A Location-Dependent Recommender System for the Web

Mauro Brunato and Roberto Battiti
DIT, Università di Trento — via Sommarive 14, I-38050 Povo (TN) — ITALY
`brunato|battiti@dit.unitn.it`

***Abstract*— User's location in a wireless mobile environment is an important information item. Knowledge of the position can be added to the existing user profile in order to provide efficient services.**

We propose a web-based recommender system that automatically adapts relevance parameters of the recommendable items (in our case web links) through implicit collaboration with users, taking into account their position both during parameter estimation and during the list generation phase. After an initial tuning phase, a specific URL will be recommended to a user in a given location in a way that considers where and how often it was accessed by the previous users. A new middleware layer, the *location broker*, associates past user positions and selected links. These data are used to develop a spatial usage pattern for each site.

We describe a data structure that permits scalability and analyze the empirical computational complexity both on a simulated scenario and in a real-world context in our province[1].

I. Introduction

Mobile and wireless applications need to be developed in a user-centric way. Reduction of the cognitive burden required to make the system provide the appropriate information is mandatory, because the average user is not willing to spend time in tuning the system, selecting information from a small output device and entering large amounts of data through slow input interfaces such as a thumb keyboard or a graffiti stylus. Adaptive context-aware systems have to develop models of the relevance of different resources (e.g. URLs) for a specific user, and to filter the information so that only a small selection of relevant and limited resources is presented.

Of course, location is a critical component of a mobile user's context. A recommender system, taking into account location among other data, and providing a few links that are considered most interesting to the user in a particular context, will provide an agile and flexible environment for a mobile user. This system can be an ideal complement to methods providing dynamic customization of content for wireless clients [1], once the relevant links are selected. Location-aware mobile commerce systems are considered for example in [2], location-aware shopping assistance is described in [3].

Mapping physical locations to Internet URIs was proposed, among others, in the HP's *CoolTown* project[2], where relevant locations are equipped with short-range infrared emitters that periodically broadcast their related URI to listening mobile devices. The context of pervasive computing in a wireless Internet framework is also explored by our WILMA Project[3] (Wireless Internet and Location Management) at the University of Trento, where the `PILGRIM` location-broker and mobility-aware recommender system is being developed.

This paper deals with scalability issues to make the system available for a global use in the Internet, where the number of resources in the system database may easily grow to reach millions of items.

The rest of the paper is organized as follows. In Section II a brief survey of recommender systems is reported, then a model for describing the preferences of mobile users is presented, together with an overview of the architecture of the `PILGRIM` (Personal Item Locator and General Recommendation Index Manager) system. In Section III the ER-tree, a spatial indexing structure tailored for the requirements of the `PILGRIM` system, is introduced and motivated. Section IV is devoted to experimental evaluation of the ER-tree structure in simulated random environments. Finally, in Section V conclusions are drawn.

II. A location-aware model of user preferences for web sites

A typical recommender system [4] answers the question: "What are the k more interesting items for the current user?"

For this purpose, a recommender system maintains a finite list of *users*, identified by unique IDs. Each user is associated to some *profile* information. A list of *items*, for instance web links, is also maintained along with relevant properties. The term *current user* will identify the user whom the recommendation list is being built for.

Techniques of *collaborative filtering* can be introduced where user profiles and evaluations are stored and used to automatically build a list of links specifically tailored for a particular user. Many recommender systems, such as Tapestry [5] or Fab [6], require users to express their evaluation of the visited item, while others can gather implicit information. For example, the GroupLens [7] USENET news recommender system uses reading times as a user interest measure. PHOAKS [8] uses data mining techniques to extract URLs or other information pointers from USENET postings or from bookmark collections.

[1]This research is partially supported by the Province of Trento (Italy) in the framework of the WILMA (Wireless Internet and Location Management) project (`http://www.wilmaproject.org/`).

[2]`http://cooltown.hp.com/`

[3]`http://www.wilmaproject.org/`

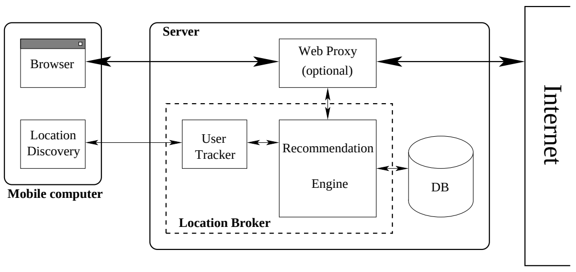

Fig. 1. Architecture of the PILGRIM system.

The PILGRIM recommender system [9] generates recommendation lists based on the user location and on web site information gathered from previous usages of the same sites by other people. For completeness we briefly summarize the PILGRIM approach in the current section.

A location-aware recommender system should be able to produce a top-k items list for a given user whose location is known with a precision ranging from a few meters to some hundreds of meters. Position estimates can be obtained by means of many systems, such as GPS (outdoor only, with a precision of about 10m), active badges [10] (precisions ranging from few centimeters to room size), or by exploiting the radio propagation properties of the wireless networking medium [11], [12] (with precisions of few meters in the Wi-Fi case). The latter solution is of particular interest because it does not need additional infrastructure, and the normal networking equipment is used both for communication and for location detection.

A mobile user is likely to handle the PDA only for the time that is strictly needed to find an interesting link and follow it and may not be willing to fumble with the device in order to give an explicit evaluation of the chosen item. So, only implicit information about the choice can be gathered, e.g., whether the recommended item was clicked or not, how long it remained on screen, what was the subsequent action of the user (she abandoned the site, or she visited also linked pages). In a mobile environment, however, another crucial piece of information is the user's *position* when she clicked the link.

The purpose of the PILGRIM system is to integrate information about the current user location into traditional recommender systems in an adaptive way.

A. Architecture of the system

The PILGRIM system is structured as an automated learning component to develop models relating resources to their spatial usage pattern by mining the historic database that records past accesses to sites.

The basic building blocks of the system are shown in Figure 1. On the client side, possibly a PDA with low computing speed, two components are active. The first, the normal off-the-shelf Internet browser, is the only component appearing on the screen during normal operation. The second component, the *location discovery* application, is a small process that enables the PDA to obtain positioning data and to send them to the server; for instance, radio signal strength from surrounding Wi-Fi access points or raw GPS data. This module is mostly transparent to the user; it will only display a startup dialog for initialization purposes, for example to change privacy settings. The two components are independent: the system could take advantage from an integrated solution, but this may not be applicable to all systems. For instance, many lightweight browsers in use on PDAs do not allow component technologies such as Java or ActiveX, and even scripting languages may not be supported.

The location discovery application running on the client sends position updates to the server-side *location broker*. Its first component, the *user tracker*, uses the location data transmitted by the client to compute an estimate of the user position (this may be done by the PDA itself, if it has enough spare CPU). The second component, the *recommendation engine*, maintains the access database, containing data about what links have been followed, and from what physical position. These data, together with the user's location provided by the user tracker module, are employed to generate a list of possibly interesting links.

B. Collaborative filtering and ranking procedure

Once the database is populated with past user accesses to items, its data can be used to build a model of user preference. Thus, the chosen approach considerably differs from other systems such as Websigns, where the database is updated and maintained by hand, and is more similar to the collaborative filtering paradigm, where the quality of recommendations shapes up as long as users interact with the system.

The models relating resources (URLs) and usage patterns in physical space are expressed in terms of a metric based on *inertial ellipsoids*. The basic motivation is that of obtaining a smooth metric, where the spatial distribution of interest for a specific URL may have a preferred orientation in space.

The recommendation engine works on a set of s links, each identified by a unique id $l = 1, \dots, s$. Suppose that site l has been visited N_l times (possibly by different users), and let the set of points $P_i^l = (x_i^l, y_i^l)$, $1 \le i \le N_l$, represent the N_l physical locations where link l was clicked. A locality measure of link l can be obtained by calculating the *inertial ellipsoid* of its points. Points can be associated to a "mass" that is related to the level of trust of the received feedback or to the length of time that a user spent on a web page. In the current version, for simplicity, all points are modeled as unit masses. The inertial ellipsoid has the following quadratic equation:

$$\begin{pmatrix} x - \bar{x}_l & y - \bar{y}_l \end{pmatrix} M_l^{-1} \begin{pmatrix} x - \bar{x}_l \\ y - \bar{y}_l \end{pmatrix} = 1,$$

where $\bar{x}_l$ and $\bar{y}_l$ are the coordinates of the center of mass, while matrix M_l is the second-order moment matrix (the covariance matrix):

$$\bar{x}_l = \frac{1}{N_l} \sum_{i=1}^{N_l} x_i^l, \qquad \bar{y}_l = \frac{1}{N_l} \sum_{i=1}^{N_l} y_i^l,$$

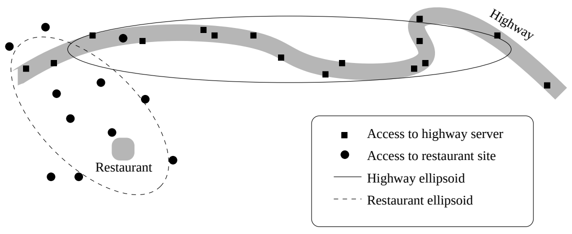

Fig. 2. Two sample sites with different access metrics.

$$M_l = \frac{1}{N_l}\begin{pmatrix} \sum_{i=1}^{N_l}(x_i^l - \bar{x}_l)^2 & \sum_{i=1}^{N_l}(x_i^l - \bar{x}_l)(y_i^l - \bar{y}_l) \\ \sum_{i=1}^{N_l}(x_i^l - \bar{x}_l)(y_i^l - \bar{y}_l) & \sum_{i=1}^{N_l}(y_i^l - \bar{y}_l)^2 \end{pmatrix}.$$

Because the matrix is positive definite, the matrix M_l^{-1} defines a distance between points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$:

$$d_l(P,Q) = \begin{pmatrix} x_P - x_Q & y_P - y_Q \end{pmatrix} M_l^{-1} \begin{pmatrix} x_P - x_Q \\ y_P - y_Q \end{pmatrix}.$$

Let $\bar{P}_l = (\bar{x}_l, \bar{y}_l)$ be the center of mass for site l. The distance d_l can be used as a measure of interest of site l for a user located at position $P = (x, y)$. The *preference* for a site l at point P is defined as:

$$r_l(P) = \frac{1}{d_l(P, \bar{P}_l)},$$

so that site l is preferable to site l' at point P if $r_l(P) > r_{l'}(P)$ (preference is $r_l(P) = +\infty$ on the center of mass).

The set of preference functions $(r_l)_{1\leq l\leq s}$ induces at every point P a permutation $\pi^P = (\pi_1^P, \dots, \pi_s^P)$ of the site IDs having the property

$$\forall i \in \{1, \dots, s\} \qquad r_{\pi_i^P}(P) \geq r_{\pi_{i+1}^P}(P).$$

The permutation is uniquely defined modulo equalities of the preference function; in this case, any tie-breaking rule, such as ID order, properly defines a unique permutation:

$$\forall i \in \{1, \dots, s\} \qquad r_{\pi_i^P}(P) = r_{\pi_{i+1}^P}(P) \Rightarrow \pi_i^P < \pi_{i+1}^P.$$

The advantages of the ellipsoid metric with respect to simpler techniques can be understood by referring to Figure 2. Consider two candidate links. The first contains information about the status of a highway, and is mostly used by people driving along that road. Almost all accesses to the site have been performed along the highway. Because of the uni-dimensionality of the road, there is a strong correlation between the x and y coordinates of the points (the small black squares in the figure), and the resulting ellipse, with the solid outline, has high eccentricity. Its preference function, $r_{\text{highway}}(P)$ decreases slowly when moving from the average access position along the highway, while it drops very rapidly when moving outside the road. On the other hand, a restaurant placed near the highway, but not directly accessible, has a less eccentric region of interest (the small black circles). The resulting ellipse, with a dashed outline, is less eccentric, even though it still shows a preferential direction, due to the physical visibility of the building, or to the terrain morphology. The preference function, $r_{\text{restaurant}}(P)$, decays more regularly with distance from the center. Note that the center of the ellipse does not coincide with the restaurant. In fact, no *a priori* information is built in the system, and the geographical relevance of a link is gradually inferred through the ellipsoid metric: every time a user clicks a link, the recommendation engine updates the database; inertial ellipsoids are periodically updated on the basis of the database records.

III. The ER-tree: a scalable implementation

The PILGRIM recommender system relies on a site database recording the site name, URL, access statistics and preference ellipsoid. To efficiently satisfy queries about site location, some spatial indexes must be implemented. The number of sites is bound to grow rapidly as the system develops and learns, and the basic algorithm based on a sequential scan of all items in order to find the nearest neighbors rapidly becomes impractical. Smarter data structures, specifically tailored for speeding up spatial queries need to be implemented.

A wide range of spatial data structures has been proposed in the literature [13]. In this paper a new data structure, the ER-tree (*Elliptic R-tree*), is introduced to handle spatial queries when objects have different metrics.

The proposed structure is based on the R-tree [14]. The two-dimensional R-tree is a hierarchy of nested rectangles. At each node, the corresponding rectangle is the bounding box of all enclosed objects. Leaves are the objects (or pointers to the objects) in the database. Every node is required to have more than m and less than M sons ($M/2 \leq m \leq M$); only the root is allowed to have as few as two children. Algorithms that build the R-tree structure try to keep the tree in a good condition for queries by minimizing interval overlapping, which is in principle unavoidable. Nearest-neighbor queries can take advantage from branch-and-bound techniques. In fact, distance from a bounding box is a lower bound for distances from all enclosed objects. The leaves of the ER-tree are the centroids of the ellipses, having pointwise extension.

Euclidean distance of a point from a bounding box is not a lower bound for the actual distances from the enclosed objects. In fact, in the aforementioned site model every site has a different elliptical metric. To make the lower bound work, every time a bounding box is calculated from the enclosed objects, a corresponding elliptical metric must be applied. The overall metric for a bounding box is calculated by translating all ellipsoids to to the origin, and by building a "bounding ellipse" such that its major axis corresponds with the largest major axis of the ellipses, while the minor axis is calibrated in order to contain all ellipses. Since every ellipse represents the unit-distance locus for the corresponding metric, it is straightforward that the overall metric is a lower bound for all enclosed metrics. Every node in the tree contains, as additional

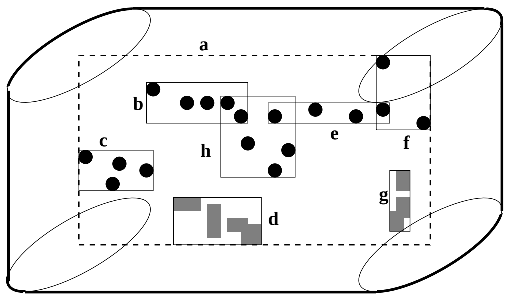

Fig. 3. An ER-tree node: bounding boxes **b** to **h** are directly contained in box **a**, and are therefore its children.

TABLE I

COMPARISON BETWEEN LINEAR ARRAY AND ER-TREE SEARCH FOR A 100000-SITE SAMPLE

	R-tree	Array	Ratio	% Differ.
Time (ms)	5.00	42.00	0.119	88.1%
Distance evals	9672.8	100000	0.097	90.3%
Memory (bytes)	22073344	21600000	1.022	2.2%

information, the overall metric corresponding to its bounding box.

Figure 3 shows part of an ER-tree by its actual spatial structure (its actual transposition to a tree structure should be evident). Node **a** is shown with its corresponding bounding box (the dashed rectangle) and its subnodes (**b** to **h**). All subnodes, with the exception of **d** and **g**, contain objects, i.e. leaves of the ER-tree (black bullets); nodes **d** and **g** contain lower-level nodes, which will be further decomposed. The thick line surrounding the bounding box of **a** is the locus of points having unit distance from the rectangle, and is determined by the overall metric of the node.

IV. EXPERIMENTAL EVALUATION

Performance of algorithms and data structures has been evaluated with respect to k-nearest-neighbor queries. Some tests have been performed on randomly-generated sites. On a square one-kilometer area a certain number N (100 to 100000) of sites are uniformly spread and their access patterns (their area of interest) are modeled by ellipses whose major axes follow a Gaussian distribution with given average μ and standard deviation σ. The inclination of the major axis and the eccentricity are uniformly distributed, the first in the interval $[-\pi/2, \pi/2)$, the second in $[e_{\min}, 1]$, where $e_{\min}$ is the smallest admitted eccentricity. We used a distribution of $N = 100$ sites with $e_{\min} = 0.1$ and $\mu = \sigma = 100$m.

The ER-tree depends on one parameter M, the node maximum degree or "node size", while the corresponding minimum for non-root nodes has been set to $m = \lfloor M/2 \rfloor$. The nearest neighbors search algorithm depends on the number k of neighbors to be returned.

Figure 4 shows the performance comparison between a

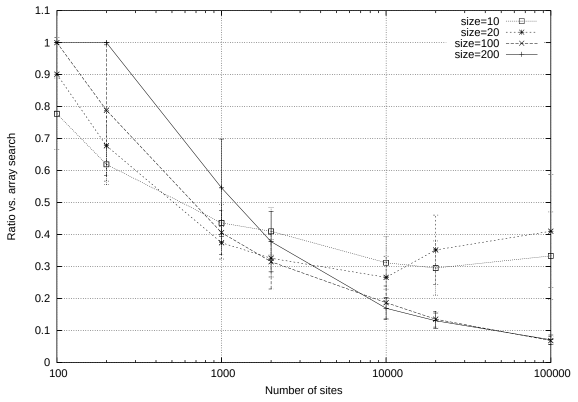

Fig. 4. Comparison between linear array and ER-tree search for different item populations and different node sizes.

linear search in an array of sites and the branch-and-bound search on the ER-tree for different node sizes. Along the x axis the number of sites is reported; the y axis represents the average performance improvement obtained by the ER-tree search. Performance is tested by counting the number of distance evaluations. In fact, from preliminary experiments such as that shown in Table I we see that counting the number of evaluations gives a good measure of the actual time improvement, while time itself, when tested on a time-shared machine, can suffer from many external influences. Parameters of the test are $k = 10$, $\mu = 100$m, $e_{\min} = .1$, $\sigma = 100$m. The different lines in the graph show the behavior for different values of M, from 10 to 200. Every point in the graph shows the average of 100 tests, and error bars indicate the 95% confidence interval.

Note that for small node sizes the advantage over sequential search is almost negligible (if $N \leq M$ all nodes must be visited anyway). Small node sizes become unpractical for a large number of nodes, probably because the depth of the tree grows considerably, and a long downwards exploration must be undertaken before arriving to the leaves. Search speedups by more than 90% are obtained as the number of items grows to $N = 100000$.

A second set of experiments has been performed in order to evaluate the performance of the system in a more realistic context. The population structure of the Trentino region has been used in order to generate a more structured pattern of web sites. Trentino is a mountain region with most people living within short distance from the bottom of the valleys and concentrated near the main towns. The system is likely to be used also by people moving along roads, or by tourists spread along the whole territory. A simulated usage test with $N = 100$ web sites resulted in the pattern shown in Figure 5. Most ellipsoids are concentrated in towns, but a few are of wide interest: they may correspond to a famous resource or to a mountain visible from a large distance.

Figure 6 shows some preliminary comparisons between linear scans and ER-Trees for the simulated environment. Every point represents the average of 100 ten-nearest-neighbor

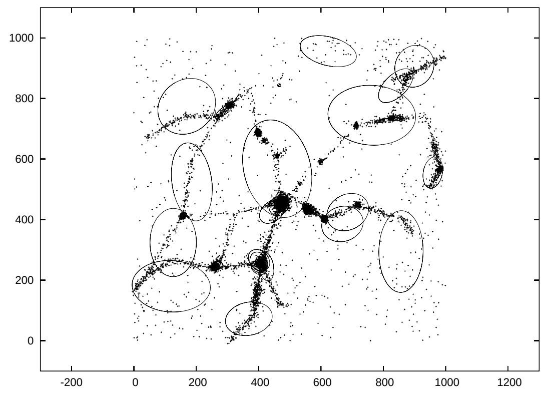

Fig. 5. Ellipsoids of interest for localized web sites in a realistic setting, the Trentino region. Small dots represent distribution of single users; while many are concentrated in towns and valleys, some are spread throughout the territory.

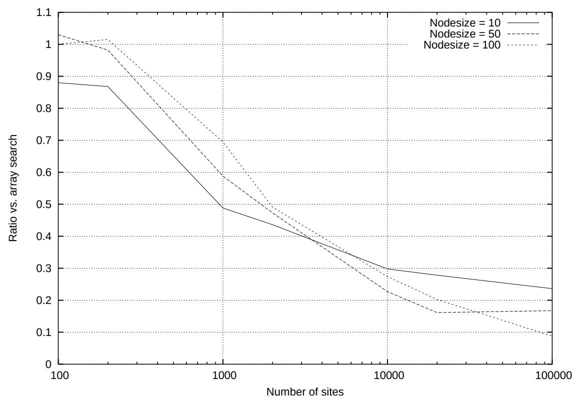

Fig. 6. Performance ratio of the R-tree data structure versus linear scan for different node sizes and different numbers of indexed sites.

searches, and the error bars show the 95% confidence interval for the mean. Note that the improvement with respect to linear scan is comparable to that obtained on the random distribution in Figure 4.

V. Discussion and conclusions

We proposed a location-aware recommender system that is integrated with a basic browser and that filters resources (URLs) for a specific user. The filter is adaptive and takes into account both the *current user location* and models relating resources to geographic locations built by mining the previous usage history. The main advantage with respect to existing systems lies in the automated creation of models (without an explicit design to couple locations and URLs), the flexibility and the independence from ad-hoc systems implemented by resource owners.

To allow the degree of scalability demanded by a large-scale implementation, a hierarchical data structure has been proposed and analyzed. The results on a model that considers the population and building distribution of our region (Trentino) show that performance of the ER-tree structure does not degrade when implemented on a more structured pattern of usage. Techniques to select the appropriate privacy level can also be easily introduced in the system.

The system is currently under development as a set of independent modules (shown in Figure 1) interacting via TCP/IP using XML/SOAP queries. User interaction is managed by CGI modules invoked by the usual HTTP mechanism. The final goal is to develop a distributed implementation where local databases contain information about items close to the server location, and a peer-to-peer content distribution scheme enables synchronization among all local servers. To this purpose, a distributed version of the ER-Tree structure and a synchronization protocol are being studied.

A future issue on the project agenda is the integration of the location broker with traditional recommender systems. This will permit specialized implementations (like for example a system dedicated to gourmet restaurants, to tourists interested in art and monuments, to mobile shopping, etc.).

References

[1] J. Steinberg and J. Pasquale, "A web middleware architecture for dynamic customization of content for wireless clients," in *Proceedings of Eleventh International World Wide Web Conference - WWW2002*, Honolulu, Hawaii, uSA, May 2002.

[2] S. Duri, A. Cole, J. Munson, and J. Christensen, "An approach to providing a seamless end-user experience for location-aware applications," in *Proceedings of the fi rst international workshop on Mobile commerce*, Rome, July 2001, pp. 20–25.

[3] T. Bohnenberger, A. Jameson, A. Krüger, and A. Butz, "User acceptance of a decision-theoretic, location-aware shopping guide," in *IUI 2002: International Conference on Intelligent User Interfaces*, Y. Gil and D. B. Leake, Eds. New York: ACM, 2002, pp. 178–179. [Online]. Available: http://dfki.de/˜jameson/abs/BohnenbergerJK+02.html

[4] P. Resnik and H. R. Varian, "Recommender systems," *Communications of the ACM, special issue*, vol. 40, no. 3, pp. 56–58, Mar. 1997. [Online]. Available: http://www.acm.org/

[5] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative fi ltering to weave an information tapestry," *Communications of the ACM*, Dec. 1992.

[6] M. Balabanovič and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, Mar. 1997. [Online]. Available: http://www.acm.org/

[7] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "GroupLens: Applying collaborative fi ltering to USENET news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, Mar. 1997. [Online]. Available: http://www.acm.org/

[8] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter, "PHOAKS: a system for sharing recommendations," *Communications of the ACM*, vol. 40, no. 3, pp. 59–62, Mar. 1997. [Online]. Available: http://www.acm.org/

[9] M. Brunato and R. Battiti, "PILGRIM: A location broker and mobility-aware recommendation system," in *Proceedings of IEEE PerCom2003*, Fort Worth, TX (USA), Mar. 2003.

[10] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transaction on Information Systems*, vol. 10, no. 1, pp. 91–102, Jan. 1992.

[11] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proceedings of IEEE INFOCOM 2000*, Mar. 2000, pp. 775–784.

[12] R. Battiti, M. Brunato, and A. Villani, "Statistical learning theory for location fi ngerprinting in wireless LANs," Dipartimento di Informatica e Telecomunicazioni, Università di Trento, Tech. Rep. DIT-02-0086, Oct. 2002. [Online]. Available: http://eprints.biblio.unitn.it/archive/00000238/

[13] V. Gaede and O. Günther, "Multidimensional access methods," Institut für Wirtschaftsinformatik, Humboldt-Universität zu Berlin, Tech. Rep., 1996. [Online]. Available: http://citeseer.nj.nec.com/gaede97multidimensional.html

[14] A. Guttman, "R-tree: A dynamic index structure for spatial searching," in *Proceedings of ACM SIGMOD*, 1984.