SurroundSense: Mobile Phone Localization via Ambience Fingerprinting

Martin Azizyan
Duke University
Durham, NC, USA
martin.azizyan@duke.edu

Ionut Constandache
Duke University
Durham, NC, USA
ionut@cs.duke.edu

Romit Roy Choudhury
Duke University
Durham, NC, USA
romit@ee.duke.edu

ABSTRACT

A growing number of mobile computing applications are centered around the user's location. The notion of location is broad, ranging from physical coordinates (latitude/longitude) to logical labels (like Starbucks, McDonalds). While extensive research has been performed in physical localization, there have been few attempts in recognizing logical locations. This paper argues that the increasing number of sensors on mobile phones presents new opportunities for logical localization. We postulate that ambient sound, light, and color in a place convey a photo-acoustic signature that can be sensed by the phone's camera and microphone. In-built accelerometers in some phones may also be useful in inferring broad classes of user-motion, often dictated by the nature of the place. By combining these optical, acoustic, and motion attributes, it may be feasible to construct an identifiable fingerprint for logical localization. Hence, users in adjacent stores can be separated logically, even when their physical positions are extremely close. We propose *SurroundSense*, a mobile phone based system that explores logical localization via ambience fingerprinting. Evaluation results from 51 different stores show that SurroundSense can achieve an average accuracy of 87% when all sensing modalities are employed. We believe this is an encouraging result, opening new possibilities in indoor localization.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software; C.2.4 [**Computer-Comunication Networks**]: Distributed Systems; H.5.5 [**Information Interfaces and Presentations**]: Sound and Music Computing

General Terms

Design, Experimentation, Performance, Algorithms

Keywords

Localization, Mobile Phones, Context, Fingerprinting

MobiCom'09, September 20–25, 2009, Beijing, China.

1. INTRODUCTION

Mobile phones are becoming a powerful platform for people-centric computing. A variety of applications are on the rise, many of which utilize the location of the phone [11, 16, 18]. For instance, GeoLife [27] is a service that plans to display shopping lists on a mobile phone when the phone is detected near a Wal-Mart. Micro-Blog plans to query users that are presently located, say, in an art gallery. Location-based advertising is on the horizon – a person entering Starbucks may receive an electronic coupon for purchasing coffee. Interestingly, all these applications operate on a *logical* notion of location, such as "Wal-Mart", "art gallery", and "Starbucks". In the absence of well-established logical localization schemes, application developers are assuming that physical coordinates (like GPS) can be reverse geo-coded to logical locations. We argue that conversion from physical to logical location is error prone. We present our arguments next, and motivate the need for independent research in logical localization.

Consider GPS, the most popular physical localization method on mobile devices. While GPS can achieve up to 10m accuracy in outdoor environments, they do not work indoors. A variety of WiFi and GSM based alternates have been proposed for indoor operation (RADAR, Place Lab, SkyHook, etc. [2–4]), each associated with distinct tradeoffs between accuracy and scalability. For argument's sake, let us imagine that future localization techniques will attain the best of both worlds. That is, a phone can be easily and ubiquitously localized to the accuracy of 5m in any indoor environment. We argue that even such an idealized scheme may not be adequate to accurately identify logical locations. This is because two logical locations may be separated by a *dividing wall*, and an error margin of 5m may place the phone on the incorrect side of this wall. As a result, a phone located near the wall of Starbucks may be erroneously placed in an adjacent RadioShack (Figure 1). Services that rely on logical locations can be affected. A coffee drinker at Starbucks may resent receiving video-game coupons intended for RadioShack customers next-door.

To discriminate between adjacent locations/contexts, some approaches have installed special transmitters in each context/room. The Cricket system achieved cm scale localization through a combination of RF and ultrasound beaconing systems installed in the surroundings [22]. Nokia deployed bluetooth based beacon-transmitters in different rooms of their lab. Unlike WiFi, bluetooth beacons can be more easily confined to a single room, and hence, are useful for localization. Installing special hardware in every room, while arguable for

Figure 1: Slight errors in physical coordinates can place a phone on the wrong side of a wall.

Figure 2: Nearby stores at our university campus exhibit diversity in wall/floor color and ambient lighting. The bookstore (left) is lit with bright white light, the boutique (middle) with dim white light, while the pub (right) is significantly darker.

specific needs, may not scale to an entire city. Solutions are necessary that obviate the need for pre-installed infrastructure. *In fact, even WiFi may not be available ubiquitously in developing regions. Mobile phones, however, are a rapidly growing platform in these regions, and localizing them even without WiFi can be enabling.*

To summarize, installing localization equipment in every logical place is unscalable, while relying solely on external infrastructure (such as GPS/GSM/WiFi) lacks the accuracy to discriminate adjacent contexts. Ideally, local attributes of a logical location need to be identified passively and exploited for accurate localization. We argue that numerous local attributes already exist in the location's ambience; sensing them through mobile phones and using them in conjunction with GSM/WiFi can be an effective solution. The central idea is presented next.

Our hypothesis is that the combined effect of ambient sound, light, and color – i.e., the overall ambience of a place – can be unique enough for localization. For example, ambient sound in Starbucks may include specific noise signatures from coffee machines and microwaves, that are different from sounds of forks and spoons clinking in restaurants. Shops may have thematic colors in their decor, such as red at Target and yellow at Panera Breads. Floors may be covered with carpets, ceramic tiles, or wooden strips, all of which are discriminating attributes of the ambience. Even lighting styles may be different in order to match with the type of service a place may provide – bars with dim yellow lights versus BlockBuster with bright white light. In addition, the movement of a person in a given place may also be a function of the layout of that place, and its type of service. Human movement in Wal-Mart (walking up and down aisles) may be different from that in Barnes and Noble (relaxed stroll with long pauses), which may in turn be different from restaurants (short queuing followed by a long duration of sitting). Even though places may not be unique based on any one attribute, the combination of all the attributes is likely to exhibit diversity. We intend to sense these attributes through the mobile phone's camera, microphone, and accelerometer, and show that their combined diversity can be exploited for localization.

A natural question is: *should ambiences of all places be unique for reliable localization?* We do not believe this is necessary. Existing indoor localization schemes, based on GSM or WiFi, effectively place a phone in a macro-location (such as a strip mall). All logical places within such a macro-location can be shortlisted. As long as the ambiences of these shortlisted places are different, SurroundSense should be able to discriminate between them correctly.

In certain cases, such as in strip malls, there is an interesting economic reason that may add to the ambience diversity. Essentially, spatially nearby businesses may have an incentive to be mutually unique. For example, multiple Chinese restaurants may not prosper if they are all located in the same strip mall, and present the same music, decor, lighting, and layout. Mutual distinctiveness reduces competition, benefiting each of the businesses financially and socially. The outcome of such economic behavior facilitates SurroundSense. Since fingerprints of spatially collocated places may be diverse by design, SurroundSense can exploit this diversity for reliable logical localization. Figure 2 presents pictures from a few adjacent stores near our university campus – the diversity in lighting and color is evident.

Translating this broad idea into a practical localization system entails a range of challenges. Recognizing fingerprints from raw ambience data is non-trivial; the ambience of a place may vary over time; a person's movement on a certain day may be atypical. Nonetheless, the availability of multiple modes of sensing may be able to cope with such variations. SurroundSense is among the early attempts to make use of these multi-modal sensing capabilities for localization. Our approaches are simple and the results look promising. Our main contributions are as follows.

(1) Identifying the possibility of fingerprinting a logical location based on ambient sound, light, color, and human movement. Cameras, microphones, and accelerometers on WiFi-enabled Nokia N95 phones were used to sense such information.

(2) An experimental framework that creates a fingerprint database and performs fingerprint matching for test samples. We performed simple feature extraction from the collected data, and combined them into a per-location fingerprint. Support vector machines (SVMs), color clustering, and other simple methods were used for location classification.

(3) Evaluation of the scheme in business locations in our university town. We covered 51 distinct stores, each store fingerprinted at various times of the day. Different students then visited each of these stores and their sensed data were used to deduce their locations. We achieved an average

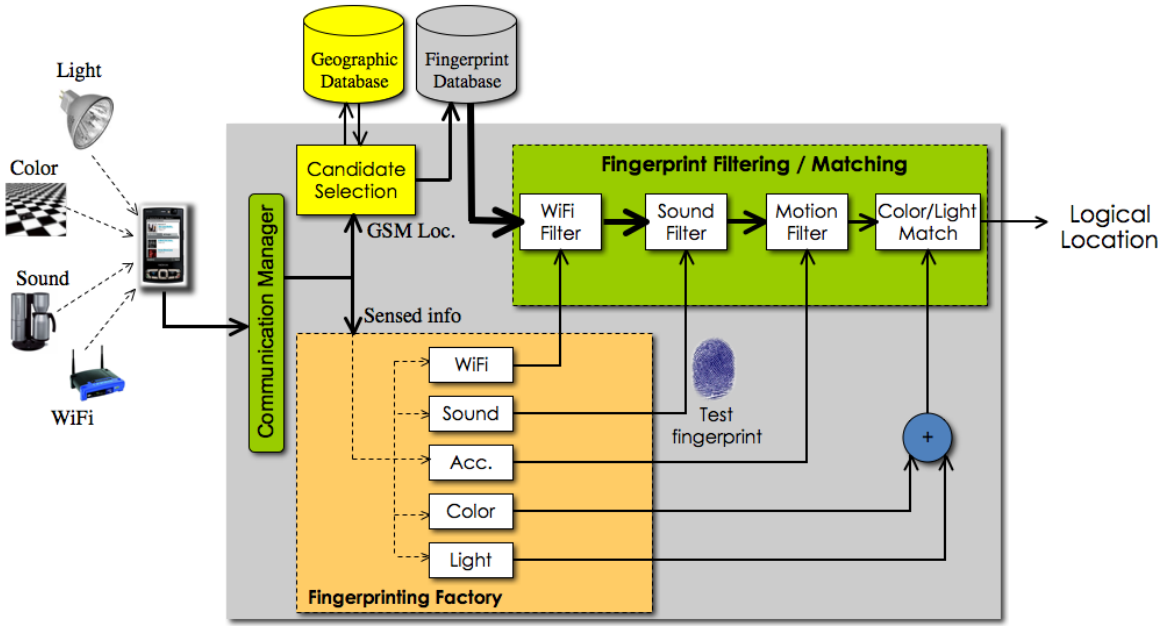

Figure 3: SurroundSense architecture: The ambience fingerprint is generated by the fingerprinting factory. This test fingerprint and candidate fingerprint (from the phone's macro-location) are matched in the fingerprint matching module.

localization accuracy of over 85% when all sensors were employed for localization.

The rest of this paper expands on each of these contributions. We discuss limitations and future work in Section 6. We present related work in Section 7 and conclude the paper in Section 8.

2. SURROUNDSENSE ARCHITECTURE

Figure 3 presents the overall architecture of SurroundSense. This is one possible architecture in the broad design space, not necessarily the optimal one. We describe the high level flow of information through this architecture, and present the internal details later. We begin with a mobile phone user visiting an unknown store. The phone senses the ambience automatically. The sensed values are recorded, pre-processed, and transmitted to a remote SurroundSense server. The goal of pre-processing is to reduce the data volume that needs to be transmitted. Once the sensor values arrive at the server, they are forwarded to the fingerprinting factory. The fingerprinting factory segregates the type of sensor data (sound, color, light, WiFi, accelerometer) and distributes them to respective fingerprinting modules. These modules perform a set of appropriate operations, including color clustering, light extraction and feature selection. The individual fingerprints from each module are logically inserted into a common data structure, called the ambience fingerprint, F. Put differently, F consists of five sub-fingerprints $< f_s, f_l, f_c, f_w, f_a >$ corresponding to sound, light, color, WiFi, and accelerometer. Since the phone's location is not known at this time, we call this the "test fingerprint". The test fingerprint is forwarded to the fingerprint matching module for localization.

The transmitted data from the phone also includes the phone's (GSM-based) physical coordinate, L_{GSM} (we assume WiFi may not be available at all locations). The L_{GSM} is a <latitude, longitude> tuple accurate to around 150m. A candidate selection module consults a geographical database (such as *geonames.org*) to shortlist all stores within 150m of L_{GSM}. Fingerprints for each of the shortlisted stores are fetched from an existing database of fingerprints (we discuss database-creation later). We call these shortlisted fingerprints "candidate fingerprints". Like the test fingerprint, each candidate fingerprint also consists of five sub-fingerprints (i.e., $F_i = < f_s^i, f_l^i, f_c^i, f_w^i, f_a^i >$). The candidate fingerprints are also forwarded to the matching module.

The matching/filtering module is expected to select one candidate fingerprint that best matches the test fingerprint. For this, it computes the pair-wise "similarity" between the test and candidate sub-fingerprints. The similarity values are used either for *filtering*, or for *matching*. Filtering means eliminating some candidate fingerprints that are not likely to be from the same location as the test fingerprint. A filter accepts the test and the candidate fingerprints, and returns a subset of the candidate set. Matching means ordering the candidate set according to the pair-wise similarity values. From the list of decreasing similarity, the top ranked candidate is declared to be from the same location as the test case. We use the WiFi, accelerometer, and sound sensors as filters; color and light are combined into a single matching operation. We describe the details next.

3. SYSTEM DESIGN

In this section we present the two main components of SurroundSense: Fingerprint Generation, and Matching.

3.1 Fingerprint Generation

The raw sensor values from phones contain a lot of information about the phone's surroundings. The important task is to extract features from them that are effective in discriminating its ambience [17]. While sophisticated data mining is one

approach, we intend to develop lightweight techniques that will eventually be executable on the phone's processor. We show that such lightweight techniques are feasible, particularly because the availability of multi-dimensional signatures obviates the need to make any single signature perfect. Of course, perfecting each of the signatures (through sophisticated machine learning techniques) is likely to offer greater benefits.

Fingerprinting Sound

The ambient sound in a place can be suggestive of the type of place. Some places play loud music in the background versus others that are quieter. Some may have a strong presence of metallic noise or A/C machine drones versus others in which there are frequent beeps at check out counters. We recorded the ambient sound of a store for one minute using the phone microphone. Our first attempt was to convert the signal to the frequency domain (through an FFT), and identify signatures of specific devices in the ambience. However, we observed that while some sound signatures were visible, in many cases it was difficult to separate them from other frequency components. Hence, we reverted back to the time domain, and used a simple fingerprinting scheme based on signal amplitude. Specifically, we divided amplitude in 100 equal intervals (50 on the positive amplitude axis, and 50 on the negative). The audio sample rate is 8 kHz (8000 samples/s). We normalized the number of samples per-interval by the total number of samples in the recording. The 100 normalized values were considered to be features of the ambient sound, together called the acoustic fingerprint. Figure 4 shows the fingerprints for 3 different stores.

Because sound from the same place can vary over time, it is unreliable to use sound as a matching scheme. Therefore, we use sound only as a filter. We compute the pair-wise distance between the test fingerprint and all candidate fingerprints. The pair-wise distance is based on an euclidean metric in the 100 dimensional space. If the distance between a candidate fingerprint and test fingerprint is greater than a filter threshold, τ, we discard that candidate fingerprint. Other fingerprints remain in the candidate set and are forwarded to successive filtering/matching modules. Of course, the question is *how do we choose* τ. For this, we collected acoustic fingerprints from each location at different times, and computed the pair-wise distances. Plots of these distances reflected the distribution of self-similarities. Smaller the distance, more self-similar they were. To eliminate outliers, for each store i, we chose a threshold distance, δ_i, that was at the 95th percentile. We chose the maximum δ_i across all shops, and assigned it to τ. In other words, τ is a measure of the maximum dissimilarity that was observed among two acoustic fingerprints from the same store. We conservatively chose the maximum to adopt a wider filter. Hence, more candidate fingerprints were allowed through the filter (more false positives). However, those eliminated, were done with greater confidence (fewer false negatives). The output of the sound-based filtering module is fed to the accelerometer filter.

Fingerprinting Motion using Accelerometers

The nature of service in a place partly influences the type of human movements in that place [21, 23]. A simple example is that people are stationary for long durations in restaurants, while grocery store customers are significantly more mobile.

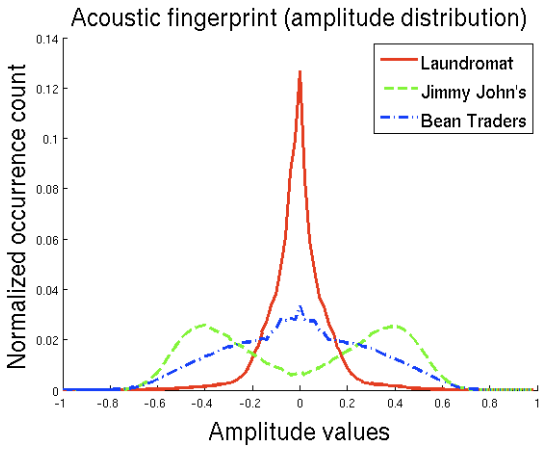

Figure 4: Sound fingerprints from 3 adjacent stores.

Capturing broad user movements within a store can be a useful fingerprint of that store. The fingerprint can be used to zero in on the test fingerprint's actual location.

We use the 3-axes accelerometer in Nokia N95 mobile phones to characterize user movements. The accelerometer is sampled 4 times per second. Each sample records the instantaneous acceleration of the phone. Unfortunately, the accelerometer has a large noise floor which makes accurate measurements difficult. To circumvent these issues, we computed a moving average over a window of 10 recent samples. The time averaging smoothened the sequence of samples, at the expense of suppressing some minor movements. We also computed the moving variance using the window of last 10 samples. The two sequence of samples were processed as follows.

We decided to identify two simple states from the accelerometer readings, namely, *stationary* and *motion*. To classify these states, we decided to use support vector machines (SVM), a popular data classification tool [7]. We first trained the SVM using readings from a statically held phone, as well as from readings in which a user walked with the phone. We used *mean* and *variance* as the features during the training phase. Once the training was accomplished, accelerometer samples from real user movements in different stores were fed to the SVM. The SVM classified each of these samples to be either in stationary (-1) or moving (+1) state. The sequence of these states can be viewed as an abstraction of the user's movement pattern.

User movement is prone to fluctuation. Some users may browse for a long time in a clothing store, while others may purchase clothes in haste. Therefore, like sound, we use accelerometers as a filtering mechanism too. Since filters are not required to provide a strict ordering, they are less prone to error. Nevertheless, it is necessary to capture a feature in user movement that is dictated more by the store, and less by the user's whim. One possibility is to first differentiate between *sitting* and *moving* places (e.g., restaurants, haircutteries, cafes, versus grocery, music stores, bookstores). In the moving category, we can further divide into *slow-browsing* and *speed-walking*. To verify this intuition, we gathered a

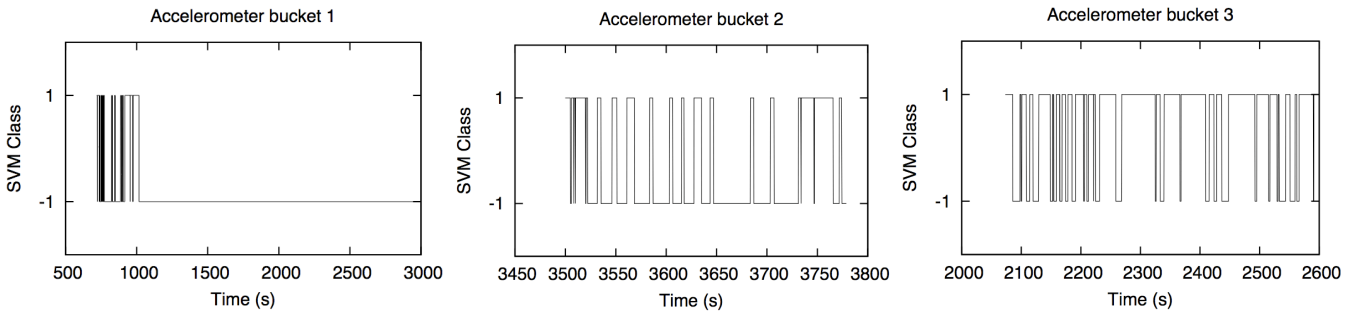

Figure 5: Sample accelerometer traces from each bucket. SVM class (-1) is static, (+1) is motion. (a) Bucket 1 stores expect users to sit for long durations (Starbucks, restaurants). (b) Bucket 2 shops carry browsing products (book stores, clothing, wine) (c) Bucket 3 stores are mostly for fast shopping (Wal-Mart, Krogers)

large number of accelerometer traces, and computed a ratio $R = \frac{t_{moving}}{t_{static}}$. The value of R is a fingerprint, where t_{moving} and t_{static} are total durations during which the SVM classified the user as moving or static. Plotting values of R on a real line revealed 3 clusters (with some outliers). We thresholded these clusters, extracting 3 buckets of accelerometer fingerprints:

- Bucket 1: $0.0 \leq R \leq 0.2$ Sitting (cafe)
- Bucket 2: $0.2 < R \leq 2.0$ Browsing (clothing)
- Bucket 3: $2.0 < R < \infty$ Walking (grocery)

Figure 5 depicts a sample accelerometer trace from each of the buckets. The difference is visually evident. However, some stores exhibited a higher spread in the R values (we noticed that Target customers could vary between long DVD browsers to quick grocery shoppers). To cope with such multi-commodity shops, we assigned them to both buckets. Upon receiving a test fingerprint from a phone, its bucket, B_{test}, is first identified. Candidate fingerprints that belong to the same bucket, B_{test}, are retained, while the others are filtered out. The retained candidates are then forwarded for color/light based fingerprint matching.

Fingerprinting Color/Light using Cameras

Empirical observations convinced us that a large number of stores have a thematic color and lighting as part of their decor. The wall and floor colors contribute significantly to this theme. Based on this, we hypothesized that automatic pictures taken from different spots in a store are likely to reflect this theme. If these colors and light intensities can be extracted from the pictures, they could form a fingerprint for localization. For now we assume that the phone is outside the user's pocket; we will visit this practical concern in the next section. The challenge, then, is to extract the appropriate features of the ambience from automatically-taken phone pictures. Of course, random pictures of the surrounding are likely to capture a variety of store items, spread over a wide range of colors [13]. This can make the pictures noisy, i.e., the dominant colors extracted from these pictures may not match the thematic colors of the ambience. To circumvent this, we focus on pictures of the floor alone. We detect that a picture is of the floor based on the orientation of the phone-camera when the picture was taken[1]. Only floor-facing pictures are enlisted for color/light extraction, while others are discarded.

Focusing only on floor pictures, i.e., those taken with the camera facing downward, offers a number of benefits. (1) Privacy concerns may prevent a user's phone from randomly clicking pictures of the surrounding. However, if pictures are taken only when the phone camera is pointing towards the floor, the concerns are partly alleviated. (2) Pictures of the floor are likely to be less punctuated with other objects. Dominant colors extracted from these pictures are expected to be less noisy. (3) There is rich diversity in the colors of carpets, tiles, marble, and wooden floors. This diversity is beneficial to localization. (4) Users may often point their cameras downward while using their phone (checking emails, typing SMS). Floor pictures may not be uncommon.

Our goal is to extract dominant colors and light intensity from pictures of floors. For this, we first analyzed the color of each pixel on a *red-green-blue (RGB)* space. The results were poor because the extracted colors were heavily biased by shadows of objects and people, and by reflections of light. The light intensities were also unreliable (we omit the details of several failed approaches). We circumvented this problem by translating the pixels to the *hue-saturation-lightness (HSL)* space. Briefly, *hue (H)* represents the naturally perceived base colors, *saturation (S)* reflects the dominance of that hue, and *lightness (L)* reflects the light intensity. As a result of this translation, the actual floor colors could be decoupled from the ambient light intensity. Shadows and reflections mattered less, and most importantly, the light intensity was separately available on the L axis.

We re-plotted the HSL pixels from all pictures of the same place. On this scatterplot, we ran the K-means clustering algorithm [14] for increasing values of K. The K-means algorithm divides the pixels into K clusters, such that the sum of distances from all pixels to their (own cluster's) centroid, is minimized. Let us call this sum, S_k. Starting from $K = 2$, we continue to compute S_k until $S_k - S_{k-1} \leq \delta$, where δ is a convergence threshold set reasonably small. At this point, we obtained the stable clusters of colors, along with the sizes of each cluster. The large clusters consisted of colors and light

[1]Six camera orientations can be obtained from the phone, namely left, right, front, back, top, bottom.

intensities that respectively reflected the thematic colors of floors and the brightness in the ambience. In certain cases, floors had multiple colors, producing multiple large clusters. Other colors from the pictures were also reflected (e.g., colors of shoes, lower end of trousers and shelves, etc.) but in proportionally small clusters. The number of clusters typically varied between 3 to 7. Figure 6 shows an example from Bean Traders Coffee shop. The centroids of these clusters, as well as the cluster sizes, were each a feature of the ambience, together forming the color-light fingerprint of that place.

We considered several fingerprint matching schemes and finalized on a simpler one (in view of our eventual goal to execute SurroundSense on the phones). The idea is to compute the similarity between fingerprints F_1 and F_2 based on the euclidean distance between their cluster centroids, and the sizes of the clusters. Large clusters that are close to each other indicate that both F_1 and F_2 have a good match in their dominant colors. When the cluster size decreases, or the distance increases, the similarity is proportionally lower. Formally, denote $C_1 = \{C_{11}, C_{12}, ...C_{1n}\}$ as the set of clusters for fingerprint F_1. Similarly, $C_2 = \{C_{21}, C_{22}, ...C_{2m}\}$ for F_2. Let $SizeOf(C_{ij})$ denote the number of pixels in cluster C_{ij}. Let T_1 and T_2 be the total number of pixels in C_1 and C_2 respectively. Also, let function $\delta(i,j)$ represent the centroid-distance between the i^{th} cluster of F_1 and the j^{th} cluster of F_2. We model the similarity S_{12} between fingerprints F_1 and F_2 as:

$$S_{12} = \sum_{i,j} \frac{1}{\delta(i,j)} \frac{SizeOf(C_{1i})}{T_1} \frac{SizeOf(C_{2j})}{T_2} \qquad (1)$$

In other words, every cluster pair contributes to the overall similarity of the two fingerprints. This contribution is proportional to the product of the two cluster sizes, and the euclidean distance between the centroids of the two clusters. The similarity between the fingerprints is a sum of all pairwise similarities. Given a set of candidate fingerprints and a test fingerprint, the similarities between the test and all candidates are computed. The candidate fingerprint with maximum similarity is declared to be the *matching fingerprint*. The unknown location of the phone is classified to be that of the matching fingerprint.

Fingerprinting Wi-Fi

While creative WiFi fingerprinting techniques exist [3,4], they do not apply directly to recognizing logical places. We adapt existing WiFi based fingerprinting to suit logical localization, and include it as the fifth sensor in SurroundSense. The intuition behind WiFi fingerprinting is simple. The MAC addresses of visible APs are some indication of the phone's location. The phone records MAC addresses from received beacons every 5 seconds. From this raw data, a fingerprint is acquired by computing the fraction of times each unique MAC address was seen over all recordings. A tuple of fractions (each tuple element corresponding to a distinct MAC address) forms the WiFi fingerprint of that place.

Fingerprint matching is performed by computing a metric of *similarity* between a test fingerprint and all candidate fingerprints. The comparison between two fingerprints, f_1 and f_2, is performed as follows. Denote M as the union of MAC addresses in f_1 and f_2. For a MAC address $m \in M$, let $f_1(m)$ and $f_2(m)$ be the fractions computed as above. Then the similarity S of f_1 and f_2 is computed as:

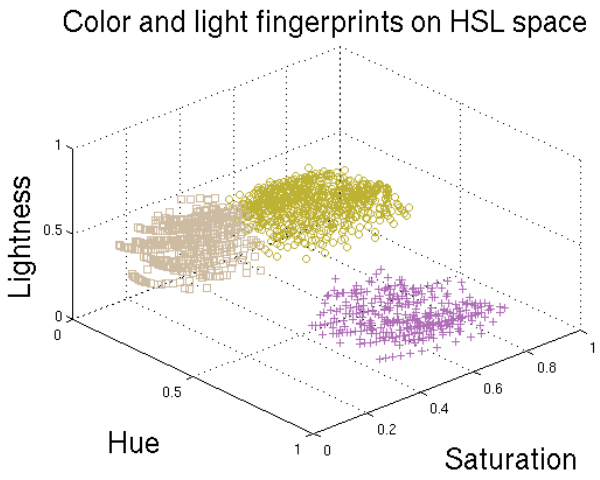

Figure 6: Color/light fingerprint in the HSL format from the Bean Trader's coffee shop. Each cluster represented by a different symbol.

$$S = \sum_{m \in M} (f_1(m) + f_2(m)) \frac{min(f_1(m), f_2(m))}{max(f_1(m), f_2(m))} \qquad (2)$$

The intuition behind this metric is to add a large value to S when a MAC address occurs frequently in both f_1 and f_2. The purpose of the fraction is to prevent adding a large value if a MAC address occurs frequently in one fingerprint, but not in the other. Locations that achieve a low *similarity* have a higher chance of discrimination, and the vice versa.

We use WiFi as a filter in SurroundSense to avoid frequent misclassifications (false negatives). However, in the absence of light/color, we use it as a matching module to obtain a precise rank of the location in question.

3.2 Fingerprint Matching

SurroundSense combines the 4 filtering/matching operations into an enveloping module that outputs the test phone's logical location. The (WiFi, sound, and accelerometer) filters are first applied to the candidate set, such that some of the members can be safely eliminated. The pruned candidate set is then fed to the color/light-based matching scheme, which outputs an ordered list of the candidates. The order of applying WiFi, sound, and accelerometer does not matter, since it is analogous to a "set intersection" operation. However, it is important to use the color/light based matching scheme last. This is because color/light performs a strict ordering, and a smaller candidate set reduces the likelihood of mistakes. The final output is an ordered list of candidates – the top ranked candidate is declared to be the location of the phone.

Feedback from the end user, if feasible, can be beneficial. Assume that SurroundSense outputs the ordered set of locations, {*A, B, C*}. The user can be asked to verify her location from the set. If the user responds that her actual location is *B*, then SurroundSense can potentially learn and train itself accordingly. Experiments in this paper, however, do not take advantage of such human feedback.

As a final note, observe that the parameters for filtering and matching can be tuned on a per-cluster basis. If certain clusters exhibit atypical behavior, SurroundSense could potentially train on that cluster and re-adjust filtering/matching parameters. The matching operation may also be jointly performed across all sensors (as opposed to our simple serial approach). However, joint approaches are complicated in view of orthogonal sensing modalities, and their unequal importance in overall localization. In that sense, the simplicity of our algorithms makes SurroundSense executable on the phone's processor.

Coping with Time-varying Ambience

Fingerprints from a particular shop may vary over time. A sound fingerprint from a busy hour may not match well if the training fingerprints were derived from low-activity periods. Colors in a picture may be different depending on daylight or electric light. We propose to divide a day into 2-hour time-windows, and index fingerprints based on the time they were created. When a time-stamped test fingerprint is sent for matching, candidate fingerprints from the appropriate time-window are selected for comparison. The time-windows may be refined, or made adaptive, as more fingerprints become available to the fingerprint database.

4. PROTOTYPE IMPLEMENTATION

SurroundSense was implemented on Nokia N95 phones using Python as the programming platform. The server consists of MATLAB and Python code, and some data mining tools for the fingerprint matching algorithms. We present relevant details next.

4.1 Client and Server

The ambience sensing script is designed such that each sensor runs on a different thread. The threads execute API calls, and are then put to sleep for a desired time duration. The time duration is chosen based on necessity. The accelerometer samples are collected at the rate of 4 readings per second. The audio sampling rate is 8 kHz. Pictures are taken every 5 seconds, and the camera was configured to the "sport" mode (to allow for better pictures while moving). A meta file is created for each fingerprint, storing information about the date, time, GSM coordinates, camera mode, etc. Figure 7 shows a few screenshots.

The server is composed of several modules. A Data Manager assimilates the raw data from different phones and formats it appropriately. The formatted data is forwarded to the Fingerprinting Factory, that employs libSVM for classifying accelerometer data, MATLAB toolkits for K-means clustering and audio processing. The $< LogicalLocation, Fingerprint >$ tuple is then inserted into the fingerprint database. A MATLAB/Python based Filtering/Matching Module accepts a test fingerprint and computes the top-ranked match.

4.2 Populating the Fingerprint Database

A natural question is *how do we build a fingerprint database?* A variety of options may be feasible, depending on the extent of coverage desired. In our case, we have performed labor-intensive *war-sensing* at 46 business locations in the university town, and at 5 locations in India. The notion of war-sensing is analogous to the notion of war-driving for WiFi and GSM based localization [3, 4]. Groups of students visited 51 stores

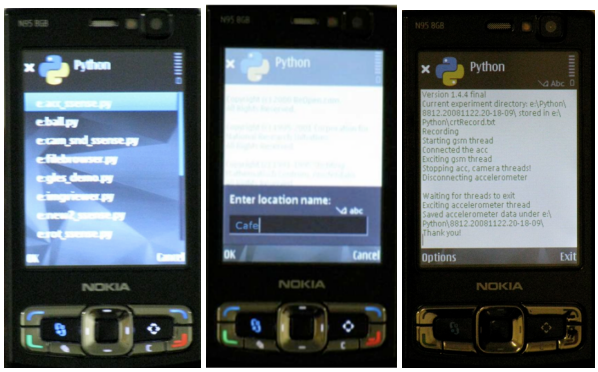

Figure 7: Nokia N95 phone running SurroundSense scripts and fingerprinting.

and collected fingerprints from each of them. The stores were visited multiple times later to collect test fingerprints and evaluate the accuracy of SurroundSense. A few details about the process are important in interpreting the performance of SurroundSense, and hence, discussed in the evaluation section later.

Of course, war-sensing is labor-intensive and may not scale to commercial-scale deployments. One possibility is to design location labeling games, like image labeling games in the Internet [28]. The structure of the game could be to have competing users record an ambience fingerprint that best matches a publicly announced fingerprint. For example, a Wal-Mart fingerprint from New York could be posted in San Diego, and people could try to match it by taking pictures and sensing sounds. The person with a best match may win a prize. More interesting variants of such a game may be possible. If large number of people play such games, the fingerprint database can be generated in a reasonable timeframe.

5. EVALUATION

We present the evaluation of SurroundSense in two parts: (1) Partially Controlled Experimentation, and (2) Performance Results.

5.1 Partially Controlled Experimentation

Our field experiments were not performed with a real user-base (difficult with limited mobile phones). We also made a few assumptions during experimentation. We report them here so that the results can be interpreted with these factors in mind.

Cameras Out of Pocket

Over the entire course of our experiments, we held the mobile phones in our hand (and not in our pockets). We used a normal grip and periodically made phones calls, browsed the Internet, and typed in SMSs. This allowed the phones to take pictures for color and light fingerprinting. In uncontrolled environments, phones may be mostly inside the pocket, preventing camera-based sensing. However, a host of wearable mobile phones have already entered the commercial market [19]. These phones are worn as wrist watches and necklaces, enabling a range of sensing/pervasive applications [10, 26]. Advances in nano-technology are further driving this trend by introducing flexible material (e.g., the Nokia

Morph [1]). We believe that wearable phones will become popular in the near future, making SurroundSense a viable application.

Mimicking Customer Behavior
While fingerprinting locations, we selected store clusters (within GSM macro locations) and visited each of them in groups of 2 people (4 people in total). Upon arriving at a cluster, individuals went to different stores so that fingerprints were time-separated. Each student fingerprinted every store in that cluster. While in a store, we tried to behave like normal customers. Of course, without any purchasing intentions, it was difficult to spend a natural amount of time in a store. Because of this, our initial data showed artificial behavior – we were moving too fast and not sitting/browsing enough as a normal customer would. We were also avoiding check-out counters and often missed signature sounds like bar-code scanning beeps. To circumvent this, we decided to purchase coffee and food in sit-down places. For other kinds of stores, we decided to mimic the movement of another customer also present in that store. We arbitrarily picked a person and moved synchronously with him or her. While he/she browsed an item on the shelf, we imitated from a distance; while he/she moved to a different shelf, or waited in a check-out queue, we tried to do the same. We believe that our fingerprints reflect the typical customer. However, they do not capture atypical behavior, such as a person picking up pre-ordered food from a restaurant, or a clothing store customer picking a dress very quickly. In that sense, one may interpret our results to be partly optimistic.

5.2 Performance

We begin by qualitatively showing that GSM based locations are macro in nature, hence, insufficient to identify the phone's logical location. Figure 8(a) shows one example. An iPhone localized itself on the parking lot of a strip mall even though the user entered the shop marked with a cross. Given that there are several shops around the parking lot (marked with circles), the phone's logical location was ambiguous. We observed similar behavior in all ten clusters we experimented with. Nonetheless, GSM was valuable because it was omnipresent and effectively identified the macro-location of the phone. SurroundSense was then applied to discriminate logical locations within this macro-location.

Per-Cluster Accuracy

To evaluate SurroundSense, we war-sensed 51 shops organized in 10 clusters. Table 1 shows the number of shops in each cluster. The first nine clusters are in a university town, and were fingerprinted by 4 different students (in groups of 2). Each group visited the clusters at very different times; while at a cluster, each student visited the shops individually with time differences of at least 10 minutes. The tenth cluster is located in India, and was visited by only two people. We evaluate SurroundSense by cross-validating the fingerprints. Specifically, for every cluster, we use person X's fingerprints as the database, and compute the other users' accuracy against it. We repeat this for all the 4 identities of X. This gives us 12 localization results per logical location. We present the results next.

Figure 9(a) illustrates the average accuracy percentages per-cluster across different localization modes (each mode comprising of a different combination of sensors). We evaluate 4 modes offering the user with multiple options to choose from. We define the modes as follows:

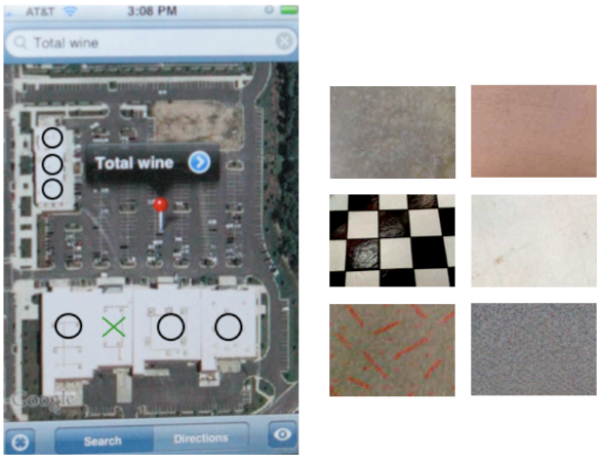

Figure 8: (Left) iPhone's GSM localization places the wine shop on the parking lot. The cross shows the wine shop, while circles show other stores within the same macro-location. (Right) Rich diversity in floor colors from a single macro-location.

1. **WiFi-only (WiFi)** is an adaptation of existing WiFi-based fingerprinting to suit logical localization.
2. **Sound, Accelerometer, Light and Color (Snd-Acc-Lt-Clr)** is the best option for places where WiFi is unavailable.
3. **Sound, Accelerometer and WiFi (Snd-Acc-WiFi)** is useful when the phone's camera is not exposed.
4. **SurroundSense (SS)** is the combined scheme with all modes of ambience fingerprinting.

The average accuracy across clusters 1 to 9 is presented in Table 2. SurroundSense achieves an average accuracy of 87%, an appreciable improvement over WiFi which achieved 70%. In the cluster in India (cluster 10), WiFi was unavailable. Nonetheless, SurroundSense achieved 100% accuracy owing to a rich diversity in the non-RF ambience. We zoom into the results from each cluster and examine the behavior/performance of each localization mode. We consistently observe that even though the locations were similar on one or more sensing dimensions, across all the dimensions their ambiences were diverse and identifiable.

The efficacy of SurroundSense is best represented in clusters 1 and 2. All the sensors (WiFi, sound, accelerometer, and color) contribute towards improving the localization accuracy to 90%. Cluster 3 reflects our hypothesis that collocated businesses may have incentives to exhibit unique ambiences. Specifically, all stores in this cluster were dining places, but the diversity in light intensities and colors was sufficient to distinguish them logically.

Cluster 4 achieves the lowest SS accuracy of around 72%. This is a strip mall in which multiple shops happened to have light brown hardwood floors. This uniformity makes the color

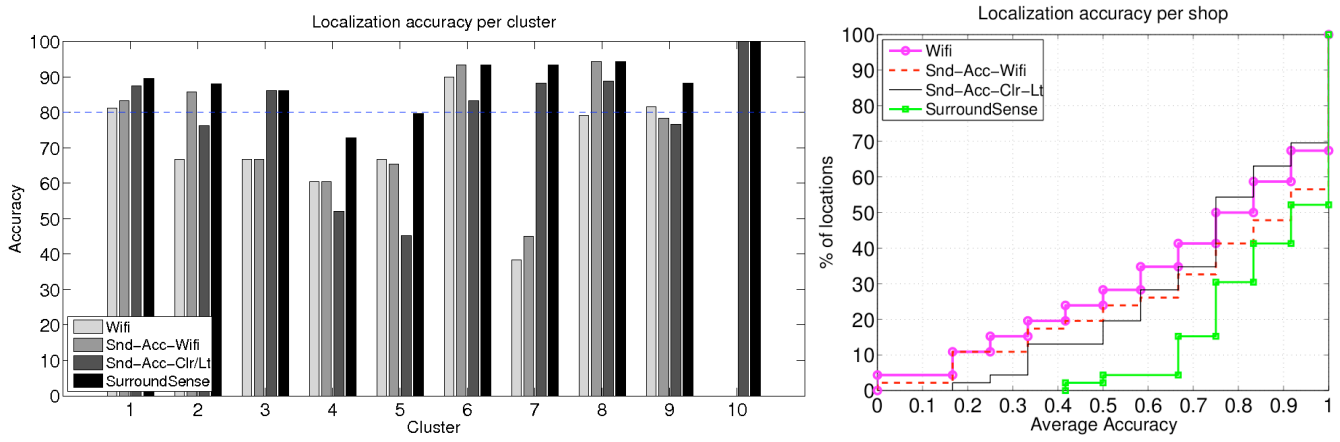

Figure 9: (a) Average accuracy per-cluster. (b) CDF of per-shop accuracy.

Cluster	1	2	3	4	5	6	7	8	9	10
No. of Shops	4	7	3	7	4	5	5	6	5	5

Table 1: Number of shops per cluster.

Mode	WiFi	Snd-Acc-WiFi	Snd-Acc-Lt-Clr	SS
Accuracy	70%	74%	76%	87%

Table 2: Average accuracy across clusters 1-9.

fingerprints less effective (we observed an average accuracy of 50% for color alone). In addition the nature of the service offered (mainly product-browsing) prevents the accelerometer from filtering out candidate shops. Sound offers a small improvement because these locations are generally crowded, and thus, exhibits similar background noise. While WiFi is available at each location, the small sizes of these shops exposed them to the dividing wall problem. However even in face of such unfavorable conditions, SS achieved a reasonable accuracy of 72%. A similar situation was sensed in cluster 5, but the combination of all the sensors again raised the accuracy to 80%.

An interesting situation occurred in cluster 7. We recorded the same audible access points in 4 of the shops. As expected, WiFi localization achieved very low accuracy (less then 40%). However Snd-Acc-Lt-Clr extracted enough diversity from the non-RF ambience to raise the localization accuracy to over 90%.

The Snd-Acc-WiFi mode achieves moderate improvement over WiFi alone. In some cases (clusters 5 and 9), sound and accelerometer filters incur false negatives causing Snd-Acc-WiFi to be less accurate than WiFi alone. Nevertheless, they adequately compensate in clusters 2 and 8, raising the accuracy by factors of 15 to 20%. Cluster 8 includes a mixture of restaurants, a loud music shop, a quiet antique book store and an art gallery. Combined with WiFi support all these shops can be accurately localized even without light and color.

Lastly, even if WiFi is unavailable (as in cluster 10 in India) SurroundSense may still be able to achieve a high accuracy by fingerprinting the non-RF ambience. This can be valuable in enabling location-based services in parts of the world where mobile phones are popular, while WiFi is not.

Per-Shop Accuracy

To understand the localization accuracy on a per-shop basis, we plot the cumulative distribution function (CDF) in Figure 9(b). Evident from the graph, 47% of the shops can be localized perfectly using SurroundSense. In contrast, RF and non-RF fingerprinting achieve perfect match for around 30% of the stores. Interestingly, Snd-Acc-WiFi displays a larger variance – it outperforms WiFi and Snd-Acc-Clr-Lt in the regime of high accuracy, but is relatively worse for low-accuracy regimes. This is because WiFi displays some type of a bimodal behavior – it either achieves a high accuracy, or suffers seriously for specific locations. The median accuracy with SS, Snd-Acc-WiFi, Snd-Acc-Lt-Clr, and WiFi are 92%, 92%, 75%, and 75% respectively. Clearly, the combination of multi-modal fingerprinting offers gain in logical localization.

Per-User Accuracy

To understand the user experience with SurroundSense, we simulated virtual users and observed the localization accuracy each may observe. Each simulated user was assigned to a random set of stores (between 4 to 8), selected randomly from the 46 stores in cluster 1 to 9. We simulated 100 such users, and report the average accuracy that each user may experience. Figure 10 presents the CDF of the average accuracy per-simulated-user. From the figure, 2% of SurroundSense users achieve between 73% and 75% accuracy. The accuracy grows to an average of 83% or more for 80% of the users. The median accuracy is around 88%, while 10% users experience 96% accuracy or more. Snd-Acc-WiFi and Snd-Acc-Lt-Clr are comparable, achieving a median of around 76%. However, they consistently outperform WiFi which achieves a median of around 68%.

Per-Sensor Accuracy

Table 3 zooms into the performance of individual and groups of sensors in SurroundSense. In the interest of space, we hand-picked 6 examples that exhibit some of the merits and demerits of each sensor. When using the filters (left part of the table), we show the average number of shops left to con-

Filters					Filters + Matching						
C#	Acc	Snd	WiFi	Snd-Acc-WF	WF	Snd-Acc-WF	C/L	Snd-C/L	Acc-C/L	Snd-Acc-C/L	SS
C7	5.00	4.83	2.25	2.08	0%	16%	100%	100%	100%	100%	100%
C2	1.00	6.25	2.33	1.00	83%	100%	75%	75%	100%	100%	100%
C3	3.00	2.50	1.92	1.58	91%	91%	75%	91%	75%	91%	100%
C8	4.00	5.58	1.33	1.00	83%	100%	100%	100%	100%	100%	100%
C6	1.25	3.33	2.00	1.00	75%	100%	33%	100%	100%	100%	100%
C3	3.00	2.25	2.33	1.67	41%	33%	91%	66%	91%	66%	66%

Table 3: Examples of average performance per sensor at different business locations. The first column shows the cluster number to which that particular business location belongs.

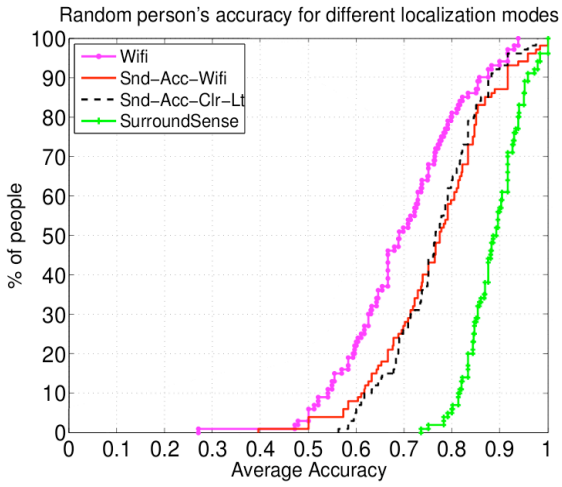

Figure 10: Random person accuracy CDF

sider after applying the respective filter. In the right side of the table, we show the percentage of tests for each location where the phone was localized correctly using only the specified sensors. For example, in row 2, the accelerometer always filters out all but one location from the cluster. As the second part of the table shows, whenever the accelerometer is used, SurroundSense's localization accuracy is always 100%. Row 4 shows a similar result for the color/light sensor. Using only the camera, SurroundSense is able to achieve 100% accuracy in this location. Finally, the last row gives an example of the high cost of false negatives in a filter. We see that using only color gives average accuracy of 91%. When sound is added, the average is reduced to 66%. Thus, sound filters out the fingerprint from the correct location in some cases. Once the correct location is filtered out, the final match will inevitably be incorrect, regardless of the color/light sensor's performance. In order to minimize the number of such cases, we were conservative when designing the filters in SurroundSense.

6. LIMITATIONS AND FUTURE WORK

We discuss some limitations with the current version of SurroundSense, along with our ongoing/future work.

Energy Considerations

This paper does not consider the energy tradeoffs with SurroundSense. Independent research is in progress on energy-efficient localization and sensing [6, 15, 29] – we believe SurroundSense will benefit from these works. In addition, we are developing simple sensing mechanisms to detect when a phone goes outdoors. One idea is to attempt GPS localization – if a GPS lock is obtained, the phone can be assumed outdoors. Variation in GSM signal strengths and temperature sensing are also promising methods. Once the phone is detected outdoors, SurroundSense can be turned off.

Non-business Locations

Our evaluation spanned business locations. Offices, libraries, airports, and other facilities may also require localization, and may lack the ambience-diversity inherent in businesses. However, these places may be considered as a broad logical location, and the *dividing wall* problem may not be as critical from the application's perspectives.

Localizing in Real Time

An accelerometer trace requires some time to converge (e.g., a person in a restaurant may need to wait in a queue before sitting). We plan to investigate faster methods of localization without compromising accuracy. Compasses and nearby Bluetooth devices can be promising.

Compass and Peer Devices

Electronic compasses already on Nokia 5140 phones can provide geographic orientation (e.g., 36.5° East). The geographic orientation may be correlated to the layout of furniture and shopping aisles in stores. For instance, users may be forced to sit on chairs or face grocery shelves while shopping. Since layouts are likely to vary between nearby stores, and because layouts are stationary over time, they may be a reliable indicator of the user's location. Of course, a user's rotation may add noise to the estimation, but the noise can be filtered out without difficulty. Discovering other phones in the neighborhood can also be suggestive of location. Classrooms may be places in which nearby phones are typically from friend lists, while traveling on a public bus may result in a high density of "unknown" phones in the surroundings. We plan to explore neighbor discovery for the purpose of localization.

Hierarchical Localization

It may be useful if logical locations can be grouped into a broader category. For example, Starbucks, Seattle's Best, Bean Traders, etc. could all be categorized as *coffee shops*. Similar categories could be *chinese restaurants, women's clothing, grocery, toys*. Certain applications may benefit from such hierarchy – imagine a coffee company advertising its brand to all coffee-shop visitors. Hierarchical categorization of logical places requires deeper insights into both the diversity and the homogeneity of fingerprints. Ambience attributes that are ho-

mogeneous across coffee shops, but diverse from other stores, need to be carefully sifted out. We leave this for future work.

7. RELATED WORK

Indoor localization and context awareness are well known problems in wireless/mobile computing. Several creative approaches have been proposed, broadly classified as active and passive, and sub-classified into RF and sensing techniques. We describe the key ideas here.

Active RF techniques refer to installing special hardwares and softwares in the environment to achieve high precision indoor localization. This category includes the Cricket [22] and Nokia systems (discussed in the introduction), as well as LEASE, PAL, PINPOINT (see [31] and references therein). Research on link signatures showed location distinction [32] using interesting techniques. Device movement can be detected based on variation in link qualities from the device to multiple listeners in the surrounding. As argued earlier, active techniques are certainly effective in high-budget applications, but are unlikely to scale over city-wide areas. Passive localization schemes bypass the need for pre-installed infrastructure.

Passive RF localization schemes sense RF signals from existing devices in the surrounding [2, 4, 20]. Place Lab [4] is a successful project where signals from different WiFi and GSM base stations are utilized for localization. A wireless map is created by war-driving a region; the mobile device localizes itself by comparing overheard APs/cell towers against the wireless map. UCSD's Active Campus project [12] adopts similar techniques of localization, but assumes that the location of the WiFi access points are known *a priori*. RADAR [2] also operates on WiFi fingerprinting, and is capable of achieving up to 5m accuracy in indoor settings. As a tradeoff to accuracy, RADAR needs to carefully calibrate WiFi signal strengths at many physical locations in the building. High resolution calibration is time-consuming and may not scale over wide areas.

Active/Passive Behavior Sensing has been utilized for context-aware computing [5, 8, 9, 24, 25, 30] An interesting work explored image matching [8] to infer context information. However, the main assumptions of these approaches is that objects in the pictures (e.g. furniture) are long lived and stable at their locations. This may not hold for products in a shop as they are sold, replaced or moved. Also, complex image processing may not be scalable for localization-related applications. SurroundSense uses floor pictures and thus avoids most short lived objects that can interfere with the data matching process. Authors in [9] have mounted cameras on shoes to achieve a vision of the floor; however, this is for applications in injury-free walking and improved navigation. We combine multiple modes of sensing towards the goal of indoor, logical localization.

8. CONCLUSION

Various mobile computing applications benefit from knowing a user's logical location, as opposed to her physical coordinates (like the latitude and longitude). We presented *SurroundSense*, a somewhat non-conventional approach to logical localization. The main idea is to fingerprint a location based on its ambient sound, light, color, RF, as well as the layout-induced user movement. This fingerprint is then used to identify the user's location. SurroundSense may not qualify as a stand-alone localization technique. However, in conjunction with GSM based macro-localization, *SurroundSense* can perform micro-localization based on the inherent properties of the ambience. We believe SurroundSense is an early step towards a long-standing research problem in indoor localization. Further research in fingerprinting techniques, sophisticated classification, and better energy management schemes could make SurroundSense a viable solution of the future.

9. ACKNOWLEDGMENTS

We sincerely thank the anonymous reviewers for their valuable feedback on this paper. We also thank Victor Bahl, Nitin Vaidya, Srihari Nelakuditi, and Suman Banerjee for their insightful comments and encouragement during the formative stages of SurroundSense. Finally, we are grateful to NSF, Nokia, Verizon, and Microsoft Research, for partially funding our research in mobile computing.

10. REFERENCES

[1] Nokia Research Center and Cambridge NanoScience Center, The Morph Concept. *http://www.nokia.com/A4852062*.

[2] P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *IEEE INFOCOM*, 2000.

[3] M. Chen, T. Sohn, D. Chmelev, D. Haehnel, J. Hightower, J. Hughes, A. LaMarca, F. Potter, I. Smith, and A. Varshavsky. Practical metropolitan-scale positioning for gsm phones. In *UbiComp*, 2006.

[4] Y. Chen, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *ACM MobiSys*, 2005.

[5] B. Clarkson, K. Mase, and A. Pentland. Recognizing user context via wearable sensors. In *The 4th IEEE International Symposium on Wearable Computers*, 2000.

[6] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox. EnLoc: Energy-efficient localization for mobile phones. In *IEEE INFOCOM Mini Conference*, 2009.

[7] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[8] R. Elias and A. Elnahas. An accurate indoor localization technique using image matching. In *Intelligent Environments*, 2007.

[9] P. Fitzpatrick and C. Kemp. Shoes as a platform for vision. In *The 7th IEEE International Symposium on Wearable Computers*, 2003.

[10] J. Funk. The future of the mobile phone internet: an analysis of technological trajectories and lead users in the japanese market. *Elsevier Science Direct*, 2004.

[11] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt. Micro-blog: Sharing and querying content through mobile phones and social participation. In *ACM MobiSys*, 2008.

[12] W. G. Griswold, P. Shanahan, S. W. Brown, R. Boyer, and M. Ratto. Activecampus - experiments in community-oriented ubiquitous computing. *IEEE Computer*, 2003.

[13] A. Kansal, M. Goraczko, and F. Zhao. Building a sensor network of mobile phones. In *IPSN*, 2007.

[14] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[15] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær. Entracked: energy-efficient robust position tracking for mobile devices. In *ACM MobiSys*, 2009.

[16] N. D. Lane, E. Miluzzo, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. Metrosense project: People-centric sensing at scale. In *First Workshop on World-Sensor-Web (WSW'2006)*, 2006.

[17] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: The design, implementation and evaluation of cenceme application. In *ACM Sensys*, 2008.

[18] P. Mohan, V. Padmanabhan, and R. Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *ACM Sensys*, 2008.

[19] Z. News. Wearable mobile phones hit US market. *http://news.zdnet.co.uk*, 2008.

[20] D. Niculescu and B. Nath. VOR base stations for indoor 802.11 positioning. In *ACM MobiCom*, 2004.

[21] A. Ofstad, E. Nicholas, R. Szcodronski, and R. R. Choudhury. Aampl: accelerometer augmented mobile phone localization. In *MELT*, 2008.

[22] N. B. Priyantha. *The cricket indoor location system*. PhD thesis, 2005.

[23] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. *American Association for Artificial Intelligence*, 2005.

[24] N. Ravi, P. Shankar, A. Frankel, A. Elgammal, and L. Iftode. Indoor localization using camera phones. In *WMCSA*, 2006.

[25] N. Ravi and L. Iftode. Fiatlux: Fingerprinting rooms using light intensity. In *Pervasive*, 2007.

[26] M. A. Smith, D. Davenport, and H. Hwa. Aura: A mobile platform for object and location annotation. In *Ubicomp*, 2003.

[27] T. Sohn, K. A. Li, G. Lee, I. E. Smith, J. Scott, and W. G. Griswold. Place-its: A study of location-based reminders on mobile phones. In *UbiComp*, 2005.

[28] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *ACM CHI*, 2004.

[29] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *Mobisys '09: Proceedings of the 7th international conference on Mobile systems, applications, and services*, 2009.

[30] C. Yiu and S. Singh. Tracking people in indoor environments. In *5th International Conference on Smart homes and health Telematics*, 2007.

[31] M. Youssef, A. Youssef, C. Reiger, A. Shankar, and A. Agrawala. Pinpoint: An asynchronous time-based location determination system. In *ACM MobiSys*, 2006.

[32] J. Zhang, M. H. Firooz, N. Patwari, and S. K. Kasera. Advancing wireless link signatures for location distinction. In *ACM MobiCom*, 2008.