

# Sherlock: Automatically Locating Objects for Humans

Aditya Nemmaluri, Mark D. Corner and Prashant Shenoy  
Department of Computer Science  
University of Massachusetts Amherst  
Amherst, MA, USA  
{adityan, mcorner, shenoy}@cs.umass.edu

## ABSTRACT

Over the course of a day a human interacts with tens or hundreds of individual objects. Many of these articles are nomadic, relying on human memory to manually index, inventory, organize, search, and locate them. However, Radio Frequency Identification (RFID) tags hold great promise for automating these tasks. While originally envisioned for managing supply chains and store inventories, RFID tags support the properties necessary for helping humans to manage their objects. This paper presents Sherlock, a system that leverages RFID tags for human-object interaction. Sherlock combines concepts from sensors, radar technology, and computer graphics to implement a novel localization and visualization system for everyday objects. At the heart of Sherlock is a new RFID localization technique that uses steerable antennas to “sweep” a room, discovering, localizing and indexing tagged objects. In response to user queries, Sherlock displays the locations of matching objects using images from a video camera. We have implemented a prototype of Sherlock to conduct experiments in a real office environment. Our results demonstrate the effectiveness of Sherlock in localizing to a volume of less than 0.55 cubic meters for 90% of objects.

## Categories and Subject Descriptors

C.3 [Special-Purpose and Application-Based Systems]: Real-time and embedded systems

## General Terms

Algorithms, Design, Experimentation, Human Factors, Measurement, Performance

## Keywords

RFID, Localization, Pervasive Computing, Ubiquitous Computing, Multimedia, Augmented Reality

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiSys'08, June 17–20, 2008, Breckenridge, Colorado, USA.  
Copyright 2008 ACM 978-1-60558-139-2/08/06 ...\$5.00.

## 1. INTRODUCTION

Each day a human interacts with tens, hundreds, or thousands of objects from papers and books to keys and coffee cups. As people move these objects, they become nomadic, making them easy to misplace and forget—relying on human memory to manually index, inventory, organize, search, and locate physical possessions is both tedious and error-prone.

As search engines have solved this problem for personal data, we should extend these systems to enable people to automatically index, catalog, organize and query their personal belongings. Applying these search and query benefits to the physical world will enable many new application domains. For instance, users will be able to correlate the use and movement of objects for activity inference [14] and employ systems that help with everyday tasks such as cooking or assisted living for the elderly [12].

Radio Frequency Identification (RFID) [1, 15] technology holds great promise for enabling such applications. Although designed as an object identification technology for managing supply chains and store inventories, RFID tags provide the properties—small-size, maintenance-free, and low-cost—necessary for helping humans to locate and manage their objects. Each RFID tag is a passive sensor containing a numeric code that uniquely identifies the object; a tag can be wirelessly powered by a reader and queried for its numeric ID. Since RFID tags are inexpensive—costing a few cents each when manufactured in large quantities—it is conceivable that every object that carries a bar-code today (e.g., books, clothing, food-items) will be equipped with self-identifying RFID tags in the near future.

RFID supply-chain applications already employ coarse-grain locationing—by using the location of a reader to approximate the location of an object. However, personal RFID applications, such as locating a particular book on a shelf, require finer-grain location information. However, inferring fine-grained location information from passive RFID tags is a relatively nascent area of research. In fact, no system to date has attempted to provide fine-grained localization for RFID tags in a realistic environment, such as a home or office setting where object occlusions and signal interference can significantly complicate the locationing problem. Our own preliminary work in the Ferret system assumed an idealized laboratory setting and depended on a user to constantly move the RFID reader [10]. Other systems use similarly artificial tag setups, with robots providing mobility [3, 13].

This paper presents a system called *Sherlock* that we have designed to support indexing and querying of physical pos-

sessions. Sherlock consists of two key components: (i) an RFID object locationing system that can pinpoint and automatically update the locations of objects, and (ii) a query and visualization system that accepts user queries for object locations and displays the results on an image of the environment. At the heart of Sherlock is a new RFID localization technique that uses *steerable antennas* inspired by pan-tilt-zoom (PTZ) cameras and steerable radars. Sherlock uses these steerable directional antennas to “sweep” a room for objects and their locations.

We have implemented a prototype of Sherlock and have used it to conduct a detailed evaluation. We have deployed Sherlock in an actual office environment with more than one hundred tagged objects. This office environment contained a variety of materials and objects that cause absorption and reflection of RF, including metal, paper, liquids, carpeting, ceilings and furniture. As we show in this paper, the performance in a controlled setting is drastically different from the office setting and Sherlock has been designed to counter these challenges.

## 2. BACKGROUND

Automatic object identification and locationing is key to human-object interactions and Radio Frequency Identification (RFID) is well suited to these tasks. RFID tags are remotely readable identification tags intended as a replacement to product barcodes. Tags can be broken into two types: *passive* and *active*. Passive tags typically contain a patch antenna and a chip that contains a small amount of processing and storage. Active tags contain these elements plus a battery to improve the range and processing power of the tag. The RFID reader uses its antenna to remotely energize the tag which then responds to the RFID reader’s request, typically for the unique identifier of the tag. RFID systems are asymmetric: Tags are cheap and have small antennas, while readers tend to be more expensive and have much larger antennas.

**Active RFID Localization:** In comparison to localization using other technologies [4], there has been relatively little research done on the localization of RFID tags. The earliest work was conducted on active RFID tags, which have much stronger, more predictable signals. For instance, SpotON [5] uses signal strength and triangulation of active tags for localization. Such techniques are less feasible with passive tags that lack a power source, which results in a weak, unpredictable signal that is less amenable to triangulation techniques. Similarly, the LANDMARC system uses active RFID tags for localization [11]. LANDMARC uses the notion of reference tags (whose locations are known a priori) and measures the tracking tag’s nearness to reference tags by the similarity of their signal received at multiple readers.

**Passive RFID Localization:** Passive RFID tags provide notable advantages over active tags. Due to their low cost, large scale deployments are possible where *every* possible object in a building is tagged. Further, although active tags are computationally more capable and have longer range, they require a power source in the form of batteries. The need to periodically replace batteries makes large scale deployments of active tags cumbersome.

The earliest application of passive RFID tags for localization was for the simultaneous location and mapping (SLAM) problem [3]. After placing reference tags, a robot equipped

with a laser scanner and RFID reader maps the environment. The RFID tags serve as solid reference points, rather than something to be localized. The work does show how to build a probabilistic model of the antenna pattern by measuring it empirically. Using this model the robot builds a probabilistic model of the tag location.

A recent commercial system, called RFID-Radar from Trolley Scan, can locate passive tags outdoors using angle of arrival (AoA) and time of flight techniques. However, RFID-Radar uses Tag Talks First (TTF) passive tags, whereas most manufacturers and standards are now using Reader Talks First (RTF) tags [2]. RTF tags are more flexible, as the protocol is driven from the reader and not from the tags, however TTF tags can be read much more often. Multiple, very fast reads of the tag are essential to establishing a time of flight distance measurement, as this requires thousands of reads of the same tag, uninterrupted by obstructions, and limited to 50 tags at a time. Also, using time of flight measurements requires careful control over all the parameters of the system, including the tag design and antenna cable lengths. Worse yet, reflections will corrupt any range measurement making this unsuitable for indoor use. Sherlock uses modern RTF tags, and does not depend on tight timing constraints. The disadvantage of RTF tags that they take longer to read, necessitating efficient scan strategies as described in this paper.

The closest work to our own is the 3D RFID system [13] which also uses a robot to control the mobility of the RFID reader. By using a combination of six tags on every object, the system can estimate the position and orientation of the object. This technique has significant disadvantages, since many objects can not accommodate six tags in different orientations.

**Ferret:** Sherlock builds on our own previous work on the Ferret system [10]. Using a Ferret system, a user walks around with a camera equipped with an RFID reader. The device records images and RFID tag identities. The camera/reader uses a separate locationing system to determine its own location, and RFID tags are localized at the granularity of reader locations (e.g., the tagged object was detected in the vicinity when the camera was at location  $(x, y, z)$ ). Over time, as the user detects the same object from different vantage points, the object location can be refined by intersecting the various readings. The refinement process can be cumbersome, since it requires user mobility to detect the tag from multiple vantage points.

The primary differences between Ferret and Sherlock lie in how the two systems are deployed. While Ferret is meant for handheld readers, Sherlock is infrastructure-based, using readers fixed in one location, capable of controlling their own movement. This gives rise to the one of the primary contributions of Sherlock, which is to control the scan strategy to detect and localize objects. The second primary difference between this work and Ferret is that our experiments were conducted under realistic conditions—in this case an office. Ferret, as well as all other prior work on passive RFID localization, has been done under idealized conditions with tags placed in controlled locations. Conditions in real environments such as homes and offices are substantially different from idealized settings—interference and low read rates due to nearby metallic objects, dynamic changes to the reader range due to surrounding objects, and object occlusions and reflections are all common in such settings.

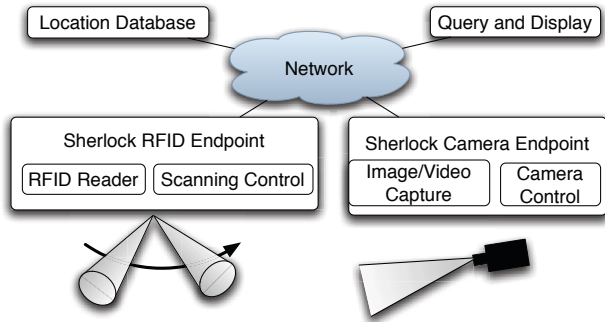


Figure 1: Sherlock Architecture.

### 3. SHERLOCK ARCHITECTURE

The goal of Sherlock is to enable human-object interactions by automatically detecting, localizing, and indexing objects present in an environment, and enabling a search and visualization interface to query this information. Sherlock assumes a world where RFID-tagged objects are pervasive—every object of interest is assumed to be tagged with low-cost passive RFID tags. Sherlock uses a network of RFID readers and pan-tilt-zoom (PTZ) cameras deployed in areas of interest. Such large scale deployments of readers are already underway [16, 7]. The goal of the readers is to detect each object, localize its position and update locations whenever nomadic objects move. The goal of the PTZ cameras is to point the camera in the vicinity of an object of interest and display an overlay of the object’s location.

We have designed Sherlock as a scalable, distributed system for computing, recording, and displaying the refined locations of a large number of objects. Shown in Figure 1, Sherlock consists of four key components:

- *RFID Endpoint:* Each RFID endpoint comprises an RFID reader with a steerable directional antenna. The RFID endpoint is responsible for continuously scanning the environment, to query for RFID tags, and to determine the tag locations. It implements multiple scan strategies, steering each antenna to determine the identities of new or moved objects and to perform localization. Each RFID endpoint operates independently of any other RFID endpoints.
- *Camera Endpoint:* Sherlock supports multiple camera endpoints. Each Sherlock camera periodically scans the location database for objects that are in within its field of view. When detecting a new object, an object that has not been photographed for some time, or an object that has moved, the camera endpoint takes a picture of the object’s location and inserts it into the database.
- *Location Database:* The location database sits between the scanner and the query components. It contains a list of all tagged objects as well as their current and recent locations. The scanner continuously updates the locations in the location database, which fuses multiple readings from multiple RFID endpoints.
- *Query and Display:* The query and display component implements a simple browser-based query inter-

face that allows a user to query for the object location and displays an image with location information overlaid onto it (see Figure 10).

Using this architecture we are interested in addressing the following questions:

- How can we efficiently localize each object using the steerable antenna? How can nomadic objects be handled? That is, what scan strategies are best suited for fast, efficient localization, and how should the system detect new objects or when a resident object moves?
- How should Sherlock counter real-world effects such as low read rates caused by interference and dynamic changes in the reader range?
- How should observations from multiple antennas be fused to improve the accuracy and effectiveness of the system?
- How can we design techniques that scale to tens or hundreds of objects that may be present in a room?
- How should the PTZ cameras be exploited to design a visual interface for querying and searching objects?

### 4. SHERLOCK SCANNER DESIGN

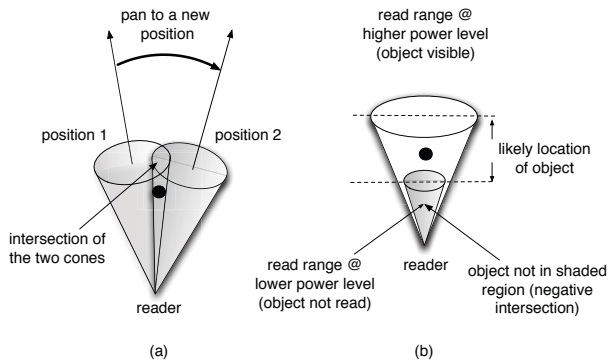
The design of Sherlock focuses on three key issues: accurate localization of object locations, efficient scanning of the environment for new objects and objects that have moved, and fusing results from multiple antennas. In this section, we present an idealized design of the localization system, while the next section focuses on incorporating real-world effects.

#### 4.1 Idealized Localization

Directional steerable antennas are key to fine-grain localization in Sherlock. To understand how passive RFID tags are localized, assume that the read range of the reader—essentially the antenna lobe—is approximated as a cone as shown in Figure 2(a). Note that this is purely an approximation, as antenna lobes are generally not cone shaped and may contain side lobes. Later sections of the paper use highly conservative assumptions on the lobe size to prevent incorrect localization.

Consider an object that is read by the reader from a particular antenna position. It follows that the object must be present in the read range (i.e., the cone) of the reader in order for it to be read. If the antenna were not steerable, this coarse-grain information would be the best estimate of the object’s likely location. The ability to steer the antenna enables us to significantly improve this location estimate. In particular, suppose that the antenna pans to a different position and can still read the object from that position. Since the object must be present in the antenna cones at both positions, it follows that it must be present in the *intersection* of the two cones—see Figure 2(a).

Thus, by steering the antenna in the pan and tilt dimensions, we obtain a sequence of positions from which the object can be detected. If we take the two extreme positions in each dimension where the object is still visible, the intersection of these allows Sherlock to narrow the possible region containing the object to a smaller volume.



**Figure 2: Localizing a tag using observations from two pan positions and refining a tag location by varying the antenna power level.**

Although we have only discussed the pan and tilt dimensions, the same intuition applies for localizing the object in the “zoom” dimension. If an object is visible at a certain power level but can not be read at a lower power level (which results in a smaller cone), it follows that the object resides in the larger cone but not in the smaller one. Thus, the region belonging to the smaller cone can be eliminated from the larger cone, allowing the object location to be refined in the zoom dimension (see Figure 2(b)).

Thus, the Sherlock scanner sweeps a room by varying the pan, tilt, and power level of the antenna to determine the objects present in the environment and estimate their locations. To be effective, the scan strategy must minimize the amount of time to detect new objects and to localize each one. Since the latency of reading tags is appreciable, efficient antenna steering is a crucial design element. Further, the strategy must scale to a large number of objects that may be present in an environment by concurrently detecting and localizing multiple objects in a single sweep. Next we present the scan algorithms employed by the Sherlock scanner to achieve these goals.

## 4.2 Scan Strategies

The scan algorithm in Sherlock is based on three building blocks: *coarse*, *localize* and *fast* scans.

*Coarse scan:* The coarse scan is used to sweep the environment to detect which objects are present and to give a rough idea of where objects are located. In a coarse scan, the overall region covered by the pan-tilt antenna is divided into  $N$  coarse positions. Each coarse scan region can read tags in a region that equals the size of the antenna beam width at the highest power level.<sup>1</sup> In a coarse scan, the antenna is pointed towards the center of each coarse region and all objects present in that position are read; the process repeats until all coarse positions are scanned. A coarse scan narrows each visible object down to a coarse region and does not attempt to refine the object location any further. For instance, if the beam width of the antenna is 60 degrees, and Sherlock detects the object in a single coarse scan position, it knows that the object is within that 60 degree cone.

*Localize scan:* A localize scan is used to determine the location of a *single* object of interest. Consider an object

<sup>1</sup>In Section 5.2, we will argue that these coarse regions must overlap with one another to improve accuracy.

that has been narrowed down to a particular coarse scan position. A localize scan performs *linear search* of the coarse scan position in small steps to localize that particular object. Consistent with the idealized localization algorithm, the goal of the search algorithm is to find the two extreme angles at which the tag can still be detected. Starting with the extreme left position of the coarse region, Sherlock incrementally pans the antenna in increments of  $\theta$  degrees until it can no longer detect the object. It performs the same procedure for the tilt direction, in increments of  $\phi$  degrees along the axis formed by the median of the two extreme pan angles—this effectively looks like a plus sign. It then centers the antenna at the middle of the plus sign and lowers the transmission power until the object is no longer visible. As an example, if the coarse region is a 60 degree cone, the localize scan could use seven, 10 degree pan increments to discover the extreme angles that the object can be detected. It would then repeat the procedure with seven 10 degree tilt positions for a total of 14 readings, plus the number of power settings.

*Fast Scan:* The current incarnation of Sherlock employs a mechanically steerable antenna in lieu of an electronically steerable one. In both the coarse and localize scans, a motor steers the antenna to a particular position and *stops*; the reader then queries the tags visible at that location, after which the motor starts up again to steer the antenna to the next position and so on. There is a mechanical latency associated with starting up the motor from a stationary position—which is incurred every time the antenna pans (or tilts) from one position to the next. Further, querying passive RFID tags at each position incurs a significant latency both from the time needed to energize the tag, as well as the completion of collision resolution protocols. While reducing the query latency requires RFID technology improvements, we can optimize the latency incurred due to the motor positioning time during a coarse scan. We do so by adding a *fast scan* phase to the system.

The basic idea behind a fast scan is to “scan while panning”—a fast scan involves a slow sweep of the environment and the reader reads continuously while the antenna is panning, unlike a coarse scan which involves stopping at each position prior to a reader query. In a fast scan, the overall region covered by the pan-tilt antenna is divided into  $n$  layers. A layer corresponds to the entire coverage area of the antenna for a given tilt level. Sherlock moves the antenna from one extreme pan end of the layer to the other extreme pan position of that layer. While a fast scan can detect the existence of tags, it gains little information about the location of the tag, only the particular tilt level where the tag was found. For tilt levels where no tags are detected, the coarse scan for that level can be skipped, reducing the overall latency of the coarse scan. Thus a fast scan quickly determines which objects are present in the environment and directs the coarse scan towards regions where they are likely to be present. Note that future incarnations of Sherlock will use electronically steerable antennas and the fast scan will not be necessary.

### 4.2.1 Efficient Scan Algorithm

Sherlock employs an overall scan algorithm that combines fast, coarse and localize scans to find tagged objects in the environment. The limiting factor in scanning the environment is the time to read the tags. When the antenna is

pointed in a particular direction it can read all of the tags, up to 200 hundred, within the cone, but it may take as long as half a second with current hardware. Because of this, Sherlock’s objective is to provide a balance between latency of detection and the accuracy of localization objects while scaling to a large number of objects. Using the individual scan strategies we have described, one can construct a host of overall scan strategies to incorporate other design concerns. For instance, if the latency in initially detecting new objects is paramount, then coarse scans should be frequent, or if certain objects need to be localized with low latency, then those should be localized first. Also, the scan strategy can be adjusted to support multiple users and applications simultaneously, for instance using a utility-driven scan strategy [8].

The current scan algorithm operates in rounds, each comprising a fast, a coarse and a localize scan. The algorithm begins with a fast scan to quickly detect tilt-levels where objects are present and performs a coarse scan at each such tilt level. The coarse scan yields a list of all objects present in the environment and the coarse scan positions where they reside. By comparing this list with the objects that were present during the previous coarse scan, Sherlock can determine which new objects have appeared in the environment and if a previously present object has moved to a different coarse scan position. After finding the coarse positions of objects, Sherlock initiates a localize scan for every new object or any object that has moved. After completing this sequence, Sherlock begins a new round with a fast scan. If no new objects are detected in a fast and coarse scan, then the localize scan can be skipped in that round.

For better scalability, all objects present within a coarse scan region are localized concurrently in a single sweep, rather than in separate localize scans. This is done by beginning at the extreme left (top) position of the coarse scan region and panning (tilting) in increments of  $\theta$  ( $\phi$ ) to determine the first and last positions where each object of interest is visible. This localizes all resident objects in a single pan and a single tilt sweep of the coarse scan region, enhancing scalability. For instance, if two objects are in exactly the same position, it requires the same number of readings as if there was only one object.

### 4.3 Handling nomadic objects

A nomadic object is one that is normally stationary but can change locations (in contrast to moving objects that are continuously mobile). Sherlock can handle nomadic objects but does not handle moving objects—moving objects need continuous tracking support, which is beyond the capabilities of the current generation of RFID readers and passive tags. The coarse scan in each round is responsible for handling nomadic objects—it detects any object that has moved from one coarse region to another. A localize scan then computes the new location estimate for each such object.

However, Sherlock needs to consider one additional scenario—*nomadic objects that move between locations within the same coarse region*. These are objects that have moved slightly but remain in the same coarse region. By itself, a coarse scan is not able to differentiate between a stationary object and a nomadic object that has moved to another location in the same coarse region—both types of objects remain visible in the same coarse region as before. To handle this case, Sherlock records the time when each object is localized. It forces

a re-localization of an object after a threshold time period elapses even if the object remains in the same coarse region (normally, previously localized objects are not subjected to a localize scan if they stay in the same coarse region). By forcing a localize scan every so often, nomadic objects that move slightly can be re-localized.

### 4.4 Fusing Multiple Readings

Each localize scan produces a three dimensional volume which is an estimate of where the object is likely to be present. If multiple antennas can detect the same object, then Sherlock can fuse multiple results to yield improved localization accuracy. To do so, the volume computed by each antenna for the object must be intersected. Sherlock employs techniques from computer graphics (and computational geometry) to perform fast convex polyhedron intersection. To do so, it converts the surface of each polyhedron into a generalized triangulated surface. The two polyhedrons are then intersected by considering each surface of the first polyhedron and clipping the second polyhedron with this surface. The output of this process yields a third polyhedron which is stored in the location database and intersected with other antenna readings if more than two RFID endpoints can detect a particular object. Currently Sherlock employs the GNU Triangulated Surface Library (libgts), an open-source implementation of several standard 3D intersection algorithms from the Computer Graphics literature to perform polyhedron intersections [9].

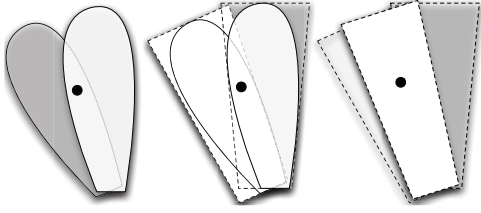
## 5. HANDLING REAL-WORLD EFFECTS

A key design goal of Sherlock is to ensure usability in real environments such as homes and offices rather than idealized lab settings. This section describes several real-world effects that arise in such environments and techniques employed by Sherlock to counter them.

Several factors conspire to make localization less than ideal. First, the antenna coverage pattern is not a perfect cone as we have assumed. Second, reflections in realistic environments cause the size of actual antenna lobe to vary dynamically and differ from the assumed antenna pattern. Third, factors such as absorption, partial occlusions, and tag orientation can all reduce the probability that a tag will respond to a reader query, thereby introducing errors and missed objects. We address each issue in turn.

### 5.1 Handling Antenna Idiosyncrasies

In reality, the antenna coverage pattern is not a regular constructive form, such as a cone, but rather a somewhat irregular balloon shaped lobe, with smaller side lobes. This implies that the system builder must empirically measure the antenna pattern of the reader to create a descriptive model. Empirical model construction involves placing tags in a number of locations within the three dimensional space in front of the antenna and measuring the probability of reading each tag [6, 10, 3]. The resulting descriptive model then takes one of two forms. The first is a three-dimensional matrix of the probability of reading a tag at each three-dimensional point in the vicinity of the reader [10, 3]. The second is a conservative boundary that encompasses the area where the read rate is non-zero [10]. This model is represented as the series of points that describe the surface of the antenna pattern.



**Figure 3: Sherlock simplifies the antenna pattern to a frustum (a trapezoid in two dimensions is shown for simplicity).**

In either case, localization involves computing the intersections of the modeled lobes at overlapping positions, which can be computationally intensive. The conservative descriptive model incurs an  $O(n^2)$  intersection overhead, where  $n$  is the granularity of the measurements, as it must iterate across two dimensions computing line intersections (each edge of the first lobe is intersected with the surface of the other lobe) [10]. The probabilistic model is worse, requiring  $O(n^3)$  time since each point must be examined to determine if it is contained within the points describing the other lobe.

To scale to thousands of objects, Sherlock seeks to reduce the complexity of the localization algorithm. Instead of using a descriptive model, Sherlock uses a *constructive* geometric model that bounds the antenna pattern with a three dimensional frustum (in two dimensions a frustum is a trapezoid such as the one shown in Figure 3). Computing the intersection of two frustums is independent of the size of the area, and can be computed in fixed time. The frustum used in Sherlock is the smallest frustum that bounds the empirical measurements of the antenna pattern from the descriptive models (see Figure 3). In the rest of the paper, we consider the angular size of the frustum to be the angle formed by intersecting the two sides of the frustum.

As explained earlier, Sherlock modulates the output power of the reader to localize the tag in the zoom dimension. It begins with the highest power setting and steadily reduces the power until the tag can no longer be detected. Using empirical measurements, we have created a conservative set of frustum boundaries that describe the coverage for each power output level. These discrete settings yield a discrete number of frustums for the possible location of an object, which are used to perform localization. An idiosyncrasy of our current reader is that it only supports four discrete output levels despite appearing to allow fine-grain variations in the power settings. If the power were tunable in finer steps, such as using power attenuation [6], the latency imposed by trying to repeatedly read the tag at various power levels would greatly slow the process of localization. Given finer steps, Sherlock would need to balance scan speed with localization accuracy.

## 5.2 Dealing with Adverse Conditions

A number of factors, such as multi-path, absorption, partial occlusion and tag orientation can all reduce the probability of a tag responding to a reader query. Sherlock’s scan algorithm incorporates a number of techniques to counter these effects.

Low read rates due to these effects are all handled by reading each position multiple times, which increases the probability of a tag response. Specifically, the coarse scan

employs *overlapping* coarse regions rather than mutually exclusive ones. In the current setup, each coarse position covers a region that is approximately equal to half the size of the lobe at the highest power level. Since the lobe is approximately 60 degrees wide, this implies that successive regions overlap with one another by 30 degrees in the pan and tilt dimensions. This overlap handles the boundary case since objects present at the boundary of the lobe tend to exhibit poor read rates. It also handles low read rates caused by effects such as absorption and partial occlusions, since each position is queried at multiple pan and tilt levels, increasing the probability of a tag reading. Similarly, localize scans can issue multiple queries at each position to improve read rates.

The orientation of the tag relative to the antenna can also have a significant impact on read rates. Sherlock uses a circularly polarized antenna, so the highest read rates are achieved whenever the tag directly faces the antenna, and read rates tend to be poor when the tag plane is perpendicular to the antenna. Sherlock handles this case by employing multiple antennas that cover each region from different vantage points. In particular, if a region is covered by two overlapping antennas that are placed perpendicular to one another, this ensures that at least one antenna has a good view of tag. Another technique is to place multiple tags on different surfaces of an object; e.g., a tag can be placed on the spine of a book and on the cover of the book. Multiple tags increase the probability of reading at least one tag regardless of the orientation of the object.

## 5.3 Dynamic Beam Width Estimation

Recall that in localizing an object, the antenna pans across an area, measuring the angle where it first detects the RFID tag and the angle where it lasts detects an object. We refer to the difference between these angles as the *measured tag beam width*. This is shown in Figure 4(a). If we assume that the RFID antenna has an antenna coverage pattern with an *assumed antenna beam width*, then in the ideal case these two beam widths are equal. In this ideal case, the object lies directly on the right-hand-side of the first pan position, and the left hand side of the last pan position where the object was detected.

However, as RF can reflect off surfaces, the RFID reader may detect tags in locations other than within the assumed antenna beam width. Since an object that would be normally outside the antenna pattern is visible due to RF reflection, the measured tag beam width is wider than the assumed antenna beam width. Figure 4(b) demonstrates how an object is read due to RF reflection even though it is not contained within the “assumed” lobe, causing the actual lobe to appear to be significantly wider than the assumed. Similarly, multi-path or absorption effects can attenuate a portion of the signal causing the object to be invisible at certain positions; this results in an actual beam that is narrower than the assumed beam as shown in Figure 4(c). To simplify later discussion, we have broken these cases into three regimes shown on Figure 4.

To experimentally demonstrate these effects, we measured the angular beam width for 30 tags arranged in a near-ideal setup: The tags were hung from string in free space in a spaced grid. We used an antenna with an approximate beam width of 60 to 70 degrees. We applied Sherlock’s localization algorithm to the tags and determined the error rate: the per-

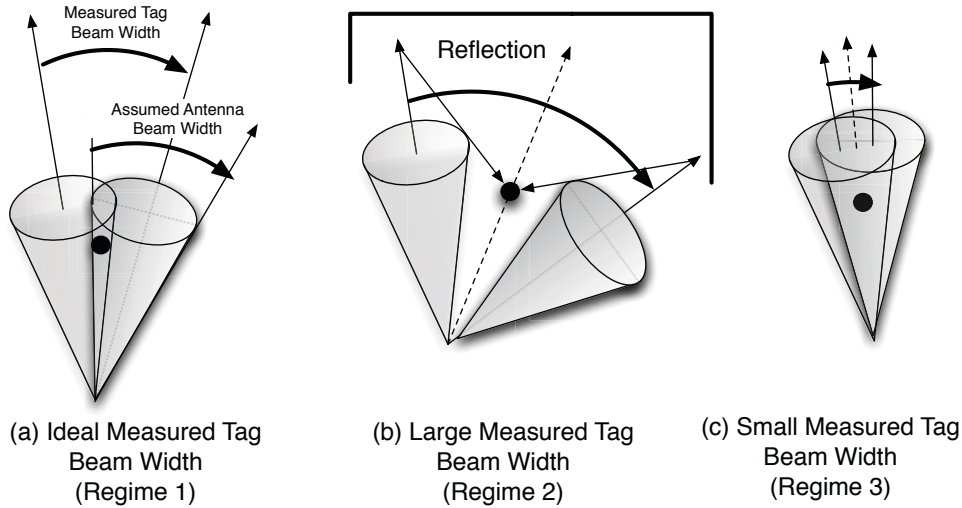


Figure 4: (a) This shows the ideal measured beam width for a tag, which is approximately equal to the antennas beam width. (b+c) These two figures show the two kinds of errors while localizing tags.

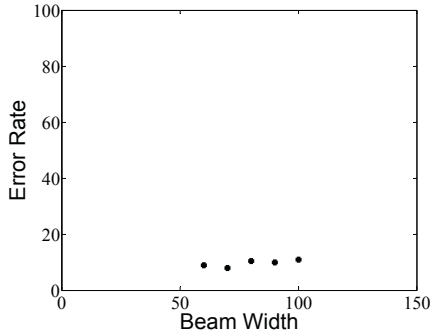


Figure 5: The antenna beam width measured under near-ideal circumstances.

centage of tags that were not actually in Sherlock’s resolved volume. The results in Figure 5 are broken down by the measured beam width. Next, we conducted the same experiment in an office room with 100 tags attached to a variety of objects, including books, doors, coffee mugs, staplers, etc. The measured error rate for each lobe width for the office experiment are shown in Figure 6(a), while Figure 6(b) plots a histogram of different observed beam widths.

In the case of the ideal experiment, the angular beam width varies between 60 and 100 degrees with an error rate less than 10%, regardless of beam width. These results are typical of our experimentation in ideal laboratory conditions with Ferret [10]. However, in the realistic experiment, the observed beam width varies from as small as 20 degree to as large at 160 degrees. Thus, there are many cases where the object remains visible when the antenna pans by 160 degrees, and many cases where it disappears from view when the antenna moves by as little as 20 degrees. Thus, if Sherlock were to assume an ideal beam width of 60 degrees for these cases, then large localization errors would result.

Since the lobe pattern can vary dynamically even within a given environment, Sherlock employs a dynamic beam width

estimation technique to improve localization accuracy. The basic idea is to *measure* the angular width of the antenna beam during localize scan, rather than assuming a fixed lobe beam. Thus, the system calibrates itself during localization. For all measured beam widths, Sherlock uses the median of the two extreme angles, and applies a frustum for a beam width of 10 degrees and adds  $\alpha$  degrees to each side of the frustum. This is shown in Figure 7.

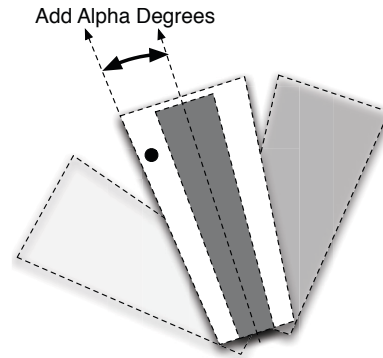


Figure 7: For objects that are difficult to localize due to measured beam widths that are too small or too large, Sherlock adds *alpha* degrees to the median of the two extreme positions. A larger alpha will encompass the object more often but increase the volume to which it is localized.

Using the data collected from our office environment, we empirically evaluated how many degrees to add versus the resulting error rate. This is shown for each regime in Figure 8. Sherlock uses a 50 degree beam width for regime 1, a 70 degree beam width for regime 2 and a 100 degree beam width for regime 3. The downside is that by adding area to the frustum, we have made the localization volume larger. However, we consider accuracy to be much more important than localization performance.

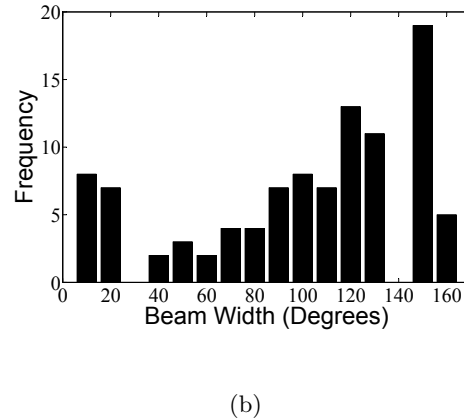
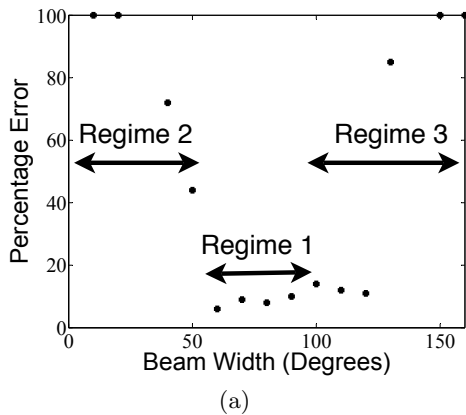


Figure 6: The antenna beam width measured under realistic circumstances and the frequency of observed beam widths.

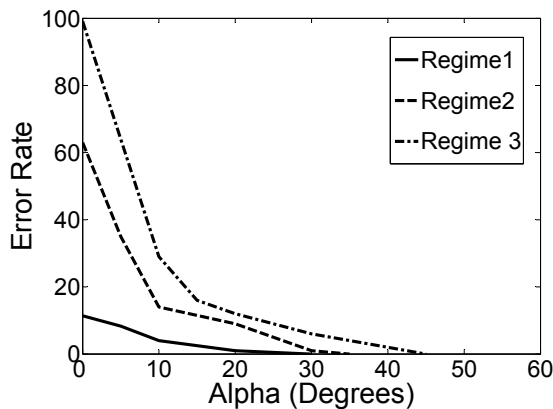


Figure 8: This figure shows the error rate vs the number of degrees added to each side of a 10 degree frustum. For instance, to achieve a zero error rate in regime one, Sherlock adds 30 degrees to each side, yielding a 70 degree frustum.

## 6. SHERLOCK IMPLEMENTATION

Our prototype of Sherlock comprises several custom hardware and software components.

### 6.1 Hardware

Our prototype consists of a ThingMagic Mercury 5 networked RFID reader. This is an enterprise-grade reader that can control four independent antennas. That is, it can behave as four independent “readers” each connected to a different antenna, with each producing an independent stream of readings. The reader has a MIPS processor that runs a Linux distribution. It exposes an API that allows custom software to be built to control the reader and the antenna. The API allows the power level of each antenna to be controlled and allows broadcast queries to be sent over an antenna and returns a list of all tags that respond to the broadcast. A PC controls the reader over an ethernet port.

At the moment there are no commercially available elec-

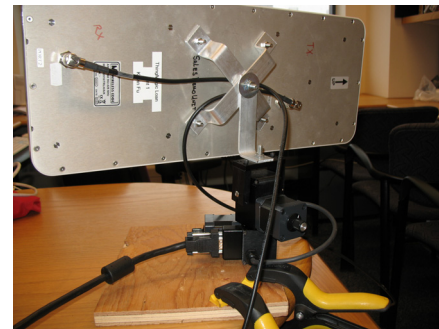


Figure 9: A photograph of the mechanically steerable antenna.

tronically steerable RFID antennas so we built our own using a pan-tilt motor. Each antenna uses a Directed Perception Pan-Tilt motor—the antenna is mounted on top of the motor using a custom mount, shown in Figure 9. The motor allows two degrees of freedom: pan and tilt, and it accepts commands from the PC over an RS-232 port. We are currently using two such antennas. The overall cost of the prototype is quite high, approaching \$10,000USD, but with electronically steerable antennas and advances in RFID reader technology the costs should eventually drop.

The coarse scan regions, in both the pan and tilt direction are  $\beta = 60$  degrees. For localize scans, Sherlock pans in steps of  $\theta = 10$  degrees and tilts in steps of  $\phi = 10$  degrees. The motor can pan the antenna over the full 360 degrees, yielding 12 coarse pan positions, and 36 localize pan positions. As the antenna cannot tilt into the base, or point directly up, it is limited to 90 degrees of tilt, yielding 4 coarse tilt positions, and 9 localize tilt positions.

In addition, we have integrated a Sony SNC-RZ25N networked PTZ camera with 40x optical zoom. We have mounted it in the office to give a clear view of the objects.

### 6.2 Software

We have also implemented four software components: the RFID endpoint that continuously scans the environment and localizes RFID-tagged objects, a location database that holds the identities and computed locations of each object,

a camera endpoint, and a query and display component for visually displaying the locations of objects.

**RFID Endpoint:** The RFID Endpoint comprises two sub-components: an antenna position controller and a query engine. The antenna controller runs on the PC and communicates with each pan-tilt motor over an RS232 (serial) interface. It issues pan and tilt commands to the motor to orient the antenna in a particular position, as directed by the scan algorithm. The antenna position controller also keeps internal state about the coarse scan positions of tags for detecting nomadic objects.

The query engine runs under Linux on the ThingMagic reader; it uses socket communication to communicate with the scan algorithm that runs on the PC. The query engine accepts commands from the scan algorithm to control the power level of each antenna. Upon being instructed, it also broadcasts a query to all tags in range. The identities of the tags that respond to the query are transmitted back to the scan algorithm over a TCP socket.

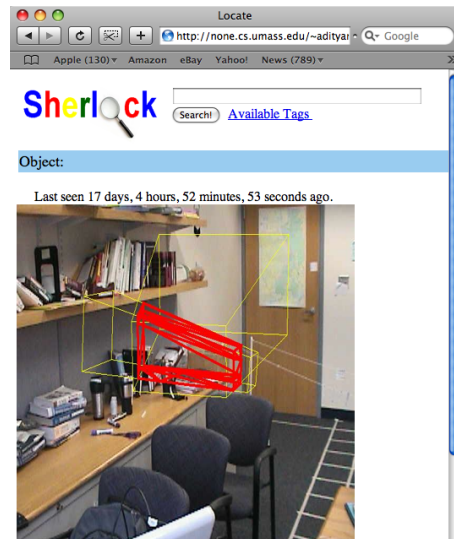
**Location Database:** The location database keeps track of the RFID identities of each object, the most recent time the object was detected, the id(s) of the antenna that detected it, and the location estimates of the tag. For objects detected by a single reader, each location estimate is essentially 8 points  $(x_i, y_i, z_i)$  that define the 8 corners of the frustum computed by the algorithm. These points are stored in the coordinate system of the antenna (which assumes that the origin is the center of the antenna). The points must be transformed back to world coordinates before answering any user query. Our current implementation stores the location database in a text file. This is sufficient for our prototype since our experiments only deal with a few hundred objects. It would be relatively trivial to use a relational database such as `mysql` to implement the location database.

A separate process fuses readings from multiple antennas. Periodically, the program examines the location database to determine if there are tags that have been detected by multiple antennas. If so, the location estimates are refined by performing polyhedron intersection of the results from each antenna, and the final results are posted to the location database.

**Camera Endpoint:** The camera endpoint grabs images from the networked camera and inserts them into the database. When an RFID endpoint adds a new object location to the database, the process pans and tilts the camera to that location, zooms the camera to encompass the three dimensional bounding volume and takes a photo. In all cases, the camera position and RFID readings must be translated into a common set of *world coordinates*. The details of how this can be done are covered by our work on Ferret [10].

**Display and Query:** Currently, Sherlock allows users to query for an object's location through a web interface, shown in Figure 10. A user can search for an object by directly specifying the name of the object (e.g., "Mobisys Proceedings 2006"). Note that lookup by name can match multiple entries in the location database: for example, a string "Mobisys Proceedings" will match multiple entries if the user has a library of conference proceedings from various years.

Once the location of each matching item has been retrieved from the location database, Sherlock retrieves the most recent image of each object. It then transforms the location coordinates into the camera coordinates and then



**Figure 10: Web interface for searching and displaying object locations.**

projects the volume containing the object onto the 2-D image plane of each image. This yields a bounding box, which Sherlock overlays onto the camera image and displays the image to the user. Thus, the user sees a visual depiction of where the object is located in her environment (see Figure 10).

## 7. EXPERIMENTAL EVALUATION

In evaluating the Sherlock system, we wanted to examine a few key points:

- What factors in our current implementation affect the performance of the system?
- How long does localization take in a realistic setting?
- To what volume are objects localized?
- How does localization performance improve with multiple antennas?
- What is the latency in detecting and localizing objects that have moved and how does the scan policy affect this latency?

### 7.1 Experimental Setup

We have deployed Sherlock in a typical office and it has been used off-and-on in this setting for 6 months. We show the placement of the RFID endpoint antennas and camera endpoint in Figure 11. We placed the antennas to maximize the coverage of the room, mounting them on desks at waist height. All of the single antenna experiments use Antenna 1 in the middle of the room. We attached passive RFID tags to at least one hundred objects, using two types of Alien Technology Gen2 Omni-Squiggle tags: a 98.2 x 12.3 mm rectangular tag for smaller objects, and a 76.2 x 76.2 mm square tag for larger objects. In the case of books, we placed rectangular tags on the spine of the book, and for other objects wherever it was convenient. We cataloged all of the objects in the database with initial photos. We have made no effort to place the objects to maximize performance, but rather left the objects where they were originally found.

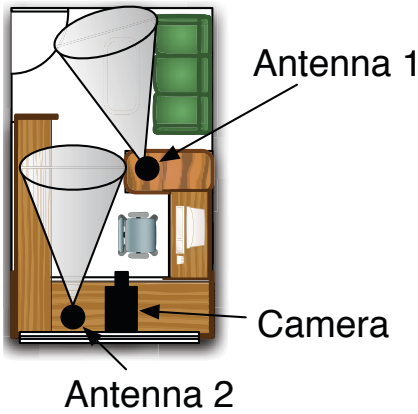


Figure 11: The experimental set up in a 16 foot by 11 foot office. The cones approximately show the coverage are for each antenna.

Characteristic	Measurement/Spec
Max angular speed of motor	60 degrees/sec
Total time for a fast scan	52 seconds
Max range of RFID reader	4-5 meters
Scan time to detect all tags	500 msec
Max tags detected per read	200

Figure 12: Characteristic measurements of the implementation.

## 7.2 Implementation characteristics

The performance results in this section must be taken in the context of the particular technology used in our experiments. To provide insight into these results, Figure 12 provides a number of measurements for the RFID reader and antenna.

### 7.3 Localization Time

The first experiment examines the latency of localizing all of the objects in the environment using a single antenna. This experiment starts with no knowledge about any objects in the room, so this may be considered a worst case. Recall that the localization algorithm proceeds from fast scan, to coarse scan, to localized scan. Using the 100 objects in the room, we randomly select a subset of the objects to include in the localization process after the fast scan. The results of this experiment for different numbers of objects in the subset are shown in Figure 13, with the scan time broken down by time spent by the reader on each phase, and the time spent in positioning the antenna using the motor. Figure 14 removes the motor latency to focus solely on the RFID reader time.

The results in Figure 13 indicate that the motor positioning time requires approximately 50% of the total time to localize objects—we consider this an artifact of the current motor-based implementation. Using the results for just the RFID reader time, in Figure 14, the time needed to localize objects in the environment is primarily determined by the number of positions the antenna must examine, driven by the number of objects that must be localized. The results show that Sherlock can localize 100 object in just under 12 minutes. There is a slight increase in the time needed for

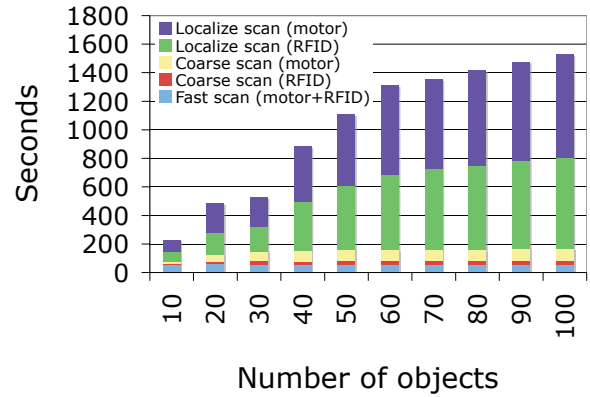


Figure 13: Time to localize all objects including the RFID reader time and the motor positioning time.

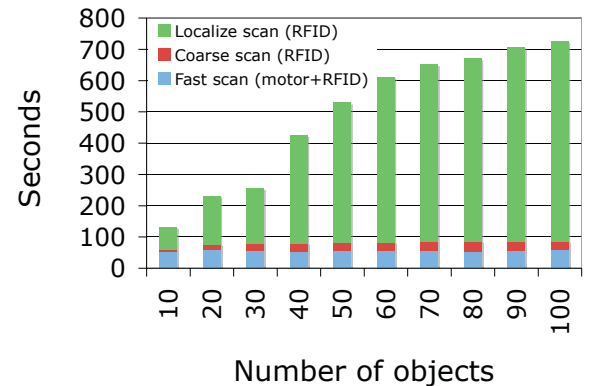


Figure 14: Time to localize all objects including just the RFID reader time.

a coarse scan, as for small numbers of objects the fast scan can eliminate certain coarse scan positions. As the number of objects increases, the latency does as well, but levels off: Once the number of objects is high, many of the positions needed for localization overlap with positions for other objects and are optimized away as described in Section 4.2.1. However, given an even larger number of objects (for instance a few thousand), it may take the reader longer to resolve the channel arbitration increasing the time needed for a scan. We have not placed enough tags to examine this possibility.

## 7.4 Localization Performance

Sherlock produces a three dimensional region that contains the localized object. The metric of success is the volume of this three dimensional space—the smaller the better. After localizing the object to a small region, the user should be able to locate it physically. Following the results of the localize scan, we measure the volume of the region and plot a CDF of the results in Figure 15. Note that due to our highly conservative localization algorithm all of the objects were correctly localized.

The results show that the worst case localization is under a meter cubed. Although the regions are not cubes, one can interpret the results as: 30% of the objects localized to a region smaller than half a meter on a side, 50% of the objects

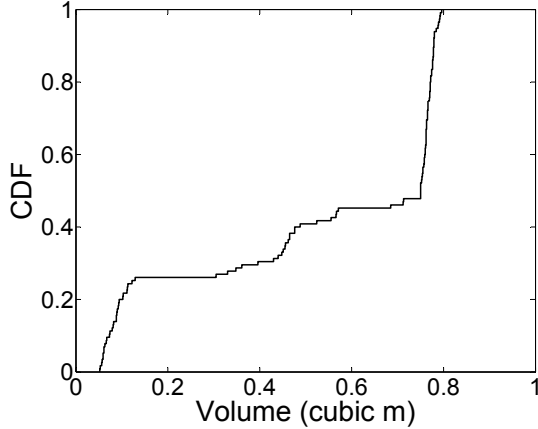


Figure 15: Volume of objects using a single antenna.

are localized to a region smaller than 0.8 meters on a side, and the rest localized to a region almost a meter on a side. The points of large slope are an artifact of the conservative design that must expand the volume for difficult to localize objects. Fully 50% of the objects have very high measured beam widths (Regime 3) due to reflections in the room and those are the objects that fall to the right side of the CDF.

## 7.5 Fusing Multiple Results

Given multiple antennas, Sherlock is able to do a better job of localizing objects. To examine this, we localize objects in the room using both antennas, and then intersect the results in three dimensions. We plot the same CDF of the final localized volume in Figure 16. We provide a deeper look at the results in Figure 17 which shows the beam width regime for each object for each antenna.

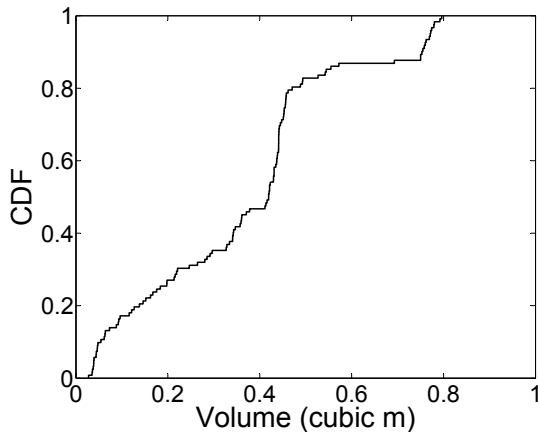


Figure 16: Volume of objects using two antennas.

The results show substantially improved localization for objects using two antennas. In fact, more than 90% of the objects are localized to an area approximately 0.8 meters on a side. This is because for any particular object, one of the two antennas may have a substantially better view of the object. This is shown in Figure 17, objects with very wide

		Antenna 1			
		R1	R2	R3	Unseen
Antenna 2	R1	10	1	21	3
	R2	4	2	2	2
	R3	16	1	34	3
	Unseen	15	1	3	x

Figure 17: Number of objects in each beam width regime per antenna. Some objects were not detectable by one antenna or the other and those are shown as undetectable for that antenna.

or narrow beam widths (Regime 2 and Regime 3), may have a more ideal beam width (Regime 1) from the other antenna. Even for objects that suffer from adverse RF conditions at both antennas, the localization improves. The two antennas both localize the object to large but different volumes due to different reflections. Also, due to the placement of the antennas, Sherlock views many objects from perpendicular directions, yielding a reduced volume after intersection. A graphical depiction of this can be seen in Figure 10 which shows two large, but perpendicular, frustums that intersect to a much smaller volume. When comparing the two CDFs, one might notice that localization for a few of the objects does not improve. This is due to the few objects that can be detected by Antenna 2, but not by Antenna 1.

## 7.6 Nomadic Objects

The last aspect of Sherlock we examine is its ability to detect and localize objects that move within the room. As before, we use 100 tagged objects in the office environment. However, to create reproducible results, and greatly speed experimentation, we simulate the movement and introduction of objects into the environment by filtering the results of each RFID scan. To simulate the movement of objects, we use two distinct objects. We filter one object in the room out of the results while leaving the other in, then at a uniformly random time swap which object we are filtering. We force Sherlock to consider these two separate tags as one object, thus simulating nomadicity.

Using this technique, we perform experiments using two different scan policies and measure both detection and localization. For detection and localization we *simulate* the movement of a number of objects at a random time and measure the time for detecting that they moved and the time to finally localize them<sup>2</sup>. We perform this experiment using two scan policies. The first is Sherlock’s default policy: fast scan, coarse scan, localize all objects, repeat. The second is a policy tuned to detect the movement of objects faster: fast scan, coarse scan, localize one object, repeat. The results for this experiment are shown in Figure 18.

The results show that the default policy detects nomadic objects slower than the fast detection policy as it spends more time localizing objects than scanning the environment for motion. However, the fast detection policy detects that objects have moved in the environment in less time. This shows the power of using different scan strategies to optimize for different criteria. We also believe that scanning partic-

<sup>2</sup>Note, that the time to detect movement is largely the same as the time to detect the introduction of new objects as Sherlock’s process to detect either is the same

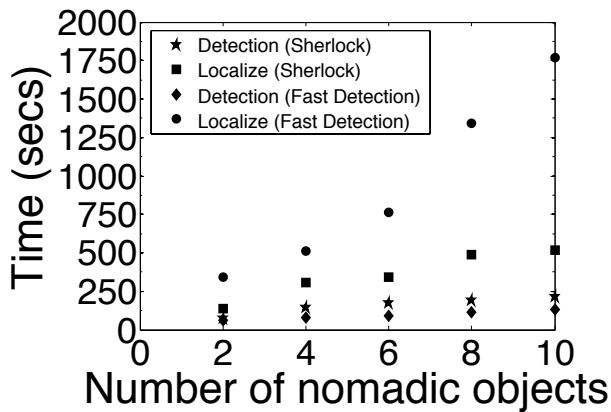


Figure 18: This figure shows the time to detect movement and the time to localize for a number of objects using two different scan policies.

ularly active parts of the room, or adapting to past history will inform even more powerful Sherlock-like systems.

## 8. CONCLUSIONS

We have presented Sherlock, a system for helping users to index and locate their personal objects. Sherlock uses a combination of passive RFID tags attached to objects, steerable antennas, and steerable cameras to help people visualize and locate their belongings. We have shown Sherlock's effectiveness in locating a hundred individual objects in a realistic setting showing that objects can always be located in less than a cubic meter, with many objects being located in a much smaller volume. We have also demonstrated one possible user interface for locating objects in an environment. While we have demonstrated Sherlock's effectiveness, we believe that this is just one step in building a multitude of systems that manage what has thus far been unmanageable: personal collections of tens of thousands of individual items.

## 9. ACKNOWLEDGMENTS

This work was funded by National Science Foundation grants CNS 0615075, CNS 0520729, and CNS 0447877. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## 10. REFERENCES

- [1] FINKENZELLER, K. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*, second ed. John Wiley & Sons, 2003.
- [2] GLOVER, B., AND BHATT, H. *RFID Essentials (Theory in Practice (O'Reilly))*. O'Reilly Media, Inc., 2006.
- [3] HÄHNEL, D., BURGARD, W., FOX, D., FISHKIN, K., AND PHILIPPOSE, M. Mapping and Localization with RFID Technology. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA '05), Barcelona, Spain* (April 2004), pp. 1015–1020.
- [4] HIGHTOWER, J., AND BORRIELLO, G. A Survey and Taxonomy of Location Sensing Systems for Ubiquitous Computing. Tech. rep., University of Washington, Department of Computer Science and Engineering, Seattle, WA, August 2001.
- [5] HIGHTOWER, J., WANT, R., AND BORRIELLO, G. SpotON: An Indoor 3D Location Sensing Technology based on RF Signal Strength. Tech. Rep. 00-02-02, University of Washington, 2000.
- [6] HODGES, S., THORNE, A., MALLINSON, H., AND FLOERKEMEIER, C. Assessing and Optimizing the Range of UHF RFID to Enable Real-World Pervasive Computing Applications. In *Proceedings of the 5th International Conference on Pervasive Computing* (2007), pp. 280–297.
- [7] KRIPLEAN, T., WELBOURNE, E., KHOUSSAINOVA, N., RASTOGI, V., BALAZINSKA, M., BORRIELLO, G., KOHNO, T., AND SUCIU, D. Physical access control for captured rfid data. *IEEE Pervasive Computing* 6, 4 (2007), 48–55.
- [8] LI, M., YAN, T., GANESAN, D., LYONS, E., SHENOY, P., VENKATARAMANI, A., AND ZINK, M. Multi-user data sharing in radar sensor networks. In *5th ACM Conference on Embedded Networked Sensor Systems (Sensys 2007)* (Nov 2007).
- [9] GNU Triangulated Surface Library. <http://gts.sourceforge.net>.
- [10] LIU, X., CORNER, M. D., AND SHENOY, P. Ferret: RFID Locationing for Pervasive Multimedia. In *Proc. ACM Ubicomp* (Irvine, CA, September 2006), pp. 422–440.
- [11] NI, L. M., LIU, Y., LAU, Y. C., AND PATIL, A. P. LANDMARC: Indoor Location Sensing using Active RFID. In *In Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom'03), Dallas-Fort Worth, TX* (March 2003), pp. 407–417.
- [12] PHILIPPOSE, M., FISHKIN, K., FOX, D., KAUTZ, H., PATTERSON, D., AND PERKOWITZ, M. Guide: Towards Understanding Daily Life Via Autoidentification and Statistical Analysis. In *Proc. of the Int. Workshop on Ubiquitous Computing for Pervasive Healthcare Applications (Ubihealth)* (2003).
- [13] ROH, S., PARK, J. H., LEE, Y. H., AND CHOI, H. R. Object Recognition of Robot using 3D RFID System. In *Proceedings of the 2005 International Conference on Control, Automation and Systems (ICCA'05), Gyeong Gi, Korea* (June 2005).
- [14] SMITH, J. R., FISHKIN, K. P., JIANG, B., MAMISHEV, A., PHILIPPOSE, M., REA, A. D., ROY, S., AND SUNDARA-RAJAN, K. RFID-Based Techniques for Human-Activity Detection. *Communications of the ACM* 48, 9 (2005), 39–44.
- [15] WANT, R. An Introduction to RFID Technology. *IEEE Pervasive Computing* 5, 1 (2006), 25.
- [16] WELBOURNE, E., BALAZINSKA, M., BORRIELLO, G., AND BRUNETTE, W. Challenges for pervasive rfid-based infrastructures. In *PERCOMW '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 388–394.