Tracking Moving Devices with the Cricket Location System*

Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Priyantha
MIT Computer Science and Artificial Intelligence Laboratory
The Stata Center, 32 Vassar Street
Cambridge, MA 02139
http://nms.csail.mit.edu/cricket/

ABSTRACT

We study the problem of tracking a moving device under two indoor location architectures: an *active mobile* architecture and a *passive mobile* architecture. In the former, the infrastructure has receivers at known locations, which estimate distances to a mobile device based on an active transmission from the device. In the latter, the infrastructure has active beacons that periodically transmit signals to a passively listening mobile device, which in turn estimates distances to the beacons. Because the active mobile architecture receives simultaneous distance estimates at multiple receivers from the mobile device, it is likely to perform better tracking than the passive mobile system in which the device obtains only one distance estimate at a time and may have moved between successive estimates. However, an passive mobile system scales better with the number of mobile devices and puts users in control of whether their whereabouts are tracked.

We answer the following question: How do the two architectures compare in tracking performance? We find that the active mobile architecture performs better at tracking, but that the passive mobile architecture has acceptable performance; moreover, we devise a hybrid approach that preserves the benefits of the passive mobile architecture while simultaneously providing the same performance as an active mobile system, suggesting a viable practical solution to the three goals of scalability, privacy, and tracking agility.

Categories and Subject Descriptors

C.3 [**Special-purpose and application-based systems**]: Real-time and embedded systems

General Terms

Design, Experimentation, Measurement

*This work was funded in part by NSF under grant number ITR ANI-0205445, and in part by Acer Inc., Delta Electronics Inc., HP Corp., NTT Inc., Nokia Research Center, and Philips Research under the MIT Project Oxygen partnership, and in part by Intel Corp.

MobiSYS'04, June 6–9, 2004, Boston, Massachusetts, USA.

Keywords

Mobility, location-awareness, pervasive computing, tracking, Cricket

1. INTRODUCTION

Determining the location of a device is a fundamental problem in mobile computing. The importance and promise of location-aware applications has led to the design and implementation of systems for providing location information, particularly in indoor and urban environments where the Global Positioning System (GPS) does not work well [1, 7, 13, 15]. In general, these systems provide more accurate location information when a mobile device is at rest than when it is in motion: tracking a moving device is harder because the inevitable errors that occur in the distance samples used to localize the device are easier to filter out if the device's position itself does not change during the averaging process.

This paper addresses the problem of tracking a moving device using the Cricket indoor location system. Our motivating applications include *human navigation*, where the goal is to direct users to their desired destinations on an active map, *robotic navigation*, where location sensors provide position information to a moving robot, and *multi-player games*, where players can move in the real world in a game like Doom or Quake, and have their moves accurately represented in the computer game. All of these applications require the position of a device moving at human speeds to be tracked.

The architecture of a location system influences its scalability, its ability to preserve user location privacy, its ease of deployment, and its device-tracking performance. We distinguish two different indoor location architectures. The *active mobile* architecture, as illustrated in Figure 1, has an active transmitter on each mobile device, which periodically broadcasts a message on a wireless channel (*e.g.*, an RF message or an RF message coupled with an ultrasonic pulse). Receivers deployed in the infrastructure (*e.g.*, on ceilings and walls) listen for such broadcasts and estimate the distance to the mobile on each broadcast they hear.[1] Typically, each receiver propagates this distance information to a central database that then updates the location of each mobile device. Examples of this architecture include the Active Badge [15], Active Bat [7], and Ubisense [3] systems.

In contrast, the *passive mobile* architecture, as illustrated in Figure 2, *inverts* the transmitter and receiver locations. Here, beacons deployed at known positions in the infrastructure periodically transmit their location (or identity) on a wireless channel, and passive

[1]Not all active mobile schemes use distance estimates; for example, the Active Badge system localizes nodes to with rooms using infrared.

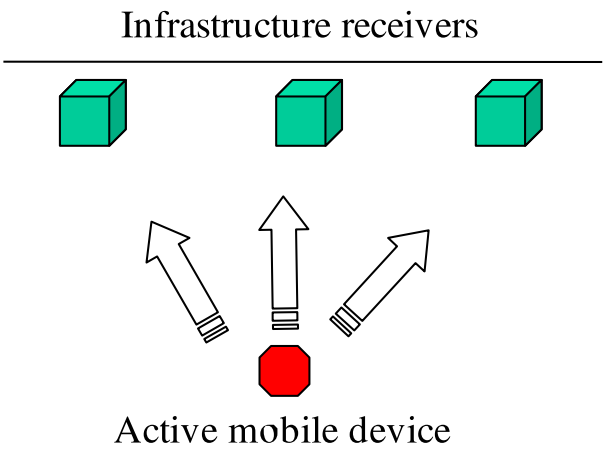

Figure 1: In an active mobile architecture, an active transmitter on each mobile device periodically broadcasts a message on a wireless channel.

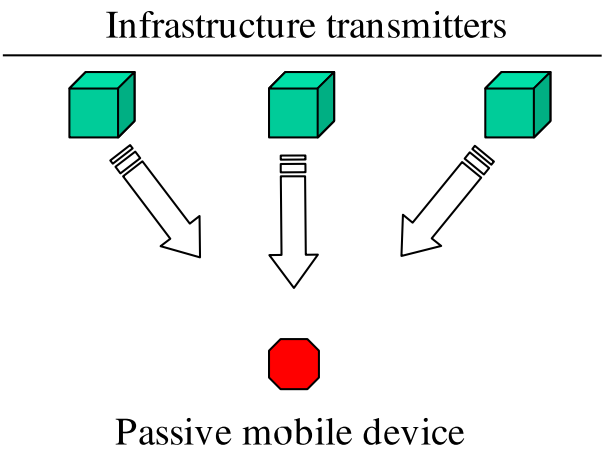

Figure 2: In a passive mobile architecture, fixed nodes at known positions periodically transmit their location (or identity) on a wireless channel, and passive receivers on mobile devices listen to each beacon.

receivers on mobile devices listen to each beacon. Each mobile device estimates its distance to every beacon it hears and uses the set of distances to estimate its position. An example of this architecture is the Cricket system [13].

Qualitatively, the passive mobile architecture scales better than the active mobile architecture as the density of devices increases, because the wireless (RF and ultrasonic) channel use is independent of the number of mobile devices. Unlike the passive mobile architecture, the active mobile architecture requires a network infrastructure to connect the deployed receivers to the central database. In addition, the active mobile architecture also allows users to be tracked more easily by the infrastructure, raising privacy concerns. In contrast, the passive mobile architecture allows a mobile device to estimate its location and control which other entities get that information.

However, the active mobile architecture solves the problem of tracking moving devices in a more natural fashion. This is because a receiver in a passive mobile system usually hears only one beacon at a time, and may move between chirps from different beacons. As a result, there is *no guaranteed simultaneity of distance samples*, unlike in the active mobile case where multiple receivers concurrently obtain distance estimates to a moving device. The absence of guaranteed simultaneity of distance estimates implies that tracking a moving object entails more than just a solution to simultaneous equations.

The natural question, then, is *How well can a passive mobile system perform at tracking a moving device?* Can we devise a method that enables the tracking performance of such a system to approach the performance of an active mobile system? If the answer to this question is "no", then it would suggest that applications requiring fast device tracking are better served with an active mobile system, but that comes at the cost of reduced scalability and increased privacy concerns. On the other hand, if the answer were "yes", then it would suggest that a passive mobile system is a tenable approach for a wide range of location-aware applications.

We show that the underlying tracking problem requires three components that are combined in different ways depending on the architecture. The first component is *outlier rejection*, wherein egregiously bad distance samples are eliminated. The second component is an *extended Kalman filter (EKF)*, which maintains the current and predicted device states and corrects the prediction each time a new distance sample is obtained. The third component is a *least-squares solver* (LSQ) that minimizes the mean-squared error of a set of simultaneous non-linear equations.

We find that the EKF is able to track movement much better than LSQ in the passive mobile system, and that it does just as well as LSQ in an active mobile system. Specifically, for speeds of up to about 0.8 m/s, an EKF model in a passive mobile system had a median error of about 10 cm, while an active mobile systems' median error was about 3 cm. At a higher speed of 1.43 m/s, the passive mobile EKF's median error was 23 cm, compared to 4 cm for the active mobile. For many applications, this error difference is unimportant.

Although the performance of the passive mobile system is acceptable for many applications, we also show how to improve it further. We improve tracking performance by developing a hybrid approach that runs the EKF in the common case and relies on an active mobile transmission when the EKF state is bad. We describe a protocol that allows the hybrid approach to not divulge device information, in an effort to alleviate privacy concerns. Our main result is that the hybrid system is nearly as accurate as our best active mobile system in tracking moving devices, while maintaining the advantages of the passive mobile system; its median error is 15 cm at a speed of 1.43 m/s.

We have implemented all the above schemes in the Cricket location system, and our measured results are in that system using a repeatable experimental setup that has both straight-line motion and radial acceleration. With the implementation of different tracking schemes, users of the Cricket system can take advantage of a variety of predictive tracking techniques for applications involving continual or unpredictable device motion.

2. RELATED WORK

The Active Bat location system is an example of a system which uses an active mobile architecture [7] . The Bat system consists of a collection of fixed nodes arranged on a grid. The fixed nodes receive ultrasonic chirps from the mobile device and compute distance estimates to the mobile using the time-of-flight of the ultra-

sonic signal. These distance samples are forwarded to a central computer which computes the mobile's position.

The Bat system employs a centralized architecture in which both mobile transmissions and mobile position estimations are handled by a central computer. The Bat system, as described, is expensive to implement in that it requires large installations, has a centralized structure, and does not preserve user privacy.

On the other hand, each of these expenses provides a direct benefit. The centralized structure allows for easy computation and implementation, since all distance estimates can be quickly shipped to a place where computational power is cheap. Moreover, the active mobile architecture facilitates the collection of multiple simultaneous distance samples at the fixed nodes, which can produce more accurate position estimates relative to a passive mobile architecture.

Some applications, such as virtual reality, require high precision tracking even in the presence of large and erratic accelerations. Two modern systems have been created for these applications. They provide very precise position estimates at the expense of infrastructure and hardware.

The HiBall head tracking system [16] uses panels of infrared LEDs that take turns flashing. Several head-mounted cameras measure the position of the flashing LED and the system uses knowledge about the geometry of the head device's cameras to compute the desired location information. The LEDs flash very quickly and thus allow very precise information to be obtained. Some of the disadvantages of this system include the difficulty of deploying a large number LED panels to cover an entire building, expensive camera hardware, high computation costs, and the possible interference from the ambient light. Nevertheless, this system provides very precise position information for specialized applications that operate in highly controlled environments. This system uses a technique called SCAAT (Single Constraint At A Time) to track movement, and is similar to our approach in that it handles one distance constraint at a time rather than obtaining multiple distance estimates to known positions simultaneously.

The Whisper system [14] uses a spread-spectrum audio approach to obtain precise distance measurements. It encodes information on an audio stream and uses time-of-arrival information to obtain distance estimates. The system can achieve high measurement rates because of the large bandwidth and the continuous nature of the spread spectrum signal. This, in turn, leads to good tracking performance. However, human-audible background noise, high computation costs, and low range are some of the disadvantages of this system.

The Global Positioning System (GPS) operates well outdoors and achieves many of the goals in positioning systems, like scalability, decentralized usage, and user privacy [6, 8]. The expensive infrastructure enables simultaneous distance estimates to be obtained, so that the tracking problem becomes easier to solve. The Kalman filter-based tracking methods presented in this paper share some similarities with the approach used in GPS.

Leonard *et al.* investigate underwater tracking of autonomous underwater vehicles [9]. Their work uses a Kalman filter, with access to more diverse observations than ours, including information from depth sensors, accelerometers, and compasses.

Another area of related work is in user tracking and user movement prediction in cellular wireless networks. In cellular phone networks, if the system can predict which cells will provide the best signal strength, the quality of service can be improved. One implementation, presented in [11], uses the so-called Robust Extended Kalman Filter (REKF), introduced in [12]. The REKF is concerned with maintaining good filter behavior in conditions with high uncertainty.

Another new class of algorithmic techniques for user tracking is given in [5], where the authors discuss Bayesian location estimation. One major difference between Kalman filtering and Bayesian estimation is that Bayesian filtering can accommodate non-Gaussian distributed measurement noise. This benefit comes at some computational complexity costs, however. Because Kalman filtering is a simpler special case of Bayesian estimation, there might be benefits to be gained from modeling our system more generally than in our current Kalman filter-based approach. Similarly, particle filters appear to be a promising method to handle the tracking problem. We leave an investigation of the performance of these approaches in Cricket to future work, noting that the current Cricket software and hardware infrastructure is well-equipped for researchers to investigate these questions.

3. TRACKING ALGORITHM

This section discusses the different components of our tracking algorithm and how they fit together. We show how the same basic algorithm works in both the passive mobile and the active mobile architectures, with appropriate parameter selection.

We start by formally defining the tracking problem. When a mobile device hears a beacon in the passive mobile architecture, or obtains a distance estimate to one or more receivers at known positions in the active mobile architecture, it gets a triple: $[t, p, d]$, where t is the current time, p is the known position of the beacon or receiver, and d the distance between the mobile device and the known beacon or receiver. Over time, the mobile device obtains a sequence of such triples: $[t_1, p_1, d_1], [t_2, p_2, d_2], \ldots, [t_n, p_n, d_n]$, where the subscripts increase by one for each distance sample. The goal of the tracking algorithm is to come up with $\hat{\phi}_n$, a good position estimate of the mobile device given the entire sequence of past triples. Of course, individual distance samples may be erroneous.

This problem formulation covers both the active and passive mobile architectures. In particular, if successive triples all have the same time t's and different p's , then the simultaneity condition is satisfied.

The flow chart of the Kalman filter tracking algorithm is shown in Figure 3. There are three procedures: a *least-squares minimization (LSQ)*, an *extended Kalman filter (EKF)*, and *outlier rejection*. The Kalman filter updates its estimate $\hat{\phi}_n$ of the device's position every time a new distance sample is acquired. It also maintains a confidence estimate in it state vector, which is used by the outlier rejection stage to reject distance samples that are "far away" from the expected value; the precise definition of "far away" depends on some heuristics, as discussed below. Finally, every once in a while, the EKF's state will be bad, and its confidence in its state low. When that happens, our tracking algorithm resets the EKF state by running LSQ on some number of current and recently observed distance samples.

We now explain the three modules in more detail.

3.1 Least Squares Minimization

If the mobile device were static, a standard way to solve the problem of estimating $\hat{\phi}_n$ is by minimizing the sum of the squares of the error terms corresponding to each distance sample. This method, called *least-squares minimization (LSQ)*, estimates $\hat{\phi}_n$ by minimizing

$$\sum_{i=1}^{n} (\|\hat{\phi}_i - p_i\| - d_i)^2 \quad (1)$$

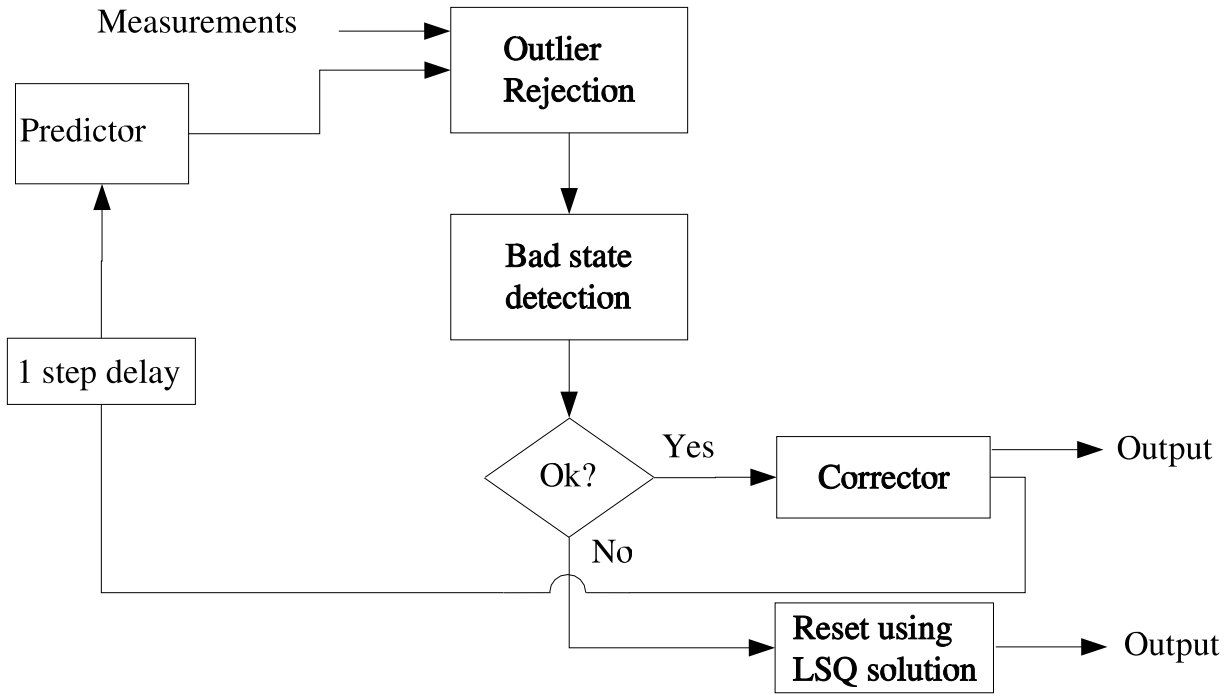

Figure 3: Flow chart for the tracking algorithm. The picture shows outlier rejection and LSQ explicitly; all the other steps are part of the extended Kalman filter. The "measurements" are the distance samples obtained as triples. The "output" at each stage is the position estimate $\hat{\phi}$.

Here, $\|\hat{\phi}_i - p_i\|$ is the Euclidean distance between the estimated coordinate of the mobile device and the beacon or receiver at position p_i.

LSQ does not always produce a good $\hat{\phi}_n$ estimate for several reasons, including:

1. If the device is moving, old tuples need to be discarded before LSQ is run. In the active mobile architecture, one might discard all samples but the ones corresponding to the latest time, and hope that there are enough samples with low distance error to produce a good position estimate for the moving device. In the passive mobile case, it is not obvious which samples, if any, to discard.

2. The simultaneity condition may not always hold for a moving device, either because the system is based on a passive mobile architecture, or because there are not enough distinct error-free samples (less than four in three dimensions) in the active mobile case (*e.g.*, because the user is in an area with ≤ 3 ceiling-mounted receivers, or because some of the receivers did not report good distance estimates).

3. The LSQ approach does not explicitly model noise terms in the distance samples.

LSQ is also computationally complex. It has been the topic of much analysis in the optimization literature, and is known to take long to converge unless we impose certain strict criteria on the function we're minimizing or on the nature of the inputs to the optimization.

These shortcomings of LSQ are well known; it is for these reasons that GPS receivers use a Kalman filter rather than LSQ. (Another reason is that the use of a Kalman filter allows a GPS receiver to predict a good current position based on previous measurements.) We also adopt a Kalman filter-based approach, modeling the system using a state vector, for the same set of reasons. However, LSQ is useful in initializing and resetting the Kalman filter, a capability we use when the system first turns on or when our filter gets into a bad state. We can think of LSQ as a way to "brute-force" the position of the mobile and the Kalman filter as a way to intelligently track the user from there.

In the least squares model we have two possibilities when distance measurements are obtained: either the problem is well-defined or it is not. In 3-D space the location problem is usually well-defined if we have four or more distances. Because our least squares minimization tools are sensitive to local minima as a function of our initial guess, x_0, it is important that we produce reasonable starting points. If our problem is well-defined we can produce a good initial guess by solving a linearized version of the problem [10]. We then provide this $\mathbf{X}_0$ along with our measurements to an interior trust-region least squares minimizer [4, 4]. On the other hand, if our problem is not well-defined we use a "line search method" with $\mathbf{X}_0$ as our last guess at our position.

Although the details of the procedures presented in this section are almost completely independent of the architecture in which they are implemented (*i.e.*, active mobile or passive mobile), there is a minor difference in our least squares calculator. In least squares under the passive mobile architecture we do not get a collection of *simultaneous* distance estimates so we maintain a buffer in which we store the last m measurements. The contents of this buffer are then processed in the same way that the least squares normally proceeds. That is, in the passive mobile case, we make the *simultaneity assumption* and assume that the distances in our buffer were collected when the device was at the same point. That assumption will

not hold in general; to reduce the chances of the assumption being invalid, we make the size of the buffer inversely proportional to the expected speed of the device.[2]

3.2 Extended Kalman Filter

We design an EKF using a state vector with six components, three position components (x, y, z) and three velocity components (v_x, v_y, v_z). The basic idea of the EKF is that after any discrete time step the filter has an idea of its state and an idea of how confident it is in that state (a *covariance matrix*). The EKF uses the most recent distance sample and its internal state to *project ahead* and produce an estimate of $\hat{\phi}$ of where the device might be in the next time-step. When the next distance sample arrives, the EKF first *corrects* its internal state based on the difference between where the device would have been had the prediction been accurate, and the actual distance sample. These steps are shown in Figure 3.

Let $\mathbf{X} = [x\ y\ z\ v_x\ v_y\ v_z]^{\mathrm{T}}$ be the state vector maintained by the EKF. If the predicted state at time-step k is $\mathbf{X}_k^{(-)}$ and the corrected state (the output of the EKF) is $\mathbf{X}_k^{(+)}$, then in the *position-velocity (PV)* filter model, which assumes that the device moves at constant velocity between time-steps, the state prediction at time ΔT after time-step k for each position component of $\mathbf{X}_k^{(-)}$ is

$$x_k^{(-)} = x_{k-1}^{(+)} + v_x\ \Delta T \tag{2}$$

$$y_k^{(-)} = y_{k-1}^{(+)} + v_y\ \Delta T \tag{3}$$

$$z_k^{(-)} = z_{k-1}^{(+)} + v_z\ \Delta T, \tag{4}$$

where the plus superscripts and $k-1$ subscripts on the velocity components are omitted for clarity.

Now, let $\mathbf{P}$ be a 6×6 covariance matrix for the EKF's state vector (we assume that errors are normally distributed). Then, the predicted covariance for time-step k is

$$\mathbf{P}_k^{(-)} = \mathbf{\Phi}\mathbf{P}_{k-1}^{(+)}\mathbf{\Phi} + \mathbf{Q}_{k-1}, \tag{5}$$

where $\mathbf{\Phi}$ is a state transition matrix specific to our model and $\mathbf{Q}_{k-1}$ reflects how we expect the quality of our state vector to degrade over time (*e.g.*, if the mobile is moving faster we expect our state to degrade more quickly).

In the correction step the EKF finds the difference between the distance it expected to hear (based on the projected state) and the measured distance sample. The EKF then adjusts the state vector to make this difference less significant. The idea is to balance the confidence in the measurement against the confidence in the state estimate to determine the amount by which the state estimate should change. An overview of the underlying mathematics is given in Appendix A.

The prediction-correction loop described above continually runs as the system obtains distance samples.

The *position (P)* filter model is exactly the same as the PV model, except that it only maintains the position of the user (x, y, z). Whereas in the PV model we assume that acceleration and higher order derivatives are zero, in the P model we assume that velocity and higher-order derivatives are zero. A P model filter is less computationally expensive and has some distinct accuracy advantages in certain situations, which we will see in our experiments (Section 5).

Finally, we introduce the *multi-modal filter*, a way of combining the output states of our PV and P models to produce a better solution. The idea here is simple. Since we have two states and their corresponding covariances, by continuing our normally-distributed assumption we can quickly come up with a way that averages our two states, weighting them by their covariances. This idea has significant promise, especially because it works across dimensions, such that if one of the filters has a better covariance only along, say, the z-axis, the multi-modal filter will weigh its z-coordinate output more heavily than the others. Multi-modal filters tend to be more accurate and track movement well, as we will show in 5.

[2] Of course, in many cases it may be impossible to know the speed *a priori*.

3.3 Outlier Rejection

The third module in our tracking algorithm implements outlier rejection. Since the EKF provides an estimate of the current position based on its state estimate, the system can compute a guess of the value of any distance measurement from a beacon or to a receiver in the infrastructure. The difference between this guess and the actual measurement defines a *residual*, r, of each measurement. If $r^2 > \gamma$, an empirically-selected parameter, then we say that the measurement is an *outlier*.

A subtle but important point lurks in this definition of an outlier. Since the residual is computed based on the EKF's state, if it gets into a bad state then we expect the residuals of the *accurate* measurements to be high. At this point, the system begins to reject all of the samples and the EKF's state becomes progressively worse because the system is no longer accepting any measurements that can help correct the state! The traditional solution to this problem is to modify our outlier rejection to test for outliers based on the modified formula $r^2 S^{-1} > \gamma$, where S^{-1} is a scalar computed based on our state which we expect reflects the confidence in our state. This formula computes what is commonly known as the Mahalanobis distance [2]. We chose not to implement this scheme because it depends on the confidence in the state vector, on which we have no guarantees of proper behavior. It also did not perform well in our empirical measurements.

The solution to the false sample rejection problem that we chose to implement is to monitor the fraction of rejections made by the system over some time window. If this fraction significantly exceeds the fraction of outliers we expect to receive, we then declare that the EKF is in a bad state. At that stage, we look to our least squares model for possible recourse. That is, we compute the residual of the LSQ output with respect to the most recent measurement. If this residual, r_{lsq} is less than the EKF's residual, r_{ekf}, then we reset the filter with the least squares estimate as our position. If the least squares residual is higher, then we do nothing, in which case we expect this testing to continue in future time-steps.

4. HYBRID ARCHITECTURE

Occasionally the Kalman filter will get into a bad state. This situation rarely happens in the active mobile approach if the mobile can hear three or more beacons, but may happen when it does not. The Kalman filter in a passive mobile system has a higher probability of reaching a bad state since the distance measurements are serialized in time. Once we detect a bad state, the listener uses a mechanism to reset its Kalman filter to a known listener position.

In an active mobile approach, the listener can easily compute a position estimate using simultaneous distance samples to multiple beacons. In contrast, in a passive mobile approach, the listener feeds the non-simultaneous distance samples to an LSQ estimator to compute a position estimate; the LSQ estimation is computationally complex and is subject to large errors (we show this in Section 5) because the LSQ estimation works with distance samples that may have been obtained when the device was in a different position.

As a solution to this inherent difficulty in reseting the Kalman

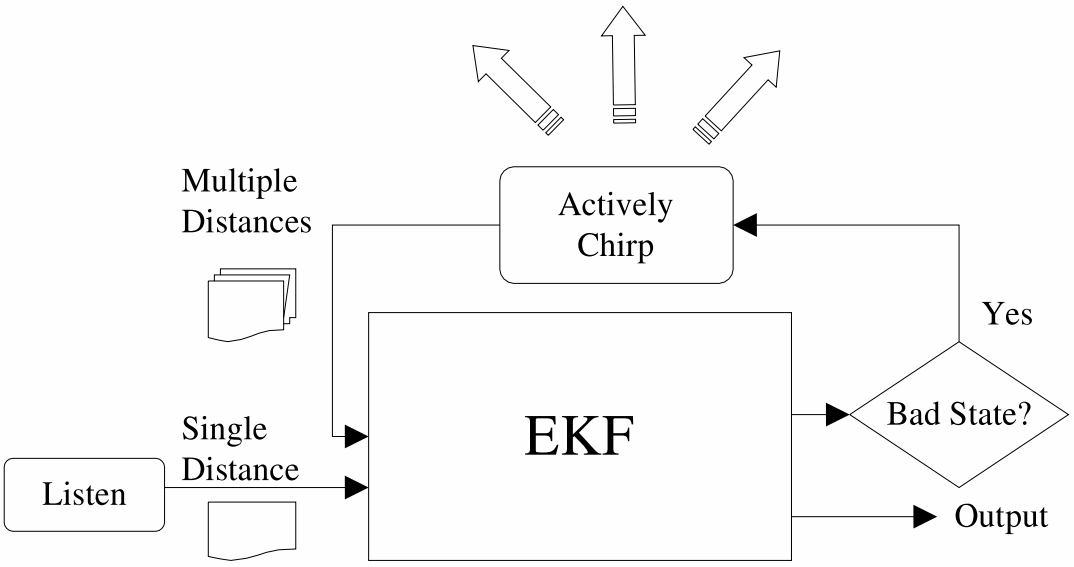

Figure 4: Flow chart of a mobile device using a hybrid architecture.

filter in a passive mobile system, we propose the following hybrid solution the tracking problem. During the normal operation, we use the passive mobile system due to its scalability and guaranteed user-privacy. At start-up time, and when we detect a bad Kalman filter state, the listener transitions to active mobile operation to obtain multiple simultaneous beacon distance samples as shown in Figure 4.

We develop the following method to transition between passive and active mode on the mobile device:

1. As long as the Kalman filter's confidence is high, the listener does not transmit any information; only beacons do.
2. If the listener's Kalman filter state is deemed bad, then it becomes an active transmitter. This transition is usually required if the device experiences sudden linear acceleration or a turn. The listener then generates a concurrent RF and US pulse, with the RF message having no information in it other than a randomly generated nonce.
3. If a beacon hears an RF message generated by a mobile and the corresponding US pulse, it waits for a short random period of time and broadcasts the nonce (set by the mobile) together with the distance estimate. During the broadcast, the beacons use a simple CSMA scheme with randomized backoff to avoid RF collisions. After receiving this information from nearby beacons, the listener can compute its position accurately since the simultaneity condition holds for these distance samples. Next, the listener uses this position estimate to reset its Kalman filter.

We can use either of the following approaches to enable a passive listener to transition to an active mode. One approach is to set aside a unique time slot for the active listener to transmit. This approach requires time synchronization among the beacons and the listeners, so that all the participants have a unified view of the time slot allocation. The other approach is to use a simple CSMA scheme in which the mobile competes with the beacons for transmitting the message requesting ranging information. One disadvantage of this scheme is that beacons need to continuously listen to the RF channel for possible mobile transmissions. We implemented the second approach due to its simplicity.

In implementing this hybrid approach, the designer can choose how frequently the mobile devices are likely to chirp by selecting an appropriate bad state detection threshold. That is, since the hybrid system would typically contain a mixture of both active and passive listeners at any given time, an appropriate balance between the two modes can be achieved with some tuning.

We now look at three important architectural properties and discuss how well the hybrid approach performs in each case.

4.1 Scalability

The transition to an active transmission state by the mobile device happens only when the Kalman filter's state is bad, which means that the system as a whole is likely to remain scalable unless a large number of mobile devices in the same neighborhood simultaneously have a bad filter state. Even if such an unfortunate situation were to arise in practice, it is possible to design a scheme to throttle the mobile's bad state threshold reporting based on channel occupancy.

It would even be possible to pick between the active mobile and hybrid architectures in real-time based on the number of other mobiles nearby. In this scenario we assume privacy isn't important, so that if there is only one mobile in the space then this mobile actively chirps. If there are two mobiles then these mobiles take turns chirping. When the number of mobiles exceeds the number of beacons, the mobiles start to actively chirp only if they get into a very bad state. Even then, the definition of "very bad state" could be a function of overall demand. If necessary, a priority scheme could be implemented that gives certain mobiles preferential access to the RF and ultrasonic channels.

4.2 Privacy

The use of a random nonce every time the mobile transmits helps to hide the mobile's identity. When the mobile device actively transmits, an eavesdropper would be able to determine that there is some mobile unit at the given position. This does not reveal the position of a given user provided there are many users in a given

region. However if there is only one user in a single space and the system knows that there is only one user, then she might be easily tracked.

The following conditions appear to be sufficient in practice for maintaining a reasonable degree of location privacy.

1. The space is densely populated *or* has high levels of traffic.
2. Even when there is only a single user, a promiscuous listener must have knowledge about the number of users in the space, since it would have to know that all of the position estimates correlate to a single device.
3. In the hybrid architecture, the percentage of active mobile transmissions is relatively low (as explained later in Section 5, especially in Figure 13), making it difficult to track the mobile continuously.

These conditions make the random nonce approach very reliable for preserving mobile privacy. Our results in the next section show that the fraction of times that a mobile must actively chirp is very small (about 1.5%) to achieve a performance close to the active mobile scheme.

Finally, note that if the mobile device is very concerned about privacy it can opt not to fall back on the active transmission mechanism, and always use LSQ instead.

4.3 Decentralization

Our protocol preserves the decentralized nature of the passive mobile architecture, so we don't have to run a cumbersome wired or wireless network infrastructure connecting the ceiling-mounted receivers to a central location to compute mobile device position estimates. Whenever a mobile actively chirps each beacon that hears this chirp sends the measured distance back to the mobile device over the RF channel (no ultrasound transmission is required). All position calculations are done on the mobile node. Not only does this significantly reduce infrastructure costs, it also solves the problem of correlating and aggregating distance estimates from different beacons.

Figure 5: Picture of the experimental setup. The beacons are shown on the ceiling; the cables emanating from the beacons are used in a subset of the active mobile experiments to report distance samples observed by infrastructure nodes.

5. EVALUATION

In this section we present the results of several experiments to evaluate the performance of different location architectures and tracking techniques. These experiments shed light on the strengths and weaknesses of the different methods. During our discussions in this section, unless stated otherwise, when we refer to a Kalman filter we are talking about the multi-modal extended Kalman filter that implements both the PV and P models and averages their outputs weighted by their covariances.

We first describe our experimental setup, followed by a discussion of the tracking performance of the three architectures discussed earlier: passive mobile, active mobile, and hybrid. We then compare the PV and P models in the EKF, showing how their relative performance differs when the device accelerates (*e.g.*, turns on the tracks). We then discuss how to scale the performance evaluation to determine what would happen at higher device movement speeds by down sampling the observed distance measurements. We conclude this section with a discussion of the computational complexity of the various schemes.

5.1 Metrics and Setup

We began testing and developing our tracking techniques using a simulator. This approach worked well while trying various ideas because we could directly measure every cause and effect in that environment. However, because the performance of the different schemes depends strongly on the nature of the erroneous distances observed in practice, we needed a real-world experimental testbed. We developed one using Cricket's hardware and software infrastructure.

Developing a testbed to facilitate accurate and repeatable tracking experiments proved more difficult than we expected. The problem was that we wanted an apparatus that would capture real-world noise, signal loss, and reflections, but at the same time permit some degree of experimental consistency so that we could run multiple runs comparing different approaches. We also needed to select an apparatus which would allow us to simulate "typical" human movement, including turns, starts, and stops, all at different speeds.

We decided to use a computer-controlled Lego train set placed in a large room, with Cricket attached to the moving train. A picture of this setup is shown in Figure 5, and the trajectory of the train in schematic form with distances is shown in Figure 6. The schematic figure was obtained by placing a listener at a number of positions on the track and calibrating its position while at rest over a long period of time at each position.

To conduct our mobile tracking experiments, we attached a Cricket listener to the train and Cricket beacons to the ceiling.[3] We calculated the beacons' positions off-line using a combination of manual and mobile-assisted measurements. We report on the results of experiments conducted at six different speeds: 0.34 m/s, 0.56 m/s, 0.78 m/s, 0.98 m/s, 1.21 m/s, and 1.43 m/s. These speeds model a range of realistic pedestrian speeds. This experimental

[3]As explained later, this setup allows us to investigate all three location architectures, not just the passive mobile one.

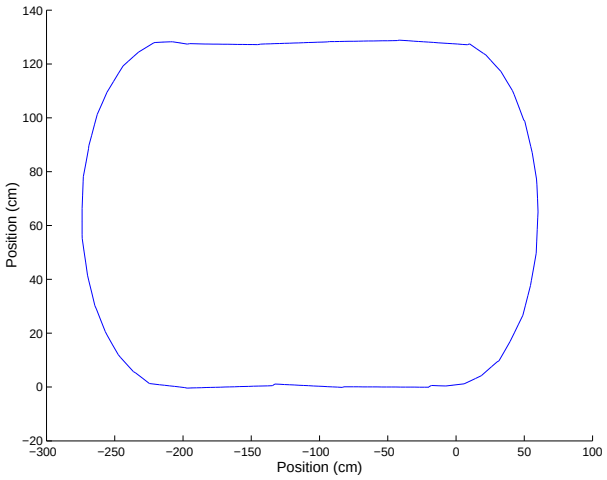

Figure 6: Schematic representation of the train's trajectory.

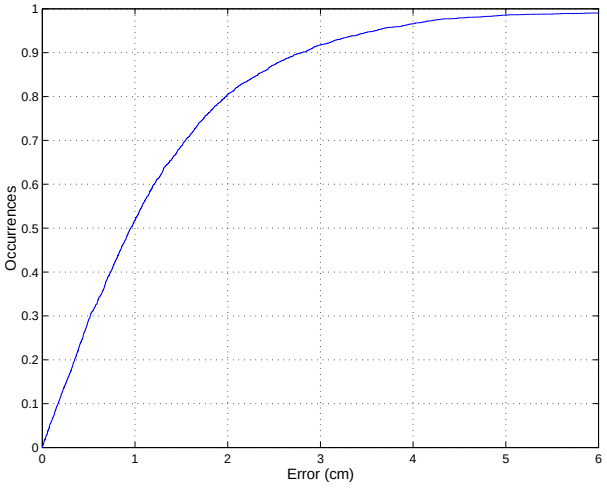

Figure 7: True positioning error, comparing distance measured to distance expected based on track counter.

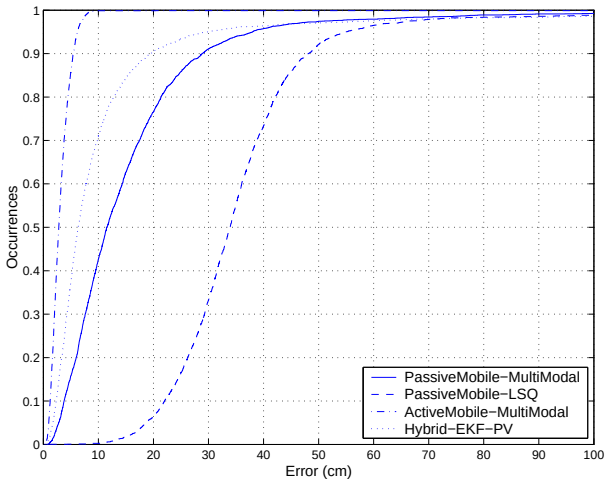

Figure 8: Error CDF of the different architectures with the device moving on the tracks at 0.78 m/s.

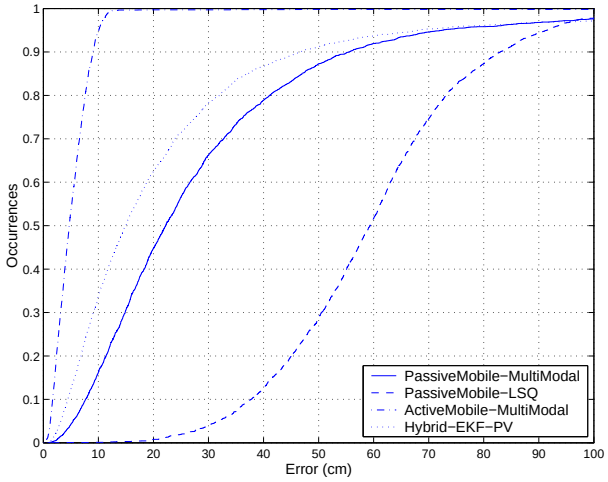

Figure 9: Error CDF of the different architectures with the device moving on the tracks at 1.43 m/s.

setup included a number of real-world effects, including multiple beacons (five or six in all experiments) interacting with one another, varying distances from the different beacons to the listener, and ultrasonic noise and reflections. For each architecture and each speed, we gathered data samples over a five-minute interval. We gathered about 15,000 individual distance estimates in the active mobile architecture and about 3,000 distance estimates in the passive mobile architecture.

All of the error values presented in this section are relative to the train's real position, which we recorded using an optical track counter mounted at the bottom of the train. The track counter counted the number of tracks traversed during its motion. We expect this counter to be accurate to 1 cm, the distance between track edges. However, because of inaccuracies in the beacon coordinate assignments and in the mapping of the optical counter values to coordinates, we expect our true position error to be larger than one centimeter some of the time. The true positioning error during movement is shown in Figure 7, where we show the difference between distances measured by the Cricket system, and the distance we expected based on the track counter. Note that the error numbers shown on this curve are different from the raw Cricket distance error (*i.e.*, relative to a laser range finder measurement when the listener is not moving); the Cricket hardware's "raw" error is usually 1 cm and almost never more than 3 cm. The figure shown embodies three sources of error: the raw Cricket distance error, the error inherent in our pre-programmed beacon coordinates, and the error in our table matching track counter values to coordinates.

5.2 Tracking Performance

We now investigate the tracking performance of the three architectures and compare them.

5.2.1 Passive Mobile Architecture

In the passive mobile architecture, every beacon in the room chirps periodically, such that we end up with a time-series of non-simultaneous distance estimates at the mobile device. The bottom two curves in Figure 8 show the error CDF for our multi-modal

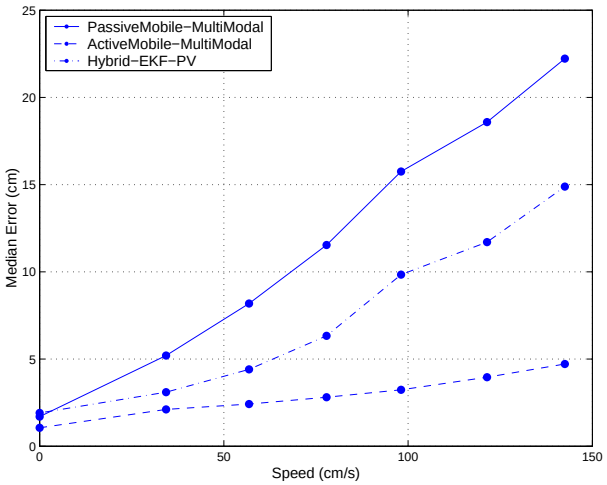

Figure 10: Median error in the passive mobile, active mobile, and hybrid architectures versus the device's speed.

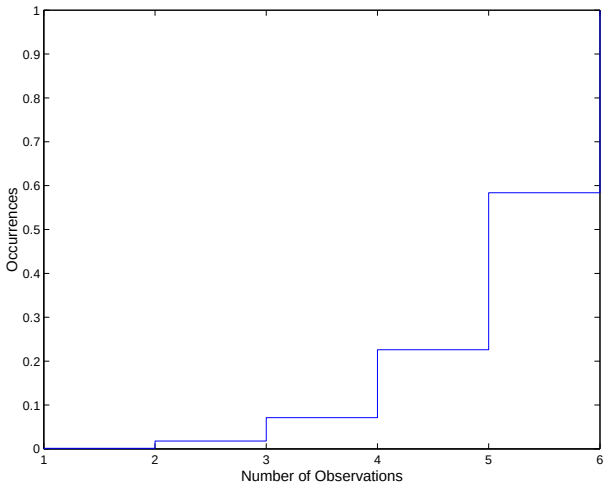

Figure 11: CDF of the number of replies from infrastructure receivers per active mobile chirp, using an RF channel for reporting distances. There were a total of six infrastructure nodes in the experiment.

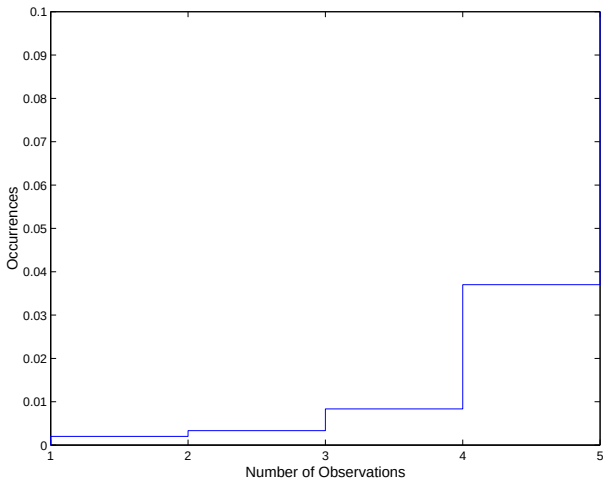

Figure 12: CDF of the number of replies from infrastructure receivers per active mobile chirp, using cabled channels for reporting distances. Notice the scale; all five infrastructure nodes in the experiment reply to more than 96% of the active mobile chirps.

EKF and least squares algorithms at a speed of 0.78 m/s while traveling on the path shown in Figure 6. Here, the multi-modal filter performs well; the 90^{th}-percentile error is an acceptable 0.3 m (note that the tracks are 2.5 m long and 1.2 m wide, with turns), whereas the least squares only achieves this level of precision only 30% of the time. The poor least squares performance is the result of the simultaneity assumption not holding, as discussed in Section 3.

The bottom two curves in Figure 9 show the results of the same passive mobile experiment conducted at the higher movement speed of 1.43 m/s. Here, the multi-modal filter maintains reasonable performance, while least squares performs much worse since the simultaneity assumption becomes increasingly invalid with increasing mobile speed. The top curve in Figure 10 shows the increase in median error of a multi-modal filter with increasing speed.

The simultaneity assumption holds, however, when the train is static. In this case the least squares error is small, but still not as good as the Kalman filter. The reason for this (slight) difference in quality is that the least squares procedure has a fixed window size and can thus be thought of as a "finite impulse response" (FIR) filter. Thus, even if we give least squares an infinite number of normally distributed samples it may never converge to the exact location because that approach discards expired samples. The Kalman filter, on the other hand, is an "Infinite Impulse Response" (IIR), so as larger numbers of samples come in, the error converges to zero.

5.2.2 Active Mobile Architecture

In the active mobile architecture, the mobile device actively chirps, and the fixed infrastructure nodes then reply either over a radio channel or a cabled infrastructure, reporting the measured distances to the mobile device or some central processor.

An important observation to make before looking at results is that throughout our experiments with this architecture *there was only one listener*, and thus no contention for the ultrasound channel. In any implementation with multiple mobile devices we expect the error of a tracking technique involving active mobile transmissions to increase as the number of mobiles increases. In this sense, the results presented for the active mobile approach are quite optimistic.

The CDF of the number of distinct ceiling-mounted receivers whose distance reports were heard by the mobile device per active chirp, in a system using a radio channel to communicate these distances from the infrastructure to the mobile device, is shown in Figure 11. The total number of ceiling-mounted receivers in this system was six. The number of responses does not depend on the device's speed. The primary limiting factor was the radio, which operated at 38.4 Kbits/s; because of the large aggregate number of messages that had to be sent from the infrastructure nodes, the preamble overhead of every radio message proved to be the bottleneck. We expect this limitation to subside in the coming years as radios for embedded devices become faster.

The same metric, for a system in which a cabled infrastructure is used to report distances, is shown in Figure 12. Here we see the expected result: if the radio bottleneck is removed, a much higher

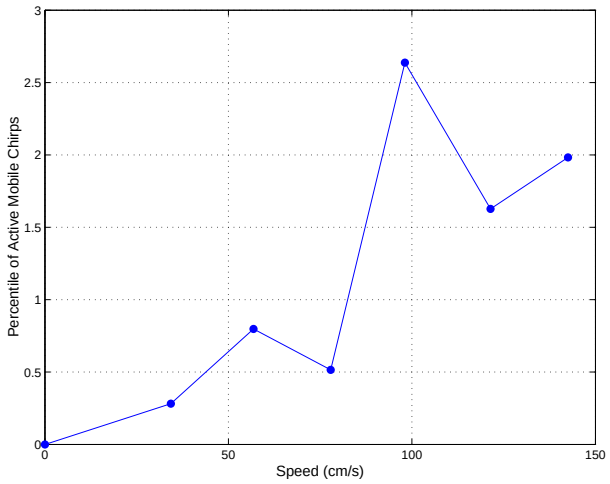

Figure 13: Frequency of mobile chirps at different speeds in the hybrid architecture, given as percentiles.

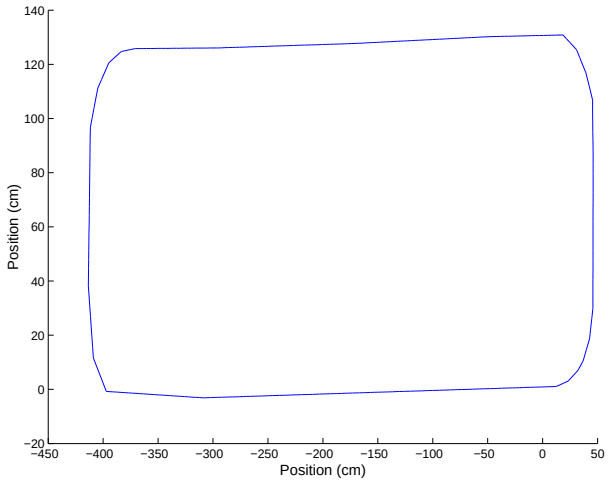

Figure 14: Apparatus B: Schematic representation of the train's trajectory.

percentage of distances are recorded. All results presented are from the more optimistic cabled system, which may be more expensive or cumbersome to deploy in practice on a large scale.

The top-most curves in Figures 8 and 9 show the error CDF of our EKF scheme in this architecture at a speed of 0.78 m/s and 1.43 m/s respectively. In both cases, the median error is less than 5 cm, comparable to the base error of our experimental setup. We do not show the results for the least squares and the multi-modal models under the active mobile architecture since the curves are indistinguishable; because the simultaneity condition is satisfied under the active mobile architecture, least squares is a viable player even at higher speeds.

5.2.3 Hybrid Architecture

We now look at the error profile of our system in an environment that allows the Kalman filter to obtain simultaneous distance estimates when they are the most useful (*i.e.*, when the filter is in a bad state).

Least squares here is not meaningful because we expect it to behave the same as it did under the passive mobile architecture, except for the (small) fraction of the time that it has a little more information. The point is that least squares already uses the simultaneity assumption, so filling its buffer half-way with simultaneous information a small fraction of the time does not have a significant impact on the output error.

Figures 8 and 9 show the error CDF graphs for the multi-modal Kalman filter under two different speeds (the relevant curves are the second from the top in both figures). This architecture shows good behavior; even at the highest movement speed of 1.43 m/s, the median error is a tolerable 15 cm.

The improvement in error (both median and tail behavior) over the passive mobile scheme comes with little cost. Figure 13 illustrates the frequency of active mobile chirps as a function of the train speed. A linear relationship between train speed and the frequency of active mobile chirps is highly desirable, since it means that we are getting this helpful (but costly) data more as error increases. Ideally this could make the error constant as our speed increases, which is an effect we come close to (considering the scale), as seen in Figure 10. The main conclusion from this graph is that the number of active chirps from a mobile device is a small fraction of the number of beacon transmissions—it is never more than 3% and at medium speeds is only 10 in 1000, while achieving a tracking error close to that of the active mobile system. Quantitatively, at the highest speed we get a 59% increase in accuracy over the passive mobile system, relative to the active mobile system. This is achieved with a 2.2% increase in the number of distance estimates used.

We arrived at the above numbers using the following calculation. At the highest speed, we were in a bad state during 62 out of the 3,506 periods. We averaged 3.7 distance measurements per active mobile chirp, and thus consumed a total of 296 extra measurements. In the active mobile case, at this speed, we averaged 4.9 measurements per period, for a total of 17,178 measurements. The passive mobile architecture consumed one measurement per period; 3,506 distances in total. So the total percent increase in distance measurements used by the hybrid architecture over the passive mobile architecture, relative to the active mobile architecture, is $296/(17178-3506)$, or 2.2%. The median error of the passive architecture EKF at this speed was 22 cm. In the active mobile architecture it was 4.7 cm, and in the hybrid architecture it was 14.9. Thus, our increase in precision is $(14.9-4.7)/(22-4.7)$, or 59%.

5.2.4 Tracking Performance Summary

The hybrid architecture turns out to perform well, incurring low overhead, because we only take on the cost of the active mobile system when the payoff will be large. As seen in Figure 9, we come significantly closer to active mobile performance by only using active mobile information only 2% percent of the time.

In Figure 8 we examine a similar graph but at lower speeds. Even though the mobile chirps half as often the hybrid filter shows better accuracy. Overall error improves as expected, but the hybrid architecture approaches the performance of the active mobile faster than the passive mobile does.

5.3 Comparing the P and PV EKF Models

In this section we will refer to a second experimental setup to illustrate some concepts. We will call the experimental setup we have seen thus far "apparatus A," and the new experimental setup "apparatus B." The track layout for apparatus B is shown in Figure 14.

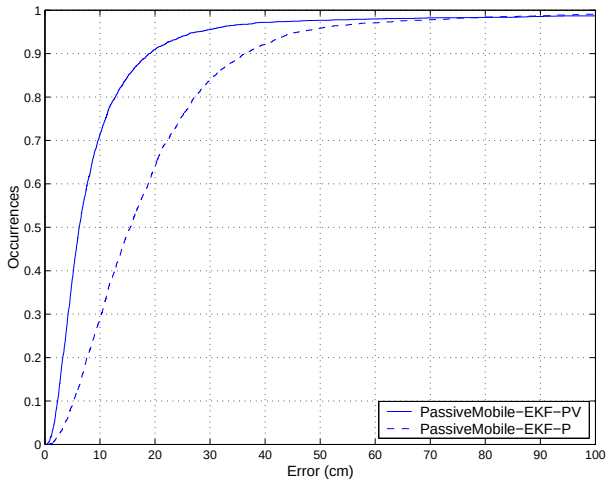

Figure 15: Apparatus A: Large-error behavior of P-model as compared to PV-model.

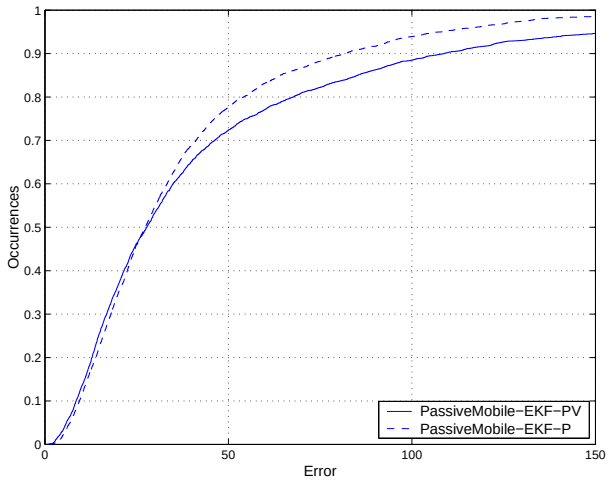

Figure 16: Apparatus B: Large-error behavior of P-model as compared to PV-model.

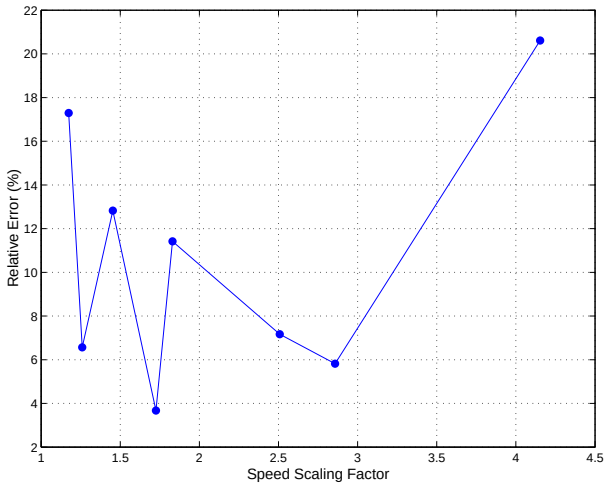

Figure 17: Percentage of difference between scaled simulations and real data at the same speed.

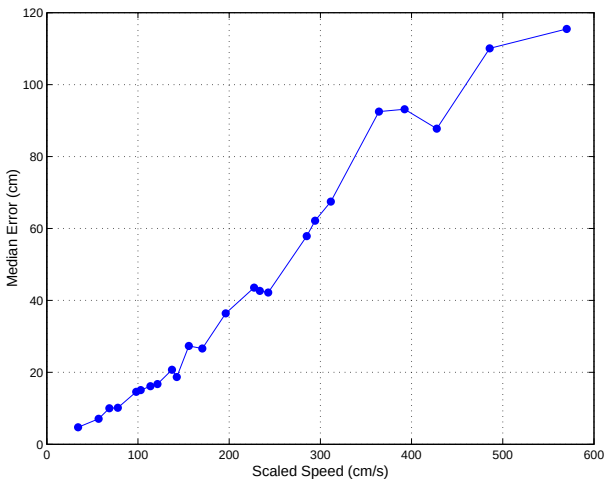

Figure 18: The positioning error of various data sets scaled to different speeds.

A general trend from our experiments is that the P-model has better *large-error* behavior than a PV-model does, even when the user is moving. This is shown in the extreme top-right corner of Figure 15, but is more apparent under apparatus B, as shown in Figure 16. This effect is best explained by the turns in our track. Because the P-model assumes that the user's velocity is zero and we know this isn't true, we tune our P-model EKF to degrade the quality of its state vector quickly (*i.e.*weigh incoming measurements more heavily than its projected state). Therefore the P-model EKF performs just as well during turns as during straight segments. On the other hand, we tune the PV model to have a higher confidence in its state vector since it has fewer assumptions. (I.e. only that acceleration and higher order derivatives are zero, as opposed to velocity, acceleration, etc. in the P-model.) Because the projected state vector in the PV-model performs poorly when acceleration levels are high, the PV model performs slightly worse than the P-model during the extreme points of turns. This explanation also tells us why the effect is more pronounced under apparatus B, since here the turns are sharper, and thus acceleration levels are higher than in apparatus A.

5.4 Scaling To Higher Speeds

We begin this section with a hypothesis. We propose that we can process down-sampled experimental data to effectively emulate higher train speeds. For example, if we have data collected while a mobile was moving at 1 m/s, we can make a good approximation to the errors we could expect at 2 m/s by only giving our filters every other distance measurement.

As illustrated in Figure 17, it seems that this hypothesis is reasonable. The shape and highly sporadic nature of this plot suggest that we cannot make any meaningful mathematical generalizations about its nature, but the fact that the largest relative error observed is less than 20% suggests that for any scaling factor of less than four in speed, this approach will be reasonably accurate.

Now, we take each set of data collected at the six speeds given in Section 5.1, and scale it up by factors of one, two, three, and

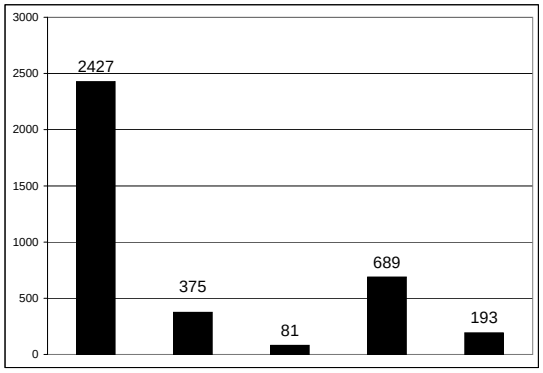

Figure 19: The number of multiply operations performed in one time step of different tracking algorithms. From left to right, the bars represent: least squares, passive mobile PV model, passive mobile P model, active mobile PV model, and active mobile P model.

four. We then run this new data (twenty-four sets in all) through our filters, which produces the trend shown in Figure 18. This data was from a passive mobile architecture and a multi-modal EKF.

In Figure 18, the points corresponding to the different speeds are roughly along a line, which suggests that the emulation of higher speeds is correct. We arrive at this conclusion by noticing that the median error linearly increases with speed in Figure 10. Moreover, as Figure 17 shows, for the speeds where we have measured data available, the emulation of scaling performance does not grow dramatically with speed.

It is important to note that the EKF filters during all of our experiments were tuned with the same parameters; *i.e.*, the parameters that govern how covariances are projected across iterations were all the same. The optimal parameter setting depends on the device's speed because as the speed increases, the assumptions of our filters become progressively worse. However, we chose to make them a constant for two reasons. First, determining how the parameters should change with speed is not clear. Second, it is not desirable to incorporate such a feedback loop into the system; if the parameters are tuned based on the state of the EKF, bad things could happen if the state degrades.

Therefore, as the speed increases in the passive mobile architecture, all of the incoming distances begin to look like outliers, and we saturate the rate of invocations of the LSQ method. We can expect that as the speed increases, our EKF behaves more and more like an LSQ filter, whose performance also severely degrades with speed as shown in Figure 9.

In contrast, recall that in the hybrid architecture, whenever the filter reaches a bad state, the device performs an active mobile chirp. In this case, as the speed increases the system would saturate the rate of fallbacks onto this active mobile chirp. That is, it would behave just as an active mobile architecture. This behavior sets an upper bound on the positioning error of a system implementing a hybrid architecture at high speeds: the hybrid architecture can always perform quite well.

5.5 Computational Complexity

A graph showing the number of multiplication and exponentiation operations for a large subset of our algorithms is given in Figure 19. The least squares bar can be generalized for all implementations since it always minimizes from some set of distances, after being buffered in the case of the passive mobile architecture. We also do not evaluate the computational performance of our multi-modal filter since it will be close to the sum of its PV and P models.

We make a few important observations from this data. The first is that our P model is about four times faster than the PV model, even though it only has half of the number of elements in its state vector. Most of the computational resources consumed by Kalman filters are used in the covariance prediction phase, since if the state vector has n elements then this calculation involves two $n \times n$ matrix multiplications. So we see that the number of operations needed in a naive Kalman filter grows as $O(n^3)$. However, because of the sparse nature of some of the computations involved, we were able to pull the increase in computational complexity of the PV model down to about $O(n^2)$.

Second, we see that even though the active mobile models have about five to six times the amount of data to process (there were either five or six beacons during these experiments) their computational complexity is only about double that of the passive mobile models. This relatively low increase in computation time is because these distance are measured simultaneously, and thus can be incorporated into a single filter time-step. Therefore covariance prediction only has to occur once.

It is obvious from the data shown that the least squares module is computationally complex. There is not much that can be done about this unless we sacrifice some accuracy. Finally, we note that we optimized our Kalman filter procedures to take advantage of the sparse nature of the covariance matrix so that multiplications in which one of the factors is zero are not performed. This provided about a two-fold increase in efficiency, so the numbers from a raw EKF implementation would be worse.

6. CONCLUSION

This paper investigated the problem of tracking a moving device in the context of two location architectures. In the active mobile architecture, fixed receivers at well-known positions periodically receive wireless signals (*e.g.*, radio and ultrasound) from a mobile device, allowing the infrastructure to track the device (or for the moving device to track itself). In the passive mobile architecture, fixed beacons in the infrastructure periodically broadcast information that allows a moving device to track itself.

The passive mobile architecture scales well with an increasing number of mobile devices, but does not allow *simultaneous* distance estimates to be obtained; as a result, its tracking has to be done one distance constraint at a time, which is less accurate than in the active mobile case. This paper investigated the relative performance of these two approaches in a real-world testbed based on the Cricket system.

We investigated the performance of an approach that uses three components: a least squares optimizer, an extended Kalman filter, and an outlier rejection method. We used our results from the active mobile and passive mobile approaches to design a hybrid approach that preserved the scalability and privacy advantages of the passive mobile approach, while providing much-improved tracking precision that came close to the performance of the active mobile architecture.

Acknowledgments

We thank the members of the Cricket project for their contributions to the Cricket hardware and software. We also thank several Cricket users for their interest in using Cricket to track moving devices.

7. REFERENCES

[1] BAHL, P., AND PADMANABHAN, V. RADAR: An In-Building RF-based User Location and Tracking System. In *Proc. IEEE INFOCOM* (Tel-Aviv, Israel, Mar. 2000).

[2] BAR-SHALOM, Y., AND FORTMANN, T. E. *Tracking and Data Association*. Academic Press, 1988.

[3] CADMAN, J. Deploying Commercial Location-Aware Systems. In *Proc. Fifth International Conference on Ubiquitous Computing* (October 2003).

[4] COLEMAN, T.F. AND Y. LI. On the Convergence of Reflective Newton Methods for Large-Scale Nonlinear Minimization Subject to Bounds. In *Mathematical Programming, Vol. 67, Number 2* (1994), pp. 189–224.

[5] FOX, D., HIGHTOWER, J., KAUZ, H., LIAO, L., AND D., P. Bayesian Techniques for Location Estimation. In *Proc. Workshop on Location-aware Computing, part of UBICOMP Conf.* (Seattle, WA, October 2003). Available from `http:www.ubicomp.org/ubicomp2003/workshops/locationaware/`.

[6] GETTING, I. The Global Positioning System. *IEEE Spectrum 30*, 12 (December 1993), 36–47.

[7] HARTER, A., HOPPER, A., STEGGLES, P., WARD, A., AND WEBSTER, P. The Anatomy of a Context-Aware Application. In *Proc. 5th ACM MOBICOM Conf.* (Seattle, WA, Aug. 1999).

[8] HOFFMANN-WELLENHOF, B., LICHTENEGGER, H., AND COLLINS, J. *Global Positioning System: Theory and Practice, Fourth Edition*. Springer-Verlag, 1997.

[9] LEONARD, J., BENNETT, A., SMITH, C., AND FEDER, H. Autonomous Underwater Vehicle Navigation. MIT Marine Robotics Laboratory Technical Memorandum 98-1, 1998.

[10] MIU, A. K. L. Design and Implementation of an Indoor Mobile Navigation System. Master's thesis, Massachusetts Institute of Technology, Jan. 2002.

[11] PATHIRANA, P., SAVKIN, A., AND JHA, S. Mobility modelling and trajectory prediction for cellular networks with mobile base stations. *MobiHoc 03* (2003).

[12] PETERSEN, I. R., AND SAVKIN, A. V. *Robust Kalman Filtering for Signals and Systems With Large Uncertainties*. Springer Verlag, 1999.

[13] PRIYANTHA, N., CHAKRABORTY, A., AND BALAKRISHNAN, H. The Cricket Location-Support System. In *Proc. 6th ACM MOBICOM Conf.* (Boston, MA, Aug. 2000).

[14] VALLIDIS, N. M. *WHISPER: A Spread Spectrum Approach to Occlusion in Acoustic Tracking*. PhD thesis, University of North Carolina at Chapel Hill, 2002.

[15] WANT, R., HOPPER, A., FALCAO, V., AND GIBBONS, J. The Active Badge Location System. *ACM Transactions on Information Systems 10*, 1 (January 1992), 91–102.

[16] WELCH, G., AND BISHOP, G. SCAAT: Incremental tracking with incomplete information. *Computer Graphics 31*, Annual Conference Series (1997), 333–344.

APPENDIX

A. EKF CORRECTION STEP

This section gives more details of the EKF correction step. After we compute our predicted state vector and covariance matrix for some time ΔT later, we process the newly measured distance sample. The idea here is that we have an estimate of the variance of the distance measurement, R_k (we assume that it is normally distributed as well), and we weigh the output between the predicted state and the new measurement based on their relative covariances. The factor that does this weighting is called the Kalman gain, K_k. We define $h(\mathbf{X})$, a function that, given a state vector $\mathbf{X}$, computes the expected distance to the appropriate beacon. We define $\mathbf{H}$ as the Jacobian of h. Denote the new measurement by z_k. We then arrive at the following three equations:

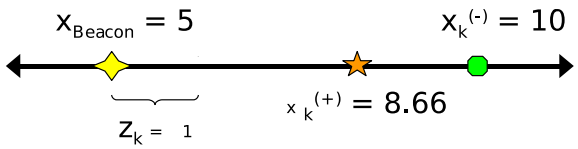

Figure 20: Illustration of our 1-D, point model, correction step example.

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{P}_k^{(-)}\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^{(-)}\mathbf{H}_k^{\mathrm{T}} + R_k)^{-1} \\
\mathbf{X}_k^{(+)} &= \mathbf{X}_k^{(-)} + \mathbf{K}_k[z_k - h(\mathbf{X}_k^{(-)})] \\
\mathbf{P}_k^{(+)} &= \mathbf{P}_k^{(-)} - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_k^{(-)}
\end{aligned}$$

We will use a very simple example to illustrate the use of these equations. We implement a Kalman filter to track an object in a 1-D space using a P ("point") model, shown in Figure 20. Suppose we have the following predicted variables:

$$\begin{aligned}
\mathbf{X}_k^{(-)} &= 10 \\
\mathbf{P}_k^{(-)} &= 2
\end{aligned}$$

Suppose the measurement and its variance from a beacon located at $x = 5$ are, respectively,

$$\begin{aligned}
z_k &= 1 \\
R_k &= 4
\end{aligned}$$

Then,

$$\begin{aligned}
h(\mathbf{X}) &= |5 - \mathbf{X}| \\
\mathbf{H}(\mathbf{X}) &= sign(5 - \mathbf{X}) \\
\mathbf{K}_k &= \frac{(2)(1)}{(1)(2)(1) + 4} = 0.33 \\
\mathbf{X}_k^{(+)} &= 10 + \frac{1}{3}(1 - 5) = 8.66 \\
\mathbf{P}_k^{(+)} &= 2 - \frac{1}{3}(1)(2) = 1.33
\end{aligned}$$

Here, the variance of the predicted state is one-half of the new measurement, so while the new measurement would suggest that the listener is at $x = 6$, the corrected state is only pulled toward that point.

Now we will observe the end-point behavior. If $R_k = P_k$, then $\mathbf{K}_k = 0.5$, and our projected state and the state dictated by the new measurement, $h(z_k)$, would be averaged to produce the corrected state. Furthermore, as R_k goes to infinity (*i.e.*, as our measurement becomes less reliable), $\mathbf{K}_k$ goes to zero, and the corrected state and covariance matrices are not changed. Conversely, as R_k goes to zero, $\mathbf{K}_k$ goes to one, and the corrected state approaches $h(z_k)$.

The simplicity of this example does not highlight the dimensions of the terms involved. If our filter has n states, and our measurement vector for the current iteration contains m measurements, then $\mathbf{X}$ is an $(n \times 1)$ vector, $\mathbf{P}$ is an $(n \times n)$ matrix, $\mathbf{z}$ is an $(m \times 1)$ vector, $\mathbf{R}$ is an $(m \times m)$ matrix, $\mathbf{H}$ is an $(m \times n)$ matrix, and $\mathbf{K}$ is an $(n \times m)$ matrix.