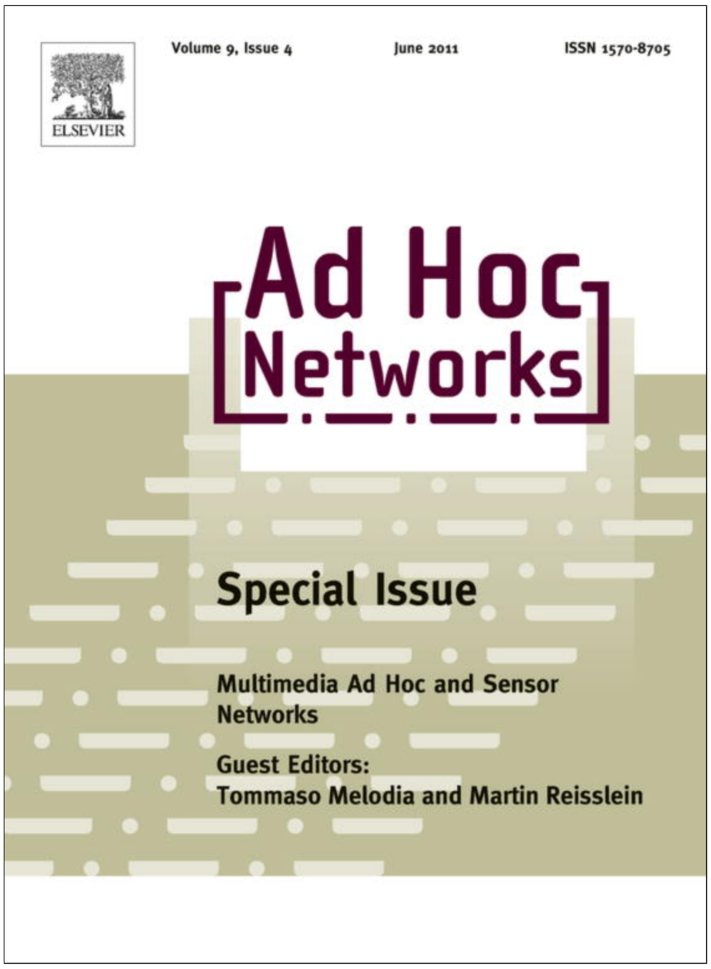
Volume 9, Issue 4
June 2011
ISSN 1570-8705
ELSEVIER
Ad Hoc Networks
Special Issue
Multimedia Ad Hoc and Sensor Networks
Guest Editors:
Tommaso Melodia and Martin Reisslein

Ad Hoc Networks 9 (2011) 565–575

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Ferret: An RFID-enabled pervasive multimedia application

Xiaotao Liu [a], Mark D. Corner [b], Prashant Shenoy [b,*]

[a] EMC Corporation, 1133 Westchester Avenue, White Plains, NY 10604, United States
[b] Department of Computer Science, University of Massachusetts, Amherst, MA 01003, United States

A R T I C L E I N F O

Article history:
Available online 18 August 2010

Keywords:
Sensor networks
RFID
Pervasive multimedia
Positioning

A B S T R A C T

This paper presents the design and implementation of Ferret, a system for locating nomadic augmented with RFID tags and visually displaying them to a user in real-time. We present a novel algorithm to infer location of tagged objects using the location of a camera and reader that observes them. We also present techniques to refine location estimates using multiple observations and a method to display and update object locations on a video camera screen. 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 min with small error rate (2.22%); (ii) Ferret can detect nomadic objects with 100% accuracy when the moving distances exceed 20 cm; and (iii) Ferret is robust against different movement patterns of user's mobility.

1. Introduction

RFID is a promising new electronic identification technology that enables objects to be equipped with lost cost identification tags. The technology, which is designed to replace bar codes [12], comprises tags with numeric codes to uniquely identify each object by querying it wirelessly. It is envisioned that all objects that have a bar code today (e.g., books, clothing, food items) will have self-identifying tags in the coming years. This paper proposes a novel mobile multimedia application that lies at the intersection of two trends: the ubiquity of RFID tags and the availability of inexpensive digital cameras and camcorders with wireless networking capabilities. Specifically we propose equipping digital cameras with mobile RFID readers so that the combined recording device can capture an image stream as well as identities and locations of all RFID-tagged objects contained within each image. Users can query the video in real-time to determine the locations of RFID tagged objects in the captured video. Thus, a user can point the camera at a bookshelf and query for the location of a misplaced book, and the application responds by displaying an outline of the likely location in the captured video. Similarly, such a device can be used by mobile robots to conduct autonomous searches operations. Retail stores can use such devices to proactively generate missing object alerts upon detecting the absence of an object from a store shelf and display a video of the action that triggered the alert.

A key hurdle in implementing such an application is that RFID tags are *self-identifying* but not *self-locating*—upon being queried, a tag can report its identify but not its location. If multiple tagged objects are present within an image, it is not possible to distinguish between these objects or pinpoint their individual locations. To accurately pinpoint an object in an image (e.g., to pinpoint a misplaced book on a bookshelf), the application needs to determine both the object identity and its location. Although numerous locationing technologies exist such as GPS and ultrasound [10,13,14], it is generally infeasible to equip every tagged with a positioning device in addition to a tag due to reasons of cost, form factor and limited battery life. Thus, we must somehow exploit passive RFID tags to provide both location and identification information. The design of such a locationing technique for passive RFID tags is the primary contribution of this paper.

* Corresponding author. Tel.: +1 413 577 0850; fax: +1 413 545 1249.
E-mail addresses: liu_xiaotao@emc.com (X. Liu), corner@cs.umass.edu (M.D. Corner), shenoy@cs.umass.edu (P. Shenoy).

1570-8705/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2010.08.001

In this paper, we present a system called *Ferret* that uses a video camera and a mobile RFID reader to provide a locationing and identification support for pervasive multimedia. Specifically, Ferret uses the location and directionality of RFID *readers* to infer the locations of nearby *tags*. It exploits the mobility of users and observations of the same object from multiple vantage points to further refine the inferred object location. Object locations are displayed by drawing an outline of the probable location on the video display. The display is continuously updated as the user moves through the environment using a continuous stream of tag readings to refine the location.

We have implemented a prototype of Ferret and have used it to conduct a detailed performance evaluation. Our results show that (i) Ferret can refine object locations to only 1% of the reader's coverage region in less than 2 min with small error rate (2.22%); (ii) Ferret can detect nomadic objects with 100% accuracy when the moving distances exceed 20 cm; and (iii) Ferret works with a wide variety of user mobility patterns.

The rest of this paper is structured as follows: Section 2 presents the architecture of Ferret, while Section 3 presents our online RFID locationing algorithm. Section 4 presents our implementation and Section 5 experimental results. Finally, Sections 6 and 7 present related work and our conclusions.

2. Ferret architecture

In this section, we first present the basic idea behind Ferret, followed by a discussion of system requirements and assumptions, and the basic algorithm used by Ferret.

2.1. Basic idea

The basic operation of Ferret is illustrated in Fig. 1. As indicated earlier, Ferret employs a handheld camera with an embedded RFID reader, and records images as well as identities of nearby RFID tags for each recorded image. The user can pose a query for an object of interest, and the system in real-time updates the camera's display with an *outline* of the probable location of that object (see Fig. 1). Since the knowledge of the object location can be imprecise, Ferret displays a bounding box rather than a single point (the centroid) to depict the object location. The figure depicts a user looking for a soup can in an office. After scanning the room, the system displays a small region containing the soup can.

2.2. System requirements

A system such as Ferret imposes several design requirements.

2.2.1. Scalability

Since RFID tags will become ubiquitous in the coming years, it is likely that even a small room will contain hundreds of tagged objects (for instance, a bookshelf with tagged books or a closet with tagged clothing). It is essential that Ferret should scale to environments with hundreds of tagged objects—in particular, it should be able to continuously detect and improve the location of each sensed object by fusing a stream of noisy, and imprecise readings from the RFID reader to formulate a proposition of the object's location.

2.2.2. Locationing

Although RFID tags do not have any means to determine their locations, we assume that the camera contains a locationing device such as GPS. This is a reasonable assumption since the latest generation of cameras contain onboard GPS receivers to stamp each image with the GPS coordinates (to record the location where the picture was taken). Specifically we assume that the Ferret recorder contains a locationing device and a digital compass to its record the location and orientation (pan, tilt and roll), respectively.

The key insight in Ferret is to use the location and orientation of the camera/reader to determine the location of objects in its vicinity. In essence, whenever a tag is read by a reader, it follows that it is contained within the sensing range of the reader; thus it's location can be narrowed to a volume comprising the reader's range. Multiple observations from different vantage points allow the location estimates to be further refined. 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.

2.2.3. Location storage and refinement

In order to refine past estimates using new observations, Ferret must store a history of past observations. The simplest approach is to store past observations on the camera's storage medium (e.g., compact flash cards). Such local storage implies that each Ferret device must start finding objects with zero initial knowledge.

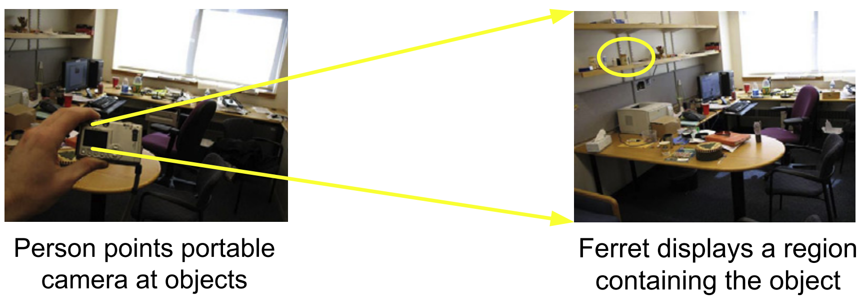
Person points portable camera at objects
Ferret displays a region containing the object

Fig. 1. Use of Ferret to discover the location of a soup can in an office.

Ferret also supports collaborative refinement of location estimates using data recorded by multiple recording devices. Sharing readings of the same object seen by multiple cameras enables faster refinement and more precise location estimates; however, it requires a means to share readings via a shared storage medium. Two approaches are possible. Since modern tags are rewritable and contain a limited amount of onboard storage, Ferret cameras can write back their location estimates to the tag (which can then be shared with other cameras). The limited amount of tag storage implies that a limited history can be maintained, and the approach does not work with read-only tags that are likely to be more common. A second approach is to employ a central database where cameras can upload their location information. 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 and requires privacy controls on the database.

2.3. Basic algorithm

Assuming the above design requirements, the basic Ferret algorithm involves a *recording* phase and a *query* phase. Recording involves four steps: (i) record image, (ii) record camera coordinates and orientation, (iii) record all tags within read range of the embedded reader, and (iv) compute (or refine) the location of each observed tag using the recorded camera's location and orientation. Clearly, step (iv) lies at the heart of Ferret and is discussed in detail in the next section. Querying involves three steps: (i) retrieve the location estimates for the object specified by the user query, (ii) overlay a bounding box on the captured image depicting this location estimate, and (iii) display the resulting image to the user.

Next we discuss Ferret's localization algorithm that can dynamically discover, update, store, and display object locations.

3. Online locationing for passive RFID

Consider an RFID reader that queries all tags in its vicinity—the reader emits a query signal which wirelessly powers the tags, and tags respond with their unique identifier. Given these tag responses, we propose a technique to determine and refine tags locations using the coordinates and orientation of the reader/camera. We begin with the basic idea behind our locationing technique, followed by the details. We also discuss how our system can handle nomadic objects—objects that move locations.

3.1. Locationing basics

Assume that the location and orientation of the camera/reader is known. When a tag is read by the reader, the reader produces a positive assertion that the object is present within its read range. Specifically, a reading indicates that the object is contained in the volume defined by the read range of the reader (see Fig. 2), and this yields an initial estimate of the object's location. Suppose that subsequently the user moves to a different location in the vicinity and the reader reads the same tag again. This additional reading can be used to refine the object location estimate; since the object must reside in the read ranges of the reader at both location, it follows that it lies in the *intersection* of the volumes representing the read range (see Fig. 3). The coverage region from each reading is intersected with all readings from the recent past, further reducing the possible region containing the object (see Fig. 3). Using this method, Ferret can continually improve its postulation of the object location so long the object remains stationary.

It is also possible to make negative assertions about an object's location. Thus, if an object is not read at a particular location, then it follows that it is not present in the reader's read range and that volume can be eliminated from the set of possible locations for the object. Although negative assertions can further narrow an object's location estimate, they are computationally more expensive than an intersection of two positive assertions. Since our goal is to implement Ferret on a mobile device, we a describe an online algorithm that only considers positive intersections to reduce computational cost.

3.2. Online locationing algorithm

For ease of exposition, we present a discrete version of our algorithm and then describe how to further reduce its computational using a continuous version.

Assume that the world is represented as a discrete grid where each coordinate represents a possible location of an object. Suppose that a reader reads an object from a certain location. Then all grid points within its read range have a non-zero probability of containing the object. This *coverage map* of a reader is shown in Fig. 2. As shown, not all points within the read range are equally likely to contain the object—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 grid points closer to the centroid have a higher likelihood of containing the object that those near the boundary. Let

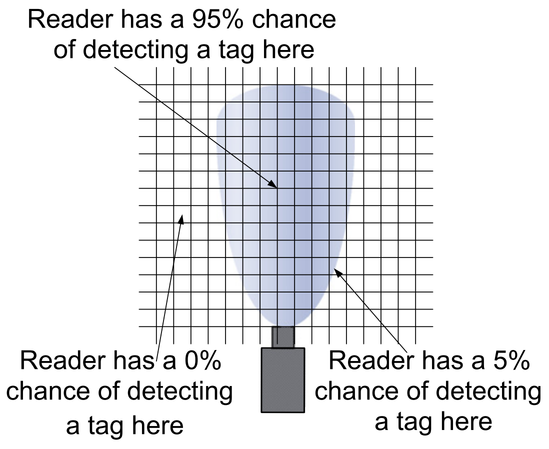
Reader has a 95% chance of detecting a tag here
Reader has a 0% chance of detecting a tag here
Reader has a 5% chance of detecting a tag here

Fig. 2. Coverage region of an RFID reader and tag detection probabilities in 2D.

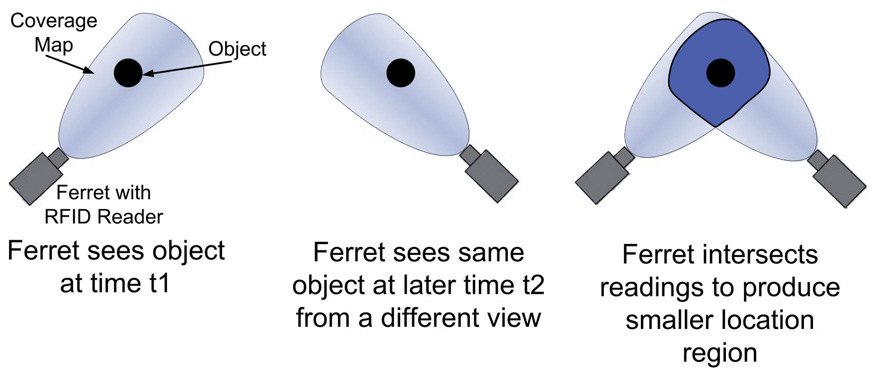
Coverage Map
Object
Ferret with RFID Reader
Ferret sees object at time t1
Ferret sees same object at later time t2 from a different view
Ferret intersects readings to produce smaller location region

Fig. 3. Refining location estimates.

$M(x,y,z)$ denote the probability that the object is located at grid point (x,y,z) within the read range; this map M can be determined from the antenna data sheet of the reader or from empirical calibration.

Our algorithm works 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 $M(x,y,z)$.

Observe that the coverage map for each new reading, as shown in Fig. 3, is represented in the *reader coordinate system*, which assumes that the origin is at the center of the reader's RFID antenna. The camera, although attached to the RFID reader, is offset from the reader, and has a slightly different coordinate system, which we refer to as the *camera coordinate system* and has its origin at the center of the camera's CCD sensor. To combine multiple readings from the reader, and subsequently display them to the user, each map M must be transformed into a common coordinate system. We refer to this as the *world coordinate system*. Without loss of generality, we assume the reader, camera, and world coordinate systems are left hand coordinate systems (see Fig. 4).

Before intersecting two readings, we must transform each to world coordinates. This transformation involves translation and rotation using techniques from linear algebra and computer graphics [4]. 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 $\alpha°$ (the rotation along the y axis, range $[-180,180]$), a tilt of $\beta°$ (the rotation along the x axis, range $[-90,90]$), a roll of $\gamma°$ (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. 4). This transformation is formulated as a rotation matrix:

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

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

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:

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \mathbf{R} \times \begin{bmatrix} x_w - x_0 \\ y_w - y_0 \\ z_w - z_0 \end{bmatrix} = \mathbf{R} \times \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T, T = -\mathbf{R} \times \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \quad (2)$$

where the composite rotation matrix $\mathbf{R}$ is given by Eq. (1).

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

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

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

When computing the intersection of coverage maps, Ferret first transforms the coverage map, M into the world coordinate systems using Eqs. (1) and (3), and computes the intersection according to the our algorithm to produce a new map L containing the likelihood of object locations.

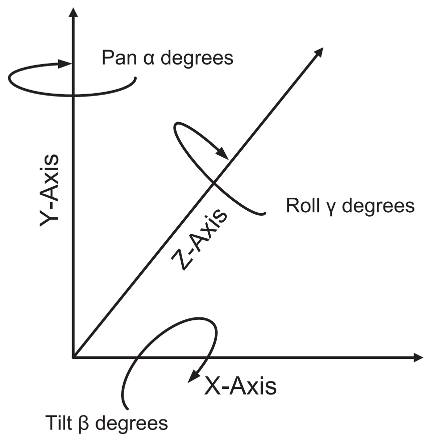

Fig. 4. Left handed coordinate system.

3.2.1. Displaying location estimates

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 chose s 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 Eqs. (1) and (2), 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:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{z_c} \times \begin{bmatrix} x_c \\ y_c \end{bmatrix} \qquad (3)$$

where u and v is the projection at the CCD sensor.

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.

3.2.2. Reducing computational overhead

Since the above discrete version of the algorithm may be too computationally intensive for small mobile devices, we present a continuous version that is simpler but at some loss of accuracy. 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.

Fig. 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 Fig. 5, we choose a constant y value $y1$. After rotating and translating the map M to match to the reader's

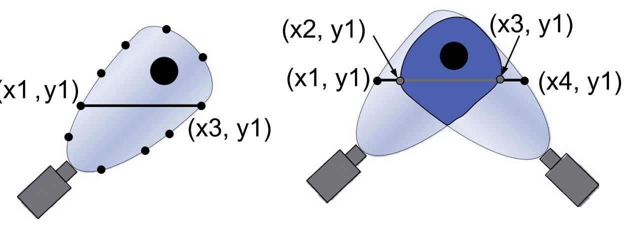

Fig. 5. Online location estimation.

current position, the system intersects the two line segments, $(x1,y1) - (x3,y1)$ from the current map L, with $(x2,y1) - (x4,y1)$ from the new map M. The resulting intersection is the segment $(x2,y1) - (x3,y1)$, 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 43.5 M bytes to 178 K 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.

3.3. Handling 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 continuous version, 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 th 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.

In contrast, discrete algorithm uses thresholding to eliminate low probability regions containing the object. It applied applying a likelihood threshold to the likelihood location map and removes 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 Fig. 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 30 dBm (1 W). This reader operates at the frequency range 909–928 MHz, 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 a 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 future 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 915 MHz.

A *Sony Motion Eye web-camera* connected to a Sony Vaio laptop. This CMOS-based camera is set to a fixed focal length of 2.75 mm, and uses a sensor size of 2.4 mm by 1.8 mm. The camera provides uncompressed 320 × 240 video at 12 frames-per-second.

Cricket [10] ultrasound 3D locationing 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.

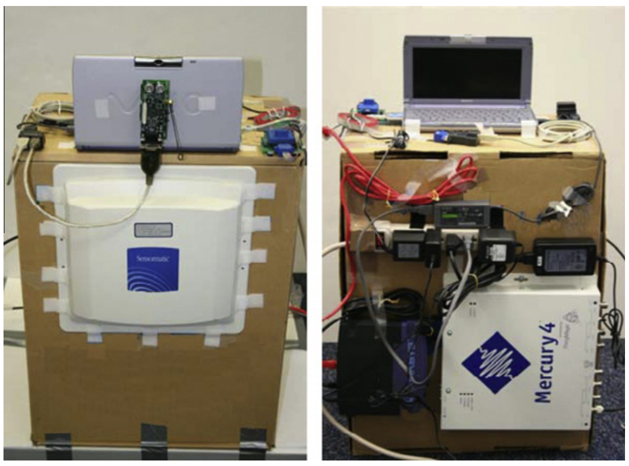

Fig. 6. Ferret prototype system.

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.

Our prototype consists of the following software modules:

4.1. 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 [2] to implement this module. We implement the projection function according to Eq. (3) 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.

4.2. 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 250 ms to the reader, and then the RFID reader periodically responds with configurable plaint text message including the tag ID, the ID of the antenna reading the tag, and so on.

4.3. 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.

4.4. Locationing module

This module implements our locationing algorithm. This implementation includes: (i) coordinate transformation functions between world coordinate system and the coordinate systems of camera and RFID reader according to Eqs. (1)–(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 also evaluate and compare the accuracy of the discrete and continuous versions, evaluate the ability to handle nomadic objects, and 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 results, a sphere with a volume of 0.01 m^3 has a diameter of 26.7 cm and a volume of 0.1 m^3 has a diameter of 57.6 cm. 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 4 m $\times$ 10 m $\times$ 3 m 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 continuous 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 2.56 m $\times$ 1.74 m $\times$ 2.56 m at the maximum and has a volume of approximately 2 m^3.

5.1. 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 Fig. 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 2 m^3 to 0.02 m^3 which is only 1% of the reader's coverage region in less than 2 min. 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* readings.

In addition to the volume size of the location estimation, we also plot the projection area versus time in Fig. 7c 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. Accuracy of our techniques

Next, we evaluate and compare the precision of the discrete and continuous methods in locating objects. While the continuous algorithm is useful for current mobile devices, the discrete algorithm uses a more precise representation of the object's location likelihood. To evaluate Ferret's precision in locating objects, we placed 30 tags in a 2.5 m $\times$ 2.5 m $\times$ 2 m region, and we move the prototype around the room for 20 min. 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 three experiments, Ferret can make between 0 and 90 location errors.

Before evaluating the discrete 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

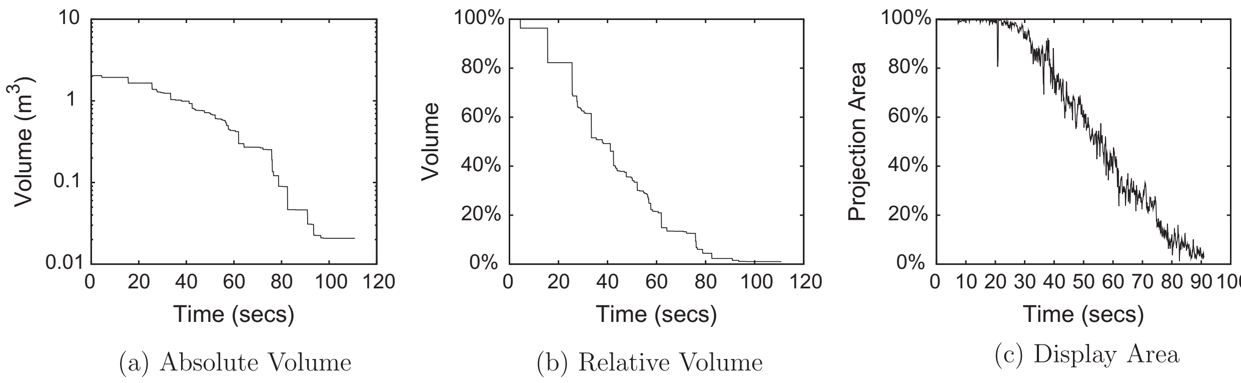

Fig. 7. Continuous refinement of location.

the object. However, a larger threshold will also increase the error rate of Ferret (the volume does not contain the object). In order to test the sensitivity of discrete Ferret to the change of likelihood threshold, we varied the likelihood threshold from 0.00001 to 0.4, and ran the discrete Ferret algorithm on the data we collected in the experiment. We show the results in Fig. 8.

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 $\leqslant$0.01, the number of errors does not 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 discrete algorithm and compare it to the performance of the continuous algorithm. In Fig. 9, we plot the CDF of Ferret's location accuracy for both algorithms.

The results show that (i) The continuous algorithm can localize an object in 0.15 m^3 and 0.05 m^3 regions with 80% and 50% probability, respectively. The 0.15 m^3 and 0.05 m^3 regions are only 7.5% and 2.5% of the reader's coverage region which is 2 m^3; (ii) The discrete algorithm outperforms the continuous algorithm by localizing an object in a 0.05 m^3 region with more then 90% probability and in a 0.1 m^3 region with 100% probability.

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

Fig. 8. Performance of Ferret under different likelihood thresholds.

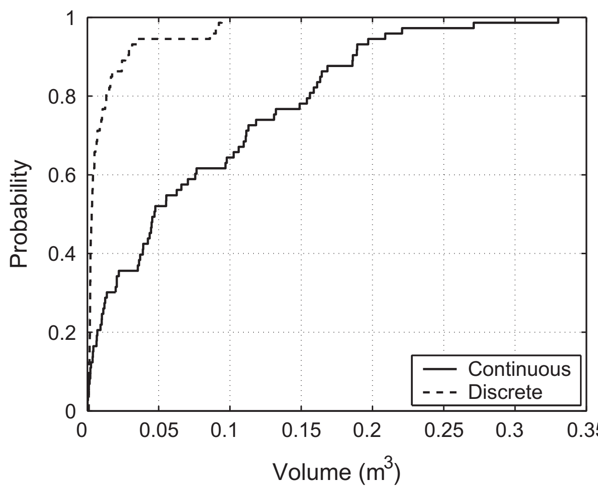

Fig. 9. CDF of locationing accuracy.

5.3. Tuning ferret

Due to the inaccuracies in estimating the location and orientation of RFID reader, Ferret might incorrectly estimate tag's location: tag is not in the region estimated by Ferret, and further Ferret might incorrectly reset the algorithm when the tag has not moved. In our experiments, the continuous algorithm has two errors and none reset, while the discrete algorithm has five errors and one reset. The larger errors of discrete algorithm is because the smaller volume of location map increases the chance of not containing object. We can compensate these inaccuracies by increasing the coverage region of RFID reader, and consequently, we are able to eliminate the errors. However, the increasing coverage region has larger volume, and therefore we pay the price of decreasing locationing accuracy. Our results in Table 1 show that as the volume size of coverage region increases from 2 m^3 to 5 m^3: (i) for the continuous Ferret, the number of errors drops from 2 to 0, and the mean volume of the location estimation increases from 0.077 m^3 to 0.30 m^3; and (ii) for the discrete Ferret, the number of errors drops from 5 to 0, the number of resets drops from 1 to 0, and the mean volume of the location estimation increases from 0.011 m^3 to 0.14 m^3.

5.4. 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 Fig. 10: (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 continuous and discrete Ferret. Our results are shown in Fig. 12.

The results show that Ferret performs similarly for each of the movement patterns; however the circular pattern

Table 1
Performance of Ferret under different volumes of coverage regions.

Volume of coverage region (m^3)	Continuous			Discrete		
	Num. errors	Num. resets	Mean volume (m^3)	Num. errors	Num. resets	Mean volume (m^3)
2	2/90	0/90	0.077	5/90	1/90	0.011
3	0/90	0/90	0.18	2/90	1/90	0.043
4	0/90	0/90	0.23	1/90	1/90	0.084
5	0/90	0/90	0.30	0/90	0/90	0.14

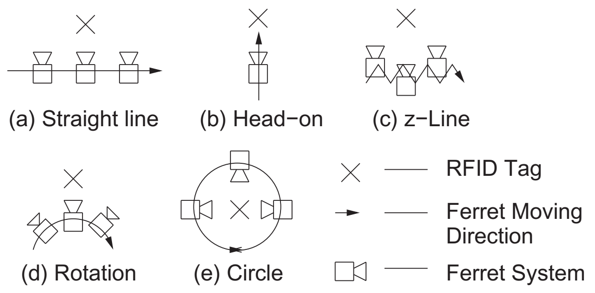

Fig. 10. Path of the Ferret device.

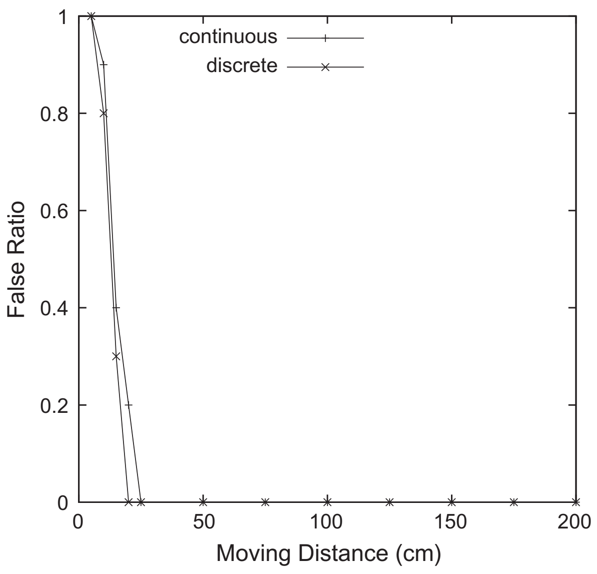

Fig. 11. Detected object movements.

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 discrete algorithm, due to its better locationing accuracy, outperforms the continuous algorithm, except in the case of the circular and head-on patterns, where the performance is similar.

5.5. 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 5 cm and 200 cm 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 did not detect a moved object. The results are shown in Fig. 11.

The figure shows that both the continuous and discrete versions can detect 100% object movements when the moving distance exceeds 25 cm and 20 cm, respectively. This is consistent with our previous results that show that Ferret can localize an object to within an region with a volume of hundredths of a m^3—this gives a radius on the order of 20 cm, 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.

5.6. Memory overheads

The prototype has a non-zero probability of detecting tags in balloon-shaped region, with maximum dimensions of 2.56 m $\times$ 2.56 m $\times$ 1.74 m—this shape has a volume of approximately 2 m^3. For the discrete algorithm we sample this coverage region every centimeter. As discussed in

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

Fig. 12. Performance of Ferret under various mobility patterns.

Section 3, the discrete algorithm requires every point in this space, while the continuous 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 discrete spatial requirements: (i) the discrete algorithm uses a **float** of four bytes to describe the probability of a sample point, and the total space is $256 * 256 * 174 * 4 = 43.5$ M bytes using a three dimensional array to store the probabilities of all sample points, and (ii) the continuous 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 $256 * 174 * 2 = 178$ K bytes. Both the discrete and continuous representations are highly compressible: the discrete can be reduced to 250 K bytes and the continuous representation to 5 K bytes using the commonly available compression tool *gzip*. For the foreseeable future, RFID tags will not contain enough storage for the discrete representation, while the continuous version is not unreasonable. If tags have more or less storage the number of sample points can be adjusted, although this will affect the precision of the system.

5.7. Computational requirements

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

6. Related work

Researchers have developed RFID-based indoor locationing systems [7,9] using active, battery powered, RFID tags. In SpotON [7], 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 [9] 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 [7,9] 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 [3]. 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 does not 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 [5], 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 [6] and Monte Carlo localization [1] 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 [11]. The 3D RFID system is equipped with a robot-controlled uni-directional 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 cannot 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.

The pervasive multimedia application envisioned in this work has similarities with our recently-proposed SEVA system [8]. SEVA assumes WiFi-based tags that are both self-identifying and self-locating—each tag has a unique identity and uses a Cricket [10] 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 Ferret. SEVA is based on active 802.11-based tags, while Ferret relies on passive RFID tags. Object tags in SEVA are assumed to have locationing capabilities and can determine their own positions. Ferret 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, Ferret requires the capability to capture and query video in real-time. Finally, SEVA scales to around a dozen *moving* objects, whereas Ferret requires the ability to scale to hundreds or thousands of *nomadic* RFID-tagged objects.

7. Conclusions

This paper presented 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 presented a novel algorithm to infer location of tagged objects using the location of a camera and reader that observes them; location estimates can be refined over time using multiple observations of an object. We also presented methods for detecting when nomadic objects move and to display and update object locations on a video camera screen. Our experiments conducted using a fully working prototype showed that (i) Ferret can refine object locations to only 1% of the reader's coverage region in less than 2 min with small error rate (2.22%); (ii) Ferret can detect nomadic objects with 100% accuracy when the moving distances exceed 20 cm; and (iii) Ferret is robust against different movement patterns of user's mobility.

We expect that future systems will build on the techniques presented in this paper, and make further improvements to the locationing algorithms. While a great number of hurdles exist in privacy issues, we contend that systems that leverage this ubiquity will provide untold utility to users.

References

[1] F. Dellaert, D. Fox, W. Burgard, S. Thrun, Monte carlo localization for mobile robots, in: Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA'99), Detroit, MI, May 1999.

[2] Ffmpeg 0.4.8. <http://ffmpeg.sourceforge.net/index.php>.

[3] K. Fishkin, B. Jiang, M. Philipose, S. Roy, I sense a disturbance in the force: long-range detection of interactions with rfid-tagged objects, in: Proceedings of UbiComp'04, September 2004, pp. 268–282.

[4] J.D. Foley, A.V. Dam, S.K. Feiner, J.F. Hughes, Computer Graphics: Principles and Practice in C, second ed., Addison-Wesley Professional, 1995.

[5] D. Hähnel, W. Burgard, D. Fox, K. Fishkin, M. Philipose, Mapping and localization with rfid technology, in: Proceedings of ICRA'05, April 2004.

[6] D. Hähnel, W. Burgard, D. Fox, S. Thrun, An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements, in: Proceedings of IROS'03, October 2003, pp. 206–211.

[7] J. Hightower, R. Want, G. Borriello, Spoton: an indoor 3d location sensing technology based on rf signal strength, Technical Report 00-02-02, University of Washington, 2000.

[8] X. Liu, M. Corner, P. Shenoy, Seva: sensor-enhanced video annotation, in: Proceedings of the 13th ACM Annual Conference on Multimedia (MM'05), Singapore, November 2005, pp. 618–627.

[9] L.M. Ni, Y. Liu, Y.C. Lau, A.P. Patil, Landmarc: indoor location sensing using active rfid, in: Proceedings of PerCom'03, March 2003, pp. 407–417.

[10] N.B. Priyantha, A. Chakraborty, H. Balakrishnan, The cricket location-support system, in: Proceedings of MobiCom'00, Boston, MA, August 2000.

[11] S. Roh, J.H. Park, Y.H. Lee, H.R. Choi, Object recognition of robot using 3d rfid system, in: Proceedings of ICCAS'05, June 2005.

[12] R. Want, An introduction to rfid technology, IEEE Pervasive Computing 5 (1) (2006) 25–33.

[13] R. Want, A. Hopper, V. Falcao, J. Gibbons, The active badge location system, ACM Transactions on Information Systems (TOIS) 10 (1) (1992) 91–102.

[14] A. Ward, A. Jones, A. Hopper, A new location technique for the active office, IEEE Personal Communications Magazine 4 (5) (1997) 42–47.

Xiaotao Liu received his B. Eng in Computer Science from the South China University of Technology, Guangzhou, China in 1998, and his M.S. and Ph.D in Computer Science from the University of Massachusetts Amherst both in 2006. He is currently a Senior Research Scientist in the CTO office of EMC Corporation. He worked as a Technical Consultant in Computer Associates Co., Ltd. from 1998 to 2001. His research interests include operating systems, computer networks and distributed systems.

Mark D. Corner is an Associate Professor in the Department of Computer Science at the University of Massachusetts Amherst. He graduated with his Ph.D in Electrical Engineering from the University of Michigan. His primary interests lie in the areas of mobile and pervasive computing and networking, file systems, and security. He was the recipient of an NSF CAREER award in 2005, Best Paper Awards at FAST 2007 and ACM Multimedia 2005, as well as the Best Student Paper Award at Mobicom 2002. Professor Corner serves on the editorial board of IEEE Pervasive.

Prashant Shenoy is a Professor of Computer Science at the University of Massachusetts Amherst. He received his Ph.D and M.S in Computer Sciences from the University of Texas at Austin and his B.Tech in Computer Science and Engineering from IIT Bombay. His research interests lie in operating and distributed systems, sensor networks, Internet systems, and multimedia. He heads the Laboratory for Advanced Systems Software at UMass where he presently leads projects in virtualization, cloud computing, and green computing. He has published more than 125 papers on these topics and has won several best paper awards at conferences such as USENIX, ACM Multimedia, and World Wide Web. He has been the recipient of the National Science Foundation Career Award, the IBM Faculty Development Award, the Lilly Foundation Teaching Fellowship, the UT Computer Science Best Dissertation Award, and an IIT Silver Medal. He is a senior member of the IEEE and a distinguished member of the ACM.