Design and Evaluation of a Hybrid Physical Space Service for Pervasive Computing Applications

Nancy Miller and Peter Steenkiste
Carnegie Mellon University
{nam, prs}@cs.cmu.edu

***Abstract*— In this paper we present the design and implementation of a space service that gives pervasive computing applications both a hierarchical and coordinate-based view of physical space. The space service supports both indoor and outdoor campus environments and, besides relational and coordinate space information, it also gives information on how spaces are connected and relevant properties of spaces. We also evaluate the effectiveness of our design using a set of diverse applications, including finding the best "walking" path between two locations, giving walking directions to people, WiFi-based localization, and drawing and annotating maps. We have found that all of our applications depend on both the hierarchical and coordinate views offered by the service, suggesting that the hybrid space model is a very powerful paradigm.**

I. Introduction

One of the more challenging problems faced by pervasive computing researchers is that of how to support applications that need to reason about physical space. In order to support mobile users effectively, we need a "physical space service" that provides space information to a wide variety of applications that are used in very diverse environments. Some applications naturally benefit from a hierarchical representation of space that uses real-world (symbolic) place names. Such a representation directly supports relational queries (e.g. whether two rooms are on the same floor) and the names are easy to interpret by users. However, a hierarchical representation lacks precision, so it is not appropriate for applications that need to calculate distances (e.g. walking distance to a printer). For such applications, a coordinate-based representation is needed.

To address these challenges, the CMU Aura project developed a "hybrid" space service that provides both a hierarchical and coordinate-based view of physical spaces in an integrated fashion [10]. The initial research focused on the development of the hybrid space model and on demonstrating its functionality and performance using simple applications. The question of whether the hybrid space model was useful for a wide variety of applications was not addressed. Since deploying the hybrid space service in 2002, we have used the service for a number of pervasive computing applications. They range from simple applications such as finding the closest printer, to more involved applications such as identifying the best walking path based on user preferences. Although most of our applications initially targeted an office environment, we have also used the space service in museum and residential environments.

In this paper we use these applications to evaluate the functionality offered by our hybrid space service. We have found that the hybrid space model is a very powerful paradigm. In fact, all the applications that use the space service benefit from having both the hierarchical and coordinate views of space. However, we have also learned that the hybrid space model is not sufficient. Extensions are needed in three areas: applications need information on connectivity between spaces (e.g. doors, stairs, etc.); we need to support both inside and outside spaces, since mobile users naturally move between buildings; and finally, applications need additional information about spaces (such as the type of space) so they can effectively communicate with users. These extensions need to be integrated carefully into the hybrid space model so the information can be retrieved easily and efficiently.

The paper is organized as follows. Section II presents a motivating application. In Sections III and IV we describe the basic hybrid space service and our extensions. In Sections V through VII we use a set of applications to evaluate both the performance and functionality of the hybrid space service. Finally we discuss related work and summarize our results.

II. Problem definition and requirements

Consider the following motivating scenario:

Mary, a new student, just arrived on campus. How might a campus guide help her? As she starts her day, she first needs to find a convenient place to eat breakfast, so she asks the campus guide for the nearest restaurants. The guide determines her location based on her WiFi device and offers her a list of options. After she has selected one, the guide gives her directions and shows her a marked up map. It picks a path that is consistent with her preferences, which she specified earlier. During the day, the campus guide also gets her directions to the closest restroom, a quiet place to study, and a nearby printer to print out her class schedule. In the afternoon, she has her first class. The campus guide estimates how long it will take to walk to the auditorium, and alerts her in time so she can get a good seat. It takes into account the fact that the elevators are busy right before class.

The campus guide in the above scenario consists of a set of applications, described in Sections V and VI, that use space information in a variety of ways. While the applications are diverse, we can identify several common requirements for the space service:

- **Diverse space information.** We need to be able to express exact points (e.g., printer location), named en-

closed spaces (e.g. an office), ill-defined unnamed spaces (e.g. the area covered by a wireless access point), and collections of spaces (e.g. a floor of offices).
- **Support for diverse environments.** Users will naturally move between inside and outside areas, so both must be supported by the space service.
- **Diverse uses of information.** Applications use space information for deriving properties of spaces (e.g. calculate distances), optimization (e.g. shortest walking path), and effective communication with users (e.g. give directions).
- **Scalability and efficiency.** The space service must respond to queries in a reasonable amount of time, even for large spaces such as an entire campus.

Besides the above requirements for the space service, the applications also share a number of requirements. For example, several applications make use of user preferences or need information not related to physical spaces (e.g. class schedules), including possibly dynamic information (e.g. when elevators are busy). In Aura, this information is provided by the Contextual Information Service (CIS), which makes a variety of information available to context-aware applications [11].

III. Space service

In this section, we describe the hybrid space service that has been used in project Aura for the past four years.

A. A Hybrid Space Model

In [10], we introduced a "hybrid" space model that captures both the hierarchical and coordinate aspects of physical space in an integrated fashion. The motivation for the hybrid design is that hierarchical and coordinate space models have complementary strengths and weaknesses [10]. As a result, pervasive computing applications can benefit from having access to both "views" of physical space. In the remainder of this section we first introduce the hierarchical location model and then elaborate on the hybrid space model.

The starting point for the hybrid model is the hierarchical location model: we view the world as a hierarchy of spaces where each level refines and subdivides the spaces of the previous level. We augment this hierarchical model with a coordinate view by allowing each space in the hierarchy to define a coordinate system that can be used to specify points or areas within that space. The coordinates allow us to define points or areas for which there is no representation in the hierarchical name system. Spaces may want to use their own coordinate systems for several reasons. First, they may use different types of coordinates (e.g. geometric versus GPS). Second, it makes it possible to enter data and serve queries for different spaces independently; this helps the system scale.

Locations are represented using the *Aura Location Identifier* (ALI). ALIs can represent a point (e.g. location of a printer), a physical space (e.g. a room), or an arbitrary area (e.g. the area covered by a wireless access point). Generally, an ALI consists of a hierarchical name, identifying a physical space. For example, the following space identifier represents room 3115 on the 3rd floor of Wean Hall at CMU:

ali://cmu/wean-hall/floor3/3100-corridor/3115

This symbolic name is optionally followed by coordinates identifying a point or area in that physical space. The coordinates are relative to the coordinate system associated with that physical space. For example, the following point identifier represents the point located at (10,4,1) within the coordinate system of room 3718:

ali://cmu/wean-hall/floor3/3700-corridor/3718#(10,4,1)

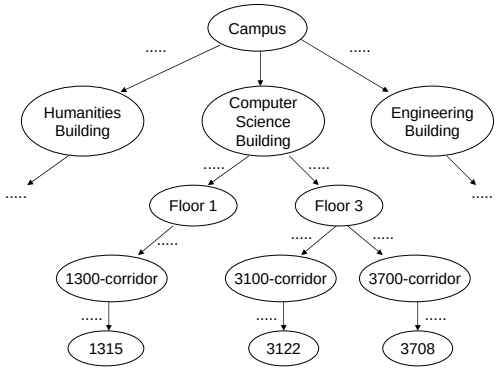

Fig. 1. Example Space Tree

The hybrid space model represents the physical space as a *hierarchical space tree*. The structure of the tree reflects the hierarchical aspect of the space model, while each node in the tree includes geometric attributes and other properties. These attributes include the shape of the space (e.g. cube, cylinder, or any other three dimensional polygon), and the origin and orientation of the space's coordinate system relative to the coordinate system of its parent. We use this information to translate between different coordinate systems. For example, if we have local coordinates for two points in different rooms, we can transform them into the shared coordinate system of the building, allowing us to calculate the distance between the two points. Figure 1 shows an example of a partial space tree.

B. A Hybrid Space Service

The space service stores the geometric space tree in a relational database as explained in [10], although we currently use MySQL instead of PostgreSQL to simplify maintenance. We populate the database using a data entry program that reads building maps in an AutoCAD file format [1]. Spatial data can be manipulated using operations that fall into three categories. Relational queries, such as "contains" and "subspaces", leverage the space hierarchy. Geometric queries, such as "distance" and coordinate translation, use the coordinate representation. Finally, we have hybrid queries that map between hierarchical names and coordinates. For example, we can translate the coordinates of a point in a building into the hierarchical name of the room that contains it.

IV. Making the Space Service Useful

Over the past several years, we have used our hybrid space service to support a variety of applications. As we describe in Sections V through VII, we have learned that the hybrid space model is very powerful, especially for more complex

From	ali://cmu/wean-hall/floor3/3100-corridor/3115
To	ali://cmu/wean-hall/floor3/3100-corridor/3117
Location	ali://cmu/wean-hall/floor3#(12,35,8)
Properties	extra information

Fig. 2. Key fields in the representation of a door

applications. However, we have also found that in order to effectively support diverse applications, the service needs to provide additional types of information. For example, many applications need information on how spaces are connected and need to interact with users in physical space. This information must be properly integrated with the hierarchical space tree so it can be accessed efficiently. Moreover, the space service needs to cover outside areas, not just buildings. We describe these extensions in this section.

A. Capturing connectivity

The original space service did not provide information on spatial connectivity. This was not needed for simple applications such as keeping track of where objects are located, but navigation applications for mobile users clearly need information on how spaces are connected.

There are many ways of adding this information to the space service. The obvious solution is to associate the connectivity information with the nodes in the space tree, i.e. we associate with each node (space) a list of the nodes (spaces) that it is connected to. This approach has the advantage that it automatically captures the hierarchical nature of the connectivity information. For example, information on doors and other connections between spaces would be stored at the "room" level, while elevator information would be stored at the "floor" level. However, the disadvantage is that this combines diverse information in the same data structure, which can complicate applications and make queries more expensive. For example, connectivity-oriented applications will have to search through the space tree, which contains much more information than they need.

Instead we opted for a representation that stores connectivity information separately and uses ALIs to link it to the hierarchical space tree. Figure 2 shows how a door is represented. It identifies the two rooms it connects ("To" and "From" fields) and the location of the door in the coordinate system of a space that is the shared parent of the two rooms (the floor, in this case). Additional fields are used to represent properties (e.g. emergency use only). The door specifies connectivity in one direction, allowing us to represent doors that are not symmetric (e.g. exit-only doors in a museum). Most doors will have two entries in the table, one for each direction.

Similar data structures are used to represent connectivity at other levels of the space hierarchy. The main difference is that the ALIs point to different levels in the hierarchical space tree. For example, a building typically has connectivity at the room level (doors), floor level (elevators and stairs) and building level (external doors). Each of these are stored in different tables in the database. However, the tables have the same format, thus simplifying development.

B. Representing outdoor space information

The goal of our original space service was to support pervasive computing applications inside buildings. We chose to work with inside spaces because of the Aura project's focus on office environments. Moreover, the representation of outside spaces had already been explored very extensively [16], [15]. However, we discovered that applications need to support users not only as they move around inside a building, but also when they walk between buildings. Instead of using a different representation for outside spaces, similar to that used by applications that provide driving directions, we decided to reuse our indoor representation for outdoor areas. The advantage of this approach is that it simplifies applications that work both inside and outside. The simple and natural way to do this is to add a level at the top of the hierarchical space tree (Figure 1). Of course, the problem is more difficult than just extending the space tree, since outdoor spaces generally do not look like buildings or even floors in buildings.

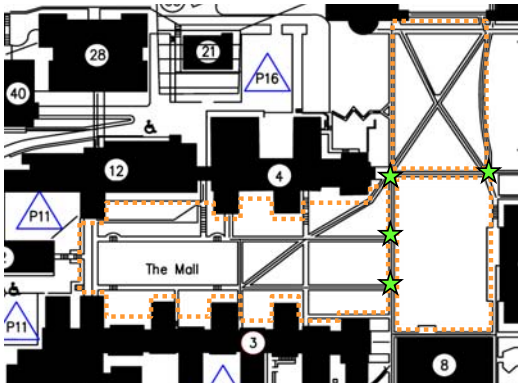

Fig. 3. Spaces in an outdoor area

When comparing indoor and outdoor space representations used by pervasive computing applications, we found that they are fundamentally different. Indoor representations tend to view the physical world as consisting of sets of *spaces*, often organized in a hierarchy. In contrast, many outdoor representations look at the physical world as consisting of sets of *paths* that provide a means of moving between two points. Paths come in many forms, such as highways, residential streets, and pedestrian walkways. The distinction between a path-based versus a space-based representation is probably the result of differences in the nature of both spaces and applications:

- Indoor spaces typically consist of open areas in which people can move around freely, while many outside spaces tend to restrict the movement of most users to specific paths.
- Most applications designed for outdoor use focus on navigation, so paths are a natural primitive for the space representation. In contrast, indoor pervasive computing applications tend to be centered around rooms; they need to know where (in what room) people and objects are located. Indoor navigation is also of interest, but it is typically not the predominant application.

Our focus for the outdoor space representation is on campus

environments, which typically consist of different areas, some of which restrict movement to paths. Figure 3 illustrates this using a partial map of the CMU campus; some areas are marked with dashed lines. We decided to represent the outside areas as spaces in the space tree, so the outside part of the campus is handled as a floor of a building. However, it is possible to add additional hierarchy, similar to the way we break up large floors into separate corridors. For the connectivity between outside spaces we use the points where walkways cross area boundaries (stars in Figure 3). When an outside space is directly accessible from a building, we use an "external" connectivity table to represent the outside door. This table is similar to the one in Figure 2, but it represents connectivity at the top level of the hierarchical space tree. Finally, all coordinates in the space service are 3-dimensional, so we automatically capture relative height between areas.

In the current implementation, we assume that the mobility model in an outside space is similar to that of a room: people can walk in a straight line from any door/intersection to any other door/intersection. While this works reasonably well for our campus, in general it may be necessary to add "paths" to the space model that specify how people can move in a space. This may also be useful for some indoor spaces, since large "rooms" (e.g. auditoriums, convention floors) often restrict movement along specific paths. If we want to extend the Aura space service to support "path centric" spaces (e.g. urban or rural areas), we will have to consider giving paths a more prominent role in the space representation.

C. Adding semantic information

The space service described so far allows applications to make decisions involving physical space. However, additional information about physical spaces is often needed, especially when applications need to interact with users. For example, while an office and an outside space have the same internal representation, it is useful to distinguish between them when giving directions to users. Similarly, it is also useful to associate additional information with "space connections" (doors, elevators). Relevant information can include usage restrictions (e.g. not handicapped accessible) and hints for people, such as the color of a door, or distinctive landmarks.

We could add this information to the space service in two ways. The first option is to embed the information into the hierarchical space tree, by adding fields to the nodes. The alternative is to create a separate database or database table which uses ALIs to link the information to the relevant spaces or connections. The best option depends on how the information will be used. If the information is fairly standard (e.g. type of space) and is likely to be used by most applications, it may be appropriate to embed it in the space tree. If the information is more specialized (e.g. waiting times for elevators, distinctive landmarks), storing it separately is likely to be more efficient.

Most of the information we need to store lends itself to having one or more "data" fields associated with spaces or connections. Information is currently represented using attribute-value pairs (e.g. space-type:office). We hope to replace this with an ontology-based representation [22] in the future.

V. Walking Distance Estimation

In this section we discuss how our space service supports the calculation of (shortest) walking paths and distances. We first describe our design for the navigation algorithm and then present our implementation and its performance.

A. Design

Calculating walking paths and distances using brute force solutions such as exhaustive search are clearly not going to scale. Instead, we achieve scalability by leveraging the hierarchical nature of the space service. Our solution finds paths at three different levels: Inter-building, Multi-floor, and Single-floor. The Inter-building level deals with queries that span multiple buildings, including outside spaces (Section IV-B). The Multi-floor level of the hierarchy concerns a single building where the starting and ending points of the path are on different floors. The Single-floor case handles queries that are contained on one floor of a single building. For each level of the hierarchy, we build a graph consisting of the spaces and connections at that level and we use a graph-based shortest path algorithm to find an optimal walking path. The three levels build on each other as is shown in Figure 4; the Inter-Building component uses Multi-Floor component.

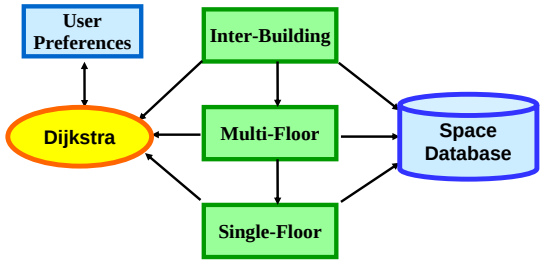

Fig. 4. Hierarchical Path Finding Algorithm

For the Single-floor case, the most obvious way to design the graph is to use the spaces as the nodes, and the doors as the links. However, this approach has the disadvantage that it is not clear how to calculate the distance (or weight) for each link, information that is needed to optimize walking distance. It is possible to use the distance between the centers of the spaces as the weight for the links, but this is at best an approximation, since people will not always walk through the center of the room. This approach also does not work well at the higher levels of the hierarchy. Therefore, we chose to do it the other way around: the doors are the nodes and the spaces are the links. Moreover, the starting point (e.g. the user's current location) and end point (e.g. the location of a printer) of the path are also nodes in the graph. A shortest path will consist of the starting point, followed by a sequence of doors, and the end point. The appropriate weight for the links in the graph is the geographic distance between the nodes, which can be readily calculated based on the coordinate information in the space service database. Note that with this design, all links are bi-directional.

The Multi-floor case uses the same design approach, except that the "connections" are now stairs and elevators instead of

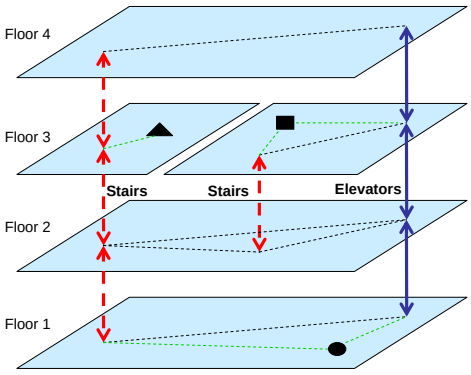

Fig. 5. Multi-floor example

doors. Specifically, the nodes in the graph include, besides the starting and ending point, the locations where people can enter or leave a stairwell or an elevator. Figure 5 shows an example of a Multi-floor graph. The solid arrows represent an elevator (on the right) and dashed arrows show stairs (in the middle and on the left). The solid and dashed vertical arrows are links in the graph and their endpoints are nodes. The weight associated with those links could be based on the user's preferences with respect the use of stairs and elevators. On each floor, the dotted lines represent the paths between the stairs and elevators, as well as the paths from the starting point and endpoint (the triangle, square, and circle) to the stairs and elevator on that floor. They are also links in the graph and their weight is determined by the Single-floor algorithm based on the walking distance.

An appropriate path can now be determined by applying a shortest path algorithm to the graph. For example, suppose the triangle and square are the starting point and end point in in Figure 5. Assuming no preference between stairs and elevators (i.e. they have equal weight), the shortest path would use the left stairwell to go to the second floor and then the central stairwell to go back to the right part of the third floor. However, if the user indicates a strong preference for using elevators, the algorithm would return a path that uses the elevators to travel from the second to the third floor (assuming walking down stairs gets a lower weight than walking up stairs).

The graph for the Inter-Building case is constructed in a manner similar to the Multi-Floor case. The nodes include the external doors of buildings, the connections between outdoor spaces, and the starting point and the end point. The weights of the links are calculated recursively using the Multi-Floor and Single-Floor algorithms.

The above algorithm uses and depends on both the hierarchical and coordinate aspects of the space service. The hierarchical information, including the hierarchical connectivity information, is used to structure the multi-level algorithm and to build the graphs at each level. The coordinate information is used to calculate the weights of the links.

B. Implementation

When it receives a query, the applications first determines which level of the hierarchy it needs to start from (Figure 4), that is, whether the start and end locations are on the same floor, on different floors of a building, or in different buildings (including outside spaces). It then builds the relevant graph for that level. This requires only connectivity information for that level, which, as we explained in Section IV-A, which is stored separately from the space tree so it can be accessed efficiently. We also need to add links for the starting and ending locations. In the multi-floor case, it may be necessary add additional links for some buildings. For example, when not all stairs and elevators reach all floors, as in Figure 5, we may also need to include links between stairs and elevators on intermediate floors.

Once the graphs have been built, weights are applied to the links. As described above, in the Single-floor case, the weights are distances, which can be calculated using the coordinate information. In the Multi-Floor (Multi-Building) case, weights are calculated by calling the Single-Floor (Multi-Floor) algorithm. It is also possible to adjust the weights to account for crowded rooms or user preferences. Once the graphs have been built and weighted, we use Dijkstra's Algorithm [7] to determine the shortest path.

This basic implementation can be improved substantially using some simple optimizations. First, when we build the Single-Floor graph, we can prune the graph by leaving out the doors that are the only entrance to spaces that are not involved in our query (these would be leaf nodes in the graph). Since many rooms have only one door, this can reduce the size of the graph substantially. For example, for buildings on the CMU campus, this pruning optimization reduces the size of the graph by about 75%. We can further improve efficiency using caching. Entire graphs, especially at the Multi-Floor and Inter-Building level, are likely to be the same for many queries, so they can cached. Similarly, assuming the weights of edges are the same, the results of queries can often be cached. For example, the paths and distances between external doors of buildings (for a Multi-Building query) and between stairs and elevators (Multi-Floor query) can be reused.

C. Performance Results

In this section we will present the results of some experiments run to test our walking path calculation. Our space service is implemented in Java (jdk 1.5) and uses MySQL as the database for storing space information. For these experiments, both were running on a 1.5GHz Intel processor with 256MB of memory.

We ran three types of experiments, one to test each level of the hierarchy: Inter-building, Multi-floor, and Single-floor. For each experiment, we ran the same query in four different configurations to show the effect of our optimizations: with and without pruning and with and without caching. The Inter-building query was run using a data set that contained two buildings, and the starting and ending points were on the same floor as the door connecting the buildings. The Single-floor and Multi-floor queries were run on a data set that consisted of two adjacent floors with three stairwells and one elevator connecting them. Each floor had about 120 rooms and corridors, and included multiple cycles.

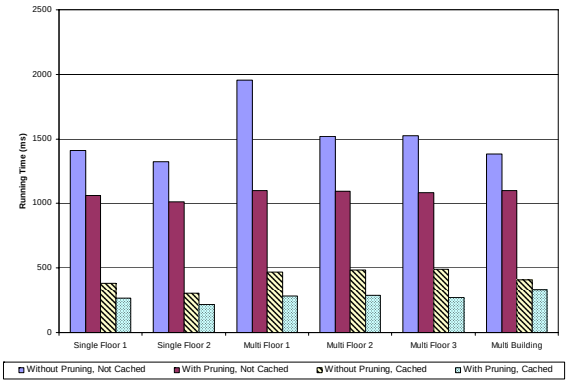

Fig. 6. Experimental Results

Figure 6 shows the running times for these queries in milliseconds. The fully optimized case took about 200-300 ms to answer each of our queries, which is a reasonable response time. As expected, both pruning and caching have a significant effect on the running time of the query, but caching clearly pays off the most. Pruning reduced the running time by about 30% on average, and caching reduced it by an average of 73%.

VI. Other Applications

In this section we give an overview of several other applications that use the space service, focusing on how they use the hierarchical and coordinate views of the space service.

A. Giving directions

Once we have identified the shortest path between two points and estimated the walking distance and time, we may want to give the user directions on how to reach the destination. We developed an application that provides written directions on how to reach the destination [3], and also overlays the path as a set of red arrows on top of a map, as is described in the next section. In this section we sketch the algorithm that is used for generating directions and highlight the role of the hybrid space service.

The starting point for this application is a path consisting of a list of ALIs as returned by the Walking Path application (Section V). The ALIs contain the coordinates of connection points (doors or hallway intersections) and, if needed, the stairs or elevators that lie along the path from the source to the destination. For each step, the application compares the x,y coordinates of the current step on the path to the next one to determine whether the user should turn left, right, or go straight. The expected orientation of the user is also maintained for each step, with the assumption that the user starts by facing in the direction of the first door on the path and turns only when prompted to by the instructions. Finally, when presenting the directions to the user, the coordinates are translated into human-understandable names for the spaces, e.g. “Turn left in the corridor and walk for 15 yards.”

The hybrid nature of our space service is very important for this application. The symbolic naming hierarchy allows us to make the directions easier to interpret for users. For example, most people would find the following instructions cryptic: “start at (3,56,21) then walk 10 yards to (8,25,21).” However, the symbolic names can easily be translated into human-readable forms, either by using the semantic information (Section IV-C) or (if needed) by parsing the symbolic name. Information about the type of space also enables the application to give more helpful instructions, e.g. “start in Office 8202 and walk 10 yards down the corridor to Office 8209.” The coordinate-based representation is important as well, as it is used to give distance information for each “segment” of the path and to provide directional information, such as whether a user should turn and if so, whether it is a left or right turn.

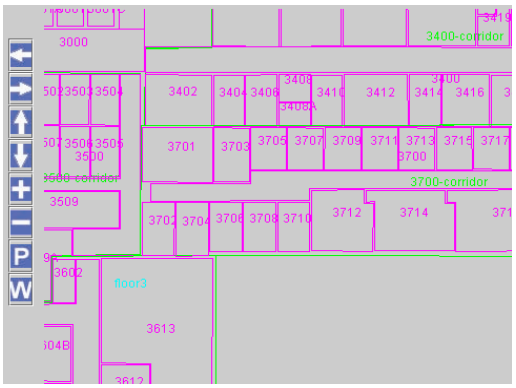

Fig. 7. Map application example

B. Drawing and annotating maps

We also use our space service to draw a visual map of an area and to annotate the map (Figure 7). The Map application traverses the hierarchical space tree and uses the coordinates of the polygon points that define each space to draw it on the map. This application relies on both the hierarchical and coordinate aspects of the hybrid space model. The traversal of the hierarchical space tree is an efficient way of identifying spaces that need to be shown on the map (e.g. a floor or corridor). The hierarchical space view also forms the basis for the “zoom” feature. For instance, if a user asks for a map of a small area, the application will present a very detailed map. When the user zooms out to view the entire floor, the same level of detail would result in a very cluttered image. Instead, the Map application “drops” one or more levels from the bottom of the space hierarchy to reduce the amount of detail, such as only showing major corridors but not individual offices. This is a very effective and simple approach to keep the generated maps readable and useful. The coordinate information in the space service is used to draw the spaces to scale.

Maps can also be annotated to highlight a location with a symbol or identify a path using a sequence of arrows. This part of the applications relies primarily on the coordinate view of the space to precisely position symbols and path annotations. However, it also needs the hierarchical view to translate the coordinates to the highest level of the hierarchy shown on the map.

C. Localization

Localization, or determining the location of a user, is a critical part of many pervasive computing applications. Many projects have explored localization based on triangulation in 802.11 wireless networks. While it is clearly possible to implement a standalone triangulation-based localization system, our WiFi localization system gains significant benefits from using the hybrid space service.

Our localization service is based on the system described in [4], which combines two techniques. The first technique creates a vector of signal strengths of nearby access points and tries to match it to the signal strength vectors of known locations in a database. A good match in the database typically results in an accurate location. If there is no good match, the system uses triangulation based on the signal strength of nearby access points with known locations. This crude estimate of the user's location is then given to the user, who can correct it. The corrected location, together with the measured signal strength vector, is then entered into the database. Over time, the system gradually accumulates accurate localization information for locations that are of interest to the user.

The triangulation component of the localization application uses both the coordinate and hierarchical views of the space service. The locations of the wireless access points are stored using geometric coordinates relative to the building's coordinate system. This allows us to easily calculate a coordinate-based position using triangulation. Next, the system uses the hybrid space service to translate the coordinates into a semantic name that can be used for interactions with the user. Sometimes triangulation can result in "impossible" locations, e.g. on the 7th floor outside of the boundaries of the building. The system detects this using the polygon information stored in the nodes of the hierarchical space tree. When this happens, it moves the result to the nearest location inside the building.

D. Nearest X

The "Nearest X" application was the first application we developed for the hybrid space service. It uses a simple iterative algorithm the find the device (e.g. printer) or location (e.g. restroom) of a certain type that is closest to the user. The algorithm uses the space hierarchy to search spaces that are increasingly further away from the user. For example, it first searches the user's corridor, then the floor, then adjacent floors, etc. At each level, the original application used euclidean distance (based on coordinates) to pick the nearest X. Clearly this is a very rough approximation and using the Walking Path application described in Section V would result in selections that are more convenient for the user.

VII. Discussion

Table I summarizes how the applications use the hybrid space service, focusing on key operations. The hierarchical view of the space service is used to help structure the application (Walking, Map, Nearest X), to identify relevant data (Map, Nearest X), and to obtain symbolic names so space information can be looked up efficiently (Directions, Localization). The coordinate information is used to calculate distances (Walking, Directions, Map, Nearest X), but also for localization (Localization) and to determine directional information (Directions). The last two columns show which applications use the space connectivity and semantic information.

Application	Hierarchy	Coordinates	Conn.	Inf.
Walking	Structure algorithm Scalability,efficiency	Calculate distance	Yes	Yes
Directions	Get symbolic names	Calculate distance Identify direction	Yes	Yes
Map	Scoping of spaces Level of detail	Draw to scale Redraw	Yes	Yes
Localization	Get symbolic names	Triangulation Points in building	No	Yes
Nearest X	Scope search	Calculate distance	No	No

TABLE I
USE OF HYBRID SPACE SERVICE FEATURES BY APPLICATIONS

VIII. Related work

Several projects have addressed issues surrounding the representation of physical space. Space clearly plays a critical role in tourist information systems such as Cyberguide [13], Deep Map [14], GUIDE [6], and the PinPoint Tourist Guide [18]. However, those projects have a greater focus on user interface design and are designed to be standalone systems. In contrast, our space service is designed as a lower level service upon on which a guide (or other pervasive computing applications) can be built.

Several projects have used hierarchical space representations. For example, Microsoft's Semantic Spaces project [5] represents the physical environment as a hierarchy of spaces. It provides support for queries about where people and objects are by corellating locations to physical spaces. However, it lacks a physical coordinate-based representation of spaces and objects, and thus is unable to compute distances or represent locations precisely. In [21], the author presents a location model based on geographic containment in which nodes in the hierarchy can be executable. This offers a very flexible and dynamic design that can be managed in a distributed fashion and link directly to computing devices. The Nexus project [2], [8] introduces an object oriented "augmented world model" that stores information about objects and people as well as spaces. Although the project defines a hierarchical representation of space, the focus of the project is not so much on this representation, but on how multiple heterogeneous "augmented" areas can be linked and on how applications interact with the augmented world.

Early space models that used both symbolic and geometric coordinates were developed by Leonhardt [12] and later formalized and extended by Narayanan [17]. However, the coordinate and hierarchical aspects of space in these projects are not as tightly integrated as in our space service.

Nimbus [19], [20] presents a formal representation of semantic names for spaces as well as a hierarchical structure that allows for operations similar to our subspace/superspace and pinpoint relations. Although Nimbus provides a way to translate spatial coordinates into symbolic names, the two

representations are kept separate. Links between spaces are logical rather than physical, i.e., the model does not contain specific connecting points or ”doors” that provide a way to move from one space to another. Dynamic changes in the environment are also not supported.

The system presented in [9] is similar to our space service in that it represents spaces as a symbolic hierarchy coupled with location information for each space. Furthermore, spaces are associated with ”exits” that connect them to other spaces. This spatial connectivity information is organized as an ”exit hierarchy” that parallels the symbolic hierarchy, with distances between exits on each level of the hierarchy pre-computed and stored. Paths are computed by walking through the exit tree and adding up the distances. However, since the graph that is used to calculate paths is a tree rather than a graph, the system does not account for spaces that are connected in more than one way. Because of this, the algorithm does not always give the correct shortest path. Furthermore, since distances (and, in effect, paths) are pre-computed, the model does not deal with dynamic changes in the environment (e.g. crowds or temporarily unpassable areas), nor is there an opportunity to take into account personal preferences, such as elevators vs. stairs. Also the spaces are stored as a logical tree without information on how they relate to each other in physical space, so there is little support for applications such as Localization, Mapping or Giving Directions.

IX. Conclusion

In this paper we discussed the design and implementation of a hybrid space service that presents pervasive computing applications with an integrated hierarchical and coordinate-based view of physical space. The same space service is used for both inside and outside space, which is important since users move between buildings. Finally, we described how we can efficiently integrate additional information into the hybrid space service, such as how spaces are connected and what they look like to people.

We also leverage our experience with the space service to evaluate the effectiveness of the hybrid space model for a variety of applications. We found that all applications used in a campus guide benefit from both the hierarchical and coordinate views offered by the service, suggesting that the hybrid space model is a very powerful paradigm.

References

[1] AutoCAD developed by Autodesk, Inc., http://www.autodesk.com.

[2] BAUER, M., BECKER, C., AND ROTHERMEL, K. Location models from the perspective of context-aware applications and mobile ad hoc networks. In *Workshop on Location Modeling for Ubiquitous Computing* (2001).

[3] BELUR, K. An Interactive Context-aware Navigation Applications. MS thesis, Information Networking Insititute, Carnegie Mellon University, May 2005.

[4] BHASKER, E., BROWN, S., AND GRISWOLD, W. Employing user feedback for fast, accurate, low-maintenance geolocationing. In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004)* (March 2004), IEEE Computer Society.

[5] BRUMITT, B., AND SHAFER, S. Topological world modeling using semantic spaces. In *Workshop on Location Modeling for Ubiquitous Computing* (2001).

[6] CHEVERST, K., DAVIES, N., MITCHELL, K., FRIDAY, A., AND EFSTRATIOU, C. Developing a context-aware electronic tourist guide: Some issues and experiences. In *Proceedings of CHI'00* (2000), ACM Press.

[7] DIJKSTRA, E. W. A note on two problems in connexion with graphs. In *Numerische Mathematik* (1959), vol. 1, pp. 269–271.

[8] HOHL, F., U.KUBACH, A.LEONHARDI, ROTHERMEL, K., AND SCHWEHM, M. Nexus - an open global infrastructure for spatial-aware applications. In *Proceedings of MobiCom* (Seattle, USA, 1999).

[9] HU, H., AND LEE, D.-L. Semantic location modeling for location navigation in mobile environment. In *Proceedings of the 2004 IEEE International Conference on Mobile Data Management* (2004).

[10] JIANG, C., AND STEENKISTE, P. A hybrid location model with a computable location identifier for ubiquitous computing. In *The Fourth International Conference on Ubiquitous Computing (UBICOMP 2002)* (Goteborg, Sweden, September 2002), vol. Lecture Notes in Computer Science 2498, Springer, pp. 246–263.

[11] JUDD, G., AND STEENKISTE, P. Providing Contextual Information to Pervasive Computing Applications. In *IEEE International Conference on Pervasive Computing (PERCOM)* (March 2003), IEEE.

[12] LEONHARDT, U. Supporting location-awareness in open distributed systems, 1998. PhD Thesis, Department of Computing, Imperial College London.

[13] LONG, S., KOOPER, R., ABOWD, G. D., AND ATKESON, C. Rapid prototyping of mobile context-aware applications: The cyberguide case study. In *Proceedings of the Second Annual International Conference on Mobile Computing and Networking* (1996), ACM Press, pp. 97–107.

[14] MALAKA, R., AND ZIPF, A. Deep map - challenging it research in the framework of a tourist information system. In *Information and Communication Technologies in Tourism* (2000), Springer-Verlag, pp. 15–27.

[15] Mapquest Home Page, http://www.mapquest.com.

[16] Microsoft Streets & Trips, http://www.microsoft.com/streets/.

[17] NARAYANAN, A. K. Realms and states: a framework for loaction aware mobile computing. In *Proceedings of the First International Workshop on Mobile Commerce* (2001).

[18] ROTH, J. Context-aware web applications using the pinpoint infrastructure. In *IADIS International Conference WWW/Internet* (Lisbon, Portugal, Nov 2002), IADIS Press, pp. 3–10.

[19] ROTH, J. Accessing location data in mobile environments – the nimbus location model. In *Mobile HCI 03 Workshop on Mobile and Ubiquitous Information Access* (Sep 2003), Springer-Verlag, pp. 256–270.

[20] ROTH, J. Flexible positioning for location-based services. In *IADIS Journal on WWW/Internet* (Dec 2003), vol. I(2), IADIS Press, pp. 18–32.

[21] SATOH, I. A location model for pervasive computing environments. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)* (March 2005), IEEE Computer Society.

[22] W3C. OWL Web Ontology Language Reference. Working Draft, 31 March 2003, http://www.w3.org/TR/owl-ref/, March 2003.