Particle Filters for Location Estimation in Ubiquitous Computing: A Case Study

Jeffrey Hightower[1,2] and Gaetano Borriello[1,2]

[1] Intel Research Seattle
1100 NE 45th St.
Seattle, WA 98105
[2] University of Washington
Computer Science & Engineering
Box 352350
Seattle, WA 98195

Abstract. Location estimation is an important part of many ubiquitous computing systems. Particle filters are simulation-based probabilistic approximations which the robotics community has shown to be effective for tracking robots' positions. This paper presents a case study of applying particle filters to location estimation for ubiquitous computing. Using trace logs from a deployed multi-sensor location system, we show that particle filters can be as accurate as common deterministic algorithms. We also present performance results showing it is practical to run particle filters on devices ranging from high-end servers to handhelds. Finally, we discuss the general advantages of using probabilistic methods in location systems for ubiquitous computing, including the ability to fuse data from different sensor types and to provide probability distributions to higher-level services and applications. Based on this case study, we conclude that particle filters are a good choice to implement location estimation for ubiquitous computing.

1 Introduction

Location estimation is important in ubiquitous computing because location is an important part of a user's context. Context-aware applications can be proactive in gathering and adapting information for the user if they have location estimates accurate to an appropriate grain-size. By being location-aware, these applications can more seamlessly blend into the user's tasks and minimize distraction. In fact, many applications may act autonomously without explicit user attention. For example, a location-aware to-do list can compare the user's position against to-do items and alert the user when she is near a location where a task can be completed.

Location estimation systems can be based on a wide range of sensing technologies such as GPS [1, 2, 3] , infrared [4, 5], ultrasound [6, 7], WiFi [8, 9], vision [10], and many others (see [11] for a survey). Many of today's deployed systems are stove-piped, that is, an application and sensing technology are coupled so it

N. Davies et al. (Eds.): UbiComp 2004, LNCS 3205, pp. 88–106, 2004.

is difficult to change sensors and still be able to use the same application. An example is GPS navigation units in automobiles that would need to be greatly revised if they were instead to use proximity to radio sources and dead reckoning from inertial sensors. In general, the location estimation problem involves:

1. a set of objects whose location must be estimated,
2. a set of potentially heterogeneous location sensors,
3. a timestamped sequence of measurements, each one generated by a sensor about an object,
4. a motion model for the various objects, and
5. an algorithm to update an object's position given a new measurement, the sensor type, and the elapsed time since the last measurement.

There are important issues to resolve for each part of the problem. How many objects can be located and at what granularity? How can heterogeneous sensor measurements be fused? How are measurements collected and where is the location estimation computation performed? Are there a limited set of motion models? How is the result of the positioning algorithm presented to applications (e.g., a single point, a region, a probability distribution)?

This paper presents a case study showing that particle filters are a good algorithmic choice for location estimation for ubiquitous computing. They work well with the sensors, motion models, hardware platforms, and queries relevant to ubiquitous computing. Particle filters are a Bayes filter implementation popularized by the robotics community. They are robust in that they can represent arbitrary probability distributions. Using trace logs from a deployed multi-sensor location system, we show that particle filters can be as accurate as common deterministic location algorithms, they can be run efficiently on a variety of mobile and stationary computing platforms used in ubiquitous computing, and they can fuse heterogeneous sensor data to support useful abstractions for higher-level services and applications.

The sections of this paper are structured to make each point independently. After introducing particle filters in section 2 we begin with accuracy comparisons in section 3, cover performance in section 4, and finally discuss the general advantages of using probabilistic methods in location systems for ubiquitous computing in section 5. Section 6 concludes and describes our current and future work.

2 Particle Filters

A particle filter is a probabilistic approximation algorithm implementing a Bayes filter. Particle filters are a member of the family of sequential Monte Carlo methods [12, 13]. For location estimation, Bayes filters maintain a probability distribution for the location estimate at time t referred to as the belief $Bel(x_t)$. Particle filters represent the belief using a set of weighted samples $Bel(x_t) = \{x_t^i, w_t^i\}, i = 1 \ldots n$. Each x_t^i is a discrete hypothesis about the location of the object. The w_t^i are non-negative weights, called *importance factors*

which sum to one. Our previous survey paper [14] provides a tutorial and more in depth discussion of the mathematics of Bayes filters for location estimation and compares particle filters to other Bayesian filtering techniques such as Kalman filters, Multi-Hypothesis Tracking, and grid and topological approaches

Particle filters have proven valuable in the robotics community for state estimation problems such as simultaneous localization and mapping (SLAM) [15, 16]. Our particle filter implementation for this case study is based on a standard approach used in the robotics community for robot localization: each new sensor measurement causes the belief samples to be updated using a procedure called Sequential Importance Sample with Resampling (SISR). In this context, SISR involves predicting each sample's motion using a *motion model*, weighting all samples by the sensor's *likelihood model* for the current measurement, and resampling using importance sampling, that is, choosing a new set of samples according to the weights of the prior samples. The appropriate number of samples is determined at each step using a procedure called *KLD adaptation*. Objects are tracked in three dimensions (x, y, z, pitch, roll, yaw, velocity, weight)[3] for maximum flexibility in locating both people and objects.

Motion Model The motion model implements the Bayes filter prediction step. Unlike in robotics where odometry information provides observations about motion, ubiquitous computing requires a motion model with no explicit input. Some systems can infer motion indirectly through assumptions about the behavior of the underlying sensor technology. For example, Locadio estimates whether a device is moving or still based on the variance of access point sightings and their signal strength values over a sliding window [17]. Our particle filter in this study, however, does not assume the use of any particular sensor technology and therefore employs a general human motion model. Each sample has velocity as part of its belief state along with its location. Each SISR iteration adjusts the velocity of each sample by introducing Gaussian acceleration noise with $\sigma = 0.5$ meters per second per second multiplied by elapsed time. Velocity is clipped to the range 0 to 10.22 meters per second—from stopped to the fastest recorded human running speed. Rotational velocity is modeled similarly. After updating velocities, samples are all moved to new positions according to their adjusted velocities and elapsed time. Sample motion is further constrained by a collision detection algorithm such that no sample may move through walls in the known map of the world.

Sensor Likelihood Model A particle filter can fuse measurements taken by heterogeneous sensor technologies. Adding a sensing technology means creating a new likelihood model characterizing the sensor. Likelihood is the conditional probability $P(z|x)$, the probability of position x of the mobile object relative to the sensor given measurement z taken by the sensor. In this case study, likelihood models are fixed and defined a priori for each sensing technology based on offline experiments to characterize sensor error. For example, our likelihood function for the VersusTech commercial infrared badge

[3] Our 3D state vector actually has 9 dimensions instead of 8 because we use a 4-element quaternion to represent the rotational (pitch, roll, and yaw) components.

system is a parametric Gaussian model of infrared range ($\mu = 0, 2\sigma = 15ft$) derived from experiments in which we verified the manufacturer's claims of a 15 foot range. Our ultrasound time-of-flight badge likelihood is a lookup table built from lab experiments characterizing the ultrasound system's measurement error. Visual examples of the infrared and ultrasound likelihood functions are illustrated in Figure 1.

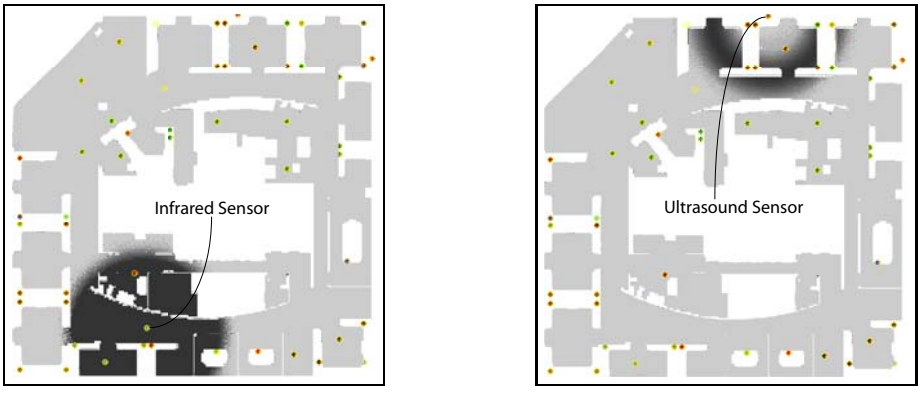

Fig. 1. Example sensor likelihood models for an infrared measurement (left) and a 4.5 meter ultrasound measurement (right) in our 30m x 30m office environment. Darker areas represent higher likelihoods.

KLD Adaptation SISR performance is determined by the number of samples. The minimal number of samples needed to represent the distribution at each iteration is determined using a method called Kullback-Leibler distance (KLD) adaptive sampling. KLD adaptation is the best-known-method in the literature to compute the minimum sample count required to represent a distribution [18]. We use KLD parameters shown by Fox and colleagues to work well with our particular map sizes and motion models: $\epsilon = 0.1$, $\Delta = 0.5m$, $z_{1-\delta} = 0.99$.

Although motion and sensor models used in this case study's particle filters are static and defined a priori, it would be possible to apply machine learning techniques to build the models from training data and even to adjust those models dynamically. Learning represents a tradeoff between accuracy and generality. Learning optimal models for an environment, for example the radio propagation characteristics or motion constraints of a particular building, can greatly increase the algorithm's accuracy in the environment represented in the training data, but naturally decreases the generality of the implementation in environments outside the training data.

3 Accurate Location Estimation

In this section, we present an experiment comparing several position estimation algorithms. We show that a particle filter can do as well as these deterministic algorithms in instantaneous position accuracy and has better dynamic properties.

3.1 Experimental Setup

We have deployed location sensors throughout our building including, among others, a commercial infrared badge system from VersusTech, an ultrasound time-of-flight badge system based on the MIT Cricket boards [6] , and a home-grown WiFi device positioning system. Under normal operation, our distributed location service fuses measurements from all these sensor technologies to track more than 30 lab residents as well as high-value and frequently lost pieces of equipment.

Comparing the accuracies of position estimation algorithms requires knowing ground-truth, which is not available during normal operation. Therefore, for these experiments we gathered measurement logs using a robot programmed to duplicate human-like motion. The robot is not in any way part of the normal configuration or operation of the location system. It is used only to collect sensor traces which also include ground truth for this paper's experimental analyses. The alternative—having a human wearing sensors continually click on a map to indicate their true position as they walk about—is both tedious and error prone in comparison.

The robot is equipped with a scanning laser range finder and can compute its position to a few centimeters and its orientation to one degree. On top of the robot we mounted the "scarecrow," a pole simulating the height and torso of a human. On the scarecrow are sensors consisting of an ultrasound badge, two types of infrared badges, RFID tags, and a WiFi client device. Figure 2 shows a picture of the robot and scarecrow.

We used the robot-plus-scarecrow setup to generate several hours of measurement logs covering our entire $900m^2$ office building. All results presented in this paper are from a 15 minute segment of this larger log. 15 minutes is sufficient length that results are clear yet generalizable. The robot's speed ranged from 0–2 meter per second—reasonable human walking speeds. The robot traveled to waypoints throughout the space on routes generated by a path planner. A collision avoidance algorithm allowed it to avoid people and other transient objects. Indeed, in the interest of realism we made no effort to clear the environment of people and other objects. Finally, to duplicate human-like motion, during data collection we would periodically override the path planner to make the robot accelerate, slow, stop and wait, turn, or "change its mind" by interjecting a new waypoint into the plan.

Sensor measurements were logged at the normal rate for each technology. Infrared badges beacon at approximately 0.5Hz, ultrasound badges at 3Hz. In both cases, packets may be seen by multiple basestations or packets may be dropped

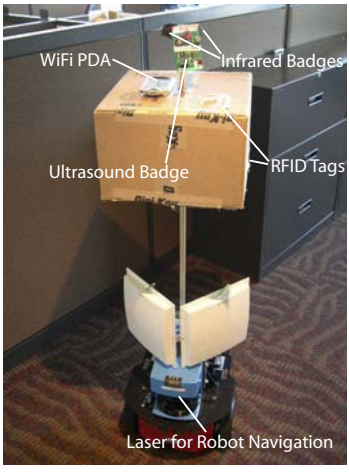

Fig. 2. This robot gathered measurement trace logs which also include ground-truth position information. The robot has a laser range finder to compute its precise position. The "scarecrow" on top simulates the torso height of a human. On the scarecrow is an ultrasonic time-of-flight badge, two types of infrared proximity badges, RFID tags, and a WiFi client device.

if no basestation is visible due to obstructions, packet collisions, or other interference. The ultrasound system is particularly susceptible to packet collisions due to reflections that act to confuse its randomized scheduling algorithm. The infrared system is prone to dropped packets due to its lower beacon rate and sparser basestation infrastructure. In total, the 15 minute log used in this paper has 2932 ultrasound measurements and 537 infrared measurements from the scarecrow-mounted sensors.

3.2 Algorithms We Compared

We compared the accuracy of particle filters described in section 2 and several deterministic position estimation algorithms:

Point The estimate is placed at the same position as the sensor generating the latest measurement. Point is the simplest algorithm for cellular location systems. Point is used by most commercial infrared badge location systems and some cellular telephony location services.

Centroid The estimate is placed at the geometric centroid of the positions of the last c sensors generating measurements. The value c is optimized offline to provide best estimates for a given environment; e.g. $c = 3$ is best for our infrared badge system.

Smooth Centroid Like Centroid, except the latest s estimates are also weighted by their age and positionally averaged to smooth the motion over a sliding

window. Smooth Centroid was the algorithm used by the SpotON ad hoc wireless location system [19].

Smooth Weighted Centroid Like Smooth Centroid, except the centroid position computation is weighted by the sensor likelihood models. Using this weighting, SWC can take into account both the error characteristics of the sensors and the parameters of the measurements, e.g. the linear distance measured by an ultrasound badge system or the propagation characteristics of radio beacons. SWC is comparable to the centroid algorithm used by Bulusu and colleagues to implement location estimation in ad hoc wireless mesh networks [20].

3.3 Accuracy Results

The particle filter's instantaneous position accuracy, computed as the weighted mean of its samples, is at least as good as the estimate produced by the deterministic algorithms. Figures 3 and 4 illustrate this result by comparing point-estimate accuracy over the 15 minute trace log for infrared alone and for combined infrared and ultrasound. The trace log and ground truth were the same for all runs and only the choice of algorithm was altered. The particle filter is as accurate as the others and much more so when sensors are combined. Because our ultrasound system is prone to significant timing and multipath errors, the sensor model has a high degree of uncertainty. Particle filtering, being an approximation which also estimates object's motions, is well suited to modeling uncertainty.

The particle filter shows better dynamic motion tracking in Figures 5 and 6. These graphs compare accumulated motion error over time using infrared alone and mixed infrared and ultrasound. The incremental error at each time step is the difference between the estimated distance moved and the actual distance moved. A slower accumulation of error implies that the algorithm better tracks the true motion dynamics of the object. The particle filter excels in cumulative error. The difference most striking in the case of combined infrared and ultrasonic sensors where the accumulated error stays near zero for the entire duration of the test. Note that the y-axis' magnitude in Figure 6 is greater than Figure 5 to capture the greater error in some algorithms when including ultrasonic sensors.

4 Practical Location Estimation

In this section we present performance results. Particle filters, like many probabilistic methods, do require more computation time and memory than simpler deterministic position estimation algorithms like weighted centroids. However, as we show in this section, performance is sufficient to make our particle filter implementation practical on real devices used in ubiquitous computing. Specifically:

- The particle filter is practical on small devices. A modern PDA can position itself using common sensors at a rate of approximately 0.5Hz using 1MB of memory.

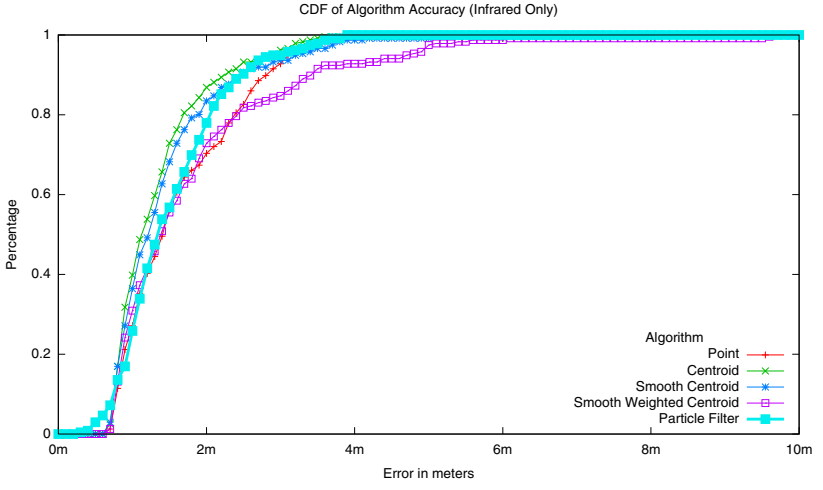

Fig. 3. This cumulative distribution compares the accuracy of several algorithms over a 15 minute log of infrared sensor measurements. The error is the distance between the algorithm's point-estimate of the most likely position and the ground truth.

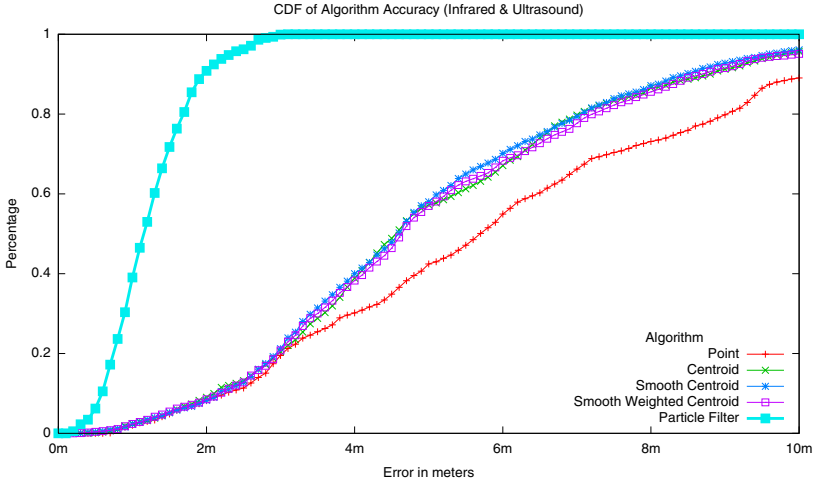

Fig. 4. This cumulative distribution shows the relative accuracy of several algorithms over a 15 minute log of fused infrared and ultrasound sensor measurements. The error is the distance between the algorithm's point-estimate of the most likely position and the ground truth.

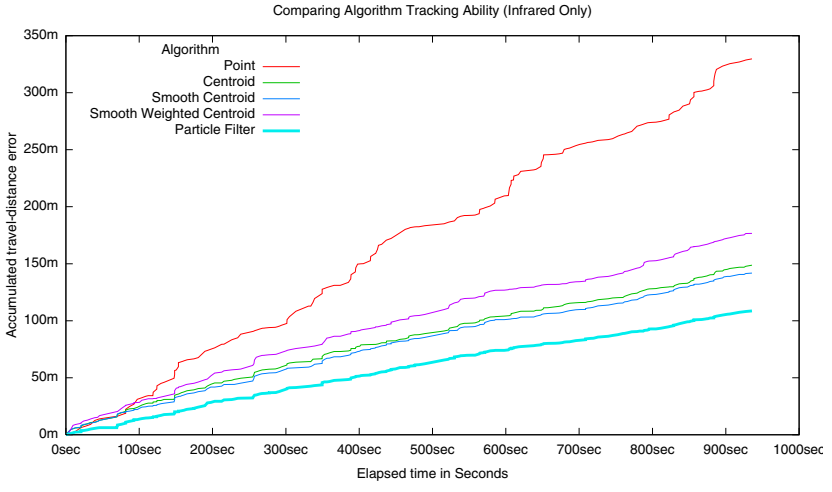

Fig. 5. This time-series shows the cumulative motion distance error of several algorithms over a 15 minute log of infrared measurements. The incremental error at each time step is the absolute value of the difference between the estimated distance moved and the actual distance moved. A slower accumulation of error implies that the algorithm better tracks the true motion dynamics of the object.

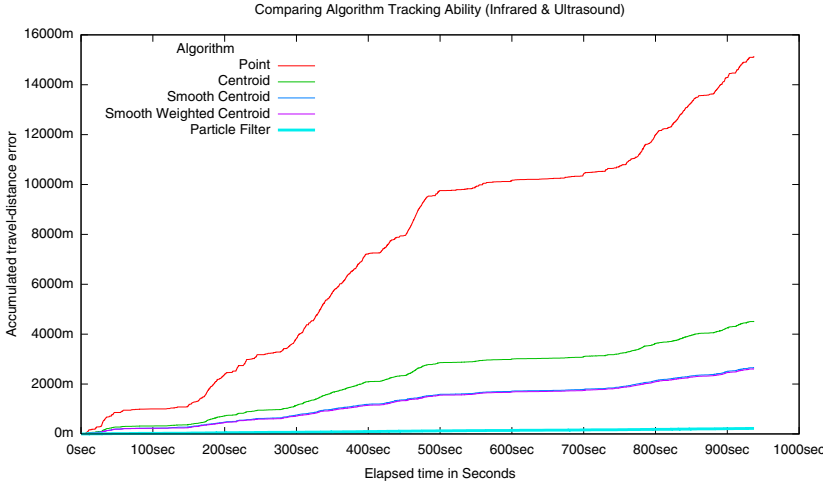

Fig. 6. This time-series shows the cumulative motion distance error several algorithms over a 15 minute trace log of both infrared and ultrasound measurements. The incremental error at each time step is the absolute value of the difference between the estimated distance moved and the actual distance moved. A slower accumulation of error implies that the algorithm better tracks the true motion dynamics of the object.

– Tablet and notebook-class devices can use the particle filter to estimate their position using multiple sensing technologies at a high update rate.
– Particle filters can be used in "enterprise" sensing systems where many tagged objects and people are tracked by central servers. A modern server can track upwards of 18 objects tagged with both infrared and ultrasound badges at a rate of 1 measurement per second per object.

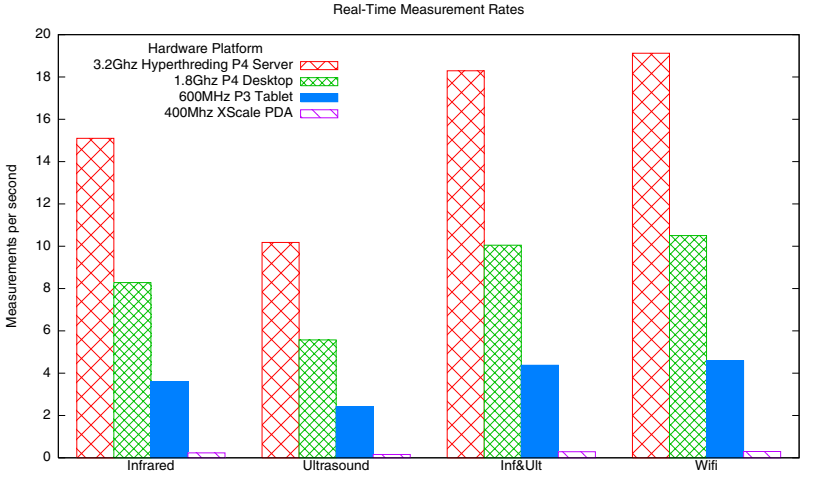

Fig. 7. This graph shows the particle filter's performance on real devices. Bars show the measurement rates that can be maintained in real time for different sensor technologies on four different hardware platforms. This graph combines the average number of samples required under each sensor technology (figure 8) with the time-performance results in figure 10.

Figure 7 summarizes the performance results. In the rest of this section, we deconstruct the respective time and space performance behind Figure 7 to illustrate the particle filter's practicality.

4.1 Memory Usage

Particle filter performance is almost entirely determined by the number of samples needed to accurately represent the probability distribution. Recall from section 2 that each step of SISR uses KLD adaptation to adjust the number of samples to the minimum number needed to represent the belief. Figure 8 shows a cumulative distribution of the KLD-adaptive sample counts and memory usage from our 15-minute trace log. Our space-optimized implementation of particle filters tracking in 3D requires approximately 500 kilobytes of constant memory plus 120 bytes per sample. For comparison to Figure 8, Figure 9 shows the sample count and memory needed to represent several reference distributions with

a particle filter. From these graphs we can conclude that on common ubiquitous computing sensor technologies, particle filters can have modest memory requirements of 1-2MB easily met by even PDA-class devices.

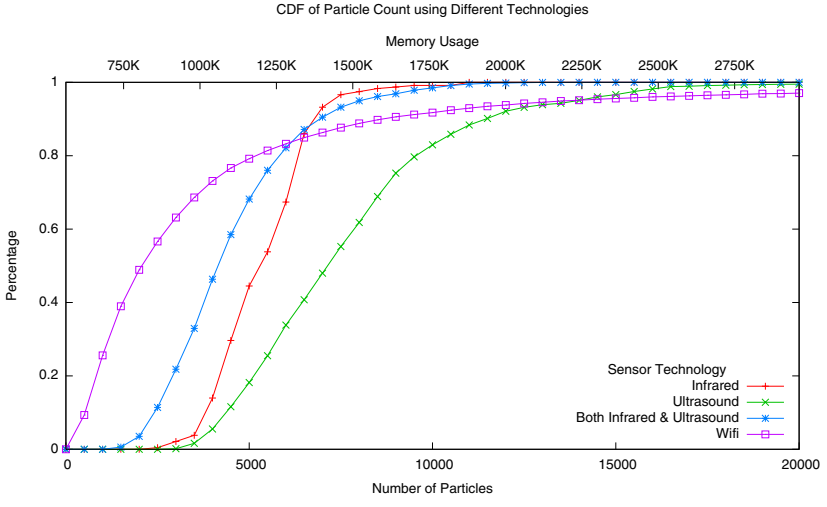

Fig. 8. This cumulative distribution show the KLD-adaptive sample counts (bottom x-axis) and memory requirements (top x-axis) for a 15 minute trace log under different sensor technologies.

4.2 Time Performance

Figure 10 shows the computation time required to perform SISR update on particle filters of different sizes on different platforms. As expected, computational performance scales linearly in the number of particles. Future increases in processor speed will linearly increase the measurement rate or number of trackable objects in a server architecture.

5 Flexible Location Estimation

Beyond the specific accuracy benefits and practical performance shown in the previous sections, in general, probabilistic approaches like particle filters are also more flexible than deterministic methods. Probabilistic methods inherently estimate the actual probability distribution of a location estimate instead of simply a single-point "you-are-here" estimate. This completeness affords low-level sensor fusion, mid-level spatial relationship computations, and high-level value to applications such as traceability and machine learning of context and human activities. These properties make particle filters, as one example of probabilistic

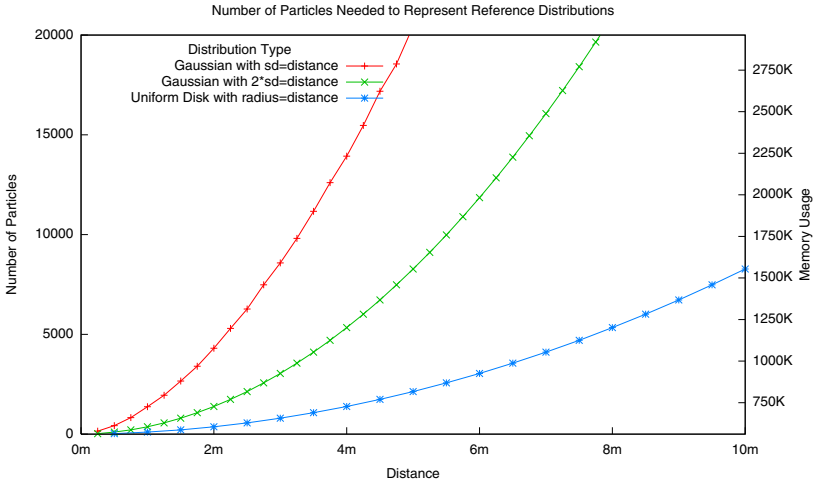

Fig. 9. This plot shows the number of particles (left y-axis) and memory required (right y-axis) to represent several well-known reference distributions.

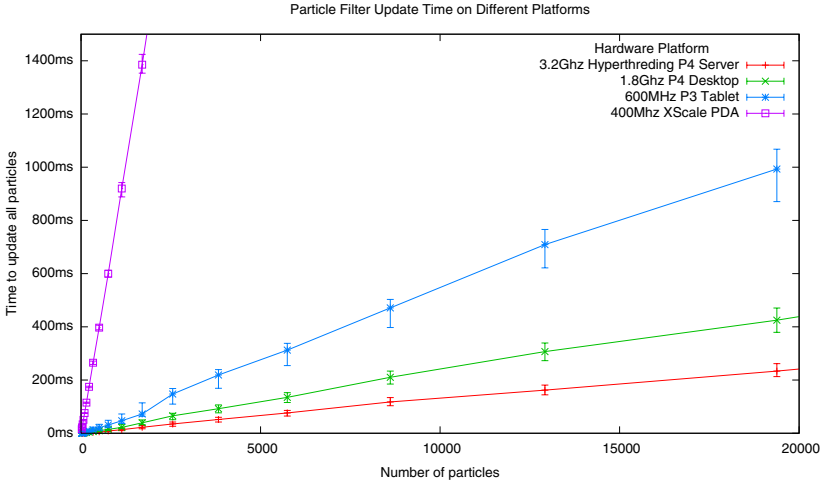

Fig. 10. This graph shows the computation time needed for SISR update on different hardware platforms ranging from a small PDA to a Pentium-4 Hyperthreading server. Particle filters scale computationally linearly in the number of particles.

methods, an ideal tool to implement established location system design abstractions such as Sentient Computing [7] or the Location Stack [21, 22], shown in Figure 11.

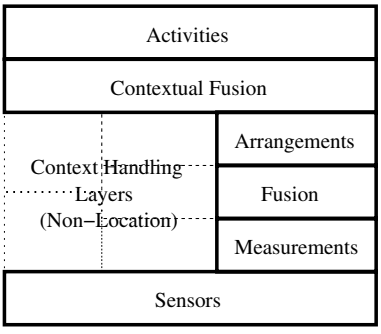

Fig. 11. The Location Stack design abstractions [21, 22] can be implemented flexibly with particle filters. Particle filters readily support multi-sensor location fusion in the Measurements and Location Fusion layers, are easy to use to answer probabilistic Arrangements queries, and are extensible to high-level Context and Activity inference.

5.1 Sensor Fusion

Figure 12 shows the benefits of sensor fusion using particle filters with infrared badges, ultrasound badges, and both. Note that unlike Figures 3 and 4, this figure compares particle filters to particle filters so the error is root-mean-square error instead of simply point error. From this graph we can see that using both sensor technologies preserves the accuracy of the more precise technology and can decrease the standard deviation below the level of either technology alone. Sensor fusion capability gives location system builders the flexibility to deploy heterogeneous sensing hardware in order to minimize cost, increase reliability, or increase coverage. Additional research has increased particle filter's flexibility even further by allowing for the incorporation of anonymous sensors like scanning laser range finders [23].

5.2 Arrangements

Using particle filters for position estimation makes it easy to implement the spatial reasoning abstractions desired by ubiquitous computing systems. A *proximity* engine can compute statistics relating multiple objects by comparing pair-wise distances of the objects' particles. A proximity query returns the estimated distance between two objects along with a confidence value or the probability that two objects are `within()` or `at-least()` a distance d from one another. A *containment* engine can compute the probability that one or more objects are in a

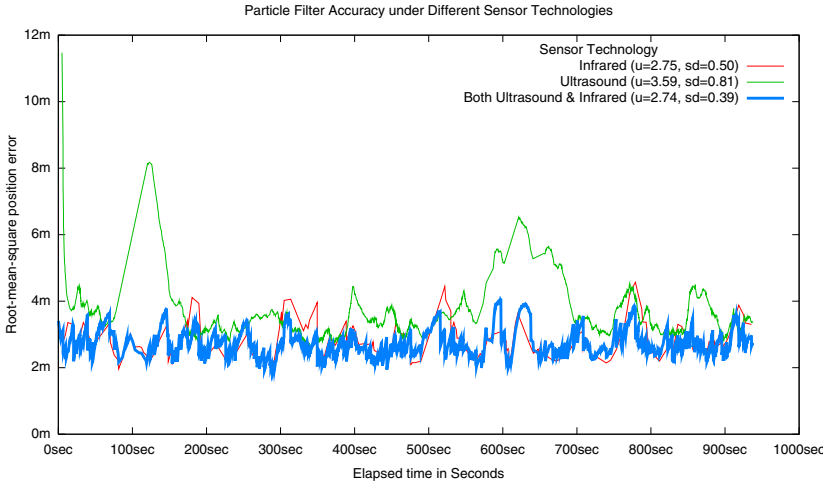

Fig. 12. This 15 minute time-series illustrates sensor fusion: particle filters can combine heterogeneous sensor measurements and achieve the accuracy of the best technology at any given time. The y-axis shows root-mean-square.

room by simply counting the proportion of particles inside the geometric volume of the polygon delineating the room. Containment is illustrated in Figure 13.

Containment and proximity built on particle filters provide a probabilistic implementation of the Arrangements layers seen in many ubiquitous computing location systems such as the "programming with space" metaphor used by location systems such as AT&T's Sentient Computing project [7]. More advanced research into using particle filters for spatial reasoning has shown how to learn and predict motion patterns [24]. In this work, we apply Expectation Maximization to learn typical motion flow of particles along Voronoi graphs of the environment—much like learning "wear lines" in the carpet. The learned motion is then used both to improve the object's motion models and to predict the destination of a moving object based on its learned motion pattern.

5.3 Applications and Activity Inference

Obviously, having probability distributions for location and spatial arrangements provides important information to applications. Application builders can exploit probability information to add understandability to user interfaces. When the system infers that a user is in a particular room, engaged in a certain activity, or existing in a specific context, it can also present traceability information such as: Why did the system make that inference? How sure is the system of the inference it made? What are the alternatives the system considered and with what probability? For example, our handheld mapping interface shows the set

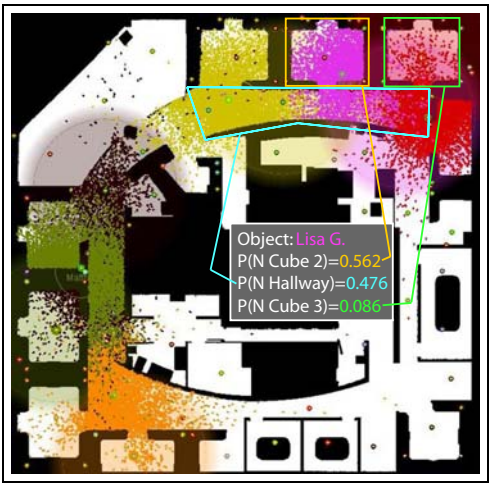

Fig. 13. A *containment* query on a snapshot of 6 people tracked by a multi-sensor location system. The callout shows probabilities for the position of "Lisa G.", the person most likely in the middle upper room. Using particle filters, containment in a room can be computed simply by counting the proportion of particles inside the room's geometric volume.

of rooms in which the user may be located and can use color or opacity cues to indicate the system's belief in each hypothesis.

Probability distributions also enable machine learning of high-level features beyond location. As their input priors, machine learning algorithms usually need to know the true probability distributions of location estimates and spatial relationships. For example, [3] shows how to augment particle filters to estimate mode of transportation in a city (walk, car, bus) based on a stream of GPS positions. Because of these capabilities, we believe particle filters are an enabling technology to ongoing ubiquitous computing work on learning significant locations and motion patterns [2] and inferring situational context other than location [25].

6 Conclusion

In this paper we have presented an in-depth case study demonstrating a particle filter can be an accurate, practical, and flexible location estimation technique for ubiquitous computing.

Accuracy Particle filter's accuracy can be as good as deterministic algorithms plus particle filters can much better estimate dynamic motion.

Practicality Probabilistic methods do require more computation and memory than deterministic algorithms, but our analyses show that particle filter's

performance is sufficient for real scenarios on real devices ranging from small handhelds to large servers.

Flexibility Particle filters' affordance for sensor fusion lets developers choose heterogeneous sensing hardware to minimize cost, increase reliability, or increase coverage as needed. Particle filters' support of spatial reasoning such as containment and proximity enables probabilistic versions of established location programming models. Because particle filters inherently represent the probability distributions of estimates, applications developers can enhance user interfaces to indicate the system's confidence in its inference and the viability of alternative hypotheses.

Our implementation of the Location Stack abstractions using the particle filter in this case study has enjoyed significant external adoption including research adoption by the Place Lab project (`www.placelab.org`), commercial adoption in Intel's Universal Location Framework (ULF), and community adoption through our publicly available location estimation library.

Place Lab is illustrative of particle filter's success in a wide-area deployment. It is an emerging global location system with low barrier to entry [26]. Place Lab allows any WiFi client device to estimate its position by listening for WiFi access point beacons and looking up beacons they hear in a local snapshot of a global access point position database. The access point database is built by users who volunteer logs of the access points they encounter. Place Lab uses particle filters to perform both the client's position calculations and the databases' access point position estimation. Particle filters allow Place Lab to provide rich programming interfaces on devices ranging from full-fledged notebook computers to small PDA and cell phone devices. Particle filters' flexibility allows Place Lab to easily explore using additional sensor technologies such as GPS and GSM telephony.

The Universal Location Framework (ULF), Intel's commercial adoption of the approach, focuses on the problem of providing users with a seamless location service and developers with a consistent API as devices move between indoor and outdoor environments. ULF uses GPS receivers when outdoors and WiFi signal-strength triangulation when indoors. Applications are unaware of the shifting reliance on the two sensing technologies. The API provides a position estimate along with a measure of the error. [22] documents the Intel engineers' experiences adopting the Location Stack and estimation library. Figure 14 shows ULF's tablet and multi-radio handset prototypes.

In the future, we seek to further increase particle filters' performance on computationally limited devices by extending the work of Kwok and colleagues on adaptive real-time particle filters [27] to take into account the characteristics of ubiquitous computing environments. More broadly, we plan to blend particle filters with probabilistic techniques such as multi-hypothesis tracking and Rao-Blackwellized particle filters. Many of these variations have characteristics similar to basic particle filters but they are better suited for complex high-level learning problems such as inferring human activities. For example, complex estimation problems often require structured versions of Bayes filters, such as hidden Markov models and dynamic Bayesian networks. Through these efforts

Fig. 14. A tablet (left) and Intel's Universal Communicator (right) are Intel's prototype multi-sensor location devices built with the Location Stack abstractions and particle filter location library.

we hope to bring to location-aware computing the same probabilistic power of representing uncertainty at different levels of abstractions that particle filters have brought to location estimation.

Acknowledgments. We appreciate the comments, advice, and insights of our reviewers and our manuscript shepherd.

References

[1] Misra, P., Burke, B.P., Pratt, M.M.: GPS performance in navigation. Proceedings of the IEEE (Special Issue on GPS) **87** (1999) 65–85
[2] Ashbrook, D., Starner, T.: Using GPS to learn significant locations and predict movement across multiple users. Personal and Ubiquitous Computing **7** (2003) 275–286
[3] Patterson, D.J., Liao, L., Fox, D., Kautz, H.A.: Inferring high-level behavior from low-level sensors. In: Proceedings of the Fifth International Conference on Ubiquitous Computing (Ubicomp 2003), Springer-Verlag (2003) 73–89
[4] Want, R., Hopper, A., Falcao, V., Gibbons, J.: The active badge location system. ACM Transactions on Information Systems **10** (1992) 91–102
[5] Want, R., Schilit, B., Adams, N., Gold, R., Petersen, K., Goldberg, D., Ellis, J., Weiser, M.: The parctab ubiquitous computing experiment. In Imielinski, T., ed.: Mobile Computing. Kluwer Publishing (1997) 45–101 ISBN 0-7923-9697-9.
[6] Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The cricket location-support system. In: Proceedings of the Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM), Boston, MA, ACM, ACM Press (2000) 32–43
[7] Addlesee, M., Curwen, R., Hodges, S., Newman, J., Steggles, P., Ward, A., Hopper, A.: Implementing a sentient computing system. Computer **34** (2001) 50–56
[8] Bahl, P., Padmanabhan, V.: RADAR: An in-building RF-based user location and tracking system. In: Proceedings of IEEE INFOCOM. Volume 2., Tel-Aviv, Isreal (2000) 775–784

[9] Bhasker, E.S., Brown, S.W., Griswold, W.G.: Employing user feedback for fast, accurate, low-maintenance geolocationing. Technical Report CS2003-0765, UC San Diego, Computer Science and Engineering (2003)
[10] Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, S.A.: Easyliving: Technologies for intelligent environments. In: 2nd Intl. Symposium on Handheld and Ubiquitous Computing. (2000) 12–27
[11] Hightower, J., Borriello, G.: Location systems for ubiquitous computing. Computer **34** (2001) 57–66 This article is also excerpted in "IT Roadmap to a Geospatial Future," a 2003 report from the Computer Science and Telecommunications Board of the National Research Council.
[12] Doucet, A., Godsill, S., Andrieu, C.: On sequential Monte Carlo sampling methods for Bayesian filtering. Statistics and Computing **10** (2000)
[13] Doucet, A., de Freitas, N., Gordon, N., eds.: Sequential Monte Carlo in Practice. Springer-Verlag, New York (2001)
[14] Fox, D., Hightower, J., Liao, L., Schulz, D., Borriello, G.: Bayesian filtering for location estimation. IEEE Pervasive Computing **2** (2003) 24–33
[15] Fox, D., Burgard, W., Kruppa, H., Thrun, S.: A probabilistic approach to collaborative multi-robot localization. Autonomous Robots **8** (2000) 325–244
[16] Haehnel, D., Burgard, W., Fox, D., Thrun, S.: An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ (2003)
[17] Krumm, J., Horvitz, E.: Locadio: Inferring motion and location from wi-fi signal strengths. In: Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, MA (2004)
[18] Fox, D.: KLD-sampling: Adaptive particle filters. In Dietterich, T.G., Becker, S., Ghahramani, Z., eds.: Advances in Neural Information Processing Systems 14 (NIPS), Cambridge, MA, MIT Press (2002) 713–720
[19] Hightower, J., Want, R., Borriello, G.: SpotON: An indoor 3d location sensing technology based on RF signal strength. UW CSE 00-02-02, University of Washington, Department of Computer Science and Engineering, Seattle, WA (2000)
[20] Bulusu, N., Heidemann, J., Estrin, D.: GPS-less low cost outdoor localization for very small devices. IEEE Personal Communications **7** (2000) 28–34 Special Issue on Smart Spaces and Environments.
[21] Hightower, J., Brumitt, B., Borriello, G.: The location stack: A layered model for location in ubiquitous computing. In: Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), Callicoon, NY, IEEE Computer Society Press (2002) 22–28
[22] Graumann, D., Lara, W., Hightower, J., Borriello, G.: Real-world implementation of the location stack: The universal location framework. In: Proceedings of the 5th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2003), IEEE Computer Society Press (2003) 122–128
[23] Schulz, D., Fox, D., Hightower, J.: People tracking with anonymous and id-sensors using rao-blackwellised particle filters. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI), Morgan Kauffman (2003) 921–926
[24] Liao, L., Fox, D., Hightower, J., Kautz, H., Schulz, D.: Voronoi tracking: Location estimation using sparse and noisy sensor data. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ (2003) 723–728

[25] Gellersen, H.W., Schmidt, A., Beigl, M.: Multi-sensor context-awareness in mobile devices and smart artifacts. In: Mobile Networks and Applications. Volume 7., New York, NY, ACM Press (2002) 341–351
[26] Schilit, B., LaMarca, A., Borriello, G., Griswold, W., McDonald, D., Lazowska, E., Balachandran, A., Hong, J., Iverson, V.: Challenge: Ubiquitous location-aware computing and the place lab initiative. In: Proceedings of the First ACM International Workshop on Wireless Mobile Applications and Services on WLAN (WMASH). (2003)
[27] Kwok, C., Fox, D., Meilă, M.: Adaptive real-time particle filters for robot localization. In: Proceedings of the IEEE International Conference on Robotics & Automation (ICRA). Volume 2. (2003) 2836–2841