If You See Something, Swipe towards It: Crowdsourced Event Localization using Smartphones

Robin Wentao Ouyang, Animesh Srivastava, Prithvi Prabahar, Romit Roy Choudhury, Merideth Addicott†, F. Joseph McClernon†
Duke University, Durham, NC, USA
†Duke University Medical Center, Durham, NC, USA
{w.ouyang, animesh.srivastava, prithvi.prabahar, romit.rc, merideth.addicott, francis.mcclernon}@duke.edu

ABSTRACT

This paper presents *iSee*, a crowdsourced approach to detecting and localizing events in outdoor environments. Upon spotting an event, an *iSee* user only needs to swipe on her smartphone's touchscreen in the direction of the event. These swiping directions are often inaccurate and so are the compass measurements. Moreover, the swipes do not encode any notion of how far the event is located from the user, neither is the GPS location of the user accurate. Furthermore, multiple events may occur simultaneously and users do not explicitly indicate which events they are swiping towards. Nonetheless, as more users start contributing data, we show that our proposed system is able to quickly detect and estimate the locations of the events. We have implemented *iSee* on Android phones and have experimented in real-world settings by planting virtual "events" in our campus and asking volunteers to swipe on seeing one. Results show that *iSee* performs appreciably better than established triangulation and clustering-based approaches, in terms of localization accuracy, detection coverage, and robustness to sensor noise.

Author Keywords

Smartphone sensing; crowdsourcing; event localization

ACM Classification Keywords

H.3.4 Information Storage and Retrieval: Systems and Software; C.2.4 Computer-Communication Networks: Distributed Systems

General Terms

Algorithms; Design; Experimentation

INTRODUCTION

This paper proposes *iSee*, a participatory sensing system that detects and localizes events in outdoor public environments. As an example event, consider smoking activity in a university campus. When an iSee user notices a person smoking, she only needs to swipe on the smartphone screen in the direction of the event, even during walking or in a vehicle. This simple gesture costs no more than one second, imposing little burden on the user. Moreover, it allows the user to conveniently indicate any direction of event without turning around. For example, if the user does not intend to reveal that she is reporting on the event, she could move to a comfortable location – say when the smoker is behind her – and then swipe backwards. Sensed data from the user's phone, including GPS location, GPS location confidence, compass direction, user-swiped angle and time, are recorded and transmitted to a central server. Using the aggregated data, iSee aims to detect and localize every occurrence of the event. As more data arrive to the server, more events are likely to get detected, while also improving each of their localization accuracies. Timestamps of user-swipes also yield temporal statistics for the events, ultimately offering a spatio-temporal heatmap of the events of interest.

A natural question one may ask is, *are certain types of events more amenable to such crowdsourcing techniques and what are the example applications?* We believe that public events with the following properties are most amenable to iSee: (1) Events that occur within visible distance from the user, but not too close (since in that case the user's own location is an adequate estimate of the event location). (2) Events that are reasonably common in everyday life (otherwise, if it is a rare occurrence, it is likely that someone will explicitly report on it anyway). (3) Events that show spatial and temporal variation in a large time scale, so that it is not easy to perform a one-time scan and record the event locations.

We find at least two events that fit these properties, namely, (1) localizing areas of smoking in campuses and cities and (2) localizing graffiti in public places. Clearly, both are important applications as evident from the studies in [16, 13, 7]. To quote briefly, millions of dollars are invested by the government and health-care departments to identify smoking regions and measure the effectiveness of anti-smoking ads and campaigns in those regions. Similarly, a huge amount of resources are invested towards early detection and prevention of city graffiti.

A number of challenges arise in translating the above goals into reality. (1) Phone sensors are noisy. GPS can lead to location errors about 10-20 meters in regular operation.

UbiComp'13, September 8–12, 2013, Zurich, Switzerland.

http://dx.doi.org/10.1145/2493432.2493455

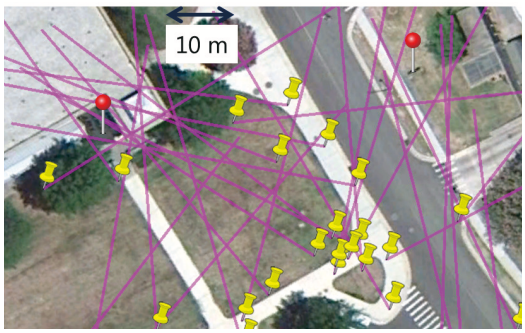

Figure 1. iSee receives a number of user-swipes with erroneous starting points (yellow pins) and directions (purple lines), and must infer the true locations of the events (red pins). The problem is non-trivial even via visual inspection.

The phone compass is affected by variations in the Earth's magnetic field and nearby ferromagnetic material. The user-swiped angle is also subject to human errors. iSee needs to cope with such heavy sensor noise. (2) User-swipes are distributed over space and time, and iSee has no prior knowledge on the correspondence between swipes and event hotspots. In other words, iSee does not know *a priori* which swipes should be grouped together to calculate an event location. This makes traditional triangulation and optimization-based localization [14] not directly applicable. (3) iSee should be able to work when only limited swipes are available (i.e., quickly detect events) and also be scalable as more swipes are received. Figure 1 captures these difficulties.

iSee tackles these problems by dividing the area of interest into grid cells and representing each user-swipe as a 2D polygon which incorporates GPS and angular errors. Projecting multiple polygons onto the cells results in a cell confidence map. Treating this map as an image, iSee is able to detect hotspots through well-established image processing techniques. By backtracking which swipes cover a detected hotspot and performing temporal clustering, the swipe-hotspot correspondence is obtained and the temporal pattern for that hotspot is analyzed. An optimization-based scheme is then followed to refine the event locations.

We implemented iSee on different models of Android smartphones and distributed them to 6 volunteers. For experimentation, we placed 20 red flags which were shuffled over time around our campus to represent virtual events. The experiment was performed for 6 days, after which iSee was able to detect 95% of the flags with around $10m$ of accuracy. Based on these results, we believe iSee demonstrates an important step towards a low-cost and near-continuous solution leveraging crowdsourcing for event reporting and localization.

The key contributions in iSee are summarized as follows.

1. We identify an opportunity in multimodal sensing and crowdsourcing to locate event hotspots in outdoor environments. The effort imposed on participants is extremely light – just a swipe on the smartphone screen.
2. We propose efficient algorithms for detecting and localizing events, and associating swipes with event hotspots to investigate events' temporal properties.
3. We implement the system on Android smartphones and perform real-world experiments to evaluate the system performance – we find iSee consistently outperforms the classical triangulation and clustering-based approach.

SYSTEM OVERVIEW

In this section, we first overview the required data input for enabling iSee and then provide a formal statement of the problems that iSee intends to solve. Finally, we introduce the proposed architecture of iSee.

Data Input

In order to detect and locate event hotspots, the following information is required.

1) The user (device) ID u.

2) The time t when the user saw an event (made the swipe).

3) The user's location $l = (x, y)$ and its accuracy a. They are obtained from the phone's GPS. a is typically defined as the radius of 68% confidence, i.e., there is a 68% probability that the true location is inside a circle centered at l with radius a.

4) The user-swiped angle θ with respect to the East (Figure 2). It is calculated by combing two angles that can be measured by the phone: i) The angle θ_c between the phone's y-axis and the magnetic North which is obtained from the phone's compass reading. ii) The user-swiped angle θ_a on the touchscreen with respect to the phone's x-axis. According to Figure 2, we can prove that θ can be calculated as $\theta = (\theta_a - \theta_c) \mod 2\pi$ no matter how the phone orientates as long as it is in the horizontal plane.

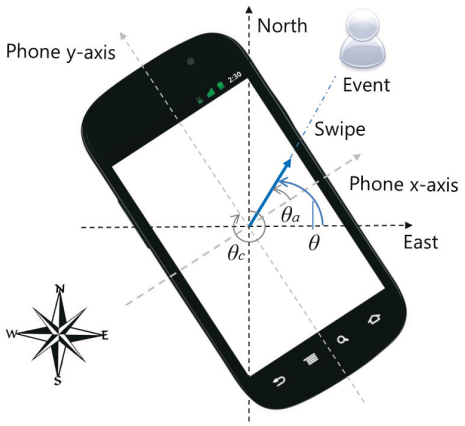

Figure 2. Illustration of user-swiped angle.

Collectively, we denote a user's swipe as $s_i = \{u_i, t_i, l_i, a_i, \theta_i\}$ and the set of all the swipes of interest over a spatial and temporal range (e.g., over the past 3 days at University A) as $\mathcal{S} = \{s_i\}$. To guarantee the proper use of iSee, we require the phone to be parallel to the ground, i.e., in the horizontal plane. We thus record the accelerometer readings as well (denoted as acc_x and acc_y; acc_z is not considered to allow people to make swipes even during walking). A swipe is considered valid only when the following conditions are satisfied while swiping.

$$\text{abs}(\text{mean}(acc_k)) \leq 0.05g,\ \text{std}(acc_k) \leq 0.1g, k \in \{x, y\},$$

where g is the gravity of Earth.

Problem Statement

Given the data set $\mathcal{S}$ contributed by crowds, iSee intends to answer the following questions:

1. How many distinct event locations were indicated by crowds?
2. Where were these events?
3. When did these events happen?

Architecture

Figure 3 depicts the architecture of iSee. iSee smartphone clients collect user-contributed data and transmit them to the iSee server. The server consists of three key components: basic data analysis (BDA), grid-based event localization (GEL), and temporal analysis and location refinement (TALR). BDA partitions the area of interest into grid cells and performs preliminary data analysis to extract useful information – a swipe-cell indicator matrix and a cell confidence matrix. Such information is then used by GEL to locate event hotspots. TALR then finds swipe-hotspot correspondence based on the output of GEL, analyzes the events' temporal properties, and performs an optimization-based location refinement. Finally, iSee server outputs the inferred event locations and occurrence time.

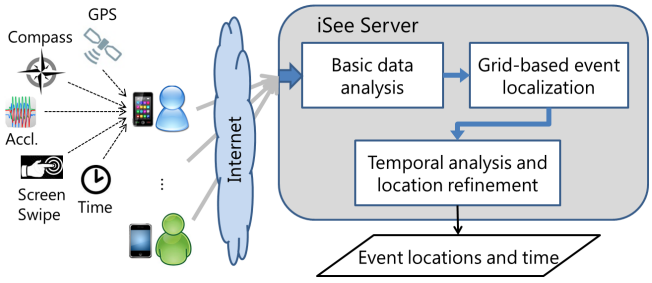

Figure 3. Architecture of iSee.

SYSTEM DESIGN

This section discusses the details of the three key components that underpin iSee. For ease of illustration, all the locations below are transformed to a local UTM coordinate system by subtracting the minimum values along respective axes.

Basic Data Analysis (BDA)

The purpose of BDA is to partition the area of interest into grid cells and perform basic analysis of collected data to generate useful information matrices – a swipe-cell indicator matrix and a cell confidence matrix, which will be used by other components. Algorithm 1 provides an overview of BDA which consists of the following steps.

Grid partitioning (Line 2). In order to systematically analyze data and detect events, this step partitions the area of interest R into grids each with $r \times r$ square meters. This results in $n_r \times n_c$ square cells, where n_r and n_c are the number of rows and columns respectively. Each cell c_i is then represented using its centroid location. After partitioning, R can be represented as $R = \{c_i\}$.

Representing each swipe as a trapezoid (Line 4). In order to reflect the errors associated with GPS location and user-swiped angle, we represent each swipe as a 2D polygon T,

Algorithm 1 Basic data analysis

Input: User-swipes $\mathcal{S} = \{s_i\}$, bounding rectangle of the area of interest R and grid size r.
Output: Swipe-cell indicator matrix I and cell confidence matrix W.

1: Initialization: $I = \mathbf{0}$, $W = \mathbf{0}$.
2: $\{c_i\}$ = GridPartitioning(R, r)
3: **for** $i = 1 : |\mathcal{S}|$ **do**
4: T_i = TrapezoidVertexComputing(s_i)
5: $[I(i,:), m_i]$ = TrapezoidCellIntersection(T_i, $\{c_j\}$)
6: W = CellConfidenceUpdating(W, $I(i,:)^{2D}$, m_i)
7: **end for**
8: **return** I and W

which has a high probability to cover the event location. We assume that a user can only clearly observe an event within a maximum visible distance d_m (it is application-dependent). Then the event's location can be in an area formed by shifting a circle centered at the user's GPS location l with radius a (GPS accuracy) along the user-swiped direction θ with distance d_m. This shape can be bounded by a rectangle shown in dashed lines in the center of Figure 4.

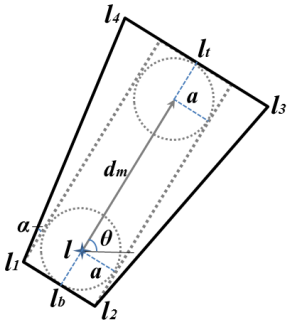

Figure 4. Trapezoid representation of a swipe.

To incorporate the angular errors from the phone compass and the user-swipe, we add 2α to the measured angle θ. In other words, the event location can be in the angular range $[\theta - \alpha, \theta + \alpha]$. We set $\alpha = \pi/18$ (i.e., $10°$). After incorporating α, the resulting polygon T is approximated as a trapezoid with vertices $l_{1:4}$ shown in Figure 4. The coordinates of these vertices are given in Table 1.

Table 1. Coordinates of a trapezoid's vertices, where $b = a + (d_m + 2a)\tan\alpha$ and $l = (x, y)$ is the user's GPS location.

loc	x-coordinate	y-coordinate
l_b	$x_b = x + a\cos(\pi + \theta)$	$y_b = y + a\sin(\pi + \theta)$
l_t	$x_t = x + (a + d_m)\cos\theta$	$y_t = y + (a + d_m)\sin\theta$
l_1	$x_1 = x_b + a\cos(\frac{\pi}{2} + \theta)$	$y_1 = y_b + a\sin(\frac{\pi}{2} + \theta)$
l_2	$x_2 = x_b + a\cos(\frac{3\pi}{2} + \theta)$	$y_2 = y_b + a\sin(\frac{3\pi}{2} + \theta)$
l_3	$x_3 = x_t + b\cos(\frac{3\pi}{2} + \theta)$	$y_3 = y_t + b\sin(\frac{3\pi}{2} + \theta)$
l_4	$x_4 = x_t + b\cos(\frac{\pi}{2} + \theta)$	$y_4 = y_t + b\sin(\frac{\pi}{2} + \theta)$

Besides our proposed trapezoid representation, a classical choice is to represent each swipe as a directed line segment with a maximum visible distance d_m. Figure 5 shows three swipes with both trapezoid ($T_{1:3}$) and line segment representation ($L_{1:3}$). As can be seen, trapezoid representation considers the uncertainty in the measurements and thus has a high

probability to cover the true event location as well as to intersect with each other. Their area of intersection provides a reasonable estimation of the true event location. In contrast, due to large GPS and angular errors, limited line segments usually result in low-quality intersections. In Figure 5, L_1 becomes useless and three line segments produce only one intersection which becomes the estimated event location. If we allow line intersection (without the maximum visible distance constraint), an even worse result will be produced.

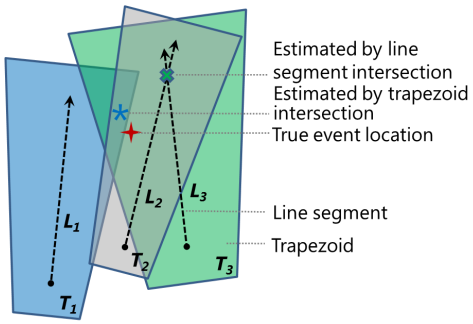

Figure 5. Trapezoid and line segment representation.

Intersecting each trapezoid with cells (Line 5). To find the possible event locations, intersecting all the trapezoids T_i is a potential solution. To achieve this, computational geometry techniques such as convex polygon intersection [9] could be used, but they are computationally expensive. Moreover, all the trapezoids need to be kept in order to intersect with newly arrived ones. This process requires considerable storage.

We propose an efficient method where each trapezoid can be processed *independently* and need not be stored after processing. Towards this end, we intersect each trapezoid T_i with all the grid cells c_j. This is essentially to determine which cell centers are *inside* the trapezoid T_i. We use the ray casting algorithm [2], which is fast and efficient, to achieve this goal. An example of intersection results is shown in Figure 6.

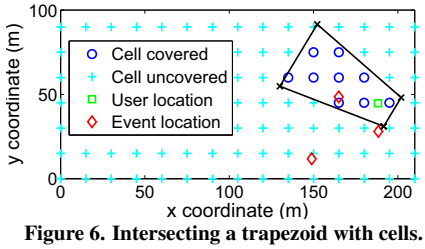

Figure 6. Intersecting a trapezoid with cells.

Assume there are altogether n_s swipes. We create a swipe-cell indicator (SCI) matrix I with n_s rows and $n_r \times n_c$ columns to record the intersection results. If swipe i (trapezoid T_i) covers cell c_j, then the (i, j) entry of the matrix I is 1. Otherwise, it is 0. From each row of I, we can examine which cells are covered by a given swipe; while from each column, we can examine which swipes contribute to a given cell. The latter is used to retrieve corresponding swipes for a detected event hotspot. Such swipe-hotspot correspondence is important since it can enable temporal analysis and optimization-based localization. Each row of the SCI matrix can also be rewritten as an equivalent $n_r \times n_c$ matrix form, which we denote as $I(i,:)^{2D}$. After each intersection, the number m_i of cells covered by trapezoid T_i is also reported.

Updating the cell confidence matrix (Line 6). In order to reflect the confidence that a cell is to be a potential event hotspot, we create a confidence variable w_k (initialized as 0) for each cell and store these variables in an $n_r \times n_c$ cell confidence matrix W. If a trapezoid T_i covers m_i cells, then the confidence w_k for each of these covered cells is updated as $w_k = w_k + 1/m_i$. This is because the size of a trapezoid is positively related to the errors in GPS and user-swiped angle. The larger a trapezoid is, the less reliable each covered cell is. The matrix W is updated until all the trapezoids are processed. Clearly, the size of W depends only on the size of the area of interest and the grid, but not the number of swipes. Processing on W is thus efficient and scalable, no matter how many swipes are contributed by crowds.

Grid-based Event Localization (GEL)

Based on the cell confidence matrix W generated by BDA, GEL intends to detect event hotspots from it. Intuitively, a hotspot should have locally high confidence over neighboring cells. GEL treats W as an image, with each cell as a pixel and the cell confidence as (scaled) image intensity. Figure 7(a) shows an example W. We term W as the cell confidence map below. Algorithm 2 provides an overview of GEL, which consists of the following steps.

Algorithm 2 Grid-based Event Localization

Input: Cell confidence matrix W and confidence threshold $thre_c$.
Output: Estimated event locations $\mathcal{E}^g$.

1: Initialization: $\mathcal{E}^g = \emptyset$.
2: W^f = MapFiltering(W, $thre_c$)
3: $\mathcal{G}$ = ConnectedComponentDetection(W^f)
4: **for** $i = 1 : |\mathcal{G}|$ **do**
5: $[b_i, ind_i]$ = LocalMaxDetection&Pruning($\mathcal{G}_i$)
6: **if** $ind_i = 0$ **then**
7: b_i = CellofMaxConfidence($\mathcal{G}_i$)
8: **end if**
9: e_i^g = CellIDtoLocation(b_i)
10: $\mathcal{E}^g = \mathcal{E}^g \bigcup \{e_i^g\}$
11: **end for**
12: **return** $\mathcal{E}^g$

Map filtering (Line 2). We filter out unreliable cells whose confidences are below a threshold $thre_c$ (set to $0.25 \max(W)$). This filtering process can also partition the large connected map into several small connected components, each of which would contain at least one event hotspot.

Connected components detection (Line 3). We use the flood-fill algorithm [1] to detect the connected components $\mathcal{G}$ on the filtered confidence map W^f.

Event location estimation (Lines 4–11). To pinpoint event locations from each connected component, GEL performs 2D local maximum detection since intuitively an event hotspot should have locally high confidence over neighboring cells. If a group of local maximums are within $2\sqrt{2}$ (Euclidean distance on pixels) to each other, we retain only the one with the maximum confidence in order to reduce false alarms. We also try to use the weighted center of these local maximums

as the detected event location, but find that it actually introduces even larger errors since false alarms are not trustable. We prune local maximums because we are not able to differentiate very close locations due to GPS and angular errors. If no local maximum b_i is detected ($ind_i = 0$), GEL chooses the cell with the maximum confidence of the connected component as the event location. If multiple cells satisfy this condition, GEL chooses the one closest to the component center. The size of a connected component and the number of event locations detected also provide certain insights about whether there are missed hotspots or false alarms. Since these locations are detected on an image, they are pixel (cell) indices. We then map those indices to the corresponding geolocations. All these estimated locations e_i^g are inserted into a set $\mathcal{E}^g$. An example of estimated hotspots on the filtered confidence map is shown in Figure 7(b).

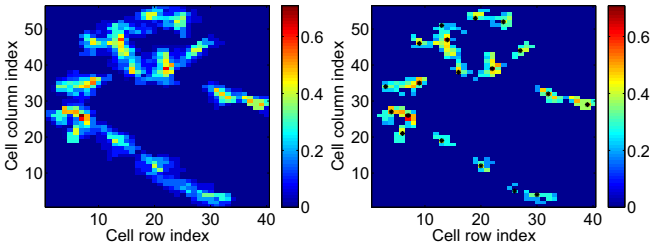

Figure 7. Left to right: (a) Original cell confidence map and (b) filtered confidence map with estimated hotspots (denoted as black stars).

Temporal Analysis and Location Refinement (TALR)

The aim of TALR is to obtain swipe-hotspot correspondence leveraging the estimated event locations $\mathcal{E}^g$ from GEL and the SCI matrix from BDA. Based on such correspondence, TALR performs temporal analysis and optimization-based localization for each hotspot. Algorithm 3 provides a formal description of TALR, which consists of the following steps.

Algorithm 3 Temporal Analysis and Location Refinement

Input: User swipes $\mathcal{S}$, estimated event locations $\mathcal{E}^g = \{e_i^g\}$ and swipe-cell indicator matrix I.
Output: Event occurrence intervals $\mathcal{I}$ and optimized event locations $\mathcal{E}^o$.

1: Initialization: $\mathcal{E}^o = \emptyset$.
2: **for** $i = 1 : |\mathcal{E}^g|$ **do**
3: $\mathcal{H}_i$ = SwipeRetrieval(e_i^g, I, $\mathcal{S}$)
4: $\mathcal{H}_i^c$ = HierarchicalClustering($\mathcal{H}_i$)
5: **end for**
6: $\{\mathcal{H}_i^r\}$ = ClusterRefinement($\{\mathcal{H}_i^c\}$)
7: **for** $i = 1 : |\mathcal{E}^g|$ **do**
8: $\mathcal{I}_i$ = TemporalAnalysis($\mathcal{H}_i^r$)
9: **if** NumOfSwipe($\mathcal{H}_i^r$) $\geq$ 3 & STD(Angle($\mathcal{H}_i^r$)) $\geq$ 0.2 **then**
10: e_i^o = OptimizeLocation($\mathcal{H}_i^r$)
11: **else**
12: $e_i^o = e_i^g$
13: **end if**
14: $\mathcal{E}^o = \mathcal{E}^o \bigcup \{e_i^o\}$
15: **end for**
16: **return** $\mathcal{I}$ and $\mathcal{E}^o$

Swipe retrieval (Line 3). This step is to retrieve the set of swipes $\mathcal{H}_i$ corresponding to a given estimated hotspot e_i^g by examining the corresponding column of the SCI matrix.

Hierarchical clustering (Line 4). This step is to purify $\mathcal{H}_i$ (filter out certain erroneous associations from nearby hotspots) and find a smaller set $\mathcal{H}_i^c$ whose elements correspond to the same hotspot with high probability. To achieve this, we perform a hierarchical clustering in the temporal domain. This is motivated by the observation that users observed the same event or consecutive events at a hotspot (confined location) should respond closely in terms of time. Moreover, for events repeatedly happened during certain intervals over days, those user-swipes could also be close in terms of time-of-day (but on different days).

We apply an agglomerative hierarchical clustering approach [10] which does not need to predefine the number of clusters and is also more robust to outliers compared with flat k-means clustering. We represent time in minutes and use absolute difference as the distance metric and shortest distance as the linkage function (measuring the distance between clusters). We finally use a predefined cutoff distance (set to 10 in this paper but it is application-dependent) on the linkage function to cut the hierarchical clustering tree and assign all the objects below each cut to a single cluster. We require that each valid cluster contains at least 3 members. Note that an event hotspot may have multiple temporal clusters. Performing temporal clustering before localization can result in only a few large clusters containing swipes to multiple event hotspots, since events can happen near-continuously over the whole area of interest.

Cluster refinement (Line 6). Since hierarchical clustering is performed for each hotspot, each swipe can possibly be assigned to multiple clusters. If this happens, we finally assign a swipe to the cluster with the largest number of members. This is to ensure that each swipe is assigned to only one cluster and thus only one hotspot. The refined clusters are denoted as $\{\mathcal{H}_i^r\}$.

Temporal analysis (Line 8). Clusters of time epochs are helpful in investigating the temporal properties at a hotspot. Figure 8 shows an example. We can simply use the range of a cluster to estimate the event occurrence interval (e.g., events happened between 12:07-12:17pm in one day from the top figure in Figure 8). We can also use histogram or kernel density estimation [6] to estimate when we can observe an event with high probability. As more swipes are aggregated, the temporal properties at a hotspot become clearer. By examining such properties over different snapshots, we can also investigate how the temporal patterns change at a hotspot.

Optimization-based location refinement (Lines 9–12). Swipes in the refined clusters are forwarded to the optimization engine. Optimization-based localization usually generates superior performance, but it may not be necessary or efficient when there are too few swipes and/or the user-indicated angles are similar. We thus first check whether the number of inputs is greater than 3 and the standard deviation of the associated angles is larger than 0.2. Otherwise, optimization will not be activated.

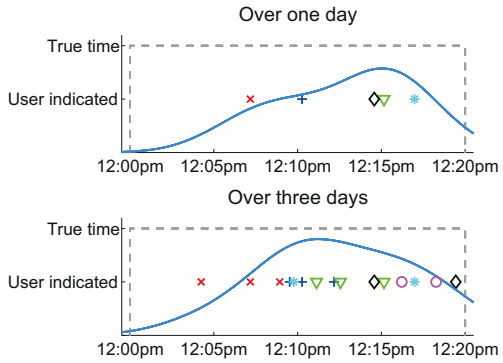

Figure 8. An example of using the range of the temporal cluster from user-swipes to estimate the event occurrence interval (e.g., events happened between 12:07-12:17pm in one day from the top figure). Different symbols denote data from different users. Blue curves show the kernel density estimation.

We formulate our optimization problem as follows:

$$\min_{x^t, y^t, x_i^t, y_i^t, \epsilon_i} \gamma \sum_i |\epsilon_i| + \sum_i w_i(|x_i^t - x_i| + |y_i^t - y_i|)$$
$$\text{s.t.} \quad (y^t - y_i^t)\cos\theta_i = (x^t - x_i^t)\sin\theta_i + \epsilon_i, \ \forall i.$$

In the above problem, (x_i^t, y_i^t) and (x^t, y^t) denote the true user and hotspot locations respectively and these values are to be optimized. (x_i, y_i), i.e., l_i, and θ_i are the user's GPS locations and swiped-angles respectively, which are known. The constraint is just an alternative expression of $\tan\theta_i = (y^t - y_i^t)/(x^t - x_i^t)$ (which should be valid at the true locations) plus an error term ϵ_i. The first sum in the objective function is to penalize the error ϵ_i in the constraint. The second sum can be interpreted as to move the users' GPS locations to their true locations as little as possible such that these swipes intersect at the same target location (satisfying the constraints). Moreover, those movements are weighted by their corresponding GPS accuracy, i.e., $w_i = 1/a_i$, which means that the more accurate a user's GPS location, the less the movement should be. In order to be robust, we use the ℓ_1 norm (sum of absolute values) in the objective function. The variable γ is positive and used to balance the emphasis on the dual objectives. We empirically find $\gamma = 1$ is a good setting. A refined location is denoted as $e^o = [x^t, y^t]$ and inserted into a location set $\mathcal{E}^o$. We term this localization algorithm as optimization-based location refinement (OLR). Note that OLR does not refine directly on the locations estimated by GEL, but uses the swipe-hotspot correspondence inferred from GEL and temporal clustering.

EVALUATION

This section presents the implementation and evaluation results of iSee.

Implementation

iSee is implemented in two parts, a smartphone client and a back-end server application. We built and tested the iSee client on the Android smartphone platform. A user swipes on the smartphone touchscreen to the directions that she sees events. Arrows are then drawn correspondingly. If the user is facing an event, she can just press the "Directly in Front" button. After submitting data, the user is asked to choose the event type from a predefined list or to enter short tags (optional). The user can also view her current location and the directions she points to on a digital map. A screenshot of the iSee user interface is shown in Figure 9.

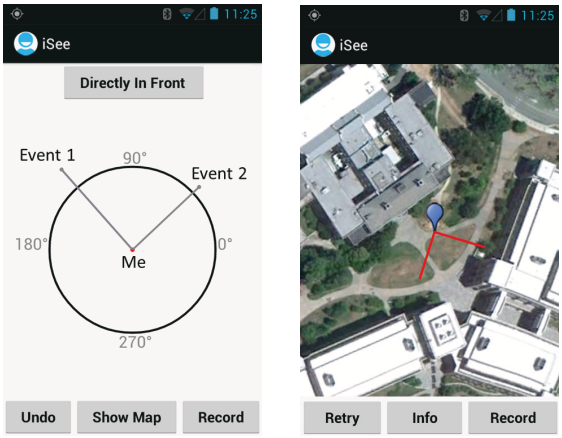

Figure 9. Screenshots of iSee user interface. (a) A user swipes on the phone's touchscreen to indicate the directions of events. (b) User location (blue pin) and the swiped directions (red lines) are shown on the map to present the user instant visual feedback.

Simple calculation of the swiped-angle and acceptance of a swipe (by examining the accelerations) are done on the phone. Then data including device ID (its usage will be discussed in the next section), time stamp, GPS location and accuracy, compass reading, and swiped-angle, are uploaded to our server (a Lenovo desktop running Windows 7) for processing via Matlab.

Experimental Setup

We designed a controlled experiment in order to conveniently collect the ground truth (location and time) to evaluate the system performance. We recruited 6 volunteers. After interviewing them, we found that they all go to a common restaurant area for lunch from respective buildings at around noon. The area covering their workplaces and restaurants is about $400 \times 550m^2$. We manually put up and removed 20 red flags at different locations during different time intervals to mimic events starting and ending along the volunteers' way to and from the restaurants in daily life. The distance between an event location to its nearest neighbor ranged from $40.4m$ to $81.6m$, with a mean of $53.4m$. The size of the red flag was chosen to reflect the equivalence between clearly observing a red flag and a human being. Where and when to put up and remove these flags was controlled and recorded by one of our authors and volunteers were unaware of such information. At any time epoch, at most 10 flags would appear simultaneously. The experiment was conducted over 6 days. The temporal patterns of 4 red flags were different for the first and last 3 days, while those of the other 16 flags were kept the same for 6 days. Each red flag appeared for 20 minutes, whereas it took about 15 minutes for a volunteer to walk to and from the restaurant. That is to say, if there was more than a 5-minute difference between the time that events started and the time that a volunteer started off, that volunteer could observe only a subset of events.

We gave each volunteer an Android smartphone. These phones included three Google Nexus S phones and three Samsung Galaxy S3 phones to reflect the heterogeneous hardware experienced in reality. Volunteers were asked to swipe to an event (red flag) when they observed it and were requested not to go to restaurants together in order to reflect the spatially and temporally distributed property of such crowdsourcing behavior. From the collected data, we found that, due to the temporal dynamics of red flags and the diversity among volunteers (in terms of their workplace locations, time for lunch, eyesight and attention), each volunteer contributed 0.75 to 1.2 swipes to each event location per day. This showed that volunteers missed events, mistakenly found more events or swiped more than once to an event. At the end of the experiment, we received 682 swipes altogether.

Methods Compared

Besides the two localization algorithms 1) GEL (grid-based event localization) and 2) OLR (optimization-based location refinement) presented in this paper, we also introduce a classical localization algorithm for comparison. We term this algorithm as 3) line segment intersection and clustering (LIC), which is just modified triangulation. It is to first find the intersections of all the line segments, then cluster those intersections, and finally use the cluster centers as the inferred event locations. We use a density-based clustering algorithm OPTICS [10] for LIC, since it is robust to outliers and can detect regions with low density. The minimum number of points required to form a cluster is set to 3. The maximum visible distance is set to $45m$ and the grid size $10m$.

Performance Metric

We denote the set of estimated and true event locations as $\mathcal{E} = \{e_i\} \in \mathbb{R}^2$ and $\mathcal{P} = \{p_j\} \in \mathbb{R}^2$, respectively. In order to evaluate the localization performance of iSee, we define the following metrics.

Reporting rate: We define the reporting rate r_r as the ratio between the number of *reported* (estimated) and true event locations. A good event detection algorithm should result in an r_r close to 1. $r_r < 1$ and $r_r > 1$ imply missed events and false alarms respectively.

Detection rate: We consider an event location p_i detected if there is an estimated location e_j within $thre_d = 25m$. Detection rate r_d is defined as the ratio between the number of *detected* and true event locations. r_d is never larger than 1. A good detection algorithm should result in an r_d close to 1. An algorithm that reports every cell as an event location can result in an $r_d = 1$ but an $r_r \gg 1$, implying lots of false alarms. Therefore, reporting rate and detection rate should be examined together to ensure that an algorithm functions well.

Localization error: For each estimated event location e_i, we define its localization error as the Euclidean distance to its nearest true event location, given by

$$d(e_i, \mathcal{P}) = \min_j \|e_i - p_j\|_2.$$

Note that, localization error is only calculated for reported locations. Missed event locations will not contribute to localization errors.

We use the association accuracy, which is defined as the percentage of hotspots with correctly associated swipes, to evaluate the effectiveness of temporal clustering. Ground truth associations are obtained by examining both location and time. For assessing the performance of estimating the occurrence interval of events, we use the Jaccard similarity between the estimated interval A and the true interval B as the performance measure which is defined as

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Performance Results

We attempt to empirically answer the following questions:

1. What are the advantages of GEL over LIC, since they have the same functionality in detecting event hotspots (while OLR is to optimize event locations after detection)? (Figures 10 and 11)
2. What are the reporting rates, detection rates and localization errors of LIC, GEL and OLR? (Figures 12)
3. How can temporal clustering improve swipe-hotspot correspondence? How can the estimated event occurrence interval approximate the true interval? (Figure 13)

Comparing GEL with LIC

We first perform a qualitative comparison between GEL and LIC to validate the necessity of GEL. Figure 10 shows examples when there are, on average, 5.5 swipes per event location. From left to right, the subfigures show a) the filtered cell confidence map with event locations estimated by GEL, b) corresponding estimated event geolocations by GEL, and c) estimated event geolocations by LIC. As can be seen, for a large portion of hotspots, there are either too few intersections (for relatively isolated locations) such that LIC is unaware of event locations (7 out of 20) or too many intersections (for locations with close neighbors) such that LIC mixes the swipes to different locations. In contrast, GEL performs much better and detects 17 out of 20 event locations.

Similar to Figure 10, Figure 11 shows examples when there are many more swipes (on average 34.1) per event location. For such a scenario, there are lots of intersections and those from nearby locations are mixed together to form several large connected areas. LIC thus cannot differentiate even distant locations and generate only a few large clusters, resulting in even worse performance than that in Figure 10. If we confine the size of a cluster, LIC will generate a reporting rate over 2, implying lots of false alarms. In contrast, GEL still performs well. The only difference is that the values on the confidence map become much higher.

These results show that LIC suffers from low detection rate and is very sensitive to the number of swipes. In contrast, since GEL takes into account the GPS and angular errors, performs localization on cells rather than intersections, and also has a mechanism to convey confidence over cells, it shows the ability to quickly detect events when there are only limited swipes available and it is also relatively stable when many more swipes arrive.

Another advantage of GEL over LIC is in the convenience of retrieving swipes for each detected hotspot (for temporal

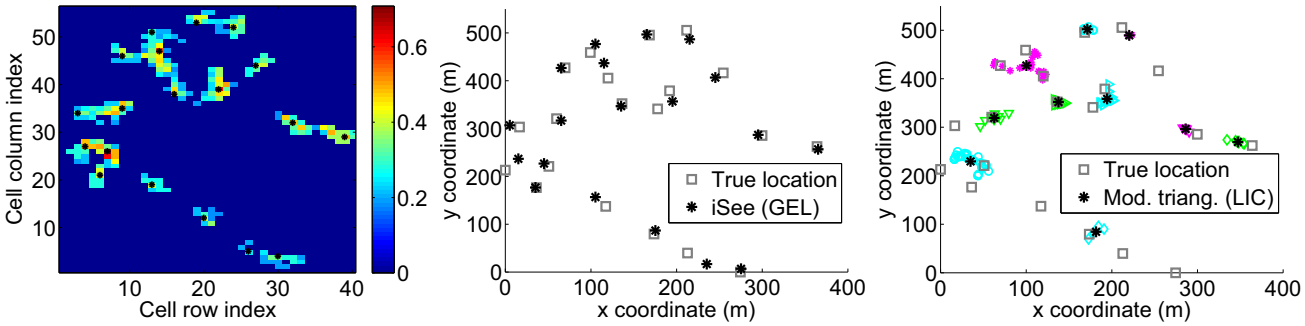

Figure 10. On average 5.5 swipes per event location are collected. Left to right: (a) the filtered cell confidence map with event locations estimated by GEL, (b) corresponding estimated event geolocations by GEL, and (c) estimated event geolocations by LIC, i.e., modified triangulation (colorful symbols denote the clusters of intersections).

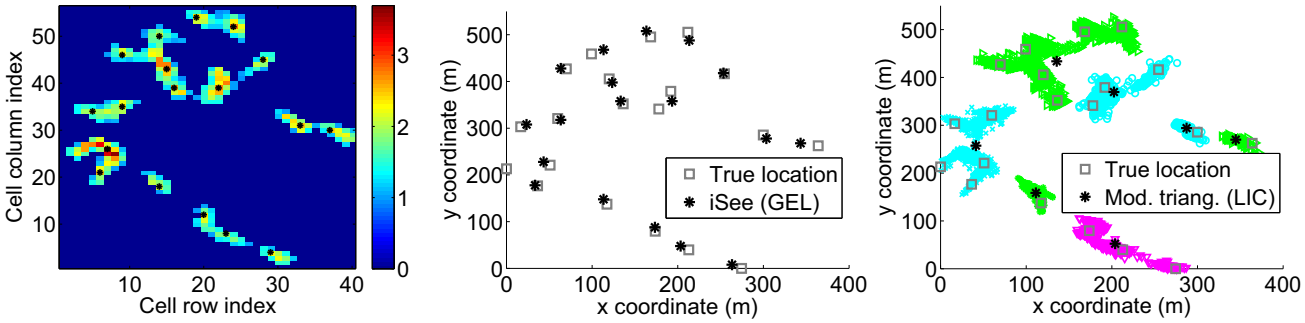

Figure 11. On average 34.1 swipes per event location are collected. Left to right: (a) the filtered cell confidence map with event locations estimated by GEL, (b) corresponding estimated event geolocations by GEL, and (c) estimated event geolocations by LIC, i.e., modified triangulation (colorful symbols denote the clusters of intersections).

analysis and location refinement). In order for LIC to retrieve swipes for each hotspot, it must record two kinds of information – which two swipes result in an intersection and which intersections are assigned to a cluster. Later, for each cluster, LIC needs to backtrack twice to find the corresponding swipes which is burdensome. Recall that when we need to retrieve the swipe-hotspot correspondence from GEL, we only need to check one column of the SCI matrix.

Finally, in terms of scalability, the number of intersections is on the order of $O(n_s^2)$ for LIC (n_s is the number of swipes). While for GEL, the number of cells is fixed and independent of n_s. In other words, GEL is more scalable.

Localization Performance

Figure 12 shows the average localization results when the swipes are cumulated over days. We find the standard deviations of the algorithms are on similar orders and thus we do not plot them to make the figures uncluttered. Each point on the figure is calculated by averaging the results from $\binom{6}{k}$ combinations on each day's data, where k is the number of cumulative days. The number of average swipes per location is 5.7, 11.2, 16.8, 22.7, 28.3 and 34.1 for cumulative 1 to 6 days, respectively. In reality, it is possible that an event receives lots of swipes during a relatively short interval because of lots of observers.

We first examine the performance of the algorithms when there are only limited swipes, i.e., during one day. As can be seen, LIC has much lower reporting rate and detection rate (below 0.6) than GEL and OLR (above 0.8), mainly due to the sparse intersection problem with only limited swipes shown in Figure 10. OLR performs slightly worse than GEL in terms of detection rate, possibly due to the large errors contained in GPS locations, reported accuracy, user-indicated angles and heterogeneous hardware. When there are only limited swipes, it is difficult for OLR to balance among them. GEL and OLR show improved localization accuracy compared with LIC, but the improvement is marginal. All the algorithms achieve a localization error around 13-14m. When there are only limited swipes, the locations reported by LIC are with relatively compact intersections and thus of relatively high confidence. With much fewer but confident locations detected, LIC can achieve reasonable localization accuracy.

We then examine the performance of the algorithms when the number of swipes increases, i.e., accumulated over 2-6 days. We can observe that GEL and OLR perform well and are relatively stable in terms of reporting rate and detection rate (around 0.9), while the performance of LIC degrades as the number of swipes increases. The worst detection rate of LIC drops to about only 0.25. The localization accuracy of LIC also quickly degrades to over 25m when accumulating 6 days' data. As more swipes arrive, intersections from neighboring hotspots mix together and form several large connected areas, LIC thus cannot accurately infer many hotspots (Figure 11). On the other hand, GEL and OLR can benefit from the increased number of swipes and their localization accuracy improves to around 10m accumulating 6 days' data. OLR results in even smaller average location errors than GEL after 4 cumulative days, showing that a reasonable number of swipes is needed for OLR to be effective.

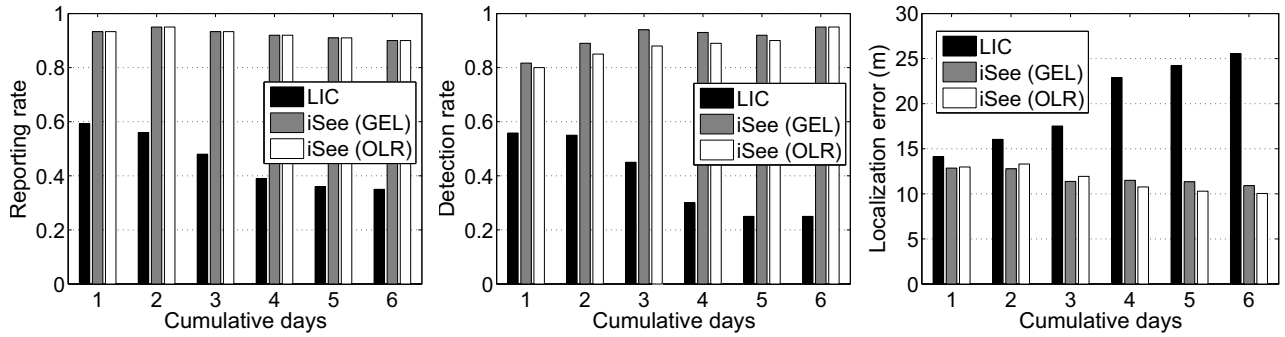

Figure 12. Left to right: Average (a) reporting rate (ratio between the number of reported and true event locations), (b) detection rate (ratio between the number of detected and true event locations), (c) localization error versus cumulative days. The average number of swipes per location is 5.7**,** 11.2**,** 16.8**,** 22.7**,** 28.3 **and** 34.1 **for cumulative** 1 **to** 6 **days, respectively.**

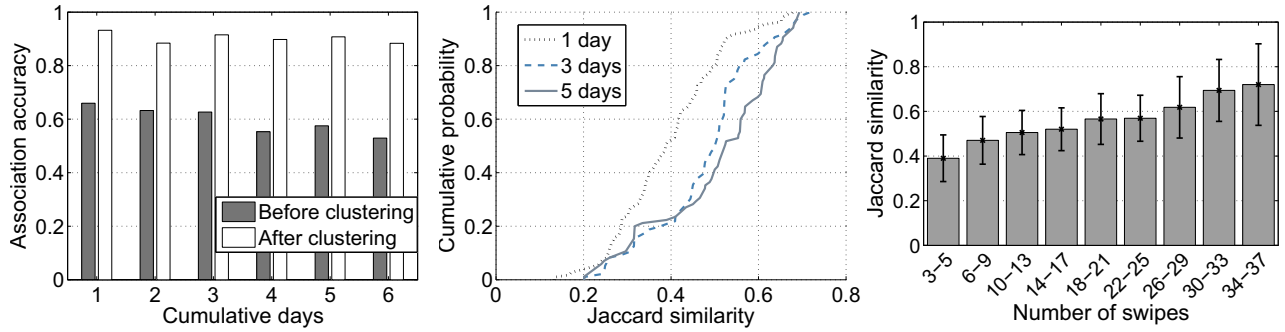

Figure 13. Left to right: (a) Mean and std of association accuracy before and after clustering. (b) CDF of Jaccard similarity between estimated and true event occurrence interval. (c) Average Jaccard similarity versus the number of swipes per event.

Temporal Clustering and Approximation

Figure 13(a) plots the association accuracy before and after temporal clustering for cumulative days. It is clear that temporal clustering can effectively improve the association accuracy, i.e., refine the swipe-hotspot correspondence, from around 60% to 90%. We find that association error occurs if a large portion of the swipes to a hotspot have been erroneously associated with nearby hotspots before clustering due to large GPS and angular errors. Figure 13(b) shows the cumulative distribution of the Jaccard similarity between the estimated and true event occurrence interval. It can be seen that as more swipes arrive, on average the Jaccard similarity increases, but the extent of improvement gradually levels off. Cumulating 1, 3 and 5 days' data can capture on average 40%, 50% and 54% of the true occurrence interval. More swipes can introduce errors as well since erroneous temporal clusters can increase the denominator of the Jaccard similarity and decrease its value. Figure 13(c) plots the mean and std of Jaccard similarity versus the number of swipes per hotspot (obtained from the inferred swipe-hotspot correspondence). It shows that on average 3-5 swipes can capture around 40% of the occurrence interval of a 20-minute event and 34-37 swipes are needed to capture over 70% of the occurrence interval, due to the temporal randomness of user-contributed swipes. Nevertheless, clusters of the time associated with swipes provide us insights about the temporal properties of events.

LIMITATION AND DISCUSSION

As a crowdsourced system, iSee may receive numerous swipes and detect numerous hotspots. Is there a mechanism that iSee can identify a smaller set of public-minded users and hotspots with public concerns in order to present the most useful information to querying users? To achieve this, we need to construct a user-hotspot indicator matrix U recording which user pointed out which hotspot. This can be done by first forming a swipe-hotspot indicator matrix from the inferred correspondence and then performing an OR operation on the swipes corresponding to the same user (each swipe was tagged with a user ID when submitted). A possible choice for inferring high-quality users and hotspots from U is to use the hypertext induced search (HITS) model [11] which has been successfully applied in web page ranking and interesting location mining [22]. Note that such rankings will change depending on the region and time of interest.

Is any further improvement of localization accuracy possible? Based on the type of event and a geo-information database, we may be able to rectify an estimated hotspot to its closest semantically meaningful location. For example, smokers may be more likely to smoke around the entrance of a building rather than in the middle of a road. Moreover, since a human being cannot see through a building, the maximum visible distance can be adjusted for each swipe such that it touches the building contour at most.

RELATED WORK

iSee draws on prior work in multiple areas, chiefly smartphone localization, crowdsoucing, and event monitoring.

Smartphone localization. Sensors on smartphones facilitate multimodal localization, e.g., ambient fingerprinting-based [8] and unsupervised indoor localization [19]. While these

schemes are to locate the mobile device itself, our task is to locate a distant spot. The most similar work is the object positioning system (OPS) [15], which combines phone camera with GPS and inertial sensors to locate a distant object. We do not utilize the camera in order to reduce the burden on the participants and also make the application less privacy-sensitive. In addition, we make use of the touchscreen swipe, which has not been explored for localization. Our proposed BDA and GEL are lightweight and can incrementally process newly-arrived data; whereas OPS uses complex computer vision techniques such as structure-from-motion. Moreover, iSee is a crowdsourced system while OPS is standalone.

Our scheme uses angular information and thus relates to triangulation and angle-of-arrival (AOA) based localization [14, 12, 21]. However, no wireless signal is transmitted in our scheme and thus swipe-hotspot correspondence is unknown. This makes classical AOA localization not directly applicable. Clustering on the intersections of swipes is a possible solution, but it is not scalable and shows poor performance in detecting and locating events. On the other hand, our proposed GEL successfully overcomes these problems and performs much better.

Crowdsourcing. Luis Von Ahn's pioneering work on re-Captcha [18] uses humans to solve difficult OCR tasks, thereby enabling digitization of old books and newspapers. His another work on the ESP game [17] uses humans for finding good labels for images. Yan et al. [20] exploit crowds for accurate real-time image search on mobile phones. Other crowdsourcing examples include the auction-based crowdsourcing (e.g. Taskcn [3]) and simultaneous crowdsourcing contests (e.g. TopCoder [4]). Our work differs in that we use crowdsourcing to discover and locate outdoor event hotspots which has not been explored previously.

Event monitoring. Wireless sensor networks have been prototyped for many military and civilian applications, such as area and environment monitoring [5]. Sensors are usually densely deployed in order to completely cover the pre-defined monitoring area. iSee provides a complementary approach that leverages crowdsourcing to report and locate outdoor events in a dynamic, scalable and distributed manner. In the domain of mapping outdoor smoking hotspots, researchers have proposed methods such as dispatching trained observers to specific public areas and then recording either the number of smokers [16] or cigarette butts observed [13]. These methods, while accurate within a specific area, do not scale and are expensive to acquire continuously, resulting in sparse characterization. In contrast, iSee proposes a low-cost and near-continuous solution.

CONCLUSION

Localizing objects or events in the environment can be useful for various applications, such as locating smoking hotspots in public places and detecting city graffiti. This paper presents a crowdsourcing-based approach where users swipe on their smartphones' touchscreens in the direction of the event of interest. By aggregating data from multiple users, we show the viability of detecting and localizing the event with consistent accuracy and robustness. Our approach is lightweight to the extent that users are likely to use it frequently, even when they are not sure that the event is worth reporting. However, it is these reports that are often invaluable for understanding the behavior of large scale systems or for early detection and prevention of large scale incidents. To this end, we believe that iSee is an important step towards low-cost and near-continuous monitoring of outdoor events, especially in densely populated public areas.

ACKNOWLEDGMENTS

We thank the program committee and anonymous reviewers for their insightful suggestions for improving the technical content and presentation of the paper. We also thank the volunteers for their help with the experiment.

REFERENCES

1. Flood fill algorithm. **http://en.wikipedia.org/wiki/Flood_fill.**
2. Ray casting algorithm. **http://en.wikipedia.org/wiki/Ray_casting_algorithm.**
3. Taskcn: A platform for outsourcing tasks. **http://www.taskcn.com/.**
4. Topcoder. **http://www.topcoder.com/.**
5. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications magazine, IEEE*, 40(8):102–114, 2002.
6. E. Alpaydin. *Introduction to machine learning (second edition)*. The MIT Press, 2010.
7. J. Austin. *Taking the train: How graffiti art became an urban crisis in New York City*. Columbia University Press, 2001.
8. M. Azizyan et al. Surroundsense: mobile phone localization via ambience fingerprinting. In *MobiCom*, pages 261–272. ACM, 2009.
9. D. P. Dobkin and D. G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoretical Computer Science*, 27(3):241–253, 1983.
10. J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.
11. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
12. J. Krumm, G. Cermak, and E. Horvitz. Rightspot: A novel sense of location for a smart personal object. In *UbiComp*, pages 36–43. Springer, 2003.
13. J. G. Lee et al. Cigarette butts near building entrances: what is the impact of smoke-free college campus policies? *Tobacco Control*, 2011.
14. H. Liu et al. Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080, 2007.
15. J. G. Manweiler et al. Satellites in our pockets: an object positioning system using smartphones. In *MobiSys*, pages 211–224. ACM, 2012.
16. G. Thomson et al. Informing outdoor smokefree policy: Methods for measuring the proportion of people smoking in outdoor public areas. *Health & place*, 2012.
17. L. Von Ahn and L. Dabbish. Labeling images with a computer game. In *CHI*, pages 319–326. ACM, 2004.
18. L. Von Ahn et al. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.
19. H. Wang et al. No need to war-drive: Unsupervised indoor localization. In *MobiSys*, pages 197–210. ACM, 2012.
20. T. Yan et al. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *MobiSys*, pages 77–90. ACM, 2010.
21. Z. Zhang et al. I am the antenna: Accurate outdoor ap location using smartphones. In *MobiCom*, pages 109–120. ACM, 2011.
22. Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *WWW*, pages 791–800. ACM, 2009.