

# Ferret: RFID Localization for Pervasive Multimedia

Xiaotao Liu Mark D. Corner Prashant Shenoy

*Department of Computer Science  
University of Massachusetts, Amherst, MA  
{xiaotaol, mcorner, shenoy}@cs.umass.edu*

## Abstract

The pervasive nature of multimedia recording devices enables novel pervasive multimedia applications with automatic, inexpensive, and ubiquitous identification and location abilities. We present the design and implementation of Ferret, a scalable system for locating nomadic objects augmented with RFID tags and displaying them to a user in real-time. We present two alternative algorithms for refining a postulation of an object's location using a stream of noisy readings from an RFID reader: an online algorithm for real-time use on a mobile device, and an offline algorithm for use in post-processing applications. We also present methods for detecting when nomadic objects move and how to reset the algorithms to restart the refinement process. An experimental evaluation of the Ferret prototype shows that (i) Ferret can refine object locations to only 1% of the reader's coverage region in less than 2 minutes with small error rate (2.22%); (ii) Ferret can detect nomadic objects with 100% accuracy when the moving distances exceed 20cm; and (iii) Ferret is robust against different movement patterns of user's mobility.

## 1 Introduction

Advances in digital imaging technologies have led to a proliferation of consumer devices with video capture capabilities. The pervasive nature of multimedia recording devices such as cellphones, digital camcorders, PDAs and laptops, has made it relatively simple to capture, transform, and share large volumes of personal video and image content. A concurrent trend is the emergence of low-cost identification technologies such as RFID tags, designed to replace bar-codes [20]. Each tag contains a numeric code that uniquely identifies the object and can be queried by a wireless reader. It is likely that in the near future many personal objects (e.g., books, clothing, food items, furniture) will be equipped with self-identifying RFID tags.

The confluence of these trends—the ubiquity of RFID tags and the pervasive nature of multimedia recording

devices—enables novel pervasive multimedia applications with automatic, inexpensive, and ubiquitous identification and location abilities. By equipping cameras with RFID readers, it is possible to record images as well as the identities and locations of all RFID-tagged objects contained within each image. The captured video can then be queried in real-time to display the location of a particular object. The ability to pinpoint objects by their RFID identities within video streams enables many new applications. For instance, users can use them to locate a misplaced book on a bookshelf. Robots can use such devices to conduct real-time identification and search operations. Vision-based applications can use them to quickly learn the structure or organization of a space. Inventory tracking applications can proactively generate missing object alerts upon detecting the absence of an object.

While the inexpensive nature of RFID tags eases large-scale deployment issues, their passive nature raises a number of hurdles. A key limitation is that passive RFID tags are *self-identifying* but not *self-locating* (i.e., upon being queried, a tag can report its identity but not its location). Consequently, if multiple objects are present in a captured image, it is not possible to distinguish between these objects or pinpoint their individual locations. Some of the above scenarios (e.g., pinpointing a misplaced book on a bookshelf) require location information in addition to object identities. While numerous locationing technologies such as GPS and ultrasound [17, 21, 22] are available, it is not possible to equip passive RFID tags with these capabilities due to reasons of cost, form-factor and limited battery life. Instead, we require a locationing technology that does not depend on modifications to tags, is easily maintained, and scales to hundreds or thousands of tagged objects.

To address the above challenges, we have designed a system called *Ferret*. Ferret combines locationing technologies with pervasive multimedia applications. Ferret can locate objects using their RFID tags and display their

locations in real-time to a mobile user. As the positions of objects are uncertain, the system overlays the video display with an outline of where the object probably is. For instance, a user with a portable camera can ask the system to display the location of every new object in the room, and the display will show an outline of all of those locations. The display is constantly updated as the user moves using the continuous stream of tag readings to update the location.

Ferret uses the *location and directionality of RFID readers* to infer the locations of nearby *tags*. Ferret leverages the user’s inherent mobility to produce readings of the tag from multiple vantage points. It does this through two novel algorithms that refine the locations of objects using a stream of noisy readings from RFID tags. One algorithm is designed for offline use, given a large amount of computational power, while the other is designed to operate in real-time on a mobile system. In the case of the offline algorithm, we also incorporate negative readings—when the reader *does not* see the object—this greatly reduces the object’s possible locations.

We have implemented a prototype of Ferret and have used it to conduct a detailed performance evaluation. Our experiments pay specific attention to how fast Ferret can refine object locations, the error rate in locating objects, and how well it handles nomadic objects. Our results show that (i) Ferret can refine object locations to only 1% of the reader’s coverage region in less than 2 minutes with small error rate (2.22%); (ii) The offline algorithm incorporates information about when it does not see the object, outperforming the online algorithm by a factor of 13 or more; (iii) Ferret can detect nomadic objects with 100% accuracy when the moving distances exceed 20cm; and (iv) Ferret works with a wide variety of user mobility patterns.

The rest of this paper is structured as follows. Section 2 presents the problem formulation and a high-level design of Ferret, while Section 3 presents the details of our RFID locationing system. Section 4 presents our implementation and Section 5 experimental results. Finally, Section 6 and 7 present related work and our conclusions.

## 2 Ferret Design

Ferret is designed to operate on a handheld video camera with a display. To use Ferret, the user selects some set of objects she would like to locate in the room and moves around the room with the camera. Using an RFID reader embedded in the video camera, Ferret samples for nearby tags, and in real-time updates the camera’s display with an *outline* of the probable location of the objects she is searching for. Ferret’s knowledge of object location can be imprecise, so rather than showing a single centroid point, Ferret displays the outline, leaving the interpreta-

tion of the precise location to the user’s cognition. For instance, if Ferret can narrow the location of a book to a small region on a shelf, a user can quickly find the precise location. Figure 1 provides a pictorial representation of how the system would work. In this scenario the user is looking for a soup can in a messy office. After scanning the room using a Ferret-based camera, the system highlights a small region that contains the soup can.

### 2.1 Nomadic Location with RFID

Many pervasive systems that rely on location are predicated on the assumption that the number of objects requiring location information is small and mobile. In contrast, we designed Ferret to support a massive number of mostly static, or nomadic objects—objects that change locations infrequently. As a fraction of all objects, nomadic ones are in the vast majority—in any given room it is likely that there are hundreds, or possibly thousands of nomadic or static objects, while there are only a few mobile ones.

The primary barrier to providing locationing information for such a large number of objects is the reliance on batteries—making objects self-locating requires the use of a battery-powered locationing hardware. Even though locationing systems such as ultrasound [17, 21, 22] and Ultra-Wide Band (UWB) are becoming more energy efficient, equipping hundreds of objects in a room with self-locating capabilities simply does not scale, since it will require changing an unmanageable number of batteries. In contrast, passive RFID provides a battery-free, inexpensive, distributed, and easily maintained method for identifying objects; Ferret adds locationing capabilities to such objects. Ferret leverages the fact that an increasing number of objects will be equipped with RFID tags as a replacement to barcodes. Further, RFID tags continue to drop in price, and one can imagine attaching tags to a large number of household or office objects.

As RFID tags are passive devices and have no notion of their own location, Ferret must continuously calculate and improve its own notion of the object locations. The system fuses a stream of noisy, and imprecise readings from an RFID reader to formulate a proposition of the object’s location. The key insight in Ferret is to exploit the location of a camera/reader to infer the location of objects in its vicinity. In essence, any tag that can be read by a reader must be contained within its sensing range; by maintaining a history of tags read by the system, Ferret can progressively narrow the region containing the object. This is a simple yet elegant technique for inferring the location of passive RFID tags without expensive, battery-powered locationing capabilities.

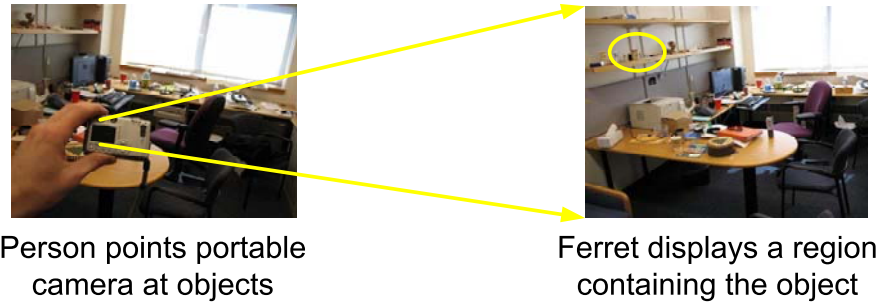


Figure 1: Use of Ferret to dynamically discover the location of a soup can in an office room.

## 2.2 Infrastructure Requirements

Strictly speaking, calculating and displaying object locations does not require any infrastructural support. Displaying a location on the video, as well as combining multiple readings of the object location, only requires *relative* locations, such as those from inertial navigation systems [1]. However, it is likely that knowledge of object locations in relation to a known coordinate system, such as GPS or a building map, will be useful for many applications. We assume that the camera/reader uses such a locationing system, such as Ultrasound or UWB, to determine its own location and then uses it to infer the location of objects in its vicinity.

As Ferret uses a directional video camera and RFID reader, it also requires an orientation system that can measure the pan (also known as heading and yaw), tilt (also known as pitch), and roll of the system. While research has proposed orientation systems for Ultrasound [18], we have chosen to use commercially available digital compass [2] to determine the directionality of the reader and the camera at any instant. Similar to the locationing system, Ferret benefits from having absolute orientation, although it can operate with only a relative orientation.

## 2.3 Location Storage

For each object, Ferret must store a description of the object's location. Considering that some RFID tags are remotely rewritable, Ferret can store the location for an object directly on the tag itself. Other options are to store the locations locally in each Ferret reader, or in an online, external database. Each option provides different privacy, performance, and management tradeoffs. Storing locations locally on each reader means that each independent Ferret device must start finding objects with zero initial knowledge. As the device moves and senses the same object from different vantage points, it can use a sequence of readings to infer and refine the object lo-

cation. The advantage of this method is that it works with read-only RFID tags and does not require any information sharing across devices. However, it prevents the device from exploiting history available from other readers that have seen the object in the recent past. In contrast, if location information can be remotely written to the RFID tags then other Ferret devices can start with better initial estimates of object location. However, this option requires writable tags and the small storage available on a tag limits the amount of history that can be maintained. In both of the above options, any device that has an RFID reader can determine object locations without needing the full complexity of the Ferret system.

A third option is to store the location information in a central database. This has the advantages of allowing offline querying and providing initial location estimates to mobile readers; further, since database storage is plentiful, the system can store long histories as well as past locations of nomadic objects. However, it requires readers to have connectivity to the database, the burden of management, and privacy controls on the database. Storing data on the tags also has implications for privacy control, however one must at least be proximate to the tag to query its location. Other systems have shown the value in using proximity as a clue for authorization [16]. We recognize that the privacy issue is much broader than this brief summary and we leave this as an issue for future work.

At the heart of Ferret is an RFID localization system that can infer the locations of individual passive RFID tagged objects. Ferret then uses this localization system to dynamically discover, update, store, and display object locations. The following section presents the design of our RFID localization technique.

## 3 RFID Locationing

Consider an RFID reader that queries all tags in its vicinity—the reader emits a signal and tags respond with

their unique identifier. Given all responses to a query, the reader can produce positive or negative assertions whether a particular tag is present within its reading range. The reader can not directly determine the exact location of the tag in relation to the reader, or even a distance measurement. However, just one positive reading of a tag greatly reduces the possible locations for that particular object—a positive reading indicates that the object is contained in the volume defined by the read range of the reader (see Figure 2). Ferret leverages the user’s mobility to produce a series of readings; the coverage region from each reading is intersected with all readings from the recent past, further reducing the possible locations for the object (see Figure 3). Using this method, Ferret can continually improve its postulation of the object location.

In addition to positive readings of an object’s RFID tag, the reader implicitly indicates a negative reading whenever it fails to get a reading for a particular tag that it is looking for. Using a similar method to positive readings, Ferret subtracts the reader’s coverage region from the postulation of the object’s location. This also improves the postulation of the object’s location. A third method to reduce the likely positions for the object is to modulate the power output of the reader. If a particular power output produces a positive reading, and a lower power produces a negative reading, the system has gained additional knowledge about the location of the object.

In general, whenever a tag is present in the read range, the reader is assumed to detect it with a certain probability—objects closer to the centroid of its read range are detected with higher probabilities, while objects at the boundary are detected with lower probabilities. Thus, each positive reading not only gives us a region that is likely to contain the object, it also associates probability values for each point within that region. This *coverage map* of a reader is shown in Figure 2. The map can be determined from the antenna data sheet, or by manually mapping the probability of detecting tags at different (x,y,z) offsets from the reader.

Given a three dimensional grid of the environment and assuming no prior history, Ferret starts with an initial postulate that associates an unknown probability of finding the object at each coordinate within the grid. For each positive reading, the probability values of each grid point contained within the coverage range are refined (by intersecting the range with past history as shown in Figure 3). Similarly, for each negative reading, the the probability values of each grid point contained within the coverage range is decreased. This results in a three-dimensional map,  $M(x, y, z)$ , that contains the probability of seeing a tag at each data point in relation to the reader. Using multiple power outputs requires building a map for each

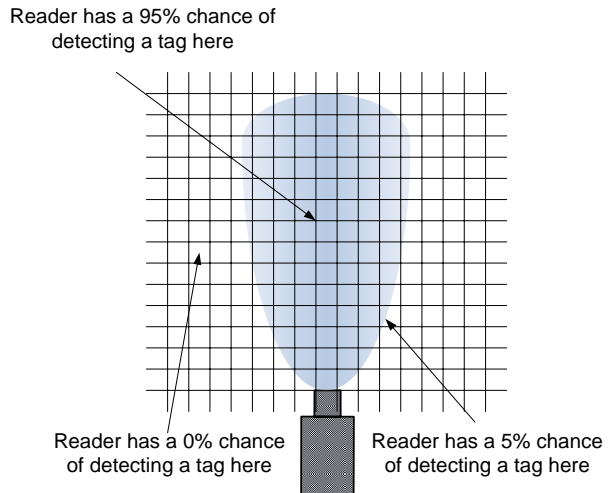


Figure 2: Coverage region of an RFID reader and tag detection probabilities in two dimensions.

power output level. Due to several constraints in our current prototype, Ferret currently does not use power modulation; however, adding this to the system will be trivial.

The amount of computation that the system can do drastically effects the location algorithm that performs intersections, the compensation for false negatives, and how it reflects the map to the user. Next we describe two alternative methods, one that is computationally intense and cannot be done in realtime on current mobile hardware. Such an offline technique is useful for describing an eventual goal for the system, or how to use the system for analyzing the data after it is collected. However, our goal is to implement Ferret on a mobile device so we also describe an online algorithm with drastically reduced computational cost.

### 3.1 Offline Locationing Algorithm

Formally, if we consider Ferret’s readings as a series of readings, both positive and negative, as a series  $D = \{D_1, D_2, D_3, \dots, D_n\}$ , and we want to derive the probability of the object being at position  $X$ , given the readings from the RFID, or  $P(X|D)$ . If we assume that each reading of the RFID reader is an independent trial, we can compute the likelihood as:

$$P(X|D) = \frac{P(X|\{D_1, \dots, D_{n-1}\})P(D_n|X)}{P(\{D_1 \dots D_N\})} \frac{1}{Z}, \quad (1)$$

where  $Z$  is a normalization factor. We omit the proof as it is a straight-forward application of conditional probability.

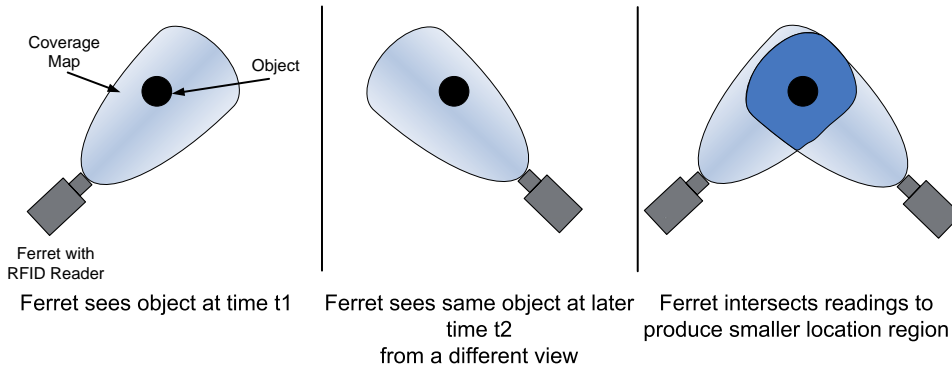


Figure 3: Refining location estimates using multiple readings.

If we first assume that the Ferret device (camera) is completely stationary, it operates as follows: i) once Ferret receives the first positive reading of a tag it initializes a three dimensional map,  $L$ , with the coverage map  $M$ , to track the probability that the object is at each of the coordinates in the map. ii) each successive reading multiplies each coordinate in  $L$  by  $M(x, y, z)$  if the reading was positive, or  $1 - M(x, y, z)$  if the reading was negative.

### 3.2 Translation, Rotation and Projection

The basic algorithm described above assumes a stationary camera/reader; Ferret’s notion of object location does not improve beyond a point, even with a large number of readings—most points in the reader’s range (i.e., within the coverage map) will continue to have a high, and equally likely probability of detecting the tag. Subsequently, multiple readings produce a large map with equally likely probabilities of the object’s location. Instead, Ferret depends on the user’s *motion* to reduce the possibilities for the object location—as the user moves in the environment, the same object is observed by the camera from multiple vantage points and intersecting these ranges allows Ferret to narrow the region containing the object. Incorporating motion is straightforward; however, the coordinates system of the coverage map  $M$  must be reconciled with that of the map  $L$  before this can be done.

The coverage map shown in Figure 3 is described in a three-dimensional coordinate system with the origin at the center of the reader’s RFID antenna, which we refer to as the *reader coordinate system*. The camera, although attached to the RFID reader, is offset from the reader, and has a slightly different coordinate system. We refer to this as the *camera coordinate system* which has its origin at the center of the camera’s CCD sensor. To combine multiple readings from the reader, and subsequently dis-

play them to the user, each map  $M$  must be transformed into a common coordinate system. We refer to this as the *world coordinate system*. The world can have its origin at any point in the space—with a locating system we can use its origin, or with an inertial location system we can use the first location of the reader. Without loss of generality, we assume the reader, camera, and world coordinate systems are left hand coordinate systems (see Figure 4).

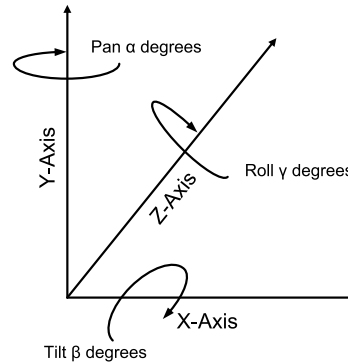


Figure 4: Left Handed Coordinate System

Performing this transformation is possible using techniques from linear algebra and computer graphics [7]. For each reading, the reader has a location and orientation with respect to the world coordinates. This is described as a location  $(x_0, y_0, z_0)$  and an orientation with a pan of  $\alpha$  degrees (the rotation along the y axis, range  $[-180, 180]$ ), a tilt of  $\beta$  degrees (the rotation along the x axis, range  $[-90, 90]$ ), a roll of  $\gamma$  degrees (the rotation along the z axis, range  $[-180, 180]$ ). The direction of the rotation is given by the left hand rule where the thumb is in the positive direction of the rotation axis and the fingers show the positive direction of rotation (see Fig-

ure 4). This transformation is formulated as a rotation matrix:

$$\mathbf{R} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \times \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \quad (2)$$

where  $\mathbf{R}$  is a 3 x 3 orthonormal matrix which has columns that are mutually orthogonal unit vectors, so that  $\mathbf{R}^{-1} = \mathbf{R}^T$ .

So, if a point is located at  $(x_w, y_w, z_w)$  in the world coordinates, the object's location in the reader coordinates  $(x_r, y_r, z_r)$  can be computed via:

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \mathbf{R} \times \begin{bmatrix} x_w - x_0 \\ y_w - y_0 \\ z_w - z_0 \end{bmatrix} = \mathbf{R} \times \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T \quad (3)$$

where the composite rotation matrix  $\mathbf{R}$  is given by Equation 2, and  $T = -\mathbf{R} \times \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$ .

Therefore, the reverse transformation from reader coordinate system to world coordinate system is given by:

$$\begin{aligned} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} &= \mathbf{R}^{-1} \times \left( \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} - T \right) \\ &= \mathbf{R}^{-1} \times \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \end{aligned} \quad (4)$$

where  $(x_0, y_0, z_0)$  is the reader's position in world coordinate system and  $\mathbf{R}^{-1} = \mathbf{R}^T$ .

When computing the intersection of coverage maps, Ferret first transforms the coverage map,  $M$  into the world coordinate systems using Equations 2 and 4, and computes the intersection according to the methods presented in Section 3.1 to produce a new map  $L$  containing the likelihood of object locations.

Once Ferret produces a three dimensional map that it believes contains a particular object, it must overlay this region onto the video screen of the camera; doing so involves projecting a 3D map onto a two dimensional display. This is done in two steps: thresholding and projection. The threshold step places a minimum value for the likelihood on the map  $L$ —by using a small, but non-zero value for the threshold, Ferret reduces the volume that encompasses the likely position of the object. However, using a larger threshold may cause Ferret to shrink the

volume excessively, thus missing the object. Currently this is a tunable parameter in Ferret—in the evaluation section we demonstrate how to choose a reasonable value.

Finally, Ferret projects the intersection map onto the image plane of the video display. Ferret must transform the intersection map from the world coordinate system into camera coordinate system. Ferret performs this transformation using Equation 2 and 3, along with the camera's current position and orientation. As stated previously, the camera coordinate system follows the left-hand convention, and the z-axis of the camera coordinate system is co-linear with the camera's optical axis. Assuming the camera has focal length  $f$ , and a point is positioned at  $(x_c, y_c, z_c)$  in camera coordinate system. The projection is given by:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{z_c} \times \begin{bmatrix} x_c \\ y_c \end{bmatrix} \quad (5)$$

where  $u$  and  $v$  is the projection at the CCD sensor [4].

For each reading the RFID reader produces, the location algorithm must perform  $O(n^3)$  operations, for a three dimensional space that is  $n \times n \times n$ , in addition to translating and rotating the coverage map, and projecting the location map onto the display.

If Ferret is searching for multiple objects, it must perform these operations for *each* individual object. In practice, we have found that each RFID reading consumes 0.7 seconds on a modern processor, while our RFID reader produces 4 readings per second. Given the speed at which a human may move the camera, this is not feasible to do in realtime, however it works well for an offline system that has less stringent latency requirements.

An offline algorithm also has the opportunity to perform these operations for the whole video, and then use the smallest region that it computed to annotate the entire video stream with that region.

### 3.3 Online Locating Algorithm

Given that the offline algorithm is too computationally intensive for a mobile device to operate in real-time, we describe a greatly simplified version of the locating algorithm. The primary goal is to reduce the representation of the probability of where the object is. Instead of a full representation that describes the probability at each location, we reduce it to describing just the convex region where the object is with very high probability. Describing such a region is very compact, as we only need to track the points that describe the perimeter of the convex region. Intersecting two maps is very fast, as it is a series of line intersections.

Figure 5 shows this in detail for two dimensions, extending it to three dimensions is straightforward. The

first half of the diagram shows sample points that describe the outside of the coverage map. Ferret rotates and translates the coverage map  $M$  as described in the previous section, and intersects it with the current map  $L$ . For each constant  $y$  value, the system finds the intersection of the two line segments and uses that as the description of the new map  $L$ . For instance in Figure 5, we choose a constant  $y$  value  $y_1$ . After rotating and translating the map  $M$  to match to the reader’s current position, the system intersects the two line segments,  $(x_1, y_1) - (x_3, y_1)$  from the current map  $L$ , with  $(x_2, y_1) - (x_4, y_1)$  from the new map  $M$ . The resulting intersection is the segment  $(x_2, y_1) - (x_3, y_1)$ , which describes the perimeter of the new location map  $L$ . Ferret repeats this process for all  $y$  values. Extending this to three dimensions is straightforward: intersect two line segments for each pair of constant  $y$  and  $z$  value. This means the complexity of the intersection is  $O(n^2)$  rather than  $O(n^3)$  as in the offline algorithm.

Also, instead of using a map of probabilities for the coverage map, we reduce it to the convex shape that describes the coverage region of the RFID reader than can read tags with some probability greater than 0. This virtually eliminates the possibility of false positives. Additionally, describing the perimeter only requires two  $x$  points for each pair of  $y$  and  $z$  values, thus the representation of the region is greatly reduced in size from  $O(n^3)$  to  $O(n^2)$ . Using our prototype as an example, this reduces the storage requirement from 43.5M bytes to 178K bytes—each of these are highly compressible. This greatly aids Ferret’s ability to store the regions directly on the storage-poor tags. The line segment representation does mean that the system cannot incorporate negative regions, as intersecting with a negative region can create a concave, rather than convex, region. A concave region would return the complexity of the representation and the intersection to  $O(n^3)$ . False negatives do not affect the system, as negative readings are not used at all.

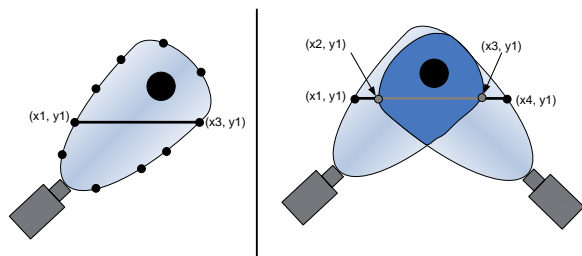


Figure 5: Online location estimation in Ferret.

### 3.4 Dealing with Nomadic Objects

We designed Ferret to deal with objects that move infrequently—commonly referred to as nomadic as opposed to mobile objects that move frequently. When objects do move, Ferret should adjust to deal with this. In the online algorithm, this is straightforward. When the location algorithm performs an intersection of two maps, it may produce a region of zero volume. This indicates that the maps were disjoint, and the object could not possibly be within the previously postulated region. The system then reinitializes the location map,  $L$ , to the most current reading, which is  $M$  rotated and translated to the reader’s current position.

However, the offline algorithm is more complicated as it produces a likelihood location map. One solution is only using the intersection of positive readings to deal with nomadic objects like what the online algorithm does, while this approach doesn’t utilize any useful information given by the negative readings. Therefore, another practical solution is applying a likelihood threshold to the likelihood location map and removing any location with a probability less than the threshold. If the resulting location map is empty, we will consider the object has moved and reinitialize the location map,  $L$ , to the most current reading. Choosing an appropriate threshold is a critical factor in this approach. Using a larger threshold will increase the likelihood that the resulting location map is empty when the object actually does not move. In Section 5, we will show the experiments on how to choose an appropriate threshold.

## 4 Implementation Considerations

We have implemented a prototype Ferret system as shown in Figure 6. Although the prototype is quite large, this is due to the combination of many separate pieces of hardware—there is nothing that would preclude a much smaller commercial version. Our prototype is based on the following hardware:

- A *ThingMagic Mercury4 RFID reader* which has a SensorMagic monostatic circular antenna connected to it. The output power of the reader is set to 30dBm (1Watt). This reader operates at the frequency range 909 – 928MHz, and supports RFID tags of EPC Class 0, EPC Class 1, and ISO 18000-6B. The reader is paired with a ThingMagic monostatic circular antenna that has a balloon shaped radiation pattern. An alternative is to use a linear antenna that has a more focused radiation pattern and longer range; however, the narrower beam will produce fewer positive readings for each tag. The tradeoff in antenna choice and the possibility of fu-

ture antennas with variable radiation patterns are interesting questions for future research. We used an orientation-insensitive, EPC Class 1, Alien Technology “M” RFID tag operating at 915MHz.

- A *Sony Motion Eye web-camera* connected to a Sony Vaio laptop. This CMOS-based camera is set to a fixed focal length of 2.75mm, and uses a sensor size of 2.4mm by 1.8mm. The camera provides uncompressed 320x240 video at 12 frames-per-second.
- *Cricket* [17] ultrasound 3D locating system to estimate the location of the camera and RFID reader. We deployed Cricket beacons (served as references) on the ceiling, and attached a Cricket sensor to our prototype system. The Cricket sensor is offset from the camera and RFID reader and we correct for this translation in software.
- A *Sparton SP3003D digital compass* to obtain the 3D orientations (pan, tilt, and roll) of the camera’s lens and the reader’s antenna. We mounted the compass, the camera’s lens, and the reader’s antenna in a way that they all have same 3D orientations.

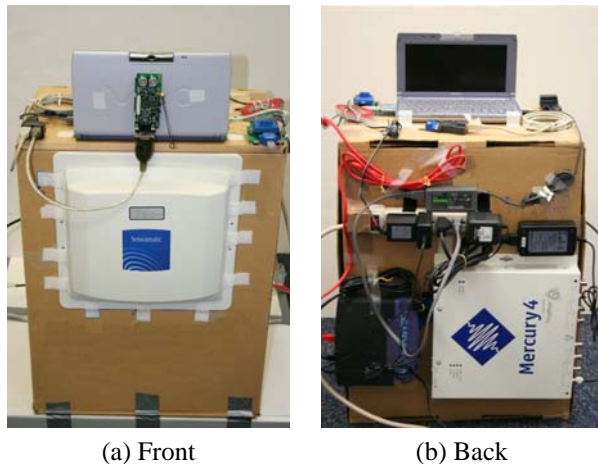


Figure 6: Ferret Prototype System

Our prototype system consists of the following software modules:

- **Video Module:** This module records the video stream from the web camera, and transcodes the video stream into MPEG-2 video clip. In addition to this functionality, the video module will project and highlight the estimated region containing the target object when displaying video stream. We modify the FFmpeg video suite [5] to implement this module. We implement the projection function according to Equation 5 to compute the projection of

the location estimation, and then intercept the display function of FFmpeg video suite to display the boundary of the projection area.

- **RFID Module:** This module controls the RFID reader, and records the readings from the RFID reader. The RFID reader provides functions of remote control and query via TCP connection using SQL-like query and control messages. The RFID module submits a query request with interval value of 250ms to the reader, and then the RFID reader periodically responds with configurable plain text message including the tag ID, the ID of the antenna reading the tag, and so on.
- **Cricket and Compass Module:** This module communicates with the Cricket sensor and digital compass to obtain the location and orientation of camera and RFID reader. The Cricket module communicates with the Cricket sensor via a serial port, and the output of the Cricket sensor is its distances to beacons. Our module records these distances, and uses them to triangulate the location of the Cricket sensor. After adding some constant offset (measured manually), we then have the location of the camera and RFID reader. The Compass module also communicates with the compass via a serial port.
- **Locationing Module:** This module implements the locationing algorithms which are discussed in Section 3. This implementation includes: (i) coordinate transformation functions between world coordinate system and the coordinate systems of camera and RFID reader according to Equation 2, 4, and 3, (ii) intersection functions to compute intersection for positive readings and negative readings, and (iii) a central database to store the location information.

## 5 Experimental Evaluation

In this section, we evaluate Ferret by focusing on the performance of locationing and projection. In particular, we concentrate on how quickly Ferret can refine the location of an object for a user. We show how to tune the offline algorithm to trade the size of the location region and the overall error rate. We then show a comparison of the on-line and offline systems. We demonstrate that Ferret can detect objects that move within a room and we show the computation and storage costs of our system.

We measure Ferret’s performance using two metrics: the size of the postulated location and the error rate. Ferret automatically provides the size, either the volume of the three-dimensional region, or the area of the two-dimensional projection on the video screen. The three-dimensional region is not a sphere, but to interpret the

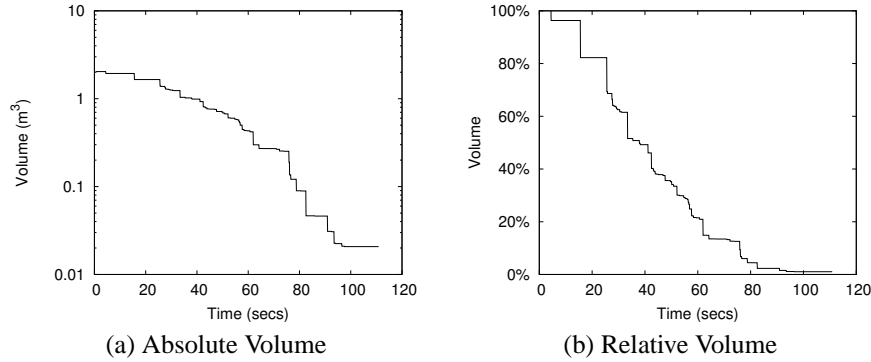


Figure 7: Online refinement of location estimation.

results, a sphere with a volume of  $0.01m^3$  has a diameter of  $26.7cm$  and a volume of  $0.1m^3$  has a diameter of  $57.6cm$ . Ferret’s error rate is the number of objects that do not appear in the area projected on to the display. The error rate is determined through manual inspection of a video recording.

All of our experiments are conducted in a  $4m \times 10m \times 3m$  room equipped with a Cricket ultrasound system. We used five beacons mounted on the ceiling which we manually calibrated. The origin of our world-coordinate system is a corner of the room. The camera records all video at 12 frames/second, and the RFID reader produces 4 readings per second. For the online system, we use a coverage map that includes all places where the tag has a non-zero probability of reading a tag. That region is an irregular shape that is  $2.56m \times 1.74m \times 2.56m$  at the maximum and has a volume of approximately  $2m^3$ .

### 5.1 Online Refinement Performance

The primary goal of Ferret is to quickly locate, refine, and display an outline on the video display that contains a particular object. As this happens online, Ferret continuously collects readings and improves its postulation of the object’s location—this is reflected as the volume of the region shrinking over time. To demonstrate this, we placed one tag in the room, and then walked around “randomly” the room with the prototype. We plot the volume of the location estimation versus time in Figure 7. The absolute volume tracks the total volume of the region, while the relative volume tracks the size of the region relative to the starting coverage region of the reader. In this case Ferret does not make any errors in locating the object. The time starts from the first positive reading of the tag and Ferret begins with no previous knowledge about object locations.

The results show that the volume size of the location estimation drops from  $2m^3$  to  $0.02m^3$  which is only 1% of the reader’s coverage region in less than 2 min-

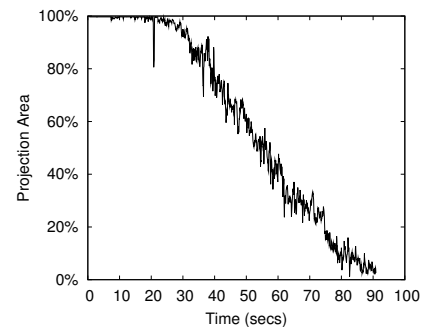


Figure 8: Online refinement on the projection display.

utes. The volume monotonically decreases, as intersecting positive readings only shrinks the area, while negative readings are ignored. Also, this is a pessimistic view of the refinement time—with prior knowledge, the process occurs much more rapidly. For instance, if the user switches to searching for another object in the same room, Ferret can take advantage of all of the previous readings. If a previous user has stored location information on the tag, this reader can also take advantage of that from the time of the first reading. Additionally, if some location information is stored in a centralized database, Ferret can immediately project an area onto the video without *any* positive readings.

In addition to the volume size of the location estimation, we also plot the projection area versus time in Figure 8 in which the projection areas are shown as a percentage of the image plane area. Our results show that the final projection area is only 3% of the whole image, or approximately a 54 pixel diameter circle on a  $320 \times 240$  frame. However, the projection area does not monotonically decrease as the volume does. This is because the camera is constantly moving, thus the point-view constantly changes, and the same volume can project different areas from different orientations.

## 5.2 Offline Algorithm Performance

While the online algorithm is useful for current mobile devices, the offline algorithm uses more information, and a more precise representation of the object’s location likelihood. To evaluate Ferret’s precision in locating objects, we placed 30 tags in a 2.5m x 2.5m x 2m region, and we move the prototype around the room for 20 minutes. We repeat the experiment 3 times and record the volume of the postulated region, and manually verify how many objects are truly contained in the area projected onto the video plane. With 30 tags and 3 experiments, Ferret can make between 0 and 90 location errors.

Before evaluating the offline algorithm, we must set a threshold for the minimum likelihood for the object as described in Section 3. Recall that a larger threshold can reduce the volume encompassing the likely position of the object. However, a larger threshold will also increase the error rate of Ferret (the volume doesn’t contain the object). In order to test the sensitivity of offline Ferret to the change of likelihood threshold, we varied the likelihood threshold from 0.00001 to 0.4, and ran the offline Ferret algorithm on the data we collected in the experiment. We show the results in Figure 9.

Threshold	Errors	Mean Volume
0.00001	5/90	0.0117m <sup>3</sup>
0.0001	5/90	0.0117m <sup>3</sup>
0.001	5/90	0.0116m <sup>3</sup>
0.01	5/90	0.0112m <sup>3</sup>
0.1	6/90	0.0108m <sup>3</sup>
0.2	7/90	0.0104m <sup>3</sup>
0.3	8/90	0.0102m <sup>3</sup>
0.4	9/90	0.0100m <sup>3</sup>

Figure 9: Performance of offline Ferret under different likelihood thresholds.

The results show that: (i) the number of errors almost doubles from 5 to 9 as threshold increase from 0.00001 to 0.4 (ii) the mean volume of the location estimation is essentially constant; and (iii) for a threshold  $\leq 0.01$ , the number of errors doesn’t change. When using too high of a threshold Ferret incorrectly shrinks the volume, leaving out possible locations for the object. Considering the balance of error rate and mean volume, we choose a likelihood threshold of 0.01. Using this threshold, we run the offline algorithm and compare it to the performance of the online algorithm. In Figure 10, we plot the CDF of Ferret’s location accuracy for both algorithms.

The results show that (i) The online algorithm can localize an object in 0.15m<sup>3</sup> and 0.05m<sup>3</sup> regions with 80% and 50% probability, respectively. The 0.15m<sup>3</sup> and 0.05m<sup>3</sup> regions are only 7.5% and 2.5% of the reader’s coverage region which is 2m<sup>3</sup>; (ii) The offline algorithm outperforms the online algorithm by localizing an object

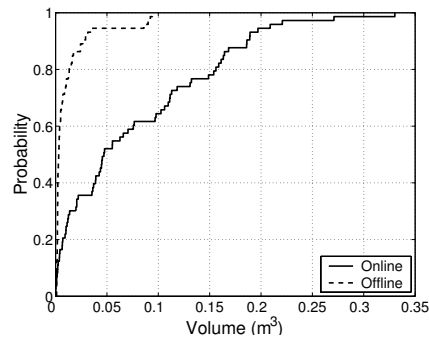


Figure 10: Empirical CDF of Ferret’s locationing accuracy

in a 0.05m<sup>3</sup> region with more than 90% probability and in a 0.1m<sup>3</sup> region with 100% probability.

However, when we verify the online algorithm’s error rate, it only makes 2 errors, as compared to the offline algorithm’s 5 errors. We believe that the slightly greater number of errors in the offline algorithm is due to our incorporation of negative readings in the algorithm. In this experimental setup, the prototype system is constantly moving and the tags are in the coverage region of the RFID reader for a small portion of the total time (less than 5%). This scenario will generate 19 times the number of negative readings than positive readings, and negative readings are weighted as heavily as positive readings. Considering that we measured the performance of the reader under ideal conditions, we have overestimated the performance of the RFID reader. The online algorithm does not exhibit the same behavior as it does not ever use negative readings. As negative readings are correlated by orientation, and location, we believe that more accurate modeling of reader performance is an important direction for future research.

## 5.3 Mobility Effects

Ferret exploits the user’s mobility to produce a series of readings from multiple positions, and further refine its location estimation via intersecting the coverage regions at these positions. The previous experiment showed the results of a human, yet uncontrolled, mobility pattern. In reality users move erratically; however, their motions are composed of smaller, discrete motion patterns. To study how individual patterns affect the performance of Ferret we placed a single tag in the room and evaluated Ferret with a small set of semi-repeatable motion patterns shown in Figure 11: (a) **straight line**, the prototype system moves in a straight line, tangential to the object, without changing the orientation of the camera lens and RFID reader; (b) **head-on**, the prototype moves

straight at the object and stops when the reader reaches the object; (c) **z-Line**, the prototype system moves in a z-shaped line without changing its orientation; (d) **rotation**, the prototype system moves in an arc, while keeping the lens orientation radial to the path; (e) **circle**, the prototype system moves in a circle, while keeping the reader facing the object. Intuitively, the circular pattern may be the least likely of the mobility patterns, whereas the head-on is probably the most likely—once the user gets one positive reading, she will tend to head towards the object in a head-on pattern. We evaluated Ferret’s performance using the volume of the resulting region. For each movement pattern we ran three experiments, averaged the results, and compared the smallest volume size of both online and offline Ferret. Our results are shown in Figure 12.

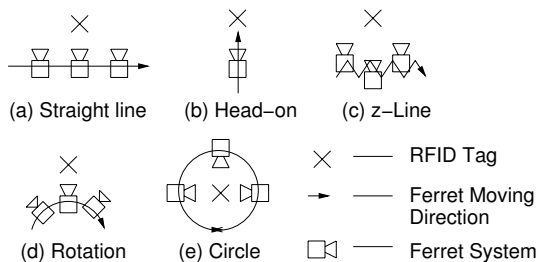


Figure 11: Path of the Ferret device

The results show that Ferret performs similarly for each of the movement patterns; however the circular pattern performs the worst. The circular pattern always keeps the object in view and generally in the center of the reader’s coverage region. This produces a set of readings that generally cover very similar regions. In each of the other cases, the mobility of the reader covers more disjoint spaces, and thus produces smaller volumes. This is true even of the head-on pattern as the first reading and the last reading have very little volume in common. Another result is that the offline algorithm widely outperforms the online algorithm, except in the case of the circular and head-on patterns, where the performance is similar. Much of the offline algorithm’s performance advantage comes from incorporating negative readings to reduce the possible locations for the object. In the case of the circular and head-on patterns, the object is always in view, producing few negative readings, yielding similar performance to the online algorithm. Although non-intuitive, this means that *not* seeing the object is as important as seeing it to narrow its location.

## 5.4 Object Motion Detection

Ferret is designed to deal with objects that move infrequently, but when the object does move, Ferret should detect this and start its refinement process over. As discussed in Section 3, whenever Ferret encounters an empty location estimation, Ferret assumes that the corresponding object has moved. To evaluate Ferret’s performance in detecting these nomadic objects we place a tag in the room and use Ferret to estimate its location. We then move the tag a distance between 5cm and 200cm and again use Ferret to estimate its location. We repeat the experiment ten times for each distance, and record the number of times that Ferret didn’t detect a moved object. The results are shown in Figure 13.

The figure shows that the online and offline Ferret can detect 100% object movements when the moving distance exceeds 25cm and 20cm, respectively. This is consistent with our previous results that show that Ferret can localize an object to within a region with a volume of hundredths of a  $m^3$ —this gives a radius on the order of 20cm, exactly how well Ferret can detect movement. As the object has not actually left the postulated area, Ferret is still correct about the object’s location.

## 5.5 Spatial Requirements

The prototype has a non-zero probability of detecting tags in balloon-shaped region, with maximum dimensions of 2.56m x 2.56m x 1.74m—this shape has a volume of approximately  $2m^3$ . For the offline algorithm we sample this coverage region every centimeter. As discussed in Section 3, the offline algorithm requires every point in this space, while the online algorithm only requires a set of points that describe the exterior of the region. This reduced representation results in much smaller spatial requirements as compared to offline spatial requirements: (i) the offline algorithm uses a **float** of four bytes to describe the probability of a sample point, and the total space is  $256 * 256 * 174 * 4 = 43.5M$  bytes using a three dimensional array to store the probabilities of all sample points, and (ii) the online algorithm uses a two dimensional array (the dimensions correspond to y and z) to represent the coverage region, and consequently, it only needs two bytes to track the x value of every outside sample point, thus the total space required is  $256 * 174 * 2 = 178K$  bytes. Both the offline and online representations are highly compressible: the offline can be reduced to 250K bytes and the online representation to 5K bytes using the commonly available compression tool *gzip*. For the foreseeable future, RFID tags will not contain enough storage for the offline representation, while the online version is not unreasonable. If tags have more or less storage the number of sample points can

	Straight line	Head-on	z-Line	Rotate	Circle
online Volume ( $m^3$ )	0.020	0.0042	0.023	0.026	0.032
offline Volume ( $m^3$ )	0.0015	0.0030	0.0017	0.0011	0.026
offline : online	13.33	1.40	13.52	23.63	1.23

Figure 12: Performance of Ferret under various mobility patterns.

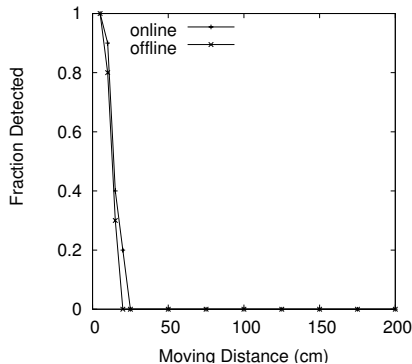


Figure 13: Fraction of object movements detected

be adjusted, although this will affect the precision of the system.

## 5.6 Computational Requirements

The computational requirements of offline and online have similar relationship. We measured the computational and spatial requirements of Ferret’s locationing algorithm on an IBM X40 laptop equipped with a 1.5GHz Pentium-M processor: (i) the offline algorithm costs 749.32ms per reading for each object, and (ii) the online algorithm only costs 6ms per positive reading for each object, which is only 1/125 of the offline computational requirements. Our results show that the online algorithm incurs small overhead and will run online to track multiple tags simultaneously on relatively inexpensive hardware, while the offline algorithm incurs large overhead and can only run offline.

## 6 Related Work

The pervasive multimedia application envisioned in this work has similarities with the recently-proposed SEVA system [13]. SEVA assumes WiFi-based tags that are both self-identifying and self-locating—each tag has a unique identity and uses a Cricket [17] ultrasound receiver to continuously determine its location. A digital camera records video as well as object locations and identities, which can be subsequently queried by the user. There are several key differences between SEVA and our

system. SEVA is based on active 802.11-based tags, while we rely on passive RFID tags. Object tags in SEVA are assumed to have locationing capabilities. Our environment does not make this assumption. SEVA is inherently designed for offline use, where users query their video collections *post-facto*; consequently, SEVA depends on post-processing capabilities. In contrast, we require the capability to capture and query video in real-time. Finally, SEVA scales to around a dozen *moving* objects, whereas we require the ability to scale to hundreds or thousands of *nomadic* RFID-tagged objects.

Similarly, a human-centric search system (MAX) of the physical world was proposed in [23]. The MAX system uses battery-powered Crossbow motes [12] to provide location information of objects with reference to identifiable landmarks (e.g., on the dining table) rather than precise coordinates.

In the FindIT Flashlight Project [14], researchers have designed a battery-powered optical sensor system to identify objects. The sensors are stimulated by a coded optical beam, and the sensor then flashes an LED once the code in the optical beam matches the code programmed in the sensor. The flashlight is then used to identify the object. While these sensors are highly energy efficient they still require a battery, and the system requires line-of-sight to operate.

Researchers have developed RFID-based indoor locationing systems [11, 15] using active, battery powered, RFID tags. In SpotON [11], Hightower, et. al, use the radio signal attenuation to estimate tag’s distance to the base stations, and triangulate the position of the tagged objects with the distance measurements to several base stations. LANDMARC [15] deploys multiple fixed RFID readers and reference tags as infrastructure, and measures the tracking tag’s nearness to reference tags by the similarity of their signal received in multiple readers. LANDMARC uses the weighted sum (the weight is proportional to the nearness) of the positions of reference tags to determine the 2D position of the tag being tracked.

All the above work [11, 13, 15, 23] use battery-powered sensors to identify and locate objects. These sensors are expensive (at least tens of dollars per sensor) and have limited lifetime (from several days to several years). These limitations have prevented them from scaling to applications dealing with hundreds and thousands of objects. In contrast, passive RFID tags are inexpensive

(less than a dollar per tag and falling) and do not require battery power source. These features make passive RFID technology ideal for such applications.

Fishkin, et.al, proposed a technique to detect human interactions with passive RFID tagged objects using static RFID readers in [6]. The proposed technique used the change of response rate of RFID tags to unobtrusively detect human activities on RFID tagged objects such as, rotating objects, moving objects, waving a hand in front of objects, and walking in front of objects. However, this doesn't consider the problem of estimating the locations of RFID tagged objects. Their experimental results show that their system could nearly always detect rotations, while the system performed poorly in detecting translation-only movement.

In [9], Hähnel, et.al, proposed a mapping and localization approach using the combination of a laser-range scanner and RFID technology. Their approach employed laser-based FastSLAM [10] and Monte Carlo localization [3, 8] to generate maps of static RFID tags using mobile robots equipped with RFID readers and laser-range scanner. Through practical experiments, they demonstrated that their system can build accurate 2D maps of RFID tags, and they further illustrated that resulting maps can be used to accurately localize the robot and moving tags.

Another system is the 3D RFID tag [19]. The 3D RFID system is equipped with a robot-controlled unidirectional antenna, and the 3D tag consists of several combined tags. Two kinds of 3D tags are developed: union tag and cubic tag. The proposed system can not only detect the existence of the 3D tag but also estimate the orientation and position of the object. However, they require usages of specific orientation-sensitive 3D tags custom-built from multiple tags. Furthermore, the system uses highly expensive robot system to control the antenna's movement and then estimate the orientation and position of the object. In contrast, Ferret only needs one standard orientation-insensitive tag per object and the user's inherent mobility to estimate the object's location.

## 7 Conclusions

This paper presents the design and implementation of Ferret, a scalable system for locating nomadic objects augmented with RFID tags and displaying them to a user in real-time. We present two alternative algorithms for refining a postulation of an objects location using a stream of noisy readings from an RFID reader: an online algorithm for real-time use on a mobile device, and an offline algorithm for use in post-processing applications. We also present methods for detecting when nomadic objects move and how to reset the algorithms to restart the

refinement process.

We present the results of experiments conducted using a fully working prototype. Our results show that (i) Ferret can refine object locations to only 1% of the reader's coverage region in less than 2 minutes with small error rate (2.22%); (ii) Ferret can detect nomadic objects with 100% accuracy when the moving distances exceed 20cm; and (iii) Ferret is robust against different movement patterns of user's mobility.

Ferret represents one possible modality in using a combination of inexpensive and ubiquitous RFID tags with real-time multimedia systems. We expect that many future systems can build on the techniques presented in this paper, and make further improvements to the localization algorithms. While a great number of hurdles exist in privacy and deployment, we contend that systems that leverage this ubiquity will provide untold utility to users.

## References

- [1] B. Barshan and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, June 1995.
- [2] Sparton sp3003d digital compass. <http://www.sparton.com/>.
- [3] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA '99)*, Detroit, MI, pages 1322–1328, May 1999.
- [4] O. Faugeras. *Three-dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, first edition, 1993.
- [5] Ffmpeg 0.4.8. <http://ffmpeg.sourceforge.net/index.php>.
- [6] K. Fishkin, B. Jiang, M. Philipose, and S. Roy. I sense a disturbance in the force: Long-range detection of interactions with rfid-tagged objects. In *Proceedings of the 6th International Conference on Ubiquitous Computing (UbiComp'04)*, Nottingham, England, pages 268–282, September 2004.
- [7] J. D. Foley, A. V. Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice in C*. Addison-Wesley Professional, second edition, 1995.
- [8] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99)*, Orlando, FL, pages 343–349, July 1999.
- [9] D. Hähnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose. Mapping and localization with rfid technology. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA '05)*, Barcelona, Spain, pages 1015–1020, April 2004.

- [10] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, Las Vegas, NV, pages 206–211, October 2003.
- [11] J. Hightower, R. Want, and G. Borriello. Spoton: An indoor 3d location sensing technology based on rf signal strength. Technical Report 00-02-02, University of Washington, 2000.
- [12] J. Hill and D. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):1224, November/December 2002.
- [13] X. Liu, M. Corner, and P. Shenoy. Seva: Sensor-enhanced video annotation. In *Proceedings of the 13th ACM Annual Conference on Multimedia (MM'05)*, Singapore, pages 618–627, November 2005.
- [14] H. Ma and J. A. Paradiso. The findit flashlight: Responsive tagging based on optically triggered microprocessor wakeup. In *Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp'02)*, Goteborg, Sweden, pages 160–167, September 2002.
- [15] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil. Landmarc: Indoor location sensing using active rfid. In *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, Dallas-Fort Worth, TX, pages 407–417, March 2003.
- [16] A. Nicholson, M. Corner, and B. D. Noble. Mobile device security using transient authentication. *IEEE Transactions on Mobile Computing*.
- [17] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the 6th annual ACM International Conference on Mobile Computing and Networking (MobiCom'00)*, Boston, MA, pages 32–43, August 2000.
- [18] N. B. Priyantha, A. K. L. Miu, H. Balakrishnan, and S. Teller. The cricket compass for context-aware mobile applications. In *Proceedings of the 7th ACM Annual Conference on Mobile Computing and Networking (MOBICOM'01)*, Rome, Italy, pages 1–14, July 2001.
- [19] S. Roh, J. H. Park, Y. H. Lee, and H. R. Choi. Object recognition of robot using 3d rfid system. In *Proceedings of the 2005 International Conference on Control, Automation and Systems (ICCAS'05)*, Gyeong Gi, Korea, June 2005.
- [20] R. Want. An introduction to rfid technology. *IEEE Pervasive Computing*, 5(1):25–33, January–March 2006.
- [21] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, January 1992.
- [22] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications Magazine*, 4(5):42–47, October 1997.
- [23] K. Yap, V. Srinivasan, and M. Motani. Max: Human-centric search of the physical world. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys'05)*, San Diego, CA, pages 166–179, November 2005.