous queries over streams and relations. In *9th International Workshop on Database Programming Languages (DBPL'03)*. Springer, Berlin/Heidelberg, 1–19.

Arasu, A., Cherniack, M., Galvez, E. F., Maier, D., Maskey, A., Ryvkina, E., Stonebraker, M., and Tibbetts, R. 2004. Linear Road: A stream data management benchmark. In *30th International Conf. on Very Large Data Bases (VLDB'04)*. Morgan Kaufmann, San Francisco, CA, USA, 480–491.

Atallah, M. J. and Frikken, K. B. 2004. Privacy-preserving location-dependent query processing. In *IEEE/ACS International Conf. on Pervasive Services (ICPS'04)*. IEEE Computer Society, Washington, DC, USA, 9–17.

Avnur, R. and Hellerstein, J. M. 2000. Eddies: Continuously adaptive query processing. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'00)*. ACM, New York, NY, USA, 261–272.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. 2002. Models and issues in data stream systems. In *21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'02)*. ACM, New York, NY, USA, 1–16.

Babu, S. and Widom, J. 2001. Continuous queries over data streams. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'01)*. ACM, New York, NY, USA, 109–120.

Bahl, P. and Padmanabhan, V. N. 2000. RADAR: An in-building RF-based user location and tracking system. In *19th Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'00)*. IEEE Computer Society, Washington, DC, USA, 775–784.

Baldauf, M., Dustdar, S., and Rosenberg, F. 2006. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing 2,* 4 (June), 263–277.

Bar-Noy, A., Kessler, I., and Sidi, M. 1995. Mobile users: To update or not to update? *Wireless Networks 1,* 2 (June), 175–185.

Barbará, D. 1999. Mobile computing and databases – a survey. *IEEE Trans. on Knowl. and Data Eng. 11,* 1 (Jan.), 108–117.

Barbará, D., DuMouchel, W., Faloutsos, C., Haas, P. J., Hellerstein, J. M., Ioannidis, Y. E., Jagadish, H. V., Johnson, T., Ng, R. T., Poosala, V., Ross, K. A., and Sevcik, K. C. 1997. The New Jersey data reduction report. *IEEE Data Eng. Bull. 20,* 4 (Dec.), 3–45.

Barbará, D. and Imielinski, T. 1995. Sleepers and workaholics: Caching strategies in mobile environments. *The VLDB Journal 4,* 4 (Oct.), 567–602.

Basagni, S., Chlamtac, I., and Syrotiuk, V. R. 1999. Geographic messaging in wireless ad hoc networks. In *49th IEEE Vehicular Technology Conf. (VTC'99)*. IEEE Computer Society, Washington, DC, USA, 1957–1961.

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

Beckmann, N., Kriegel, H., Schneider, R., and Seeger, B. 1990. The R*-tree: an efficient and robust access method for points and rectangles. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'90)*. ACM, New York, NY, USA, 322–331.

Benetis, R., Jensen, S., Karciauskas, G., and Saltenis, S. 2006. Nearest and reverse nearest neighbor queries for moving objects. *The VLDB Journal 15,* 3 (Sept.), 229–249.

Berchtold, S., Keim, D. A., and Kriegel, H. 1996. The X-tree: An index structure for high-dimensional data. In *22th International Conf. on Very Large Data Bases (VLDB'96)*. Morgan Kaufmann, San Francisco, CA, USA, 28–39.

Biveinis, L., Saltenis, S., and Jensen, C. S. 2007. Main-memory operation buffering for efficient R-tree update. In *33rd International Conf. on Very Large Data Bases (VLDB'07)*. ACM, New York, NY, USA, 591–602.

Bohn, J., Coroama, V., Langheinrich, M., Mattern, F., and Rohs, M. 2004. Living in a world of smart everyday objects – social, economic, and ethical implications. *Journal of Human and Ecological Risk Assessment 10,* 5 (Oct.), 763–786.

Bonnet, P., Gehrke, J., and Seshadri, P. 2000. Querying the physical world. *IEEE Personal Comms. 7,* 5 (Oct.), 10–15.

Brinkhoff, T. 2002. The impact of filtering on spatial continuous queries. In *International Symposium on Spatial Data Handling (SDH'02)*. Springer, 41–54.

Budiarto, Harumoto, K., Tsukamoto, M., and Nishio, S. 1997. Position locking: Handling location dependent queries in mobile computing environment. In *International Conf. on Worldwide Computing and Its Applications (WWCA'97)*. Springer, London, UK, 363–378.

Burcea, I. and Jacobsen, H. 2003. L-ToPSS – push-oriented location-based services. In *4th International Workshop of Technologies for E-Services (TES'03)*. Springer, Berlin/Heidelberg, 131–142.

Burcea, I., Jacobsen, H., de Lara, E., Muthusamy, V., and Petrovic, M. 2004. Disconnected operation in publish/subscribe middleware. In *5th International Conf. on Mobile Data Management (MDM'04)*. IEEE Computer Society, Washington, DC, USA, 39–50.

Burrough, P. A. and McDonnell, R. 1998. *Principles of Geographical Information Systems.* Oxford University Press, Oxford, UK.

Cai, Y. and Hua, K. A. 2002. An adaptive query management technique for efficient real-time monitoring of spatial regions in mobile environments. In *21st IEEE International Performance, Computing, and Communication Conf. (IPCCC'02)*. IEEE Computer Society, Washington, DC, USA, 259–266.

Cai, Y., Hua, K. A., and Cao, G. 2004a. Processing range-monitoring queries on heterogeneous mobile objects. In *5th International Conf. on Mobile Data Management (MDM'04)*. IEEE Computer Society, Washington, DC, USA, 27–38.

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

Cai, Y. D., Clutter, D., Pape, G., Han, J., Welge, M., and Auvil, L. 2004b. MAIDS: Mining alarming incidents from data streams. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'04)*. ACM, New York, NY, USA, 919–920.

Cao, H., Wang, S., and Li, L. 2003. Location dependent query in a mobile environment. *Information Sciences 154,* 1–2 (Aug.), 71–83.

Carzaniga, A., Picco, G. P., and Vigna, G. 1997. Designing distributed applications with a mobile code paradigm. In *19th International Conf. on Software Engineering (ICSE'97)*. ACM, New York, NY, USA, 22–32.

Chang, C.-C., Lin, I.-C., and Lin, C.-C. 2008. A novel location tracking scheme for reducing location updating traffic in a personal communication system. *Wireless Personal Communications 44,* 2 (Jan.), 139–152.

Chang, J.-W., Um, J.-H., and Lee, W.-C. 2006. A new trajectory indexing scheme for moving objects on road networks. In *23rd British National Conf. on Databases (BNCOD'06)*. Springer, Berlin/Heidelberg, 291–294.

Chang, R.-S. and Lee, A.-C. 2007. An energy efficient data query architecture for large scale sensor networks. *IEICE Transactions on Communications E90-B,* 2 (Feb.), 217–227.

Chang, Y. 2003. A graphical query language for mobile information systems. *SIGMOD Record 32,* 1 (Mar.), 20–25.

Chen, G. and Kotz, D. 2000. A survey of context-aware mobile computing research. Tech. rep., Dartmouth College.

Chen, Y., Chen, X. Y., Rao, F. Y., Yu, X. L., Li, Y., and Liu, D. 2004. LORE: An infrastructure to support location-aware services. *IBM J. Res. and Development 48,* 5/6 (Sept.), 601–615.

Chen, Y., Rao, F., Yu, X., Liu, D., and Zhang, L. 2003. Managing location stream using moving object database. In *6th International DEXA Workshop on Mobility in Databases and Distributed Systems (MDDS'03)*. IEEE Computer Society, Washington, DC, USA, 916–920.

Cheng, R., Chen, J., Mokbel, M. F., and Chow, C.-Y. 2008. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *24th IEEE International Conf. on Data Engineering (ICDE'08)*. IEEE Computer Society, Washington, DC, USA, 973–982.

Cheng, R., Kalashnikov, D. V., and Prabhakar, S. 2003a. Evaluating probabilistic queries over imprecise data. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'03)*. ACM, New York, NY, USA, 551–562.

Cheng, R., Kalashnikov, D. V., and Prabhakar, S. 2004a. Querying imprecise data in moving object environments. *IEEE Trans. on Knowl. and Data Eng. 16,* 9 (Sept.), 1112–1127.

Cheng, R., Prabhakar, S., and Kalashnikov, D. V. 2003b. Querying imprecise data in moving object environments. In *19th International Conf. on Data Engineering (ICDE'03)*. IEEE Computer Society, Washington, DC, USA, 723–725.

Cheng, R., Xia, Y., Prabhakar, S., and Shah, R. 2005. Change tolerant indexing for constantly evolving data. In *21st International Conf. on Data Engineering (ICDE'05)*. IEEE Computer Society, Washington, DC, USA, 391–402.

Cheng, R., Xia, Y., Prabhakar, S., Shah, R., and Vitter, J. S. 2004b. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *30th International Conf. on Very Large Data Bases (VLDB'04)*. Morgan Kaufmann, San Francisco, CA, USA, 876–887.

Cheng, R., yiu Lam, K., Prabhakar, S., and Liang, B. 2007. An efficient location update mechanism for continuous queries over moving objects. *Inf. Syst. 32,* 4 (June), 593–620.

Cheverst, K., Davies, N., Mitchell, K., and Friday, A. 2000. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. In *6th International Conf. on Mobile Computing and Networking (MobiCom'00)*. ACM, New York, NY, USA, 20–31.

Choi, Y.-J. and Chung, C.-W. 2002. Selectivity estimation for spatio-temporal queries to moving objects. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'02)*. ACM, New York, NY, USA, 440–451.

Chon, H. D., Agrawal, D., and Abbadi, A. E. 2001a. Storage and retrieval of moving objects. In *2nd International Conf. on Mobile Data Management (MDM'01)*. Springer, Berlin/Heidelberg, 173–184.

Chon, H. D., Agrawal, D., and Abbadi, A. E. 2001b. Using space-time grid for efficient management of moving objects. In *2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'01)*. ACM, New York, NY, USA, 59–65.

Chon, H. D., Agrawal, D., and Abbadi, A. E. 2002a. Data management for moving objects. *IEEE Data Eng. Bull. 25,* 2 (June), 41–47.

Chon, H. D., Agrawal, D., and Abbadi, A. E. 2002b. NAPA: Nearest available parking lot application. In *18th International Conf. on Data Engineering (ICDE'02)*. IEEE Computer Society, Washington, DC, USA, 496–497.

Chon, H. D., Agrawal, D., and Abbadi, A. E. 2003. Range and kNN query processing for moving objects in grid model. *Mobile Networks and Applications 8,* 4 (Aug.), 401–412.

Chow, C., Leong, H. V., and Chan, A. T. S. 2004. Peer-to-peer cooperative caching in a hybrid data delivery environment. In *7th International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN'04)*. IEEE Computer Society, Washington, DC, USA, 79–85.

Civilis, A., Jensen, C. S., Nenortaite, J., and Pakalnis, S. 2004. Efficient tracking of moving objects with precision guarantees. In *1st Annual International Conf. on Mobile and Ubiquitous Systems (MobiQuitous'04)*. IEEE Computer Society, Washington, DC, USA, 164–173.

Civilis, A., Jensen, C. S., and Pakalnis, S. 2005. Techniques for efficient road-network-based tracking of moving objects. *IEEE Trans. on Knowl. and Data Eng. 17,* 5 (May), 698–712.

Clementini, E. and Felice, P. D. 2000. Spatial operators. *SIGMOD Record 29,* 3 (Sept.), 31–38.

Cook, D. and Das, S. 2004. *Smart Environments: Technology, Protocols and Applications.* Wiley, New York, NY, USA.

Coombs, R. and Steele, R. 1999. Introducing microcells into macrocellular networks: A case study. *IEEE Trans. on Commun. 47,* 4 (Apr.), 568–576.

Cooperstock, J. R., Tanikoshi, K., Beirne, G., Narine, T., and Buxton, W. A. S. 1995. Evolution of a reactive environment. In *Conf. on Human factors in Computing Systems (CHI'95)*. ACM/Addison-Wesley, New York, NY, USA, 170–177.

Corral, A., Manolopoulos, Y., Theodoridis, Y., and Vassilakopoulos, M. 2000. Closest pair queries in spatial databases. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'00)*. ACM, New York, NY, USA, 189–200.

Dai, X., Yiu, M. L., Mamoulis, N., Tao, Y., and Vaitis, M. 2005. Probabilistic spatial queries on existentially uncertain data. In *9th International Symposium on Advances in Spatial and Temporal Databases (SSTD'05)*. Springer, Berlin/Heidelberg, 400–417.

Dalvi, N. and Suciu, D. 2004. Efficient query evaluation on probabilistic databases. In *30th International Conf. on Very Large Data Bases (VLDB'04)*. Morgan Kaufmann, San Francisco, CA, USA, 864–875.

Dar, S., Franklin, M. J., Jónsson, B. T., Srivastava, D., and Tan, M. 1996. Semantic data caching and replacement. In *22th International Conf. on Very Large Data Bases (VLDB'96)*. Morgan Kaufmann, San Francisco, CA, USA, 330–341.

Deitel, H. M., Deitel, P. J., Nieto, T. R., and Steinbuhler, K. 2001. *Wireless Internet and Mobile Business.* Prentice Hall, Upper Saddle River, NJ, USA.

Ding, H., Trajcevski, G., and Scheuermann, P. 2006. OMCAT: optimal maintenance of continuous queries' answers for trajectories. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'06)*. ACM, New York, NY, USA, 748–750.

Ding, Z. and Güting, R. H. 2004. Managing moving objects on dynamic transportation networks. In *16th International Conf. on Scientific and Statistical Database Management (SSDBM'04)*. IEEE Computer Society, Washington, DC, USA, 287–296.

Djuknic, G. M. and Richton, R. E. 2001. Geolocation and Assisted GPS. *IEEE Comput. 34,* 2 (Feb.), 123–125.

Dobra, A., Garofalakis, M., Gehrke, J., and Rastogi, R. 2002. Processing complex aggregate queries over data streams. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'02)*. ACM, New York, NY, USA, 61–72.

Drab, S. A. and Binder, G. 2005. Spacerace – a location based game for mobile phones using assisted GPS. In *2nd International PERVASIVE'05 Workshop on Pervasive Gaming Applications (PerGames'05)*.

Ducatel, K., Bogdanowicz, M., Scapolo, F., Leitjen, J., and Burgelman, J. C. 2001. Scenarios for ambient intelligence in 2010. Tech. rep., IST Advisory Group, European Commission.

Dunham, M. and Helal, A. 1995. Mobile computing and databases: Anything new? *SIGMOD Record 24,* 4 (Dec.), 5–9.

Dunham, M. H. and Kumar, V. 1998. Location dependent data and its management in mobile databases. In *1st International DEXA Workshop on Mobility in Databases and Distributed Systems (MDDS'98)*. IEEE Computer Society, Washington, DC, USA, 414–419.

Dye, S. and Buckingham, S. 1999. *Mobile Positioning.* Mobile Lifestreams.

Ehrlich, N. 1979. The Advanced Mobile Phone Service. *IEEE Communications Magazine 17,* 2 (Mar.), 9–16.

Elmasri, R. and Navathe, S. B. 2003. *Fundamental of Database Systems.* Addison Wesley.

Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M. 2003. The many faces of publish/subscribe. *ACM Comput. Surv. 35,* 2 (June), 114–131.

Evangelidis, G., Lomet, D. B., and Salzberg, B. 1995. The hB^{π}-tree: A modified hB-tree supporting concurrency, recovery and node consolidation. In *21st International Conf. on Very Large Data Bases (VLDB'95)*. Morgan Kaufmann, San Francisco, CA, USA, 551–561.

Feldman, A., Tapia, E. M., Sadi, S., Maes, P., and Schmandt, C. 2005. ReachMedia: On-the-move interaction with everyday objects. In *9th IEEE International Symposium on Wearable Computers (ISWC'05)*. IEEE Computer Society, Washington, DC, USA, 52–59.

Ferhatosmanoglu, H., Stanoi, I., Agrawal, D., and Abbadi, A. E. 2001. Constrained nearest neighbor queries. In *7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*. Springer, Berlin/Heidelberg, 257–278.

Forman, G. H. and Zahorjan, J. 1994. The challenges of mobile computing. *IEEE Comput. 27,* 4 (Apr.), 38–47.

Frentzos, E., Gratsias, K., Pelekis, N., and Theodoridis, Y. 2007. Algorithms for nearest neighbor search on moving object trajectories. *GeoInformatica 11,* 2 (June), 159–193.

Frentzos, R. 2003. Indexing moving objects on fixed networks. In *8th International Symposium on Advances in Spatial and Temporal Databases (SSTD'03)*. Springer, Berlin/Heidelberg, 289–305.

Gaede, V. and Günther, O. 1998. Multidimensional access methods. *ACM Comput. Surv. 30,* 2 (June), 170–231.

Gao, X. and Hurson, A. R. 2005. Location dependent query proxy. In *ACM Symposium on Applied Computing (SAC'05)*. ACM, New York, NY, USA, 1120–1124.

Gedik, B. and Liu, L. 2005. Location privacy in mobile systems: A personalized anonymization model. In *25th IEEE International Conf. on Distributed Computing Systems (ICDCS'05)*. IEEE Computer Society, Washington, DC, USA, 620–629.

Gedik, B. and Liu, L. 2006. MobiEyes: A distributed location monitoring service using moving location queries. *IEEE Trans. on Mobile Computing 5,* 10 (Oct.), 1384–1402.

Gedik, B., Singh, A., and Liu, L. 2004. Energy efficient exact kNN search in wireless broadcast environments. In *12th ACM International Symposium on Advances in Geographic Information Systems (GIS'04)*. ACM, New York, NY, USA, 137–146.

Gedik, B., Wu, K., Yu, P. S., and Liu, L. 2006. Processing moving queries over moving objects using motion adaptive indexes. *IEEE Trans. on Knowl. and Data Eng. 18,* 5 (May), 651–668.

Ghanem, T. M., Aref, W. G., and Elmagarmid, A. K. 2006. Exploiting predicate-window semantics over data streams. *SIGMOD Record 35,* 1 (Mar.), 3–8.

Giaglis, G. M., Kourouthanassis, P., and Doukidis, G. J. 2002. Intelligent tagging and automatic home replenishment schemes: The MyGrocer innovative business and technology framework. In *International Conf. on New Technologies and Strategies to Enhance Packaging Supply Chains*.

Giaglis, G. M., Kourouthanassis, P., and Tsamakos, A. 2003. Towards a classification framework for mobile location services. In *Mobile commerce: technology, theory, and applications*. IDEA Group Publishing, Hershey, PA, USA, 67–85.

Gietz, W. and Dupree, C. 2002. *Oracle 9i Data Cartridge Developer's Guide System Documentation Release 2*. Oracle Corp., Redwood Shores, CA, USA.

Gilbert, A. C., Kotidis, Y., Muthukrishnan, S., and Strauss, M. 2001. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *27th International Conf. on Very Large Data Bases (VLDB'01)*. Morgan Kaufmann, San Francisco, CA, USA, 79–88.

Gök, H. G. and Ulusoy, O. 2000. Transmission of continuous query results in mobile computing systems. *Information Sciences 125,* 1–4 (June), 37–63.

Göker, A., Watt, S., Myrhaug, H. I., Whitehead, N., Yakici, M., Bierig, R., Nuti, S. K., and Cumming, H. 2004. An ambient, personalised, and context-sensitive information system for mobile users. In *2nd European Union Symposium on Ambient intelligence (EUSAI'04)*. ACM, New York, NY, USA, 19–24.

Golab, L. and Özsu, M. T. 2003. Issues in data stream management. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'03)*. ACM, New York, NY, USA, 5–14.

Görlach, A., Heinemann, A., and Terpstra, W. W. 2005. Survey on location privacy in pervasive computing. In *Privacy, Security and Trust within the Context of Pervasive Computing*. Springer, New York, NY, USA, 23–34.

Gowrisankar, H. and Nittel, S. 2002. Reducing uncertainty in location prediction of moving objects in road networks. In *2nd International Conf. on Geographic Information Science (GIScience'02)*.

Graumann, D., Hightower, J., Lara, W., and Borriello, G. 2003. Real-world implementation of the Location Stack: The Universal Location Framework. In *5th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'03)*. IEEE Computer Society, Washington, DC, USA, 122–128.

Gruteser, M. and Grunwald, D. 2003. Anonymous usage of location-based services through spatial and temporal cloaking. In *1st International Conf. on Mobile Systems, Applications and Services (MobiSys'03)*. ACM, New York, NY, USA, 31–42.

Guha, S., Koudas, N., and Shim, K. 2001. Data-streams and histograms. In *33rd ACM Symposium on Theory of Computing (STOC'01)*. ACM, New York, NY, USA, 471–475.

Guha, S., Mishra, N., Motwani, R., and O'Callaghan, L. 2000. Clustering data streams. In *41st Symposium on Foundations of Computer Science (FOCS'00)*. IEEE Computer Society, Washington, DC, USA, 359.

Gupta, M., Tu, M., Khan, L., Bastani, F. B., and Yen, I.-L. 2005. A study of the model and algorithms for handling location-dependent continuous queries. *Knowl. Inform. Sys. 8,* 4 (Nov.), 414–437.

Güting, H., Almeida, V. T., and Ding, Z. 2006. Modeling and querying moving objects in networks. *The VLDB Journal 15,* 2, 165–190.

Güting, R. H., Böhlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., and Vazirgiannis, M. 2000. A foundation for representing and querying moving objects. *ACM Trans. Database Syst. 25,* 1 (Mar.), 1–42.

Güting, R. H. and Schneider, M. 2005. *Moving Objects Databases*. Morgan Kaufmann, San Francisco, CA, USA.

Guttman, A. 1984. R-trees: a dynamic index structure for spatial searching. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'84)*. ACM, New York, NY, USA, 47–57.

Hadjieleftheriou, M., Kollios, G., and Tsotras, V. J. 2003. Performance evaluation of spatio-temporal selectivity estimation techniques. In *15th International Conf. on Scientific*

and Statistical Database Management (SSDBM'03). IEEE Computer Society, Washington, DC, USA, 202–211.

Hadjieleftheriou, M., Kollios, G., Tsotras, V. J., and Gunopulos, D. 2002. Efficient indexing of spatiotemporal objects. In *8th International Conf. on Extending Database Technology (EDBT'02)*. Springer, Berlin/Heidelberg, 251–268.

Hadjieleftheriou, M., Kollios, G., Tsotras, V. J., and Gunopulos, D. 2006. Indexing spatiotemporal archives. *The VLDB Journal 15,* 2 (June), 143–164.

Hage, C., Jensen, C. S., Pedersen, T. B., Speicys, L., and Timko, I. 2003. Integrated data management for mobile services in the real world. In *29th International Conf. on Very Large Data Bases (VLDB'03)*. Morgan Kaufmann, San Francisco, CA, USA, 1019–1030.

Haghighat, A., Lopes, C., Givargis, T., and Mandal, A. 2004. Location-aware web system. In *OOPSLA'04 International Workshop on Building Software for Pervasive Computing*.

Hagras, H., Callaghan, V., Colley, M., Clarke, G., Pounds-Cornish, A., and Duman, H. 2004. Creating an ambient-intelligence environment using embedded agents. *IEEE Intelligent Syst. 19,* 6 (Nov.), 12–20.

Hambrusch, S. E., Liu, C., Aref, W. G., and Prabhakar, S. 2001. Query processing in broadcasted spatial index trees. In *7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*. Springer, Berlin/Heidelberg, 502–521.

Hartzman, C. S. and Watters, C. R. 1990. A relational approach to querying data streams. *IEEE Trans. on Knowl. and Data Eng. 2,* 4 (Dec.), 401–409.

Henrich, A. 1994. A distance scan algorithm for spatial access structures. In *2nd ACM Workshop on Advances in Geographic Information Systems (GIS'94)*. ACM, New York, NY, USA, 136–143.

Hinrichs, K. 1985. Implementation of the grid file: Design concepts and experience. *Bit 25,* 4, 569–592.

Hjaltason, G. R. and Samet, H. 1999. Distance browsing in spatial databases. *ACM Trans. Database Syst. 24,* 2 (June), 265–318.

Hjelm, J. 2002. *Creating Location Services for the Wireless Web*. Wiley, New York, NY, USA.

Hoareau, C. and Satoh, I. 2007. A model checking-based approach for location query processing in pervasive computing environments. In *2nd International OTM'07 Workshop on Pervasive Systems (PerSys'07)*. Springer, Berlin/Heidelberg, 866–875.

Housel, B. C. and Lindquist, D. B. 1996. WebExpress: a system for optimizing web browsing in a wireless environment. In *2nd Annual International Conf. on Mobile Computing and Networking (MobiCom'96)*. ACM, New York, NY, USA, 108–116.

Hu, H. and Lee, D. L. 2005. Energy-efficient monitoring of spatial predicates over moving objects. *IEEE Data Eng. Bull. 28,* 3 (Sept.), 19–26.

Hu, H., Xu, J., and Lee, D. L. 2005a. A generic framework for monitoring continuous spatial queries over moving objects. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'05)*. ACM, New York, NY, USA, 479–490.

Hu, H., Xu, J., Wong, W. S., Zheng, B., Lee, D. L., and Lee, W.-C. 2005b. Proactive caching for spatial queries in mobile environments. In *21st International Conf. on Data Engineering (ICDE'05)*. IEEE Computer Society, Washington, DC, USA, 403–414.

Huang, X. and Jensen, C. S. 2004. Towards a streams-based framework for defining location-based queries. In *2nd Workshop on Spatio-Temporal Database Management (STDBM'04)*. 73–80.

Huang, X., Jensen, C. S., and Saltenis, S. 2005. The Islands approach to nearest neighbor querying in spatial networks. In *9th International Symposium on Advances in Spatial and Temporal Databases (SSTD'05)*. Springer, Berlin/Heidelberg, 73–90.

Hung, D., Lam, K., Chan, E., and Ramamritham, K. 2003. Processing of location-dependent continuous queries on real-time spatial data: The view from RETINA. In *6th International DEXA Workshop on Mobility in Databases and Distributed Systems (MDDS'03)*. IEEE Computer Society, Washington, DC, USA, 961–965.

Ilarri, S., Mena, E., and Bobed, C. 2007. Processing location-dependent queries with location granules. In *2nd International OTM'07 Workshop on Pervasive Systems (PerSys'07)*. Springer, Berlin/Heidelberg, 856–866.

Ilarri, S., Mena, E., and Illarramendi, A. 2002. Monitoring continuous location queries using mobile agents. In *6th East-European Conf. on Advances in Databases and Information Systems (ADBIS'02)*. Springer, Berlin/Heidelberg, 92–105.

Ilarri, S., Mena, E., and Illarramendi, A. 2003. Dealing with continuous location-dependent queries: Just-in-time data refreshment. In *1st IEEE International Conf. on Pervasive Computing and Communications (PerCom'03)*. IEEE Computer Society, Washington, DC, USA, 279–286.

Ilarri, S., Mena, E., and Illarramendi, A. 2004. Testing agent-based mobile computing applications using distributed simulations. In *7th International DEXA Workshop on Mobility in Databases and Distributed Systems (MDDS'04)*. IEEE Computer Society, Washington, DC, USA, 652–656.

Ilarri, S., Mena, E., and Illarramendi, A. 2006a. Location-dependent queries in mobile contexts: Distributed processing using mobile agents. *IEEE Trans. on Mobile Computing 5,* 8 (Aug.), 1029–1043.

Ilarri, S., Trillo, R., and Mena, E. 2006b. SPRINGS: A scalable platform for highly mobile agents in distributed computing environments. In *4th International WoWMoM'06 Workshop on Mobile Distributed Computing (MDC'06)*. IEEE Computer Society, Washington, DC, USA, 633–637.

Imielinski, T. 1996a. *Mobile Computing*. Kluwer, Boston, MA, USA.

Imielinski, T. 1996b. Mobile computing: DataMan Project perspective. *Mobile Networks and Applications 1,* 4 (Dec.), 359–369.

Imielinski, T. and Nath, B. 1992. Querying in highly mobile distributed environments. In *18th International Conf. on Very Large Data Bases (VLDB'92)*. Morgan Kaufmann, San Francisco, CA, USA, 41–52.

Imielinski, T. and Nath, B. 1993. Data management for mobile computing. *SIGMOD Record 22,* 1 (Mar.), 34–39.

Imielinski, T. and Nath, B. 2002. Wireless graffiti: data, data everywhere. In *28th International Conf. on Very Large Data Bases (VLDB'02)*. Morgan Kaufmann, San Francisco, CA, USA, 9–19.

Imielinski, T. and Navas, J. C. 1999. GPS-based geographic addressing, routing, and resource discovery. *Commun. ACM 42,* 4 (Apr.), 86–92.

Ioannidis, J., Duchamp, D., and Maguire, G. Q. 1991. IP-based protocols for mobile internetworking. In *Conf. on Communications architecture and protocols (SIGCOMM'91)*. ACM, New York, NY, USA, 235–245.

Iwerks, G. S., Samet, H., and Smith, K. 2003. Continuous K-nearest neighbor queries for continuously moving points with updates. In *29th International Conf. on Very Large Data Bases (VLDB'03)*. Morgan Kaufmann, San Francisco, CA, USA, 512–523.

Iwerks, G. S., Samet, H., and Smith, K. P. 2006. Maintenance of K-nn and spatial join queries on continuously moving points. *ACM Trans. Database Syst. 31,* 2 (June), 485–536.

Jayaputera, J. and Taniar, D. 2004. Defining scope of query for location-dependent information services. In *International Conf. on Embedded and Ubiquitous Computing (EUC'04)*. Springer, Berlin/Heidelberg, 366–376.

Jayaputera, J. and Taniar, D. 2005a. Data retrieval for location-dependent query 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 services. *International Journal of Business Data Communications and Networking 1,* 2, 17–40.

Jensen, C. S. (Ed.) 2002. Special issue on indexing of moving objects. *IEEE Data Eng. Bull. 25,* 2 (June).

Jensen, C. S., Friis-Christensen, A., Pedersen, T. B., Pfoser, D., Saltenis, S., and Tryfona, N. 2001. Location-based services: A database perspective. In *8th Scandinavian Research Conf. on Geographical Information Science (ScanGIS'01)*. 59–68.

Jensen, C. S., Kligys, A., Pedersen, T. B., and Timko, I. 2002. Multidimensional data modeling for location-based services. In *10th ACM International Symposium on Advances in Geographic Information Systems (GIS'02)*. ACM, New York, NY, USA, 55–61.

Jensen, C. S., Lin, D., and Ooi, B. C. 2004. Query and update efficient B+-tree based indexing of moving objects. In *30th International Conf. on Very Large Data Bases (VLDB'04)*. Morgan Kaufmann, San Francisco, CA, USA, 768–779.

Jensen, C. S. and Pakalnis, S. 2007. TRAX: real-world tracking of moving objects. In *33rd International Conf. on Very Large Data Bases (VLDB'07)*. ACM, New York, NY, USA, 1362–1365.

Jiang, C. and Steenkiste, P. 2002. A hybrid location model with a computable location identifier for ubiquitous computing. In *4th International Conf. on Ubiquitous Computing (UbiComp'02)*. Springer, Berlin/Heidelberg, 246–263.

Jing, J., Helal, A. S., and Elmagarmid, A. 1999. Client-server computing in mobile environments. *ACM Comput. Surv. 31,* 2 (June), 117–157.

Jones, C. E., Sivalingam, K. M., Agrawal, P., and Chen, J. C. 2001. A survey of energy efficient network protocols for wireless networks. *Wireless Networks 7,* 4 (Aug.), 343–358.

Jung, I.-D., You, Y.-H., Lee, J.-H., and Kim, K. 2002. Broadcasting and caching policies for location-dependent queries in urban areas. In *2nd International Workshop on Mobile Commerce (WMC'02)*. ACM, New York, NY, USA, 54–60.

Junglas, I. A. 2005. An experimental investigation of location-based services. In *38th Hawaii International Conf. on System Sciences (HICSS'05)*. IEEE Computer Society, Washington, DC, USA, 85.

Kaaranen, H., Naghian, S., Laitinen, L., Ahtiainen, A., and Niemi, V. 2001. *UMTS Networks: Architecture, Mobility and Services*. Wiley, New York, NY, USA.

Kalashnikov, D. V., Prabhakar, S., and Hambrusch, S. E. 2004. Main memory evaluation of monitoring queries over moving objects. *Distrib. and Parall. Databases 15,* 2 (Mar.), 117–135.

Karimi, H. A. and Liu, X. 2003. A predictive location model for location-based services. In *11th ACM International Symposium on Advances in Geographic Information Systems (GIS'03)*. ACM, New York, NY, USA, 126–133.

Kayan, E. and Ulusoy, Ö. 1999. Real-time transaction management in mobile computing systems. In *6th International Conf. on Database Systems for Advanced Applications (DASFAA'99)*. IEEE Computer Society, Washington, DC, USA, 127–134.

Kirchner, H., Krummenacher, R., Risse, T., and Edwards-May, D. 2004. A location-aware prefetching mechanism. In *4th International Network Conf. (INC'04)*. 453–460.

Kolahdouzan, M. R. and Shahabi, C. 2004. Voronoi-based K nearest neighbor search for spatial network databases. In *30th International Conf. on Very Large Data Bases (VLDB'04)*. Morgan Kaufmann, San Francisco, CA, USA, 840–851.

Kolahdouzan, M. R. and Shahabi, C. 2005. Alternative solutions for continuous K nearest neighbor queries in spatial network databases. *GeoInformatica 9,* 4 (Dec.), 321–341.

Kollios, G., Gunopulos, D., and Tsotras, V. J. 1999a. Nearest neighbor queries in a mobile environment. In *1st International Workshop on Spatio-Temporal Database Management (STDBM'99)*. Springer, Berlin/Heidelberg, 119–134.

Kollios, G., Gunopulos, D., and Tsotras, V. J. 1999b. On indexing mobile objects. In *18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'99)*. ACM, New York, NY, USA, 261–272.

Kollios, G., Papadopoulos, D., Gunopulos, D., and Tsotras, J. 2005. Indexing mobile objects using dual transformations. *The VLDB Journal 14,* 2 (Apr.), 238–256.

Kollios, G., Tsotras, V. J., Gunopulos, D., Delis, A., and Hadjieleftheriou, M. 2001. Indexing animated objects using spatiotemporal access methods. *IEEE Trans. on Knowl. and Data Eng. 13,* 5 (Sept.), 758–777.

Koubarakis, M., Sellis, T. K., Frank, A. U., Grumbach, S., Güting, R. H., Jensen, C. S., Lorentzos, N. A., Manolopoulos, Y., Nardelli, E., Pernici, B., Schek, H.-J., Scholl, M., Theodoulidis, B., and Tryfona, N., Eds. 2003. *Spatio-Temporal Databases: The CHOROCHRONOS Approach*. Springer, Berlin/Heidelberg.

Kriegel, H.-P., Kunath, P., and Renz, M. 2007. Probabilistic nearest-neighbor query on uncertain objects. In *12th International Conf. on Database Systems for Advanced Applications (DASFAA'07)*. Springer, Berlin/Heidelberg, 337–348.

Ku, W., Zimmermann, R., and Wan, C. 2005. Location-based spatial queries with data sharing in mobile environments. Tech. Rep. USC-CS-TR05-843, University of Southern California.

Kubach, U. and Rothermel, K. 2001a. Exploiting location information for infostation-based hoarding. In *7th International Conf. on Mobile Computing and Networking (MobiCom'01)*. ACM, New York, NY, USA, 15–27.

Kubach, U. and Rothermel, K. 2001b. A map-based hoarding mechanism for location-dependent information. In *2nd International Conf. on Mobile Data Management (MDM'01)*. Springer, Berlin/Heidelberg, 145–157.

Kumar, S. and Stokkeland, J. 2003. Evolution of GPS technology and its subsequent use in commercial markets. *International Journal of Mobile Communications 1,* 1/2 (July), 180–193.

Kwon, D., Lee, S., and Lee, S. 2002. Indexing the current positions of moving objects using the lazy update R-tree. In *3rd International Conf. on Mobile Data Management (MDM'02)*. IEEE Computer Society, Washington, DC, USA, 113–120.

Lam, K. and Ulusoy, O. 2006. Adaptive schemes for location update generation in execution location-dependent continuous queries. *J. Syst. Softw. 79,* 4 (Apr.), 441–453.

Lam, K., Ulusoy, Ö., Lee, T. S. H., Chan, E., and Li, G. 2001. An efficient method for generating location updates for processing of location-dependent continuous queries. In *7th International Conf. on Database Systems for Advanced Applications (DASFAA'01)*. IEEE Computer Society, Washington, DC, USA, 218–225.

Lange, D. B. and Oshima, M. 1999. Seven good reasons for mobile agents. *Commun. ACM 42,* 3 (Mar.), 88–89.

Lazaridis, I., Porkaew, K., and Mehrotra, S. 2002. Dynamic queries over mobile objects. In *8th International Conf. on Extending Database Technology (EDBT'02)*. Springer, Berlin/Heidelberg, 269–286.

Lee, D. L. 2007. On searching continuous k nearest neighbors in wireless data broadcast systems. *IEEE Trans. on Mobile Computing 6,* 7 (July), 748–761.

Lee, K., Lee, W., Zheng, B., and Winter, J. 2006. Processing multiple aggregation queries in geo-sensor networks. In *11th International Conf. on Database Systems for Advanced Applications (DASFAA'06)*. Springer, Berlin/Heidelberg, 20–34.

Lee, K. C. K., Leong, H. V., and Si, A. 1999. Semantic query caching in a mobile environment. *SIGMOBILE Mobile Computing and Communications Review 3,* 2 (Apr.), 28–36.

Lee, M., Hsu, W., Jensen, C. S., Cui, B., and Teo, K. L. 2003. Supporting frequent updates in R-trees: A bottom-up approach. In *29th International Conf. on Very Large Data Bases (VLDB'03)*. Morgan Kaufmann, San Francisco, CA, USA, 608–619.

Lee, V. C. S., Lam, K. W., and Kuo, T. 2004. Efficient validation of mobile transactions in wireless environments. *J. Syst. Softw. 69,* 1–2 (Jan.), 183–193.

Lee, W. and Zheng, B. 2005. DSI: A fully distributed spatial index for location-based wireless broadcast services. In *25th International Conf. on Distributed Computing Systems (ICDCS'05)*. IEEE Computer Society, Washington, DC, USA, 349–358.

Lehmann, O., Bauer, M., Becker, C., and Nicklas, D. 2004. From home to world – supporting context-aware applications through world models. In *2nd IEEE International Conf. on Pervasive Computing and Communications (PerCom'04)*. IEEE Computer Society, Washington, DC, USA, 297–308.

Li, F., Cheng, D., Hadjieleftheriou, M., Kollios, G., and Teng, S.-H. 2005. On trip planning queries in spatial databases. In *9th International Symposium on Advances in Spatial and Temporal Databases (SSTD'05)*. Springer, Berlin/Heidelberg, 273–290.

Lin, D., Jensen, C. S., Ooi, B. C., and Saltenis, S. 2005. Efficient indexing of the historical, present, and future positions of moving objects. In *6th International Conf. on Mobile Data Management (MDM'05)*. ACM, New York, NY, USA, 59–66.

Lin, D., Zhang, R., and Zhou, A. 2006. Indexing fast moving objects for kNN queries based on nearest landmarks. *GeoInformatica 10,* 4 (Dec.), 423–445.

Lindemann, C. and Thümmler, A. 2003. Performance analysis of the general packet radio service. *Comput. Netw. 41,* 1 (Jan.), 1–17.

Liu, L. 2007. From data privacy to location privacy: models and algorithms. In *33rd International Conf. on Very Large Data Bases (VLDB'07)*. ACM, New York, NY, USA, 1429–1430.

Loney, K. and Koch, G. 2002. *Oracle9i: The Complete Reference*. McGraw-Hill/Osborne, Berkeley, CA, USA.

Lu, H., Ng, Y. Y., and Tian, Z. 2000. T-tree or B-tree: Main memory database index structure revisited. In *11th Australasian Database Conf. (ADC'00))*. IEEE Computer Society, Washington, DC, USA, 65–73.

Madria, S. K., Bhargava, B. K., Pitoura, E., and Kumar, V. 2000. Data organization issues for location-dependent queries in mobile computing. In *4th East-European Conf. on Advances in Databases and Information Systems (ADBIS'00)*. Springer, Berlin/Heidelberg, 142–156.

Markopoulos, A., Pissaris, P., Kyriazakos, S., and Sykas, E. 2004. Efficient location-based hard handoff algorithms for cellular systems. In *3rd International IFIP-TC6 Networking Conf.* Springer, Berlin/Heidelberg, 476–489.

Marsit, N., Hameurlain, A., Mammeri, Z., and Morvan, F. 2005. Query processing in mobile environments: A survey and open problems. In *1st International Conf. on Distributed Frameworks for Multimedia Applications (DFMA'05)*. IEEE Computer Society, Washington, DC, USA, 150–157.

Michael, K., McNamee, A., and Michael, M. G. 2006. The emerging ethics of humancentric GPS tracking and monitoring. In *International Conf. on Mobile Business (ICMB'06)*. IEEE Computer Society, Washington, DC, USA, 34.

Milner, R. 2004. Theories for the global ubiquitous computer. In *Foundations of Software Science and Computation Structure (FoSSaCS'04)*. Springer, Berlin/Heidelberg, 5–11.

Milojicic, D. S., Douglis, F., Paindaveine, Y., Wheeler, R., and Zhou, S. 2000. Process migration. *ACM Comput. Surv. 32,* 3 (Sept.), 241–299.

Mokbel, M. F. and Aref, W. G. 2005. GPAC: generic and progressive processing of mobile queries over mobile data. In *6th International Conf. on Mobile Data Management (MDM'05)*. ACM, New York, NY, USA, 155–163.

Mokbel, M. F. and Aref, W. G. 2008. SOLE: scalable on-line execution of continuous queries on spatio-temporal data streams. *The VLDB Journal 17,* 5 (Aug.), 971–995.

Mokbel, M. F., Chow, C.-Y., and Aref, W. G. 2006. The new Casper: query processing for location services without compromising privacy. In *32nd International Conf. on Very Large Data Bases (VLDB'06)*. ACM, New York, NY, USA, 763–774.

Mokbel, M. F., Ghanem, T. M., and Aref, W. G. 2003. Spatio-temporal access methods. *IEEE Data Eng. Bull. 26,* 2 (June), 40–49.

Mokbel, M. F., Xiong, X., and Aref, W. G. 2004. SINA: Scalable incremental processing of continuous queries in spatio-temporal databases. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'04))*. ACM, New York, NY, USA, 623–634.

Mokbel, M. F., Xiong, X., Hammad, M. A., and Aref, W. G. 2005. Continuous query processing of spatio-temporal data streams in PLACE. *GeoInformatica 9,* 4 (Dec.), 343–365.

Mokhtar, H. M. O. and Su, J. 2005. A query language for moving object trajectories. In *17th International Conf. on Scientific and Statistical Database Management (SSDBM'05)*. Lawrence Berkeley Laboratory, Berkeley, CA, US, 173–184.

Moreira, J., Ribeiro, C., and Abdessalem, T. 2000. Query operations for moving objects database systems. In *8th ACM International Symposium on Advances in Geographic Information Systems (GIS'00)*. ACM, New York, NY, USA, 108–114.

Morrow, R. 2002. *Bluetooth: Operation and Use*. McGraw-Hill Professional, New York, NY, USA.

Mouly, M. and Pautet, M. 1992. *The GSM System for Mobile Communications*. Telecom Publishing, Alexandria, VA, USA.

Mouratidis, K. and Papadias, D. 2007. Continuous nearest neighbor queries over sliding windows. *IEEE Trans. on Knowl. and Data Eng. 19,* 6 (June), 789–803.

Mouratidis, K., Papadias, D., Bakiras, S., and Tao, Y. 2005a. A threshold-based algorithm for continuous monitoring of k nearest neighbors. *IEEE Trans. on Knowl. and Data Eng. 17,* 11 (Nov.), 1451–1464.

Mouratidis, K., Papadias, D., and Hadjieleftheriou, M. 2005b. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'05)*. ACM, New York, NY, USA, 634–645.

Mukherjee, A., Saha, D., and Jha, S. 2003. Location management in mobile wireless networks. In *Wireless Internet Handbook: Technologies, Standards, and Application*. CRC Press, Boca Raton, FL, USA, 351–380.

Myllymaki, J. and Edlund, S. 2002. Location aggregation from multiple sources. In *3rd International Conf. on Mobile Data Management (MDM'02)*. IEEE Computer Society, Washington, DC, USA, 131–138.

Nascimento, M. A. and Silva, J. R. O. 1998. Towards historical R-trees. In *ACM Symposium on Applied computing (SAC'98)*. ACM, New York, NY, USA, 235–240.

Nascimento, M. A., Silva, J. R. O., and Theodoridis, Y. 1999. Evaluation of access structures for discretely moving points. In *1st International Workshop on Spatio-Temporal Database Management (STDBM'99)*. Springer, Berlin/Heidelberg, 171–188.

Nath, B., Acharya, A., and Imielinski, T. 1993a. Impact of mobility on distributed computations. *ACM Operating Systems Review 27,* 2 (Apr.), 15–20.

Nath, B., Bakre, A. V., Imielinski, T., and Maramtz, R. 1993b. Handling mobile clients: A case for indirect interaction. In *Workshop on Workstation Operating Systems (WWOS'93)*. IEEE Computer Society, Washington, DC, USA, 91–97.

Nehme, R. and Rundensteiner, E. A. 2006. SCUBA: Scalable cluster-based algorithm for evaluating continuous spatio-temporal queries on moving objects. In *10th International Conf. on Extending Database Technology (EDBT'06)*. Springer, Berlin/Heidelberg, 1001–1019.

Nehme, R. V. and Rundensteiner, E. A. 2007. *ClusterSheddy*: Load shedding using moving clusters over spatio-temporal data streams. In *12th International Conf. on Database Systems for Advanced Applications (DASFAA'07)*. Springer, Berlin/Heidelberg, 637–651.

Ng, C. K. and Chan, H. W. 2005. Enhanced distance-based location management of mobile communication systems using a cell coordinates approach. *IEEE Trans. on Mobile Computing 4,* 1 (Jan.), 41–55.

Ni, J., Ravishankar, C. V., and Bhanu, B. 2003. Probabilistic spatial database operations. In *8th International Symposium on Advances in Spatial and Temporal Databases (SSTD'03)*. Springer, Berlin/Heidelberg, 140–158.

Ni, L. M., Liu, Y., Lau, Y. C., and Patil, A. P. 2004. LANDMARC: Indoor location sensing using active RFID. *Wireless Networks 10,* 6 (Nov.), 701–710.

Ohrtman, F. 2005. *WiMAX Handbook*. McGraw-Hill Professional, New York, NY, USA.

Ohrtman, F. and Roeder, K. 2003. *Wi-Fi Handbook: Building 802.11b Wireless Networks*. McGraw-Hill Professional, New York, NY, USA.

Okabe, A., Boots, B., Sugihara, K., and Chiu, S. N. 2000. *Spatial tessellations: Concepts and applications of Voronoi diagrams*. Wiley, New York, NY, USA.

Özsu, M. T. and Valduriez, P. 1999. *Principles of Distributed Database Systems*. Prentice Hall, Upper Saddle River, NJ, USA.

Pagès-Zamora, A., Manzano, J. V., and Brooks, D. H. 2002. Closed-form solution for positioning based on angle of arrival measurements. In *13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'02)*. IEEE Computer Society, Washington, DC, USA, 1522–1526.

Papadias, D., Tao, Y., Kalnis, P., and Zhang, J. 2002. Indexing spatio-temporal data warehouses. In *18th International Conf. on Data Engineering (ICDE'02)*. IEEE Computer Society, Washington, DC, USA, 166–175.

Papadias, D., Zhang, J., Mamoulis, N., and Tao, Y. 2003. Query processing in spatial network databases. In *29th International Conf. on Very Large Data Bases (VLDB'03)*. Morgan Kaufmann, San Francisco, CA, USA, 802–813.

Pashtan, A., Blattler, R., Heusser, A., and Scheuermann, P. 2003. CATIS: A context-aware tourist information system. In *4th International Workshop of Mobile Computing (IMC'03)*.

Patel, J. M., Chen, Y., and Chakka, V. P. 2004. STRIPES: An efficient index for predicted trajectories. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'04)*. ACM, New York, NY, USA, 635–646.

Paton, N. W., Ed. 1999. *Active Rules in Database Systems*. Springer, New York, NY, USA.

Patroumpas, K. and Sellis, T. K. 2004. Managing trajectories of moving objects as data streams. In *2nd Workshop on Spatio-Temporal Database Management (STDBM'04)*. 41–48.

Pelanis, M., Saltenis, S., and Jensen, C. S. 2006. Indexing the past, present, and anticipated future positions of moving objects. *ACM Trans. Database Syst. 31,* 1 (Mar.), 255–298.

Peng, W. and Chen, M. 2005. Query processing in a mobile computing environment: Exploiting the features of asymmetry. *IEEE Trans. on Knowl. and Data Eng. 17,* 7 (July), 982–996.

Perkins, C. E. 1998. Mobile networking through mobile IP. *IEEE Internet Computing 2,* 1 (Jan.), 58–69.

Perkins, C. E. and Bhagwat, P. 1994. A mobile networking system based on Internet protocol. *IEEE Personal Comms. 1,* 1, 32–41.

Perusco, L. and Michael, K. 2007. Control, trust, privacy, and security: Evaluating location-based services. *IEEE Technology and Society Magazine 26,* 1 (Mar.), 4–16.

Pfoser, D. and Jensen, C. S. 1999. Capturing the uncertainty of moving-object representations. In *6th International Symposium on Advances in Spatial Databases (SSD'99)*. Springer, Berlin/Heidelberg, 111–132.

Pfoser, D. and Jensen, C. S. 2003. Indexing of network constrained moving objects. In *11th ACM International Symposium on Advances in Geographic Information Systems (GIS'03)*. ACM, New York, NY, USA, 25–32.

Pfoser, D., Jensen, C. S., and Theodoridis, Y. 2000. Novel approaches to the indexing of moving object trajectories. In *26th International Conf. on Very Large Data Bases (VLDB'00)*. Morgan Kaufmann, San Francisco, CA, USA, 395–406.

Pitoura, E. and Bhargava, B. 1999. Data consistency in intermittently connected distributed systems. *IEEE Trans. on Knowl. and Data Eng. 11,* 6 (Nov.), 896–915.

Pitoura, E. and Samaras, G. 1998. *Data Management for Mobile Computing*. Kluwer, Boston, MA, USA.

Pitoura, E. and Samaras, G. 2001. Locating objects in mobile computing. *IEEE Trans. on Knowl. and Data Eng. 13,* 4 (July), 571–592.

Porkaew, K. 2000. Database support for similarity retrieval and querying mobile objects. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Porta, T. F. L., Sabnani, K. K., and Gitlin, R. D. 1996. Challenges for nomadic computing: Mobility management and wireless communications. *Mobile Networks and Applications 1,* 1 (Aug.), 3–16.

Prabhakar, S., Xia, Y., Kalashnikov, D. V., Aref, W. G., and Hambrusch, S. E. 2002. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. *IEEE Trans. Comput. 51,* 10 (Oct.), 1124–1140.

Prager, S. D. 2007. Environmental contextualization of uncertainty for moving objects. *Computers, Environment and Urban Systems 31,* 3 (May), 303–316.

Predic, B. and Stojanovic, D. 2005. A framework for handling mobile objects in location based services. In *8th Conf. on Geographic Information Science (AGILE'05)*. 419–427.

Procopiuc, C. M., Agarwal, P. K., and Har-Peled, S. 2002. STAR-tree: An efficient self-adjusting index for moving objects. In *4th International Workshop on Algorithm Engineering and Experiments (ALENEX'02)*. Springer, Berlin/Heidelberg, 178–193.

Randell, C. and Muller, H. 2000. The Shopping Jacket: Wearable computing for the consumer. *Personal Ubiquitous Computing 4,* 4 (Aug.), 241–244.

Ranganathan, A., Al-Muhtadi, J., Chetan, S., Campbell, R. H., and Mickunas, M. D. 2004. MiddleWhere: A middleware for location awareness in ubiquitous computing applications. In *ACM/IFIP/USENIX International Middleware Conf.* Springer, Berlin/Heidelberg, 397–416.

Raptis, D., Tselios, N., and Avouris, N. 2005. Context-based design of mobile applications for museums: a survey of existing practices. In *7th International Conf. on Human Computer Interaction with Mobile Devices and Services (MobileHCI'05).* ACM, New York, NY, USA, 153–160.

Raptopoulou, K., Papadopoulos, A., and Manolopoulos, Y. 2003. Fast nearest-neighbor query processing in moving-object databases. *GeoInformatica 7,* 2 (June), 113–137.

Ray, A. and Kurkovsky, S. 2003. A survey of intelligent pervasive computing. In *International Conf. on Artificial Intelligence (IC-AI'03).* CSREA Press, 30–35.

Ray, S., Ungrangsi, R., Pellegrini, F. D., Trachtenberg, A., and Starobinski, D. 2003. Robust location detection in emergency sensor networks. In *22nd Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'03).* IEEE Computer Society, Washington, DC, USA.

Reiher, P., Popek, J., Gunter, M., Salomone, J., and Ratner, D. 1996. Peer-to-peer reconciliation based replication for mobile computers. In *2nd ECOOP'06 International Workshop on Mobility and Replication.*

Ren, Q. and Dunham, M. H. 2000. Using semantic caching to manage location dependent data in mobile computing. In *6th International Conf. on Mobile computing and networking (MobiCom'00).* ACM, New York, NY, USA, 210–221.

Rhodes, B. J., Minar, N., and Weaver, J. 1999. Wearable computing meets ubiquitous computing: Reaping the best of both worlds. In *3rd IEEE International Symposium on Wearable Computers (ISWC'99).* IEEE Computer Society, Washington, DC, USA, 141–150.

Rizos, C. 2005. Trends in geopositioning for LBS, navigation and mapping. In *4th International Symposium and Exhibition on Geoinformation.*

Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R. H., and Nahrstedt, K. 2002. A middleware infrastructure for active spaces. *IEEE Pervasive Computing 1,* 4 (Oct.), 74–83.

Roth, J. 2003a. Accessing location data in mobile environments – the Nimbus location model. In *5th International Workshop on Mobile and Ubiquitous Information Access (MobileHCI'03).* Springer, Berlin/Heidelberg, 256–270.

Roth, J. 2003b. Flexible positioning for location-based services. *IADIS International Journal on WWW/Internet 1,* 2 (Dec.), 18–32.

Roussopoulos, N., Kelley, S., and Vincent, F. 1995. Nearest neighbor queries. *SIGMOD Record 24,* 2 (May), 71–79.

Rundensteiner, E. A., Ding, L., Sutherland, T. M., Zhu, Y., Pielech, B., and Mehta, N. 2004. CAPE: Continuous query engine with heterogeneous-grained adaptivity. In *30th International Conf. on Very Large Data Bases (VLDB'04).* Morgan Kaufmann, San Francisco, CA, USA, 1353–1356.

Ryu, J., Song, M., and Hwang, C. 2003. Organizing the LDD in mobile environments. *IEICE Transactions on Information and Systems E86–D,* 9 (Sept.), 1504–1512.

Saltenis, S. and Jensen, C. S. 2002. Indexing of moving objects for location-based services. In *18th International Conf. on Data Engineering (ICDE'02).* IEEE Computer Society, Washington, DC, USA, 463–472.

Saltenis, S., Jensen, C. S., Leutenegger, S. T., and Lopez, M. A. 2000. Indexing the positions of continuously moving objects. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'00).* ACM, New York, NY, USA, 331–342.

Samaras, G. and Pitsillides, A. 1997. Client/intercept: a computational model for wireless environments. In *4th International Conf. on Telecommunications (ICT'97).*

Satyanarayanan, M. 2001. Pervasive computing: Vision and challenges. *IEEE Personal Comms. 8,* 4 (Aug.), 10–17.

Schilit, B., Adams, N., and Want, R. 1994. Context-aware computing applications. In *1st IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94)*. IEEE Computer Society, Washington, DC, USA, 85–90.

Schiller, J. and Voisard, A. 2004. *Location-Based Services*. Morgan Kaufmann, San Francisco, CA, USA.

Sesay, S., Yang, Z., and He, J. 2004. A survey on mobile ad hoc wireless network. *Information Technology Journal 3,* 2 (May), 168–175.

Seshan, S. 1995. Low-latency handoff for cellular data networks. Ph.D. thesis, University of California at Berkeley.

Seydim, A. Y. and Dunham, M. H. 2002. A location dependent benchmark with mobility behavior. In *International Symposium on Database Engineering and Applications (IDEAS'02)*. IEEE Computer Society, Washington, DC, USA, 74–85.

Seydim, A. Y., Dunham, M. H., and Kumar, V. 2001. Location dependent query processing. In *2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'01)*. ACM, New York, NY, USA, 47–53.

Shahabi, C., Kolahdouzan, M. R., and Sharifzadeh, M. 2002. A road network embedding technique for k-nearest neighbor search in moving object databases. In *10th ACM International Symposium on Advances in Geographic Information Systems (GIS'02)*. ACM, New York, NY, USA, 94–100.

Sistla, A. P., Wolfson, O., Chamberlain, S., and Dao, S. 1997a. Modeling and querying moving objects. In *13th International Conf. on Data Engineering (ICDE'97)*. IEEE Computer Society, Washington, DC, USA, 422–432.

Sistla, A. P., Wolfson, O., Chamberlain, S., and Dao, S. 1997b. Querying the uncertain position of moving objects. In *Temporal Databases*. Springer, Berlin/Heidelberg, 310–337.

Smailagic, A., Siewiorek, D. P., and Anhalt, J. 2001. Towards context aware computing: Experiences and lessons learned. *IEEE Intelligent Syst. 16,* 3 (June), 38–46.

Song, Z. and Roussopoulos, N. 2001a. Hashing moving objects. In *2nd International Conf. on Mobile Data Management (MDM'01)*. Springer, Berlin/Heidelberg, 161–172.

Song, Z. and Roussopoulos, N. 2001b. K-nearest neighbor search for moving query point. In *7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*. Springer, Berlin/Heidelberg, 79–96.

Spyrou, C., Samaras, G., Evripidou, P., and Pitoura, E. 1999. Wireless computational models: Mobile agents to the rescue. In *2nd International DEXA Workshop on Mobility in Databases and Distributed Systems (MDDS'99)*. IEEE Computer Society, Washington, DC, USA, 127–133.

Spyrou, C., Samaras, G., Pitoura, E., and Evripidou, P. 2004. Mobile agents for wireless computing: The convergence of wireless computational models with mobile-agent technologies. *Mobile Networks and Applications 9,* 5 (Oct.), 517–528.

Stathatos, K., Roussopoulos, N., and Baras, J. S. 1997. Adaptive data broadcast in hybrid networks. In *23rd International Conf. on Very Large Data Bases (VLDB'97)*. Morgan Kaufmann, San Francisco, CA, USA, 326–335.

Stojanovic, D. and Djordjevic-Kajan, S. 2003. Processing continuous range queries on mobile objects in location-based services. In *1st Balkan Conf. in Informatics (BCI'03)*. 280–292.

Stojanovic, D., Djordjevic-Kajan, S., Papadopoulos, A. N., and Nanopoulos, A. 2006. Continuous range query processing for network constrained mobile objects. In *8th International Conf. on Enterprise Information Systems (ICEIS'06)*. 63–70.

Stojanovic, D., Papadopoulos, A. N., Predic, B., Djordjevic-Kajan, S., and Nanopoulos, A. 2008. Continuous range monitoring of mobile objects in road networks. *Data and Knowl. Eng. 64,* 1 (Jan.), 77–100.

Su, J., Xu, H., and Ibarra, O. H. 2001. Moving objects: Logical relationships and queries. In *7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*. Springer, Berlin/Heidelberg, 3–19.

Sun, G., Chen, J., Guo, W., and Liu, K. J. R. 2005. Signal processing techniques in network-aided positioning: A survey. *IEEE Signal Processing Magazine 22,* 4 (July), 12–23.

Sun, J., Papadias, D., Tao, Y., and Liu, B. 2004. Querying about the past, the present, and the future in spatio-temporal databases. In *20th International Conf. on Data Engineering (ICDE'04)*. IEEE Computer Society, Washington, DC, USA, 202–213.

Sun, J., Tao, Y., Papadias, D., and Kollios, G. 2006. Spatio-temporal join selectivity. *Inf. Syst. 31,* 8 (Dec.), 793–813.

Swann, J., Chatre, E., and Ludwig, D. 2003. Galileo: Benefits for location based services. *Journal of Global Positioning Systems 2,* 1 (June), 57–66.

Tao, Y., Cheng, R., Xiao, X., Ngai, W. K., Kao, B., and Prabhakar, S. 2005a. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *31st International Conf. on Very Large Data Bases (VLDB'05)*. ACM, New York, NY, USA, 922–933.

Tao, Y., Faloutsos, C., Papadias, D., and Liu, B. 2004a. Prediction and indexing of moving objects with unknown motion patterns. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'04)*. ACM, New York, NY, USA, 611–622.

Tao, Y., Kollios, G., Considine, J., Li, F., and Papadias, D. 2004b. Spatio-temporal aggregation using sketches. In *20th International Conf. on Data Engineering (ICDE'04)*. IEEE Computer Society, Washington, DC, USA, 214–226.

Tao, Y., Mamoulis, N., and Papadias, D. 2003a. Validity information retrieval for spatio-temporal queries: Theoretical performance bounds. In *8th International Symposium on Advances in Spatial and Temporal Databases (SSTD'03)*. Springer, Berlin/Heidelberg, 159–178.

Tao, Y. and Papadias, D. 2001a. Efficient historical R-trees. In *13th International Conf. on Scientific and Statistical Database Management (SSDBM'01)*. IEEE Computer Society, Washington, DC, USA, 223–232.

Tao, Y. and Papadias, D. 2001b. MV3R-tree: A spatio-temporal access method for timestamp and interval queries. In *27th International Conf. on Very Large Data Bases (VLDB'01)*. Morgan Kaufmann, San Francisco, CA, USA, 431–440.

Tao, Y. and Papadias, D. 2003. Spatial queries in dynamic environments. *ACM Trans. Database Syst. 28,* 2 (June), 101–139.

Tao, Y. and Papadias, D. 2005. Historical spatio-temporal aggregation. *ACM Trans. Inf. Syst. 23,* 1 (Jan.), 61–102.

Tao, Y., Papadias, D., and Shen, Q. 2002. Continuous nearest neighbor search. In *28th International Conf. on Very Large Data Bases (VLDB'02)*. Morgan Kauffman, San Francisco, CA, USA, 287–298.

Tao, Y., Papadias, D., and Sun, J. 2003b. The TPR*-tree: An optimized spatio-temporal access method for predictive queries. In *29th International Conf. on Very Large Data Bases (VLDB'03)*. Morgan Kaufmann, San Francisco, CA, USA, 790–801.

Tao, Y., Papadias, D., Zhai, J., and Li, Q. 2005b. Venn sampling: A novel prediction technique for moving objects. In *21st International Conf. on Data Engineering (ICDE'05)*. IEEE Computer Society, Washington, DC, USA, 680–691.

Tao, Y., Sun, J., and Papadias, D. 2003c. Analysis of predictive spatio-temporal queries. *ACM Trans. Database Syst. 28,* 4 (Dec.), 295–336.

Tao, Y., Sun, J., and Papadias, D. 2003d. Selectivity estimation for predictive spatio-temporal queries. In *19th International Conf. on Data Engineering (ICDE'03)*. IEEE Computer Society, Washington, DC, USA, 417–428.

Tao, Y. and Xiao, X. 2008. Primal or dual: Which promises faster spatiotemporal search? *The VLDB Journal 17,* 5 (Aug.), 1253–1270.

Tao, Y., Xiao, X., and Cheng, R. 2007. Range search on multidimensional uncertain data. *ACM Trans. Database Syst. 32,* 3 (Aug.), 15.

Tayeb, J., Ulusoy, O., and Wolfson, O. 1998. A quadtree-based dynamic attribute indexing method. *Comput. J. 41,* 3, 185–200.

Teraoka, F., Yokore, Y., and Tokoro, M. 1991. A network architecture providing host migration transparency. In *Conf. on Communications Architecture and Protocols (SIGCOMM'91)*. ACM, New York, NY, USA, 209–220.

Terry, D., Goldberg, D., Nichols, D., and Oki, B. 1992. Continuous queries over append-only databases. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'92)*. ACM, New York, NY, USA, 321–330.

Theodoridis, Y. 2003. Ten benchmark database queries for location-based services. *Comput. J. 46,* 6 (Nov.), 713–725.

Theodoridis, Y., Sellis, T. K., Papadopoulos, A., and Manolopoulos, Y. 1998. Specifications for efficient indexing in spatiotemporal databases. In *10th International Conf. on Scientific and Statistical Database Management (SSDBM'98)*. IEEE Computer Society, Washington, DC, USA, 123–132.

Trajcevski, G., Ding, H., and Scheuermann, P. 2005a. Context-aware optimization of continuous range queries maintenance for trajectories. In *4th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'05)*. ACM, New York, NY, USA, 1–8.

Trajcevski, G. and Scheuermann, P. 2003. Triggers and continuous queries in moving objects database. In *6th International DEXA Workshop on Mobility in Databases and Distributed Systems (MDDS'03)*. IEEE Computer Society, Washington, DC, USA, 905–910.

Trajcevski, G. and Scheuermann, P. 2004. Reactive maintenance of continuous queries. *SIGMOBILE Mobile Computing and Communications Review 8,* 3 (July), 20–31.

Trajcevski, G., Scheuermann, P., Brönnimann, H., and Voisard, A. 2005b. Dynamic topological predicates and notifications in moving objects databases. In *6th International Conf. on Mobile Data Management (MDM'05)*. ACM, New York, NY, USA, 77–85.

Trajcevski, G., Scheuermann, P., Ghica, O., Hinze, A., and Voisard, A. 2006. Evolving triggers for dynamic environments. In *10th International Conf. on Extending Database Technology (EDBT'06)*. Springer, Berlin/Heidelberg, 1039–1048.

Trajcevski, G., Scheuermann, P., Wolfson, O., and Nedungadi, N. 2004a. CAT: Correct answers of continuous queries using triggers. In *9th International Conf. on Extending Database Technology (EDBT'04)*. Springer, Berlin/Heidelberg, 837–840.

Trajcevski, G., Wolfson, O., Hinrichs, K., and Chamberlain, S. 2004b. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst. 29,* 3 (Sept.), 463–507.

Trajcevski, G., Wolfson, O., Xu, B., and Nelson, P. 2002a. Real-time traffic updates in moving objects databases. In *5th International DEXA Workshop on Mobility in Databases and Distributed Systems (MDDS'02)*. IEEE Computer Society, Washington, DC, USA, 698–704.

Trajcevski, G., Wolfson, O., Zhang, F., and Chamberlain, S. 2002b. The geometry of uncertainty in moving objects databases. In *8th International Conf. on Extending Database Technology (EDBT'02)*. Springer, Berlin/Heidelberg, 233–250.

Trevisani, E. and Vitaletti, A. 2004. Cell-ID location technique, limits and benefits: An experimental study. In *6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*. IEEE Computer Society, Washington, DC, USA, 51–60.

Tsugawa, S. 2005. Issues and recent trends in vehicle safety communication systems. *IATTS Research 29,* 1, 7–15.

Tucker, P. A., Maier, D., Sheard, T., and Fegaras, L. 2003. Exploiting punctuation semantics in continuous data streams. *IEEE Trans. on Knowl. and Data Eng. 15,* 3 (May), 555–568.

Tucker, P. A., Maier, D., Sheard, T., and Stephens, P. 2007. Using punctuation schemes to characterize strategies for querying over data streams. *IEEE Trans. on Knowl. and Data Eng. 19,* 9 (Sept.), 1227–1240.

Tull, C. 2002. *WAP 2.0 Development.* Que, Indianapolis, IN, USA.

Ulusoy, Ö., Yoncaci, I., and Lam, K. Y. 2003. Evaluation of a criticality-based method for generating location updates. In *Workshop on Database Mechanisms for Mobile Applications.* GI, 94–105.

Vallino, J. 1998. Interactive augmented reality. Ph.D. thesis, Department of Computer Science, University of Rochester.

Vazirgiannis, M. and Wolfson, O. 2001. A spatiotemporal model and language for moving objects on road networks. In *7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*. Springer, Berlin/Heidelberg, 20–35.

Veijalainen, J. and Weske, M. 2003. Modeling static aspects of mobile electronic commerce environments. In *Advances in Mobile Commerce Technologies*. IDEA Group Publishing, Hershey, PA, USA, 137–170.

Viredaz, M. A., Brakmo, L. S., and Hamburgen, W. R. 2003. Energy management on handheld devices. *ACM Queue 1,* 7 (Oct.), 44–52.

Waluyo, A. B., Srinivasan, B., and Taniar, D. 2005. Global indexing scheme for location-dependent queries in multi channels mobile broadcast environment. In *19th International Conf. on Advanced Information Networking and Applications (AINA'05)*. IEEE Computer Society, Washington, DC, USA, 1011–1016.

Want, R., Hopper, A., Falcão, V., and Gibbons, J. 1992. The Active Badge location system. *ACM Trans. Inf. Syst. 10,* 1 (Jan.), 91–102.

Weiser, M. 1999a. The computer for the 21st century. *SIGMOBILE Mobile Computing and Communications Review 3,* 3 (July), 3–11.

Weiser, M. 1999b. Some computer science issues in ubiquitous computing. *SIGMOBILE Mobile Computing and Communications Review 3,* 3 (July), 12–20.

Wolfson, O., Chamberlain, S., Kapalkis, K., and Yesha, Y. 2001. Modeling moving objects for location based services. In *NSF Workshop Infrastructure for Mobile and Wireless Systems (IMWS'01)*. Springer, Berlin/Heidelberg, 46–58.

Wolfson, O., Jiang, L., Sistla, A. P., Chamberlain, S., Rishe, N., and Deng, M. 1999a. Databases for tracking mobile units in real time. In *7th International Conf. on Database Theory (ICDT'99)*. Springer, Berlin/Heidelberg, 169–186.

Wolfson, O. and Mena, E. 2004. Applications of moving objects databases. In *Spatial Databases: Technologies, Techniques and Trends*. IDEA Group Publishing, Hershey, PA, USA, 186–203.

Wolfson, O., Sistla, A. P., Chamberlain, S., and Yesha, Y. 1999b. Updating and querying databases that track mobile units. *Distrib. and Parall. Databases 7,* 3 (July), 257–287.

Wolfson, O., Sistla, P., Dao, S., Narayanan, K., and Raj, R. 1995. View maintenance in mobile computing. *SIGMOD Record 24,* 4 (Dec.), 22–27.

Wu, S.-Y. and Wu, K.-T. 2006. Effective location based services with dynamic data management in mobile environments. *Wireless Networks 12,* 3 (May), 369–381.

Wu, W., Yang, F., Chan, C. Y., and Tan, K.-L. 2008. Continuous reverse k-nearest-neighbor monitoring. In *9th International Conf. on Mobile Data Management (MDM'08)*. IEEE Computer Society, Washington, DC, USA, 132–139.

Xiao, Z., Meng, X., and Xu, J. 2007. Quality aware privacy protection for location-based services. In *12th International Conf. on Database Systems for Advanced Applications (DASFAA'07)*. Springer, Berlin/Heidelberg, 434–446.

Xiong, X. and Aref, W. G. 2006. R-trees with update memos. In *22nd IEEE International Conf. on Data Engineering (ICDE'06)*. IEEE Computer Society, Washington, DC, USA, 22.

Xiong, X., Mokbel, M. F., and Aref, W. G. 2005. SEA-CNN: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *21st International Conf. on Data Engineering (ICDE'05)*. IEEE Computer Society, Washington, DC, USA, 643–654.

Xiong, X., Mokbel, M. F., and Aref, W. G. 2006. LUGrid: Update-tolerant grid-based indexing for moving objects. In *7th International Conf. on Mobile Data Management (MDM'06)*. IEEE Computer Society, Washington, DC, USA, 13.

Xiong, X., Mokbel, M. F., Aref, W. G., Hambrusch, S., and Prabhakar, S. 2004. Scalable spatio-temporal continuous query processing for location-aware services. In *16th International Conf. on Scientific and Statistical Database Management (SSDBM'04)*. IEEE Computer Society, Washington, DC, USA, 317–328.

Xu, B., Ouksel, A. M., and Wolfson, O. 2004. Opportunistic resource exchange in inter-vehicle ad-hoc networks. In *5th IEEE International Conf. on Mobile Data Management (MDM'04)*. IEEE Computer Society, Washington, DC, USA, 4–12.

Xu, B. and Wolfson, O. 2003. Time-series prediction with applications to traffic and moving objects databases. In *3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'03)*. ACM, New York, NY, USA, 56–60.

Xu, J., Tang, X., and Lee, D. L. 2003a. Performance analysis of location-dependent cache invalidation schemes for mobile environments. *IEEE Trans. on Knowl. and Data Eng. 15,* 2 (Feb.), 474–488.

Xu, J., Tang, X., Lee, D. L., and Hu, Q. 1999. Cache coherency in location-dependent information services for mobile environment. In *1st International Conf. on Mobile Data Access (MDA'99)*. Springer, Berlin/Heidelberg, 182–193.

Xu, J., Tang, X., and Lee, W.-C. 2008. A new storage scheme for approximate location queries in object-tracking sensor networks. *IEEE Trans. Parall. Distrib. Syst. 19,* 2, 262–275.

Xu, J., Zheng, B., Lee, W., and Lee, D. L. 2003b. Energy efficient index for querying location-dependent data in mobile broadcast environments. In *19th International Conf. on Data Engineering (ICDE'03)*. IEEE Computer Society, Washington, DC, USA, 239–250.

Xu, X., Han, J., and Lu, W. 1990. RT-tree: An improved R-tree indexing structure for temporal spatial databases. In *International Symposium on Spatial Data Handling (SDH'90)*. 1040–1049.

Xu, Y., Lee, W., Xu, J., and Mitchell, G. 2003c. Window query processing in highly dynamic geo-sensor networks: Issues and solutions. In *NSF Workshop on Geo Sensor Network (GSN'03)*. CRC Press, Boca Raton, FL, USA, 31–52.

Xu, Y., Lee, W., Xu, J., and Mitchell, G. 2006. Processing window queries in wireless sensor networks. In *22nd IEEE International Conf. on Data Engineering (ICDE'06)*. IEEE Computer Society, Washington, DC, USA, 70.

Xu, Z. and Jacobsen, A. 2007. Adaptive location constraint processing. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'07)*. ACM, New York, NY, USA, 581–592.

Yavas, G., Katsaros, D., Ulusoy, O., and Manolopoulos, Y. 2005. A data mining approach for location prediction in mobile environments. *Data and Knowl. Eng. 54,* 2 (Aug.), 121–146.

Yin, H. and Wolfson, O. 2004. A weight-based map matching method in moving objects databases. In *16th International Conf. on Scientific and Statistical Database Management (SSDBM'04)*. IEEE Computer Society, Washington, DC, USA, 437–438.

Yiu, M. L., Tao, Y., and Mamoulis, N. 2008. The B^{dual}-tree: Indexing moving objects by space-filling curves in the dual space. *The VLDB Journal 17,* 3 (May), 379–400.

Yu, B. and Kim, S. H. 2006. Interpolating and using most likely trajectories in moving-objects databases. In *17th International Conf. on Database and Expert Systems Applications (DEXA'06)*. Springer, Berlin/Heidelberg, 718–727.

Yu, P. S., Chen, S.-K., and Wu, K.-L. 2006. Incremental processing of continual range queries over moving objects. *IEEE Trans. on Knowl. and Data Eng. 18,* 11 (Nov.), 1560–1575.

Yu, X., Chen, Y., Rao, F., and Liu, D. 2004. Filtering location stream in moving object database. In *7th International DEXA Workshop on Mobility in Databases and Distributed Systems (MDDS'04)*. IEEE Computer Society, Washington, DC, USA, 645–649.

Yu, X., Pu, K. Q., and Koudas, N. 2005. Monitoring k-nearest neighbor queries over moving objects. In *21st International Conf. on Data Engineering (ICDE'05)*. IEEE Computer Society, Washington, DC, USA, 631–642.

Zeidler, A. 2004. A distributed publish/subscribe notification service for pervasive environments. Ph.D. thesis, Technischen Universität Darmstadt.

Zhang, J., Zhu, M., Papadias, D., Tao, Y., and Lee, D. L. 2003. Location-based spatial queries. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'03)*. ACM, New York, NY, USA, 443–454.

Zheng, B. and Lee, D. L. 2001. Semantic caching in location-dependent query processing. In *7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*. Springer, Berlin/Heidelberg, 97–116.

Zheng, B., Lee, W., and Lee, D. L. 2003a. Selecting the best valid scopes for wireless dissemination of location-dependent data. In *ACM Symposium on Applied computing (SAC'03)*. ACM, New York, NY, USA, 860–865.

Zheng, B., Lee, W., and Lee, D. L. 2003b. Spatial index on air. In *1st IEEE International Conf. on Pervasive Computing and Communications (PerCom'03)*. IEEE Computer Society, Washington, DC, USA, 297–304.

Zheng, B., Lee, W., and Lee, D. L. 2004. On semantic caching and query scheduling for mobile nearest-neighbor search. *Wireless Networks 10,* 6 (Nov.), 653–664.

Zheng, B., Xu, J., and Lee, D. L. 2002. Cache invalidation and replacement strategies for location-dependent data in mobile environments. *IEEE Trans. Comput. 51,* 10 (Oct.), 1141–1153.

Zheng, B., Xu, J., Lee, W.-C., and Lee, L. 2006. Grid-partition index: A hybrid method for nearest-neighbor queries in wireless location-based services. *The VLDB Journal 15,* 1 (Jan.), 21–39.

Received February 2008; Revised September 2008; Accepted November 2008