When Location-Based Services Meet Databases

Dik Lun Lee Manli Zhu Haibo Hu

Department of Computer Science, Hong Kong University of Science and Technology

Clear Water Bay, Hong Kong

{dlee, cszhuml, haibo}@cs.ust.hk

Abstract:

As location-based services (LBSs) grow to support a larger and larger user community and to provide more and more intelligent services, they must face a few fundamental challenges, including the ability to not only accept coordinates as location data but also manipulate high-level semantics of the physical environment. They must also handle a large amount of location updates and client requests and be able to scale up as their coverage increases. This paper describes some of our research in location modeling and updates and techniques for enhancing system performance by caching and batch processing. It can be observed that the challenges facing LBSs share a lot of similarity with traditional database research (i.e., data modeling, indexing, caching, and query optimization) but the fact that LBSs are built into the physical space and the opportunity to exploit spatial locality in system design shed new light on LBS research.

Keyword:

Location-based service, Databases, Mobile Devices

1. Introduction

Technological advances, especially those in wireless communication and mobile devices, have fueled the proliferation of location-based services (LBS). In return, the demands from sophisticated applications with large user populations pose new challenges in LBS research systems:

- As the name implies, LBSs are built on the notion of “location.” In the past, definitions of “location” were very simple, ranging from coordinates such as longitude and latitude to statements like “I am in New York City.” These simple descriptions, especially those that model the physical space as a Euclidean space, are not enough for sophisticated LBSs. For example, in an indoor environment, doors, corridors, lifts, escalators, and staircases not only restrict physical reachability but impose different costs and constraints on movements. Road networks are another example where Euclidean distance does not apply.
- Mobile devices can only provide a computing environment with limited CPU, memory, network bandwidth, and battery resources. As such, mobile clients must be designed to balance the utilization between these resources and the loads between the client and the server. For example, pushing more computation on the client can reduce bandwidth consumption but increase CPU load and memory consumption. Given the rapidly increasing capability of mobile devices, mobile applications must make reasonable assumptions about the client’s computational capability and be able to adapt to it.
- LBSs must be able to handle a large user population and scale up in the face of

increasing demands. An LBS server must be able to process a large number of requests simultaneously. Fortunately, requests to the servers are likely to exhibit spatial locality and perhaps to a lesser degree temporal locality as well. For example, there will be a large number of requests about local traffic conditions in the morning and for shopping guides in downtown areas. The effective scheduling and distribution of requests to LBS servers based on spatial locality will directly improve the performance and user experience of an LBS application.

In this article, we address each issue by first reviewing existing work, then describing some of our solutions to them, and finally pointing to some future research directions. In the last part of this article, we describe some ongoing LBS projects involving the research issues described in this paper.

2. Applications

Logistics management such as mobile work force and fleet management is a natural – and arguably the first – application for LBSs. However, the most significant impact of LBSs lies on personal applications. In the following, we summarize some promising applications that may change our daily lives.

- **Location-based instant messaging**

Internet-based instant messaging (IM) has already proven to be one of the most popular internet applications. Location-based instant messaging is a natural extension of internet-based instant messaging that benefits from the huge existing market for IM. In location-based messaging, each message is associated with a location, which can be either the sender’s current location or a manually input location specification. The message will be sent to users who meet the location specification. For example, a user who is sitting in a coffee shop may want to send a message to his friends nearby to invite them to join him for a cup of coffee.

Many applications can be derived from location-based messaging. An example which has quickly become popular is location-based dating. In addition to all the traditional dating services, location-based dating provides instant and convenient dating service to people based on their locations in real time. According to a marketing study carried out by ComScore Networks in the United States, the number of visitors to online personal sites grew from 20.3 million in 2002 to 26.6 million in 2003 (a 31% increase) and the number of registered users on major dating sites including Match.com, Yahoo's Personals, One2onematch.com, and TerraLycos' Matchmaker exceeded 16 million in 2003.

- **Location-based alerts**

Most alerts that we use today are time-based; for example, reminding the user of the time of a meeting. In contrast, location-based alerts remind users when specified spatial conditions are met. For example, reminding a person to buy milk when he walks by a supermarket, or alerting him when he walks by a jewelry store one month before his wife's birthday. A more serious application is family safety alerts and emergency safety information, which is expected to account for 60 percent of location-based revenue by 2006 according to I-Pacific Partners [Web].

- **Location-based advertisement**s

On the Internet, non-intrusive, well-targeted advertisements, as exemplified by Google

Adword and Adsense, have been so successful that they have practically revived the entire online advertisement industry. A drive on the Internet is to make the advertisements relevant to the user's location (e.g., Google Local). Location-based advertisements make advertising even more targeted and relevant. For example, coffee coupons are served to users when they walk by a coffee shop close to 9am and store discount coupons and shopping information when they walk into a shopping mall. This apparently simple application requires a huge network to drive the collection and distribution of advertisements and to build in a mechanism to respect privacy and measure the success (and billing) of ads.

3. Location Model

Location is an important element in LBSs. A location model allows us to describe the physical space and properties of the objects contained in it. For example, it defines the coordinate system and therefore the locations of the objects and their spatial relationships. In addition, operations such as location sensing, object counting, browsing, navigation, and search are affected by the location model chosen.

Existing location models can be classified into geometric or symbolic models. A geometric model represents the physical space as a Euclidean space and the objects therein as points, lines, shapes, and volumes in the Euclidean space. With the coordinates of each object defined in the Euclidean space, operations such as distance and shortest path computations can be supported. A symbolic model's main objective is to capture the semantics of the entities in a physical space by expressing not only the capturing of the objects but also the relationship among them in some form of graph. For example, buildings, roads, offices, and rooms can be represented as nodes in a graph and their relationships such as containment and accessibility as edges. Most of the existing models are designed for outdoor applications but they are not appropriate for indoor environments for the following reasons:

- Indoor environments are intrinsically diverse. Each environment has its own way to describe locations. For example, a room may be described as "next to the elevator on the 3rd floor of the academic building," which embeds a lot of semantics about the space. In comparison, the outdoor environment typically adopts the universal longitude-latitude-altitude coordinate system.
- Many more entities of diverse types need to be modeled in an indoor environment because they may be needed to describe the context of the environment. As a system scales to millions of entities, a conventional graph is no longer efficient in terms of storage, processing, and maintenance costs. A more concise structure is needed to capture the entities and their relationships.
- The next generation of LBSs focuses on context awareness. The Euclidean and graph models are not capable of capturing the rich semantics of the user's context.

To address these problems, we developed a semantic location model SLM for indoor navigation [HL04]. The model is symbolic in nature, but it is unique in that it can be derived from the geometric representation of the indoor space. The model is composed of two types of entities: locations and exits. A location is defined as a bounded geographic area with one or more exits. An exit is a connector between two locations.

Two exits are “direct reachable” if the user can move between them without passing through a third entity. To compute this relation from a digitalized map (e.g., a floor plan), we developed an algorithm which uses the visibility graph in computational geometry to perform a lossless reduction from the resulting reachability graph to a compact exit hierarchy (CEH). The reduction is based on the observation that nearby exits tend to be directly reachable from each other and thus they should be clustered as a “super” exit. We proved that the topology on these super exits forms a forest and proposed detailed algorithms to compute the CEH from the reachability graph. The CEH can not only save the computational cost for the shortest path search but also impose a hierarchy on the locations. Such a hierarchy explicitly describes the relations between locations: the child location must be reached through its parent location from outside. The location hierarchy coincides with human cognition during navigation, so it is more suitable to be displayed on the mobile device than on a plain floor plan.

To extend our research on indoor location modeling, we focus on the following issues:

- Our model does not fully support object moving. In other words, the position change might drastically change the topology as the objects (the users in particular) move. If we need to adapt it to a dynamic environment, the model should support incremental and local updates as the positions of the objects change.
- Our model can handle only one type of relation; that is, reachability. Nonetheless, the context in indoor environments is more complicated than mere physical

reachability. For example, we may need to know the temperature and humidity distribution in the environments. This problem of multi-relation modeling poses a new challenge to our research.

4. Query Processing: Broadcast vs. Point-to-Point

LBSs by and large assume wireless communication since both the clients and the data (e.g., vehicles being tracked) move. Wireless communication supports two basic data dissemination methods. In *periodic broadcast*, data are periodically broadcast on a wireless channel accessible to all clients in range. A mobile client listens to the broadcast channel and downloads the data that matches the query. In *on-demand* access, a mobile client establishes a point-to-point connection to the server and submits requests to and receives results from the server on the established channel.

The role of periodic broadcast is crucial in LBSs. This is because both the number of clients and the amount of requests to be supported by the LBS could be huge. Periodic broadcast can be used to disseminate information to all participants at very low cost. Early work on data broadcast focused on non-spatial data, but in order to be useful in LBSs, a data broadcast system must support spatial queries and spatial data. For example, in an intelligent transportation system, the locations of all monitored vehicles can be broadcast so that an onboard navigation system can decide autonomously how to navigate through a crowded downtown area. Similar needs can be envisaged in the coordination of mobile robots on a factory floor.

In this section, we investigate the spatial query processing technique in a broadcast environment followed by location-based spatial queries processing in a demand access mode.

4.1 General Spatial Query Processing on Broadcast Data

Search algorithms and index methods for wireless broadcast channels should avoid back-tracking. This is because, in wireless broadcast, data are available “on air” in the sense that they are available on the channel transiently. When a data item is missed, the client has to wait for the next broadcast cycle, which takes a lot of time. Existing database indexes were obviously not designed to meet this requirement because data are stored on disks and can be randomly accessed any time. Hence, they perform poorly on broadcast data. Since most spatial queries involve searching on objects that are close to each other, a space-filling curve such as a Hilbert Curve can be applied to arrange objects on the broadcast so that the objects in proximity are close together. Fig.1 shows a Hilbert Curve of order 2 which arranges the 16 spatial objects in a 1-dimensional data broadcast.

Algorithms can be developed to answer window and *kNN* queries on the broadcast [ZL05]. Given a query window, we can identify the first and the last objects within the window that the Hilbert Curve passes through. In the example shown in Fig. 1, the dashed rectangle is the query window, and objects 2 and 13 are, respectively, the first and the last objects within the query window visited by the Hilbert Curve. Any objects outside this range are guaranteed to not fall within the query window. Thus, the client only needs to download and verify the objects between objects 2 and 13, inclusive, to eliminate objects that fall outside the window (i.e., objects 10, 11, and 12). *kNN* queries can be answered in a similar manner by first estimating the bounds within which the *k* nearest neighbors can be found followed by a detailed checking of the Euclidean distances between the candidate objects and the query.

The adoption of a space-filling curve can avoid the clients' back-tracking the broadcast channel several times when retrieving objects for spatial queries. This is of essential importance to save the power consumption of the clients and improve the response time. However, in on-demand access mode, the processing of location-based spatial queries raises different issues due to the mobility of the clients.

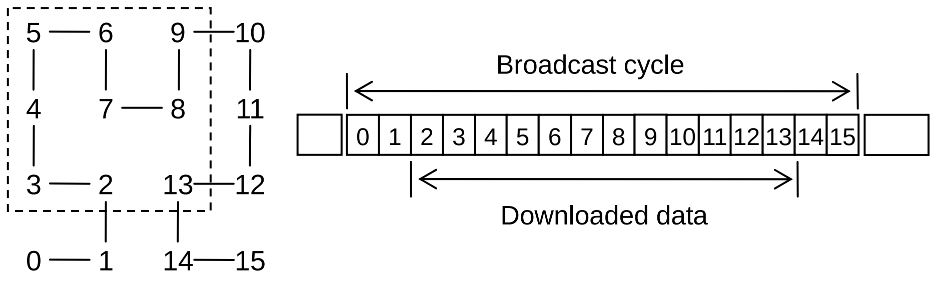
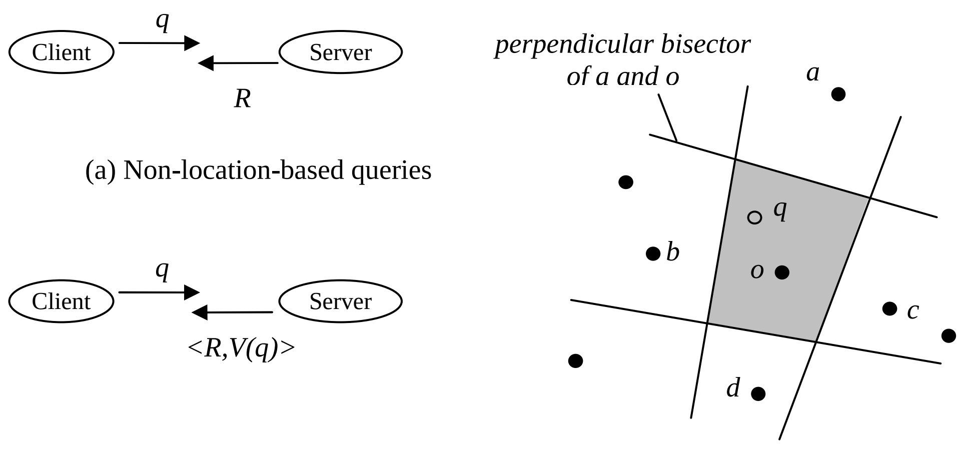


Fig. 1: Answering Window Queries based on Hilbert Curve

4.2 Location-based Spatial Queries

In contrast to conventional spatial processing, queries in LBSs are mostly concerned about objects around the user's current position. Nearest neighbor queries are one example. In a non-location-based system, responses to queries can be cached at the clients and are reusable if the same queries are asked again. In LBSs, however, users are moving frequently. Since the result of a query is only valid for a particular user location, new queries have to be sent to the server whenever there is a location update, resulting in high network transfer and server processing cost. To alleviate this problem, the concept of *validity region* can be used. The validity region of a query indicates the geographic area(s) within which the result of the query remains valid and is returned to the user together with the query result. The mobile client is then able to determine whether a new query should be issued by verifying whether it is still inside the validity region. The utilization of *validity region* reduces significantly the number of new queries issued to the server and thus the communication via the wireless channel.



(a) Non-location-based queries

(b) Location-based queries

(c) Validity region for a *NN* query

Fig. 2: An Example of Single Nearest-Neighbor Query

The validity region of course depends on the type of queries. Methods for determining the validity region for *NN*, *kNN*, and window queries have been developed [ZZP+03]. Fig. 2 (c) illustrates the validity region of *NN* queries which are the same as Voronoi cells. The representation scheme for validity regions should (i) be as small as possible to reduce the network cost (provided that it captures accurately the shape of the validity region) and (ii) facilitate the validity checking which is performed on a client with limited computational capability. Representing a validity region as a polygon may cause a high overhead. A better way is to identify the *influence objects* of a query

q as the data points that *contribute* at least one to the validity region $V(q)$. In other words, the influence set is the minimal set of objects which determines the validity region. In Fig. 2, the influence objects of query q are those whose perpendicular bisectors with object o constitute the edges of the Voronoi cell. Finally, the server returns the query result to the client along with the influence objects. The validity checking at the client side is performed by examining whether the new user location is inside the half-plane formed with respect to each influence point, which is equivalent to checking whether the query focus lies in the Voronoi cell of the result.

5. Scheduling and Monitoring Spatial Queries

5.1 Scheduling Spatial Queries Processing

One unmistakable challenge that LBSs have to face is the huge user population and the resulting large workload generated. Traditional spatial database research focuses on optimizing the I/O cost for a single query. However, in LBSs where a large spatial locality could be expected, queries received in an interval may access the same portion of the database and thus share some common result objects. For instance, users in a busy shopping area may want to display on their PDAs a downtown map with shopping malls overlaid on it. These are equivalent to a number of window queries around the users' positions. Although the windows are different, they likely overlap, resulting in the accessing of overlapping objects from the map database. Inter-query optimization can be utilized at the server to reduce the I/O cost and response time of the queries.

To achieve this objective, multiple spatial window queries can be parallelized, decomposed, scheduled, and processed under a real-time workload in order to enhance system runtime performance; for example, I/O cost and response time. Query locality can be used to decompose and group overlapping queries into independent jobs, which are then combined to minimize redundant I/Os. The essential idea is to eliminate duplicate I/O accesses to common index nodes and data pages. In addition, jobs may be scheduled to minimize the mean query response time. In principle, processing queries close to one another will save more I/O cost because there is a high chance that they can share some MBRs in the R-tree index as well as data objects. An innovative method to quantify the closeness and degree of overlapping in terms of I/O has been developed based on window query decomposition in [HZL03]. In addition, in a practical implementation where a fair amount of main memory is available, caching can be used to reduce the I/O cost. For spatial data indexed by an R-tree, high-level R-tree nodes can be cached in memory.

5.2 Monitoring Continuous Queries on Moving Objects

Most of the existing LBSs assume that data objects are static (e.g., shopping malls and gas stations) and the location-based queries are applied on these static objects. As discussed in Section 2, applications such as location-based alerts and surveillance systems require continuous monitoring of the locations of certain moving objects, such as cargos, security guards, children, and so on. For instance, we may issue a window query, monitor the number of security guards in the window, and send out an alert if the number falls below a certain threshold. Likewise, we can issue several nearest neighbor queries centered around the major facilities and monitor the nearest police patrols so that they can be dispatched quickly should any problem arise. Generally, we are interested in the result changes over time for some spatial queries. This is the problem known as “monitoring continuous spatial queries.”

Most of the existing work assumed that clients are autonomous in that they continuously measure their positions and report location updates to the server periodically. Thus, server-side indexing and processing methods have been devised to minimize the CPU and I/O costs for handling these updates and re-evaluating the query results [ISS03,MXA04,YPK05]. In these methods, the update frequency is crucial in determining the system performance – high update frequency would overload both the clients and servers in terms of high communication and processing costs while low update frequency would introduce errors into the monitored results.

An alternative to periodic location update is to let the clients be aware of the spatial queries that are being monitored so that they would update their locations on the server only when the query results were about to change. The main idea is that the server maintains a “safe region” for each moving client. The “safe region” is created to guarantee that the results of the monitoring queries remain unchanged and as such need not be re-evaluated as long as the clients are moving within their own safe regions. Once a client moves out of its safe region, it initiates a location update. The server identifies the queries being affected by this update and re-evaluates them incrementally. At the same time, the server re-computes a new safe region of this client and sends it back to the client. Fig. 3 (a) shows four window queries, Q_1 through Q_4. The number of clients inside each window is being monitored. It is clear that when a client (shown as a dot in the figure) moves within the shaded rectangle S_1, it will not affect the results of the four queries. However, when it moves out of S_1, its new location must be sent to the server in order to re-evaluate the queries and re-compute a new safe region for the client accordingly.

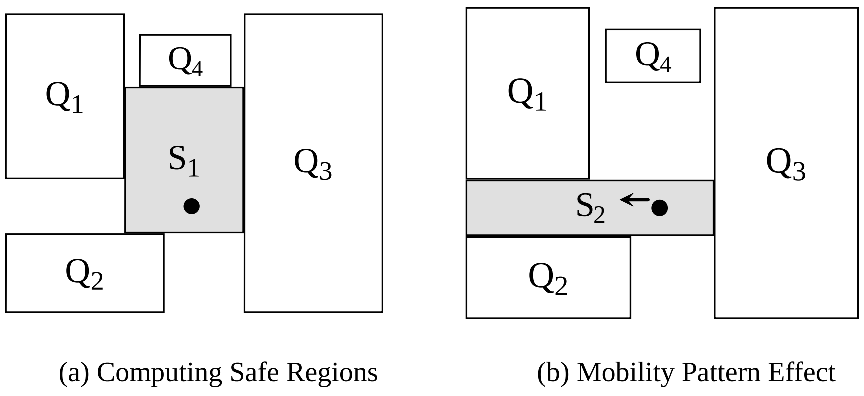


Fig. 3: Computing the Safe Region to Reduce Location Update Workload

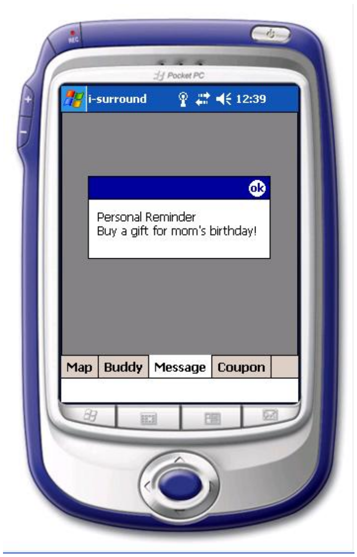
For *kNN* queries, the exact locations of the clients must be known in order to determine which client is closer to the query point. However, since a client can move freely inside a safe region without informing the server, the safe regions can only be used to identify a number of potential candidates and the server still needs to probe these candidates to request their exact positions.

In order to reduce the communication cost between the clients and the server, we need to reduce the number of location updates and server probes during monitoring. As such, we need to find the largest possible safe regions for the clients to reduce the number of location updates and devise efficient algorithms for re-evaluating the query results to reduce the number of server probes. When computing the safe regions, we also need to consider the effect of the client’s mobility pattern. This is because a safe region does not have to occupy the largest area to be the best. For example, in Fig. 3 (a), S_1 is the largest safe region for the client. However, as shown in Fig. 3 (b), if we know that the client is moving toward the west, S_2 is a better safe region than S_1 because it takes longer for the client to move out of S_2 and thus defer the next location update.

6. System Prototypes

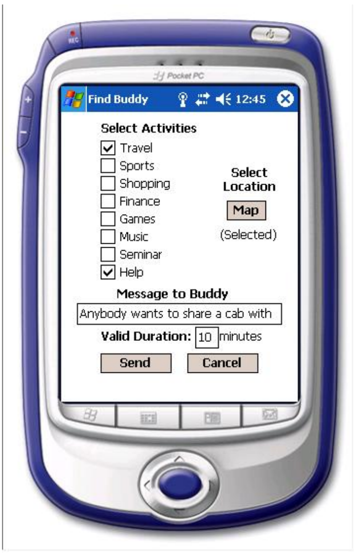
In this section, we describe two location-based prototypes that incorporate most of the research ideas described in this paper. The first prototype is essentially a location-based messaging platform called *i-surround*. It provides users with context-aware services that allow them to communicate with each other and to obtain relevant information based on their current locations. *i-surround* supports interaction with buddies, location-aware personal reminder, and information publishing and intelligent filtering.

Several screen dumps are shown in Fig. 4. *i-surround* was designed to support various kinds of client devices, such as PDAs, cell phones, smart phones, and PCs. Therefore, it is required to support different location update schemes, such as automatic location updates for the clients equipped with location sensing devices and manual location updates for users without location sensing devices. For the latter, *i-surround* allows users to specify their locations by drawing on the map. In order to protect user privacy (i.e., disclosure of user location and profile) when he/she is using the buddy-finding service, a two-way-hand-shaking technique is employed. The system allows a user to set his/her profile and mode, based on which type of information can be filtered. In addition, the “smart” interface on the client side keeps track of the user’s recent actions on various types of messages and accordingly ranks the messages so that the most relevant are displayed first.



Personal Reminder
Buy a gift for mom's birthday!

(a) Personal reminder when the user passes by a predefined location



Select Activities

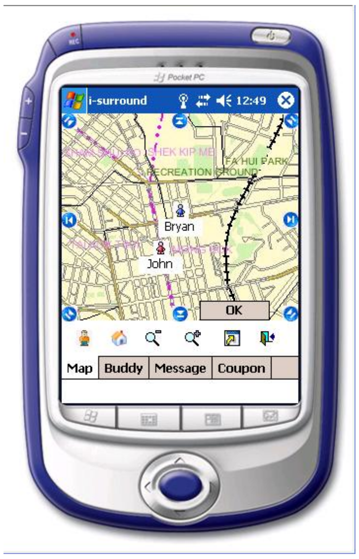
- [✓] Travel
- [ ] Sports
- [ ] Shopping
- [ ] Finance
- [ ] Games
- [ ] Music
- [ ] Seminar
- [✓] Help

Select Location: Map (Selected)

Message to Buddy: Anybody wants to share a cab with

Valid Duration: 10 minutes

(b) Interface for buddy finder (specify location, activities, and the time duration)

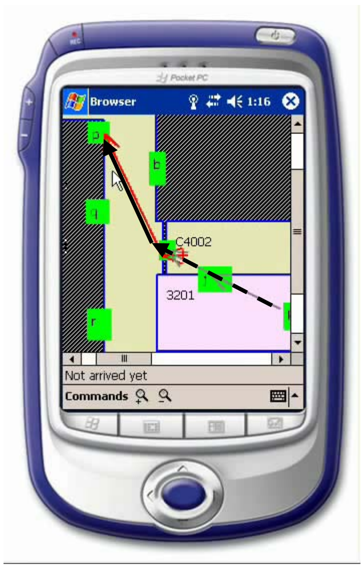


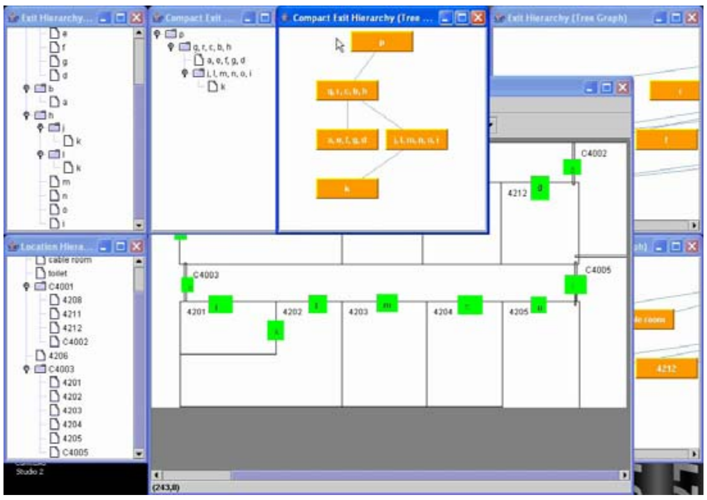
(c) Display the location of a buddy on the map

Fig. 4: Screen-dumps of *i-surround*

The second prototype is an indoor location navigation system for PDAs. It is based on the semantic location model described in Section 3. The client module on the PDA can display a context-aware floor map, which shows only the locations the user can reach directly from his/her current location (ref. Fig. 5 (a)). This map is more suitable for a small screen size than the traditional static map as it helps the user focus on those locations that are more relevant to the current navigation. We implement this feature by only drawing the current location with its parent, siblings, and children in the location hierarchy of the semantic location model described in Section 3. In Fig. 5 (a), since the user is in Room 3201, only this room and the neighboring corridors are displayed while all the other rooms are in shadow as they are not directly reachable.

Therefore, the user can concentrate on all the exits that are available to him/her.





(a) Client side (b) Location model creation

Fig. 5: Indoor Navigation System on a Mobile PDA

The second characteristic of this prototype is the shortest path search. By applying the CEH of the semantic location model, the client can efficiently compute the shortest path to the destination. In Fig. 5 (a), the dashed arrows show the path that has been passed through and the solid arrows show the next exit along the shortest path. An interesting feature of our prototype is that the shortest path is recomputed every time the user moves to a new location so that the user can adjust his/her movement at anytime, even after he/she goes the wrong way. The CPU cost for recomputation is reasonable thanks to the efficiency of the CEH.

To create the location model, the prototype system has a floor plan editing tool to allow the user to create a floor plan by drawing rooms, corridors, or other types of locations, and connecting them through exits. The “direct reachable” relation between the exits is obtained on the fly so that the tool can automatically maintain the up-to-date location hierarchy and the exit hierarchy (see Fig. 5 (b)). The resulting location model is stored in a MySQL database and is accessed wirelessly by the client module.

7. Conclusion and Future Research Direction

This paper has reviewed some of our research on location modeling, location updates, caching of location-dependent data, and batch processing of spatial queries. A couple of prototypes were briefly described to illustrate where the research may be applied in real-life applications. LBSs draw from many different research areas, ranging from systems to applications to user modeling. Research in these different areas must cope with the "location" aspects of LBSs. Regarding the system architecture, while most traditional research is based on the client-server architecture, the mobility and limited communication ranges of the clients make it natural to consider LBSs on peer-to-peer and mobile ad-hoc architectures. The monitoring of the moving peers and the dynamic formation of mobile communities to accomplish a common goal will be a major research challenge to us. In terms of data dissemination, data broadcast has been a cornerstone technology in mobile computing. Work on scheduling, caching and indexing of broadcast data will allow LBSs to efficiently push data to users who need them. Techniques for distributing location-based data to access points and scheduling them for broadcast according to local interests are essential to the efficient utilization of bandwidth and reduction of access time. At the backend, LBSs must be able to acquire and model, among other things, the location properties of all concerned objects and be able to do so in a large scale. This will require a distributed and collaborative infrastructure where “spheres of interest” can be developed and maintained locally and at the same time collaborate globally. These research issues are definitely very difficult by themselves but to integrate the results synergistically to meet the demands of the fast evolving LBS marketplace is even more challenging to researchers and application builders alike.

References

[HL04] H. Hu and D.L. Lee, Semantic Location Modeling for Location Navigation in Mobile Environment, *Proceedings of the 5th International Conference on Mobile Data Management 2004 (MDM’04)*, pp. 52-61.

[HZL03] H. Hu, M. Zhu, and D.L. Lee, Towards Real-time Parallel Processing of Spatial Queries, *Proceedings of the 2003 International Conference on Parallel Processing*, pp. 565-572.

[ISS03] G.S. Iwerks, H. Samet, and K. Smith, Continuous K-Nearest Neighbor Queries for Continuously Moving Points with Updates, *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB’ 03)*, pp. 512-523.

[MXA04] M.F. Mokbel, X. Xiong, and W.G. Aref, SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases, *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD’04)*, pp. 623-634.

[Web] http://www.wavemarket.com

[YPK05] X. Yu, K.Q. Pu, and N. Koudas. Monitoring K-Nearest Neighbor Queries Over Moving Objects, *Proceedings of the 21st International Conference on Data Engineering (ICDE' 05)*, to appear.

[ZL05] B. Zheng and D.L. Lee, Information Dissemination via Wireless Broadcast, *Communications of the ACM*, 2005, to appear.

[ZZP+03] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D.L. Lee, Location-Based Spatial Queries, *Proceedings of ACM SIGMOD Conference on Management of Data*, *2003*, pp. 443-454.