Geo Referenced Dynamic Bayesian Networks for User Positioning on Mobile Systems

Boris Brandherm and Tim Schwartz

Department of Computer Science, Saarland University,
Stuhlsatzenhausweg, Bau 36.1, D-66123 Saarbrücken, Germany
{brandherm, schwartz}@cs.uni-sb.de

Abstract. The knowledge of the position of a user is valuable for a broad range of applications in the field of pervasive computing. Different techniques have been developed to cope with the problem of uncertainty, noisy sensors, and sensor fusion.

In this paper we present a method, which is efficient in time- and space-complexity, and that provides a high scalability for in- and outdoor-positioning. The so-called geo referenced dynamic Bayesian networks enable the calculation of a user's position on his own small hand-held device (e.g., Pocket PC) without a connection to an external server. Thus, privacy issues are considered and completely in the hand of the user.

1 Introduction

In order to compute the position of a user, some kind of sensory data has to be evaluated. Car navigation or out-door localization systems typically use GPS (Global Positioning System) to determine the coordinates of a user. Unfortunately, GPS does not work properly in in-door environments because of the fundamental physics of GPS satellite signals, so other technologies have to be used. Examples are infrared beacons, radio frequency emitters, laser range scanners or video cameras. These senders (and their respective sensors) differ in various ways, like sending-characteristics, sending-range, accuracy, price, form factor, and power consumption. Of course there is the usual trade-off between the obviously technical important factors (accuracy, power consumption) versus the limiting factors (price, form factor, sending characteristics), so that the decision, which sender to use, depends heavily on the environment and the planned application as well as on financial conditions. It may even be the case that an already existing installation of positioning senders and/or sensors should be improved by installing additional senders with better accuracy. A location system should therefore not only be able to process different kinds of sensory data but should also be able to fuse the data together in order to calculate the best possible position information.

Another problem is the noisy nature of sensors; therefore, a probabilistic approach for calculating user positions is highly preferable. Commonly Bayesian filters [1] are used to estimate the true location of the user out of the received sensor signals. Several techniques are known to implement Bayes filters, which differ from each other in complexity and certain restrictions (see [2] for a detailed description of different Bayes filter techniques). Particle filters are such an implementation and they are widely used in localization systems. Although overcoming most restrictions of other filters, they still have to be trained or calibrated for their environment.

We were trying to find a new way how to fuse sensory data and how to cope with the problem of inaccurate data. This new approach should be low in calculation and memory costs, so it can be implemented and executed on a mobile system (e.g. Hewlett-Packard iPAQ). We came up with the idea of *geo referenced dynamic Bayesian networks* and we want to share and discuss this idea with the research community.

The paper is organized as follows: Section 2 describes some technical issues and design decisions of our localization system. Section 3 explains the idea of geo referenced Bayesian networks, gives a detailed description of the algorithm and illustrates it with an example. Our working implementation of the system is introduced as a proof of concept in Sect. 4. Section 5 resumes our research and gives an outlook to our future work.

2 Technical Issues

Although our localization system is not the main issue of this paper we will first explain a few technical details of it because we think that this may be helpful for understanding the concept of geo referenced dynamic Bayesian networks.

2.1 Exocentric Localization

In localization systems like the Active Badge, the Active Bat or Ubisense (see [3, 4, 5]) the users wear some kind of sender (infrared, ultrasound, ultrawideband radio) and the respective sensors are installed in the environment. We call these systems exocentric localization systems, because the sender actively reveals the user's position to its surroundings (like calling out "I'm here, I'm here"). A user of this system does not know what the collected localization data is used for or who has access to this data. Of course, functions can be implemented that give the user a choice on who will be allowed to access his position data, but this means that the user has to trust the system. If he does not, his only alternative is not to wear the sender (or to switch it off). By doing this, the user will not only prohibit others to gain information about his current position, but he will also be unable to use the system for his own needs (e.g. for navigational purposes).

2.2 Egocentric Localization

An egocentric localization calculates the user's position on a mobile, personal device of the user. This can be accomplished by placing the *senders* in the environment (now, the senders call out "You're near me, you're near me") and by equipping the user's mobile device with the respective sensors. With this technique the position calculation and the location information is literally in the hands of the user and he can decide if he wants to share this information through a WiFi-connection with other people or applications. If he does not trust the systems privacy control, he can switch off his WiFi-Connection and will still be able to determine his own position for navigational purposes.

We favor this approach and thus we designed our example system in that way, using infrared beacons and active RFID tags as senders (that are installed in the environment and *not* worn by the users).

2.3 Infrared Beacons

We use infrared beacons that are manufactured by eyeled GmbH[1]. These beacons are powered by batteries and send out a 16-bit wide identification code that can individually be adjusted for each beacon. The emitted infrared beam has a range of about 6 meters and has, due to the physical attributes of light, a conical sending characteristic. The price for such a beacon is about 80 Euro. The required infrared sensor is often already integrated in a PDA for data exchanging purposes.

2.4 Active RFID Tags

Radio Frequency IDentification (RFID) tags are available as passive and active parts. In both forms, the tags store some identification code that can be read out with a special RFID reader that sends out a radio signal. The passive tags get their power out of the reader's radio signal and therefore have a very low range (usually up to 15 cm). Active RFID tags have their own power supply through a battery. We use active RFID tag from Identec Solutions AG[2], which have a range of up to 10 meters. Due to the physical attributes of radio waves, the sending characteristic is radial. One active tag costs about 20 Euro. The reading devices for active RFID tags come in various form factors. In conjunction with the PDA, we use a PCMCIA reader card, which costs about 1500 Euro. (The high costs mainly arise from the fact that these readers are manufactured in very low quantities.)

In [6] the authors describe a localization system that uses active RFID tags. Their system fits more into the exocentric localization approach (see 2.1), because they install the readers in the environment and equip the user with a tag. The authors also describe an experiment, where they find out that receiving active RFID tags highly depends on different factors, like static obstructions and human movement. Therefore, they also install reference tags in the environment to improve their system's accuracy.

A system that fits in the category of egocentric localization is described in [7]. Their approach is similar to ours, because they install the tags in the environment

[1] http://www.eyeled.de
[2] http://www.identecsolutions.com

(on the floor, we place our tags at the ceiling) and let the user wear one or more readers. They derive the user's position at time t by calculating the sum of the known positions of the received RFID tags and the previous known location of the user, followed by a division through the number n of received tags plus one (for the previous location):

$$\mathrm{UserPos}(t) = \frac{1}{n+1}\left(\mathrm{UserPos}(t-1) + \sum_{i=1}^{n} \mathrm{Coord}\left(\mathrm{receivedTag}[i]\right)\right) \tag{1}$$

(The equation is simplified and adapted to our notation, for the original version see [7].) Our experience with active RFID tags shows that the reader often detects tags that are far away from the user. Therefore, we try to cancel out these false readings with the use of dynamic Bayesian networks.

3 Geo Referenced Dynamic Bayesian Networks

In the following, we describe the idea of geo referenced dynamic Bayesian networks and how they can be used for sensor fusion and to cancel out false readings.

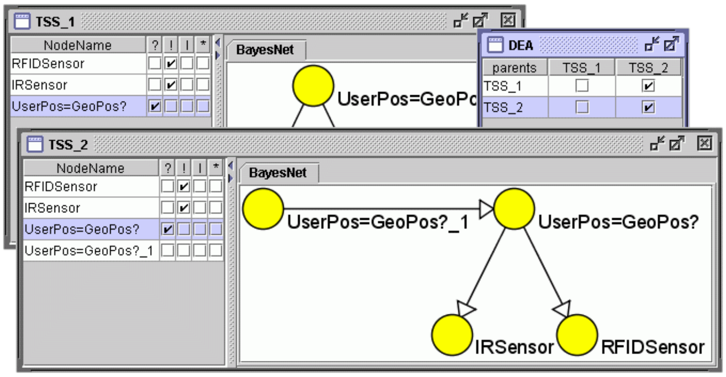
3.1 Idea

Bayesian Networks (BNs) are a computational framework for the representation and the inference of uncertain knowledge. A BN can be represented graphically as a directed acyclic graph (see Fig. 1). The nodes of the graph represent probability variables. The edges joining the nodes represent the dependencies among them. For each node, a conditional probability table (CPT) quantifies these dependencies.

Dynamic Bayesian networks (DBNs) are an extension of Bayesian networks. With a DBN, it is possible to model dynamic processes: Each time the DBN receives new evidence a new time slice is added to the existing DBN. In principle, DBNs can be evaluated with the same inference procedures as normal BNs; but their dynamic nature places heavy demands on computation time and memory. Therefore, it is necessary to apply roll-up procedures that cut off old time slices without eliminating their influence on the newer time slices.

Our idea is to let such a DBN represent the characteristics of the used senders (in our case IR beacons and RFID tags). Note that we do not use the DBN to represent all the senders that are actually installed in the environment. We use one small DBN that prototypically describes the reliability of the sender types (assuming that all senders of a certain type have the same reliability). This prototypical DBN gets instantiated several times during the runtime of the system and each instantiation gets assigned to a geo coordinate *GeoPos*.

Figure 1 shows the network that we use in our example application. The top right node (labeled UserPos=GeoPos?) represents the probability that the user is standing at the assigned geo coordinate *GeoPos*. The node to the left of it (UserPos=GeoPos?_1) is the probability that was calculated in the previous time



TSS_1

NodeName	?	!	I	*
RFIDSensor	☐	✓	☐	☐
IRSensor	☐	✓	☐	☐
UserPos=GeoPos?	✓	☐	☐	☐

DEA

parents	TSS_1	TSS_2
TSS_1	☐	✓
TSS_2	☐	✓

TSS_2

NodeName	?	!	I	*
RFIDSensor	☐	✓	☐	☐
IRSensor	☐	✓	☐	☐
UserPos=GeoPos?	✓	☐	☐	☐
UserPos=GeoPos?_1	☐	☐	☐	☐

Fig. 1. Screenshot of JavaDBN with directed graph (DEA) and time slice schemata (TSS_1 and TSS_2)

slice. The two bottom nodes (IRSensor and RFIDSensor) represent the probability that an IR beacon and/or a RFID tag installed at *GeoPos* can be detected under the condition that the user is standing at *GeoPos*.

Receiving an infrared signal gives very high evidence that the user is standing near the respective beacon (the infrared sensory data is nearly noise free). Receiving a RFID tag gives smaller evidence that the user is standing near the tag (the reader data is very noisy). These characteristics are coded in the CPTs of the IRSensor- and RFIDSensor-nodes. The actual geoDBN that we use in our example application is described in more detail in Section 3.3.

These networks with their assigned coordinates are the geo referenced dynamic Bayesian networks (geoDBNs). Each geoDBN represents the believe that the user is standing at the associated coordinate.

The active RFID tags have a small internal memory that can be freely used to read and write data. We use this memory to store the coordinate of the tag. IR beacons are always combined with one RFID tag that provides the coordinates of the nearby beacon and the tag itself. When a tag or beacon is sensed by the PDA, geoDBNs are instantiated and associated with the induced coordinates. These induced coordinates depend on the stored coordinate and on the sender type.

The calculation of the user position is somewhat similar to (1) but we weight the coordinates with the calculated probabilities of the existing geoDBNs:

$$\mathrm{UserPos}(t) = \sum_{i=1}^{n} \alpha\ w(\mathrm{GeoDBN}[i])\ \mathrm{GeoPos}(\mathrm{GeoDBN}[i]). \tag{2}$$

Here n is the number of existing geoDBNs at time t ($n \geq \#\mathit{ReceivedSenders}_t$), GeoPos(GeoDBN[$i$]) is the coordinate and w(GeoDBN[i]) the weight of the ith geoDBN. α is a normalization factor that ensures that the sum of all weights multiplied with α is one.

To reduce calculation cost and memory usage the number of instantiated geoDBNs must be as low as possible. To achieve this goal, geoDBNs with a weight lower than $threshold_{use}$ are marked as unused (these geoDBNs provide only little evidence that the user is in the vicinity of their geo coordinate). This threshold should match the a priori probability for the geoDBN at its first instantiation. To cope with resource restrictions a maximum number of possible geoDBNs can be specified. If this number is exceeded those geoDBNs that provide the least estimation will be deleted. To keep the overhead for memory management low (or to prevent garbage collection if the system is implemented in languages like Java or C#) GeoDBNs that are marked as unused can be "recycled" by resetting them to initial values and new coordinates.

The following section describes the algorithm and explains it with an example.

3.2 Estimation of the User Position

A new estimation of $\mathcal{U}$'s current position is calculated after each new measurement and is based on the current sensory data as well as on the previous data. A weighted combination of the old and new data is achieved through inference in the respective geoDBN. The schematic approach looks like this:

1. Perform a new measurement and store the received coordinates
2. Extend every existing geoDBN with a new time slice and cut off the old time slice.
3. Insert the new evidences of the sensors:
   (a) If there is not already a geoDBN at a received coordinate, create a new geoDBN and insert the evidence.
   (b) If there is a geoDBN at a received coordinate, insert the evidence in the current time slice.
4. Go through all geoDBNs and calculate the estimation that the user is at the associated coordinate.
5. Sort the geoDBNs in descending order of their belief.
6. Mark geoDBNs as unused that provide an estimation that is lower than $threshold_{use}$.
7. Calculate the user position by considering only those geoDBNs that provide an estimation above $threshold_{consider}$.

An example run illustrates the approach and the algorithm (the shown values are for demonstration purposes only; they do not represent real data):

Let the following table describe the current situation. The array GeoDBN[.] contains the existing geoDBNs geoDBN$_a$ to geoDBN$_d$ with their respective coordinates sorted in descending order of their belief.

i	GeoDBN[i]	GeoPos(.)	Belief(.)	w(.)
1	geoDBN$_a$	$(10, 5, 0)$	$(0.7, 0.3)$	0.7
2	geoDBN$_b$	$(12, 5, 0)$	$(0.7, 0.3)$	0.7
$n = 3$	geoDBN$_c$	$(8, 5, 0)$	$(0.5, 0.5)$	0.5
4	geoDBN$_d$	$(14, 5, 0)$	$(0.1, 0.9)$	0.1

Using (2) the computed user position is $(10.21, 5.00, 0.00)$. Note that only the first three geoDBNs contribute to the calculation since the belief of the fourth geoDBN lies below $\mathit{threshold}_{consider}$ (Step 7 in the algorithm).

The next measurement (Step 1) receives the senders RFID_b, RFID_d, IR_d, RFID_e, and RFID_f. A quick look at the preceding table reveals that geoDBNs already exist for the first three senders whereas the last two are new. The new data is inserted in the geoDBNs (for the first three senders, Step 3b) or new geoDBNs are created respectively (the last three senders, Step 3a). After an update of all geoDBNs (Step 4) and the final sorting (Step 5) we get the following situation:

i	GeoDBN$[i]$	GeoPos(.)	Belief(.)	$w(.)$
1	geoDBN_b	$(12, 5, 0)$	$(0.85, 0.15)$	0.85
2	geoDBN_d	$(14, 5, 0)$	$(0.75, 0.25)$	0.75
3	geoDBN_a	$(10, 5, 0)$	$(0.5, 0.5)$	0.5
$n = 4$	geoDBN_c	$(8, 5, 0)$	$(0.25, 0.75)$	0.25
5	geoDBN_e	$(20, 5, 0)$	$(0.06, 0.94)$	0.06
6	geoDBN_f	$(16, 5, 0)$	$(0.06, 0.94)$	0.06

The fifth and sixth geoDBN are below $\mathit{threshold}_{consider}$, so only the first four geoDBNs are used for the calculation (Step 7). Equation (2) evaluates to $(11.78, 5.00, 0.00)$. Another measurement detects senders RFID_b, RFID_d, and RFID_f. Updating and sorting gives the following:

i	GeoDBN$[i]$	GeoPos(.)	Belief(.)	$w(.)$
1	geoDBN_d	$(14, 5, 0)$	$(0.89, 0.11)$	0.89
2	geoDBN_b	$(12, 5, 0)$	$(0.85, 0.15)$	0.85
3	geoDBN_a	$(10, 5, 0)$	$(0.25, 0.75)$	0.25
$n = 4$	geoDBN_f	$(16, 5, 0)$	$(0.25, 0.75)$	0.25
5	geoDBN_c	$(8, 5, 0)$	$(0.05, 0.95)$	0.05
6	geoDBN_e	$(20, 5, 0)$	$(0.04, 0.96)$	0.04

The beliefs of geoDBN_c and geoDBN_e are now below $\mathit{threshold}_{use}$, so both are marked unused (Step 6, unused values are gray in the table). Only geoDBN_d, geoDBN_b, geoDBN_a, and geoDBN_f are considered in the calculation, so the user position is now $(13.02, 5.00, 0.00)$.

The following section is a detailed description of the geoDBN that we use in our current implementation of the system.

3.3 The Geo Referenced Dynamic Bayesian Network of the Example Application

To model the time slice schemes of the DBN we use our own developed Java application called *JavaDBN* (see Fig. 1). A directed graph (DEA) determines in which orders the time slice schemes can be instantiated, furthermore the user can specify the query and evidence nodes of each time slice schema (see the tabular in each time slice schema window in Fig. 1). Then JavaDBN generates source code (so far Java and C++) that represents the DBN and that contains routines for the inferences. This code permits a constant-space evaluation of DBNs and contains a non-approximative roll-up method, which cuts off older time slices and that incorporates their impact without loss of information on the remaining time slices of the DBN (see [8, 9] for the theoretical background). To our knowledge, this is the first system that allows DBNs to run on mobile systems.

The source code has the following additional features: All for the computation necessary variables are created on initialization of the DBN. No variables are created or disposed at run-time to minimize overhead for memory management (C++) or to avoid the activation of a garbage-collection (Java). The inference in the DBN is optimized regarding the selected query and evidence nodes. Special functions to set evidence are provided. Since variables are freely accessible from outside the class, evidence of a node can also be set directly.

The geo referenced DBN consists of a time slice schema for instantiation of the initial time slice (TSS_1) and a time slice schema for instantiation of succeeding time slices (TSS_2). TSS_1 and TSS_2 are visualized in Fig. 1. TSS_1 differs from TSS_2 in that it does not contain a node UserPos=GeoPos?_1, which models the influence of the precedent time slice on the current one. The node UserPos=GeoPos? in TSS_2 represents the motion model and contains the two states a_1 = "yes" and a_2 = "no". The a-posteriori probability distribution over both these states (also called the belief) represents the network's estimation of the knowledge of whether $\mathcal{U}$ is located at the associated geo coordinate. The state "yes" stands for "knowing if $\mathcal{U}$ is located at the geo coordinate" and the state "no" for "not knowing if $\mathcal{U}$ is located at the geo coordinate". The conditional probability table (CPT) of the node is as follows: $\left[\begin{array}{c|cc} & \underline{a_1} & \underline{a_2} \\ \hline a_1 & 0.7 & 0.001 \\ a_2 & 0.3 & 0.999 \end{array}\right]$. The conditional probabilities of the node are adapted to the mean walking speed of a pedestrian, which causes an accordingly fast decrease of the network's belief if the respective sender is not received for a few subsequent measurements (i.e. the user is not in the vicinity of the sender). The belief increases in an according way if the user enters (or re-enters) the range of the sender.

The node UserPos=GeoPos? in TSS_1 also contains the two states a_1 = "yes" and a_2 = "no", but it lacks the preceding node, so its CPT reduces to the a-priori probabilities $\left[\begin{array}{c|c} a_1 & 0.05 \\ a_2 & 0.95 \end{array}\right]$.

The nodes IRSensor and RFIDSensor correspond to the sensors in the real world. These nodes will be instantiated with the results of the measurement. The conditional probabilities of the nodes model the reliability of the sensors (perceptual model) according to the real world situation. The node IRSensor has the states b_1 = "yes", and b_2 = "no". The following CPT is associated with this node: $\left[\begin{array}{c|cc} & a_1 & a_2 \\ \hline b_1 & 0.9 & 0.05 \\ b_2 & 0.1 & 0.95 \end{array}\right]$. The CPT has the following interpretation: Assumed that the user is in the vicinity of the IR beacon (up to about 2 m distance) then the sensor of the PDA will detect the sender in 90% of all cases and it won't detect in 10% of all cases. In reverse, if the user has a greater distance, the sensor will still detect the IR signal in 5% of all cases but won't detect it in 95% of all cases. Note that the IR beacon sends its signal every 500 ms whereas an RFID tag will send only after receiving a ping-signal from the PDA of the user. Node RFIDSensor, like Node IRSensor, contains the states c_1 = "yes", and c_2 = "no". The lower precision of the RFID-Sensor compared to the IR-Tag is reflected in the associated CPT: $\left[\begin{array}{c|cc} & a_1 & a_2 \\ \hline c_1 & 0.6 & 0.3 \\ c_2 & 0.4 & 0.7 \end{array}\right]$. The imprecision of the RFID signal is, among other things, due to its high sending range. This accuracy can be increased by decreasing the transmitting power of the RFID reader, which sends the above-mentioned ping-signal.

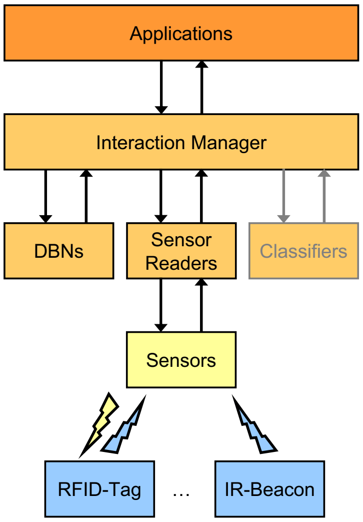
A new time slice is instantiated whenever data arrives and then the estimation is calculated, followed by the roll-up.

4 Example Application

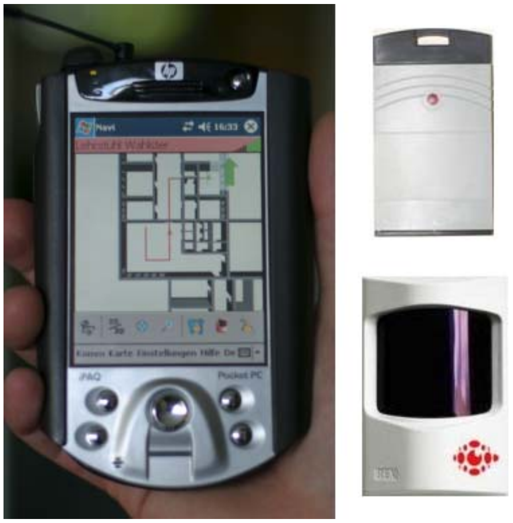
To test our approach we implemented a localization system and equipped parts of our lab with IR beacons and RFID tags. The software is running on an HP iPAQ h5550 with 128 MB ram and Windows CE 4.2 as operating system. The iPAQ is settled in an expansion pack that provides an additional battery and a PCMCIA slot that hosts the active RFID tag reader card. IR signals are received through the internal IR port of the iPAQ. The system itself consists of two applications, which both run on the iPAQ simultaneously: The first application is the PositionSensor, which takes a measurement of both sensors every 500 ms and that reads the stored coordinates of every received RFID tag. The collected data is then processed via the described calculations and the estimated coordinate of the user is send to the second application, the BMW Personal Navigator (see [10]), which visualizes the position to the user.

Figure 2 shows how the PositionSensor is integrated into a framework where applications only query the layer of the Interaction Manager. This Interaction Manager is responsible for forwarding raw sensor data to its corresponding classifiers or DBNs and synchronizing DBN rollups with processing and reading sensors. It also registers new classifiers and sensors and handles their access. This separates the development of applications from extending sensors, classifiers, and DBN libraries — which can still be used independently from all other modules or the framework itself.

Figure 3 shows a test walk of a user $\mathcal{U}$ from Room 121 (lower left in Fig. 3) to Room 119.3 (upper right in Fig. 3) in our lab. The stickman indicates the system's estimation of $\mathcal{U}$'s position. The corresponding circle around each stickman shows the area of the possible real positions. RFID tags are marked as green squares, IR beacons as red squares with cones that indicate the sending direction. Both rooms are equipped with RFID tags in each of the four corners and additional IR beacons at the entrances. Room 119.3 also contains a bookshelf that bears



(a)



(b)

Fig. 2. (a) System Architecture and (b) the iPAQ with PCMCIA RFID-Reader Card, built-in WLAN and IR-Reader (left) and RFID-Tag and IR-Beacon (right)

an IR beacon. The hallway has RFID tags at about every 2 meters and two IR beacons (one in each direction) at about every 4 meters. $\mathcal{U}$ moved with slow to normal walking speed. The user position is very accurate on room level, i.e. if $\mathcal{U}$ is in a particular room, the system will estimate the position somewhere in that room. The position in the room itself is rather coarse but the accuracy can be greatly increased by placing IR beacons at points of special interest (like the bookshelf in Room 119.3). The position estimation in the hallway varies about 1 meter from the actual position.

5 Conclusion and Future Work

We presented a new approach to estimate a user position by probabilistic sensor fusion. Our method is low in space and time complexity and can therefore run completely on a PDA. The calculation of the position does not need a model of the environment and the time-consuming task of calibration is superfluous. Any arbitrary environment can be enhanced by simply putting up tags and beacons and writing the respective coordinates in the memory of these devices. Regions with coarser or finer granularity can be established by using senders with the appropriate precision (e.g. IR beacons for higher precision and RFID tags for lower precision).

We think that the concept of geo referenced dynamic Bayesian networks can be generalized to referenced dynamic Bayesian networks, because DBNs can also

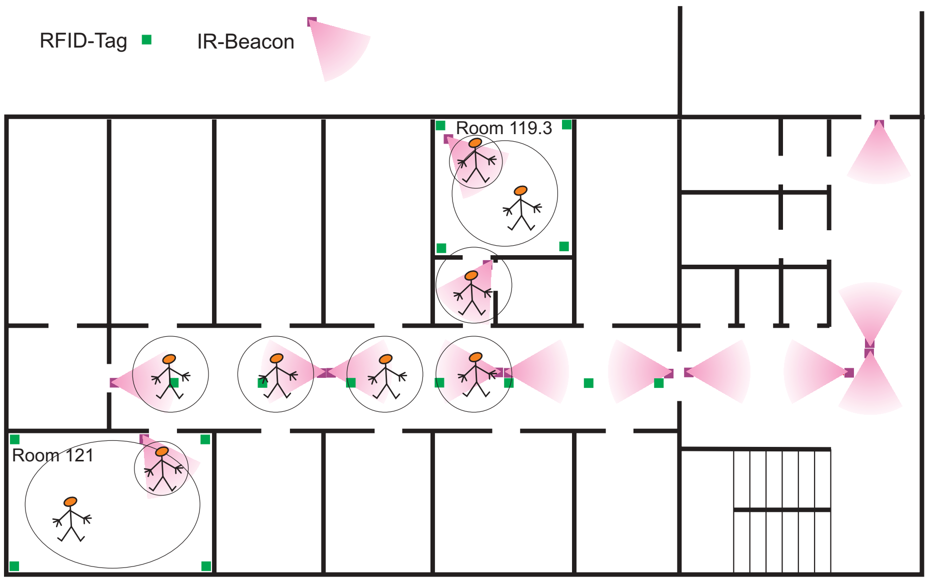


Fig. 3. A test walk through our lab. The stickman indicates the estimated position and the circle the area of the real positions

be referenced to actions, plans or symbolic locations. Future work will be the integration of other sensor types (e.g. video cameras, microphone arrays) and a system to determine the optimal placing of the senders and sensors. Another project at our lab researches the usability of biophysiological sensors. Part of that work (e.g. accelerometers) can be used to adapt the motion model of the geoDBNs to the current state of the user (e.g. sitting, walking fast or walking slow).

Acknowledgments

We would like to thank Anthony Jameson, Antonio Krüger, Jörg Baus, and Christoph Stahl for their many helpful suggestions.

This research is being supported by the German Science Foundation (Deutsche Forschungsgemeinschaft (DFG)) in its Collaborative Research Center on Resource-Adaptive Cognitive Processes, SFB 378, Project EM 4, BAIR, and its Transfer Unit on Cognitive Technologies for Real-Life Applications, TFB 53, Project TB 2, RENA.

References

1. Russell, S.J., Norvig, P. In: Artificial Intelligence, A Modern Approach. second edn. Pearson Education (2003) 492–580

2. Fox, D., Hightower, J., Liao, L., Schulz, D., Borriello, G.: Bayesian filtering for location estimation. IEEE Pervasive Computing **2** (2002) 24–33
3. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The Active Badge Location System. ACM Transanctions on Information Systems **10** (1992) 91–102
4. Harter, A., Hopper, A., Steggles, P., Wart, A., Webster, P.: The anatomy of a context-aware application. In: 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom '99). (1999)
5. Ubisense Unlimited: Ubisense. http://www.ubisense.net (2004)
6. Ni, L.M., Liu, Y., Lau, Y.C., Patil, A.P.: Landmarc: Indoor location sensing using active RFID. In: IEEE International Conference in Pervasive Computing and Communications 2003 (Percom 2003). (2003)
7. Amemiya, T., Yamashita, J., Hirota, K., Hirose, M.: Virtual Leading Blocks for the Deaf-Blind: A Real-Time Way-Finder by Verbal-Nonverbal Hybrid Interface and High-Density RFID Tag Space. In: Proceedings of the 2004 Virtual Reality (VR'04), IEEE Virtual Reality (2004) 165–172
8. Darwiche, A.: A differential approach to inference in Bayesian networks. Journal of the Association for Computing Machinery **50** (2003) 280–305
9. Brandherm, B., Jameson, A.: An extension of the differential approach for Bayesian network inference to dynamic Bayesian networks. International Journal of Intelligent Systems **19** (2004) 727–748
10. Krüger, A., Butz, A., Müller, C., Stahl, C., Wasinger, R., Steinberg, K.E., Dirschl, A.: The Connected User Interface: Realizing a Personal Situated Navigation Service. In: IUI 04 - 2004 International Conference on Intelligent User Interfaces, ACM Press (2004)