Intelligent Energy-Efficient Triggering of Geolocation Fix Acquisitions Based on Transitions Between Activity Recognition States

Thomas Phan

Samsung Research America - Silicon Valley
San Jose, CA USA
thomas.phan@samsung.com

Abstract. Location-based applications (LBAs) running on smartphones offer features that leverage the user's geolocation to provide enhanced services. While there exist LBAs that require continuous geolocation tracking, we instead focus on LBAs such as location-based reminders or location-based advertisements that need a geolocation fix only at rare points during the day. Automatically and intelligently triggering geolocation acquisition just as it is needed for these types of applications produces the tangible benefit of increased battery life. To that end, we implemented a scheme to intelligently trigger geolocation fixes only on transitions between specific modes of transportation (such as driving, walking, and running), where these modes are detected on the smartphone using a low-power, high-resolution activity recognition system. Our experiments show that this approach consumes little power (approximately 225 mW for the activity recognition system) and correctly triggers geolocation acquisition at transitional moments with a median delay of 9 seconds from ground-truth observations. Most significantly, our system performs 41x fewer acquisitions than a competitive accelerometer-assisted binary classification scheme and 243x fewer than continuous tracking over our collected data set.

1 Introduction

Location-based applications (LBAs) running on modern commodity smartphones offer features that leverage the user's current or past physical location to provide some enhanced service. For example, commercial smartphone LBAs currently exist that automatically acquire geolocation fixes (represented as latitude, longitude coordinates) in order to keep the user's health statistics [10], derive vehicular traffic conditions [38], and warn drivers of nearby hazards [36].

While such capabilities are compelling, they are limited in practice due to the high rate of energy consumption from the geolocation fix acquisition process. A fix can be attained through GPS, which has high precision outdoors but cannot function indoors, or trilateration using WiFi or cellular towers, which can work indoors but with typically lower precision depending on the density of surrounding signal sources [5, 35, 13].

Although our work is relevant to either means of geolocation positioning, we note that GPS achieves the most accuracy but at the highest battery cost. Studies have shown that continuous use of GPS can deplete a smartphone's battery within one day [20, 29, 21]. As a result, researchers in the area of energy-efficient geopositioning have looked at better ways of performing fix acquisitions, for example through duty-cycled scheduling with accelerometer-assisted motion detection [1, 39, 22]. However, such approaches are based on the fundamental assumption that LBAs need *continuous geolocation tracking*.

In this paper we focus on LBAs that require automated geolocation fixes only at specific points of the day to facilitate user engagement, thereby allowing an underlying system to trigger fix acquisition *very rarely*. Examples are:

- Location-based reminders: Smartphone applications like Apple's voice-driven Siri [2] allow the user to set reminders whose UI alert is activated only at a target location, such as an alert for "Remind me to finish writing the report when I get home."
- Location-based advertising: Advertisement offers, such as coupons, can be proactively pushed to users who are in the vicinity of a business [32].
- Location-based tourist recommendations: Tourists who drive and walk through a city can be shown recommended places to visit [6].

Because these types of LBAs require only rare geolocation fixes, namely at the location where the user becomes engaged with the LBA, continuous tracking is wasteful. Instead, battery power could be conserved if the geolocation software could trigger a fix acquisition in a just-in-time manner. Using a location-based reminder as an example, an ideal oracle system would acquire a fix right as the user reaches the target location, and the resulting proximity would then activate the reminder.

In this paper we approximate such an oracle by defining transitional periods between transportation modes to be opportune points to acquire a geolocation fix. For example, if the user has been driving and then transitions to walking, then most likely the user has parked his car and gotten out to walk, and that moment would be an opportunity to acquire a fix. Other identifiable transitions would also be good opportunities.

We implemented the above scheme to intelligently trigger geolocation fixes only on transitions between specific modes of transportation, where the modes are detected on the smartphone using a low-power, high-resolution activity recognition system that acquires real-time 3-D accelerometer data, performs signal processing to extract features, and applies a trained machine learning model to determine the most likely means of transportation (such as driving, walking, running, bicycling, or idling). Our experiments show that this approach consumes little power (approximately 225 mW for the activity recognition system) and correctly triggers geolocation acquisition at the correct transitional moments with a median delay of 9 seconds from ground-truth observations. Most significantly, it performs 41x fewer acquisitions than a competitive accelerometer-assisted binary classification scheme and 243x fewer than continuous tracking over our collected data set.

The rest of this paper is organized in the following manner. In Section 2 we discuss related work in the area of energy-efficient geolocation positioning. In Section 3 we provide an overview of our system, and in Sections 4 and 5 we describe our high-resolution activity recognition component and our transition-triggered geolocation fix acquisition systems, respectively. We show experimental results in Section 6 and then conclude in Section 7.

2 Related work

Our work differs from previous research in the area of energy-efficient geolocation acquisition (e.g. [8, 17, 39]) in that we do not make the assumption that LBAs require continuous tracking. Instead, we focus on LBAs that need a geolocation fix only at specific and rare points in the day.

Previous researchers in this area have observed that (continuous) geolocation acquisitions can be disabled during phases of the day when the user is idle, such as when the user is asleep. One approach to implementing this scheme is to use the accelerometer at a coarse-grained resolution to detect only two states: idle versus non-idle [1, 39, 34, 16, 27]. These works do not exploit the fact that non-idle states can be broken down further to identify transitions between them. Other approaches detect non-idle states by using GPS itself as a sensor [17, 37, 11].

The work in [22] uses high-resolution activity recognition with the accelerometer to identify different modes of activity, which can then be used to inform GPS duty-cycle scheduling. Again, this work differs from ours is that they assume a need for continuous tracking; specifically, they look to expend a battery energy budget. The work in [26] uses the accelerometer to distinguish between standing and sitting to assist localization down to stores in which people tend to stand or sit (such as grocery stores and coffee shops).

In our work we define a finite state machine (FSM) that uses specific transitions between known user state vertices to trigger geolocation acquisition. The work in [37] also uses a FSM but for the opposite problem: they use sensors (including GPS) to determine the unknown user state.

Previous efforts have also looked at exploiting historical data. For example, the work in [28] adapts GPS duty-cycling based on whether location uncertainty has exceeded a threshold for a given location and time historical combination. The work in [25] discovers association rules that allow low-power sensors to replace high-power sensors. Our work differs in that our system does not require accumulated history.

Our system uses an activity recognition component to determine the user's mode of transportation so that transitions between them can be detected. Activity recognition is the process of inferring the user's physical behavior from sensor data and stems from work in the body-sensor community. Early research (e.g. [4, 31, 19, 7, 15]) relied on custom sensor hardware mounted on the human body. Modern smartphones with built-in sensors are now the platform of choice for mobile activity recognition research. Recent work (e.g. [23, 22, 18, 33]) has

demonstrated that on-device recognition is feasible and benefits from the fact that users carry smartphones with them throughout most of the day.

3 Design overview

While there exist many location-based application (LBA) domains that require continuous geolocation tracking, such as health monitoring and vehicular traffic aggregation, we instead focus on LBAs for which continuous tracking is superfluous and unnecessarily depletes the battery; indeed, such LBAs may need a geolocation fix only at rare points during the day. Automatically and intelligently triggering geolocation acquisition just as it is needed for these types of applications produces the tangible benefit of increased battery life for users.

As a concrete example, consider a user who wants to set a location-based reminder. While the user is at work, he creates an alert that should activate when he arrives at his targeted home. We make the assumption that his home's address is in his smartphone's addressbook and has already been resolved to a geocoordinate. Such a reminder can be implemented in a number of ways:

- **Continuous tracking.** The system can continuously track the user as he drives from work to home. To save power, it can use coarse-grained but lower-power cellular tower geopositioning to determine the user's location with an error of about 300m to 400m [8]. When the user is within that error range to his home, the system may switch to GPS to get accuracy to within 10m. Once the user is within a proximity tolerance of the target, the reminder system activates the alert.
- **Just-in-time fix.** The system can, somehow, trigger *one* geolocation fix just as the user arrives at his home, and because he is within a proximity tolerance, the reminder system activates the alert.
- **Sensor fingerprinting to derive semantic location.** The system can use sensor fingerprints (i.e. features extracted from on-device sensors) to infer that the user has arrived at his home, which is a semantic location rather than a geocoordinate. Previous work has looked at WiFi, Bluetooth, audio, light, and other sensors as inputs to a classifier [3, 9]. Since this approach requires training fingerprints to be taken, its generalizability is still an open question.

Our scheme follows the just-in-time approach. While triggering a geolocation fix acquisition before knowing that the user has arrived at a location may seem oracular, we posit that such triggering can be implemented in a general manner by observing transitions between user activity states, where each state represents a mode of transportation. Our key assumption is that particular transitions are indicative of moments in time when the user is amenable to engagement with an LBA, and so a fix should be triggered on these transitions.

We implemented our scheme on Android smartphones, and Figure 1 shows a block diagram of the system. At the bottom-most layer, a low-power accelerometer-only activity recognition component generates a continuous stream of inferred

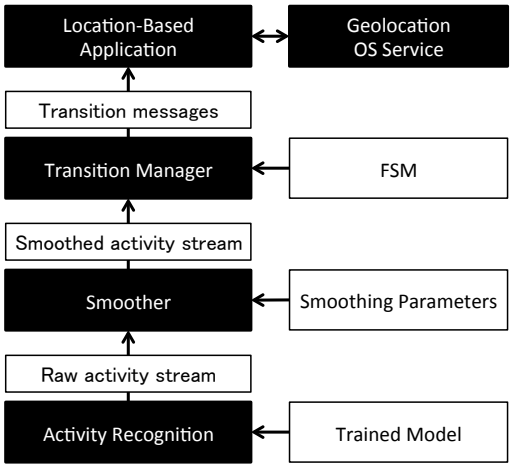

Fig. 1. High-level architecture of our system. The black boxes are software components while white boxes are data or models.

human modes of transportation. This component uses a supervised machine learning classifier that loads a trained C4.5 decision tree model into memory, reads real-time accelerometer signals, and produces a classification label. While the activity recognition is accurate on single-activity behavior in lab environments, it produces endemic misclassification errors when faced with real-world, mixed-activity behavior. To address this problem, we apply a smoothing window. The activity recognition is discussed in Section 4.

The transition manager listens to the stream of smoothed activity labels and manages a finite state machine (FSM) to detect relevant transitions. The vertices and edges in the FSM have labels that correspond to the possible activity recognition labels. The transition manager is discussed in Section 5.

The output of the transition manager are transition messages sent to any listening application, and in our Android-specific implementation, these messages are realized as Android Intents. Upon receiving such a message, the listening LBA should then acquire a geolocation fix and check to see if the the resulting geolocation coordinate satisfies its LBA-specific logic. For example, the coordinate could trigger a location-based reminder or a location-based advertisement.

An alternative implementation could have the transition manager acquire the geolocation and then include the resulting coordinates in its Intent message to all listeners. However, this approach would bypass Android's security and privacy model where each application must declare its use of geolocation services (in its manifest file), which is then surfaced to the user in the application's description before it can be downloaded from the Google Play store.

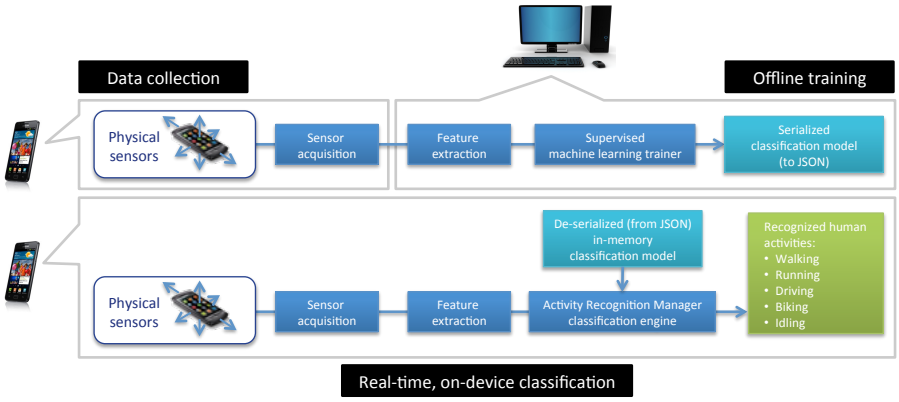

Fig. 2. Activity recognition end-to-end data collection, training, and classification.

4 High-resolution activity recognition

Smartphone-based physical activity recognition [4, 22, 18] infers human activities by analyzing on-board smartphone sensor signals, and in our framework we use supervised machine learning to make these inferences. The end-to-end system comprises two distinct phases, as shown in Figure 2. First, in an offline training phase, test participants perform various physical activities while wearing smartphones that run data-collection software. After extracting relevant sensor features, the model is then trained that maps such features back to the physical activities. Second, in an online classification phase that runs on a smartphone, the model is deployed onto smartphones and loaded by our activity recognition manager component to perform on-board, real-time classification of sensor data that can drive higher-level applications.

The activity recognition manager's sole input is the accelerometer, and the periodic output is a callback containing an activity represented as a string, such as *Running*; downstream clients then implement callback listener functions to receive them. We defined the following activities: *Walking*, *Running*, *Driving*, *Bicycling*, and *Idling* (which comprises standing and sitting).

4.1 Offline model training

To train the machine learning model, we had users perform single-activity, scripted behavior while an Android application (running on a Samsung Nexus S smartphone) recorded tri-axis accelerometer readings at 32 Hz and labelled them with the user's activity.

In each resulting file, we trimmed off the first 10 and the last 10 seconds' worth of samples. There were 17 total users, all from our research lab and in the age range of late-20s to 50s. The data totalled 397 minutes, or over 6 hours, across the activities mentioned earlier: walking (on a flat surface), walking up

stairs, walking down stairs, running, bicycling, driving, standing, and sitting. We asked the users to hold the phones in any way they wanted, but most ended up placing the phone in their front right pants pocket. For sitting, users placed the phone either in that pocket or on their desk. For driving, users placed the phone either in that pocket, in a bag, or on a windshield-attached cradle. For bicycling, users placed the phone in a belt-mounted hip holster.

Because the phone can be oriented in different directions on a person's body, we normalize the accelerometer readings into three orientation-independent time series: Cartesian magnitude of the acceleration vector; magnitude of projection onto the true horizontal plane; and projection onto the true vertical axis. To derive the latter two time series, our projection algorithm computes the dynamic component of device acceleration in the true global vertical plane and the true horizontal plane perpendicular to the vertical plane. Our projection algorithm is based on past work [24, 22].

The resulting three time series (Cartesian magnitude of the acceleration vectors, magnitude of projection onto the true horizontal plane, and projection onto the true vertical axis) are then captured with a 128-sample sliding window with 50% overlap.

For each of the three time series, we computed features using both the time and frequency domains. To obtain frequency domain information, we ran an FFT and follow standard steps to compute a correctly-scaled magnitude frequency response. We are constantly evaluating new features, but at the time of this writing, the features we use are:

- time-domain mean
- time-domain standard deviation
- time-domain real-valued discrete power
- time-domain entropy
- frequency-domain energy
- the frequency f_{max} with the highest magnitude
- the magnitude of f_{max}
- the frequency $f_{weighted-mean}$ calculated as the weighted mean of the top-5 highest-magnitude frequencies weighted by magnitude
- the weighted variance of the top-5 highest-magnitude frequencies weighted by magnitude.

These 9 features are extracted from each of the 3 time series, resulting in 27 total features that are concatenated together into one feature vector.

We used the Weka [14] off-the-shelf machine learning program to build our model offline on a desktop computer. The software provides a variety of different machine algorithms, including Naive Bayes, SVM, and kNN, and after testing them, we found that the C4.5 multi-class decision tree [30] produced the best results. Additional benefits of using a decision tree are that its memory footprint is low (less than 10 kB resident in memory) and its time to classify a feature vector is small (on the order of milliseconds), characteristics that are important when executing on a phone. Classifying a real-valued feature vector is a matter

of walking down a tree structure and making a single floating-point comparison at each level, and in our work the decision tree never grew taller than 10 levels. The resulting trained model is serialized to a JSON file that contains data structures, statistics, and other metadata that are deserialized later on the phone by the activity recognition manager. The serialization format is JSON rather than binary, resulting in a file that is portable and can be can be deployed and updated separately from the other activity recognition components.

4.2 On-device classification system architecture

The on-device classification system comprises several software stages and a trained model. Referring back to Figure 2, the sensor acquisition layer is the interface between the classification system and the sensor API provided by the smartphone's operating system. This layer is responsible for (i) reading the physical hardware sensors, such as the accelerometer (and other sensors like the microphone and compass in future work), and (ii) preparing the data into a structure for later processing.

The activity recognition manager runs as a system-wide singleton service that (i) loads the JSON-encoded classication model built during the training phase, (ii) reads accelerometer input from the sensor acquisition layer to extract features, and (iii) executes the real-time activity classification. Clients register themselves as listeners to the manager, which then invokes client callback functions whenever a relevant activity has been recognized by the classifier.

4.3 The need for smoothing

We augmented our classifier with an smoothing algorithm that improves classification for real-world usage. As we show later in Section 6, the system achieves a 10-fold cross-validated accuracy of 98.4%. While this result is appealing and is in alignment with the methodology and results reported in other activity recognition work (e.g. [22, 18]), it is misleading because it applies only to the single-activity, scripted activity behavior recorded during the training phase. Even with the use of cross-validation, two problems arise: (1) the built model may be overfit for the data; and (2) mixed-activity, naturalistic human behavior contains many motions and transitions that were not captured during training. As a result, the classifier can return incorrect labels in real-world use.

To avoid any such choppiness, we implemented a simple smoothing algorithm with a majority vote within a trailing window (of parameter size N). As we show later, this choice of N affects the trade-off between the number of geolocation fix requests and the delay (in terms of time and distance) between the fix request and ground-truth observed transitions.

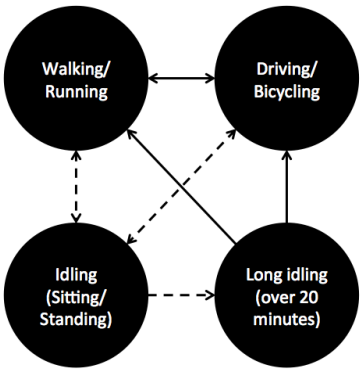

Fig. 3. Geolocation fixes are triggered only on major activity transitions controlled by a finite state machine. A fix is acquired on transitions indicated by the solid edges.

5 Activity recognition transition-triggered geolocation fix

Given the availability of the low-power activity recognition system described in the previous section, we can use its output states (comprising recognized modes of transportation) as input into our just-in-time geolocation fix component.

As mentioned in Section 3, we posit that certain transitions between modes of transportation are indicative of points in time which are more conducive to user engagement with LBAs. For example, when the user makes the transition $Driving \rightarrow Walking$, it may indicate that he has gotten out of his car after having arrived at a desired destination. This spot would then be a good opportunity for an LBA to acquire a geolocation fix to see if it satisfies its LBA-specific logic.

Note that in the case of a location-based reminder with the user's home as the target destination, this approach will still trigger a geolocation acquisition if the user stops somewhere else first, such as at a gas station. Nonetheless, as we show show in Section 6, these additional acquisitions are still less frequent than continuous tracking.

Referring back to Figure 1, a transition manager component observes the smoothed activity recognition stream and keeps track of the current and immediately previous user state. By observing these two states, the manager can identify transitions between certain activities, where these transitions are represented in our system as edges within a finite state machine (FSM).

Our reference FSM is shown in Figure 3. Note that most of the vertices in the system correspond to the available known activity labels that can be produced by the activity recognition component. The only special vertex is the one for long idling, which we define to be idling (that is, sitting or standing) for more than 20 minutes.

When a transition between states is detected by the transition manager, the FSM is checked to determine if the transition qualifies for a geolocation fix acquisition. The solid edges in the FSM are those transitions that we selected

to trigger an acquisition. They were heuristically selected to be those where an LBA would most likely take advantage of a new geolocation.

Different FSMs defining different transitions can be used in place of this one. In the future we will also look at ways to discover appropriate transitions automatically rather than heuristically; building a Hidden Markov Model is one such approach.

Note here the advantage of using a high-resolution accelerometer-only activation recognition system such as the one we implemented. Unlike previous work [1, 39, 34, 27] where accelerometers are used to differentiate between only binary states (idling vs. non-idling), our work leverages the distinction between multiple classes of non-idling, such as between driving and running. Note further that being able to be detect these distinctions using only the accelerometer provides a battery consumption advantage over other approaches that use GPS as a sensor [17, 37, 33, 11].

However, the use of transitions to trigger geolocation acquisition suggests a clear limitation. If the user does not change his mode of transportation, then our system will not be able to detect any transitions at all. For example, it may be the case that the user continuously walks from his home to work, never stopping at any point. To address this problem, we additionally implemented an optional background service that performs fixed duty-cycling with a parameterized period to serve as a backup.

6 Evaluation

We implemented and ran our system on two commodity Android smartphones, a Samsung Galaxy S II (released in the U.S. in May 2011) and a Samsung Nexus S (released in the U.S. in December 2010). The key result from our experiments is that while our activity recognition transition-triggered design incurs more delay to identify ground-truth events versus competing schemes, it requires *significantly* fewer geolocation fix acquisitions, up to 41x fewer acquisitions than an accelerometer-assisted binary classification scheme and 243x fewer than continuous tracking over our collected data.

6.1 Activity recognition results

Our activity recognition system, described in Section 4, generates the raw user activity stream indicating the user's mode of transportation. Using our training data set collected from 17 users, our system demonstrates a 98.4% per-window classification accuracy over our activities with 10-fold cross-validation. The confusion matrix is shown in Table 1. We found that walking up and down stairs were confused often with walking on a flat surface, so we aggregate them together into our final classifier. Similarly, standing and sitting are aggregated into idling. The end result is an output of five activities: $Walking$, $Running$, $Driving$, $Biking$, and $Idling$.

		Predicted							
		Walking	Sitting	Running	Standing	Driving	StairsDown	StairsUp	Biking
Actual	Walking	**99.05%**	0.00%	0.09%	0.06%	0.12%	0.15%	0.34%	0.18%
	Sitting	0.00%	**99.24%**	0.00%	0.51%	0.13%	0.00%	0.00%	0.13%
	Running	0.42%	0.00%	**99.48%**	0.00%	0.00%	0.00%	0.00%	0.10%
	Standing	0.00%	1.33%	0.00%	**97.52%**	1.14%	0.00%	0.00%	0.00%
	Driving	0.16%	0.06%	0.00%	0.13%	**99.00%**	0.00%	0.00%	0.65%
	StairsDown	16.83%	0.00%	0.00%	0.00%	0.00%	**80.20%**	2.97%	0.00%
	StairsUp	29.63%	0.00%	0.00%	0.00%	0.00%	7.41%	**62.96%**	0.00%
	Biking	0.56%	0.00%	0.00%	0.00%	2.33%	0.00%	0.00%	**97.11%**

Table 1. Confusion matrix for activity classification. Overall accuracy is 98.4%.

We further evaluated the power consumption of the activity recognition system using a Monsoon Solutions Power Monitor, a hardware power meter that directly provides electricity through connected leads and measures the resulting power draw. The results from both test devices are shown in Table 2. We found that the continuous recognition consumes up to approximately 225 mW. Note that the activity recognition's signal processing, feature extraction, and classification consume very little power, whereas the baseline idle CPU (kept alive with a wake lock to ensure that the system continues to run even if the screen is off) consumes the most. As point of reference, the power consumption of the Galaxy S II to acquire a GPS fix is 771 mW, including the baseline CPU.

Smartphone	Baseline idle CPU with wake lock	Accelerometer sampling to 32 Hz	Feature extraction and classification
Samsung Nexus S	167.15 mW	18.03 mW	4.49 mW
Samsung Galaxy S II	184.34 mW	35.53 mW	5.05 mW

Table 2. Power consumption of the activity recognition system on commodity smartphones, where the phones were run in "airplane mode" with no WiFi and the screen off. Total power on the Galaxy S II is 224.92 mW.

6.2 Activity recognition transition-triggered geolocation fix

The following subsections provide an evaluation to quantify the performance of activity recognition transition-triggering against competing schemes. Note that the system runs successfully on our test smartphones and correctly triggers geolocation acquisition fixes on transitions between specific modes of transportation; however, to fully evaluate the system with varying parameters, we recorded data traces and ran offline experiments to expedite our work. Both the online and offline execution used the same software code base.

In these experiments we asked users to carry two clock-aligned Android phones to collect two streams of data over the course of entire days, including overnight. First, to collect **ground truth transitions**, users carried a phone running a program that could record the immediate time and geocoordinate

when a button is pressed. We asked users to press this button at transition moments between modes of transportation which could be appropriate for a LBA to acquire a fix. Second, to collect **traces of sensor data**, users carried another phone which they kept with them in any position they wanted. The program on this phone continuously recorded accelerometer data and geolocation data. Note that both the ground truth transitions and sensor data collection could have been performed on the same phone, but we did not want the physical interaction for ground-truth marking to produce anomalous sensor data.

In this data set we collected sensor readings totaling 547,583 minutes (or 6.34 days). During this time, users also collected 124 ground-truth transition events of their choice that they believed represented opportunistic locations where a LBA could provide value. These locations turned out to include the users' home, workplace, shopping malls, gas stations, restaurants, and other places. Geolocation fixes were acquired through either GPS or WiFi/cell tower trilateration.

Given the two data streams, we evaluated our system against competing geolocation-triggering schemes along the following dimensions:

1. What is the recall of each scheme – that is, what fraction of ground-truth transitions were correctly recognized?
2. What is the delay between the ground-truth observation and the triggered geolocation fix acquisition in terms of time and distance?
3. How many fix acquisitions are triggered by each scheme?

6.3 Geolocation fix acquisition triggering schemes

We evaluated four algorithms for triggering a geolocation fix acquisition:

- **Continuous Tracking.** This approach triggers a geolocation fix as often as possible, where the collected data came directly from the sensor-collection phone as described earlier. To enable this approach, we used Android's LocationManager class and asked for callbacks from either GPS or network services, where the callbacks are set with a minimum time of 1 millisecond and minimum distance of 0.1 meters. However, as we show later, the callback minimum time is not necessarily honored by the underlying location service.
- **Continuous Tracking with Fixed Duty Cycling.** This approach triggers acquisitions with a 15-second inter-acquisition period.
- **Accelerometer-Assisted, Binary States.** This approach is representative of current state-of-the-art research work [1, 39, 34, 27] where the accelerometer is used to detect idling versus non-idling. Here, we evaluate a 4-second segment of accelerometer data, and if the user is idling, then geolocalization is deactivated, whereas if the user is not idling (that is, moving), then the geolocation fixes are acquired as often as possible (using the same settings as Continuous Tracking).
- **Activity Recognition-Triggered, Window N.** This approach represents our work, where we apply our activity recognition software on the collected sensor data and apply a smoothing window of size N (which varies between

5 and 40). The system then triggers a geolocation fix acquisition only on the transitions shown in the FSM of Figure 3.

6.4 Recall

We first evaluate the recall of each triggering scheme by quantifying the fraction of the 124 ground-truth transition events that were correctly identified for geolocation triggering. We say that a true-positive identification of a ground-truth event occurs when the scheme performs triggering within 60 seconds of the ground-truth, where a particular triggering can be associated with only one other ground-truth event. In Table 3 we show the recall of each triggering scheme.

Triggering Scheme	Recall
Continuous Tracking	0.8952
Continuous Tracking w/Fixed Duty Cycling	0.8952
Accelerometer-Assisted, Binary States	1.0000
Activity Recognition-Triggered, Window 5	1.0000
Activity Recognition-Triggered, Window 10	0.9677
Activity Recognition-Triggered, Window 20	0.9274
Activity Recognition-Triggered, Window 30	0.8306
Activity Recognition-Triggered, Window 40	0.7097

Table 3. Recall of ground-truth observations for each triggering scheme.

Most surprisingly, Continuous Tracking, with and without Fixed Duty Cycling, performed relatively poorly despite the fact that they supposedly trigger geolocation fixes often. The problem is that while the Android API allows the developer to set the minimum time between the automatic geolocation callbacks, it is only a *hint* to the underlying system software or device driver [12], which may instead provide the callbacks as it sees fit (for example, it may aggressively try to reduce power by conserving geolocation fix invocations). Indeed, we occasionally observed long lapses (on the order of 15 minutes) between consecutive geolocation fixes with Continuous Tracking regardless of whether the user was moving or not. In the future we look to explore this issue more fully.

The Accelerometer-Assisted, Binary States scheme performs well with 100% recall, which is representative of the fact that transitions between transportation modes are captured by the binary accelerometer classifier.

Our Activity Recognition-Triggered, Window N scheme exhibits 100% recall with a window size of 5 but suffers decreasing recall with increasing N. The recall diminishes because a longer smoothing window causes activity recognition convergence to take longer, resulting in missed ground-truth transitions.

6.5 Delay

We next evaluated each scheme with respect to the delay between the ground-truth transition events and the triggered geolocation acquisition. Figure 4 shows

the median time delay in units of seconds. We observe that the Continuous Tracking scheme has a low time delay, and as expected the Continuous Tracking with Fixed Duty Cycling set to a 15-second period has a longer delay. The Acceleration-Assisted, Binary States has the lowest time delay, again illustrating its ability to turn on triggered geolocation fixes as soon as it detects any movement. Our Activity Recognition-Triggered, Window N scheme incurs longer delay than, but is competitive against, both Continuous Tracking and the Accelerometer-Assisted Binary States scheme with N=5. Note, though, that it again demonstrates decreasing performance with increasing window size. Activity recognition simply responds slower when the smoothing window is large.

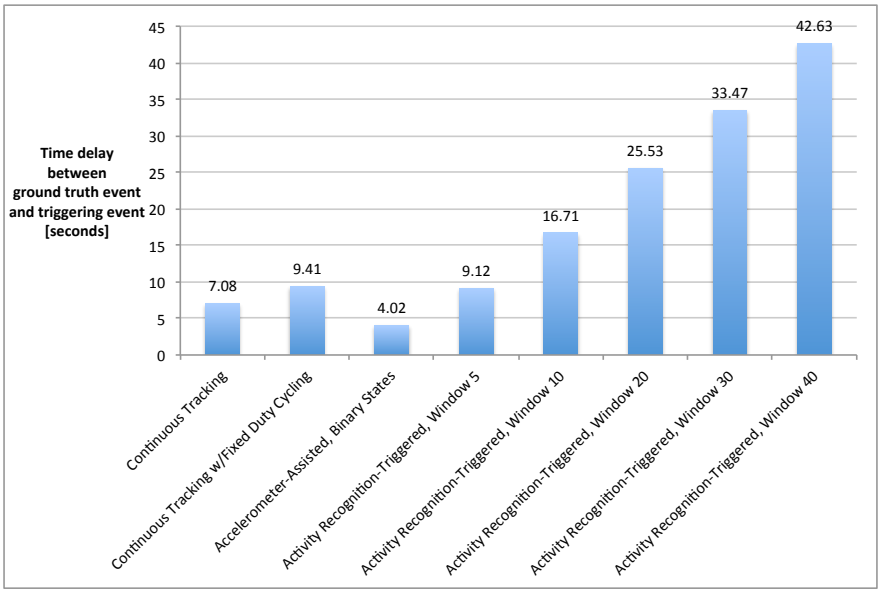

Fig. 4. Median time delay in units of seconds between ground-truth event and triggered geolocation acquisition.

Figure 5 shows the median distance delay in units of meters, where we calculated the Haversine distance from the latitude and longitude coordinates. Note that any distance between the time of the ground-truth event and the time of the triggering is dependent on the mode of transportation; for example, walking will most likely produce a shorter distance than driving. Nonetheless, we observe similar relative behavior between the schemes as with the time delay. It is important to further note that the small distances involved here, on the order of 30-40 meters, is close to the 10m accuracy of GPS and at or below the known accuracy of cellular network trilateration.

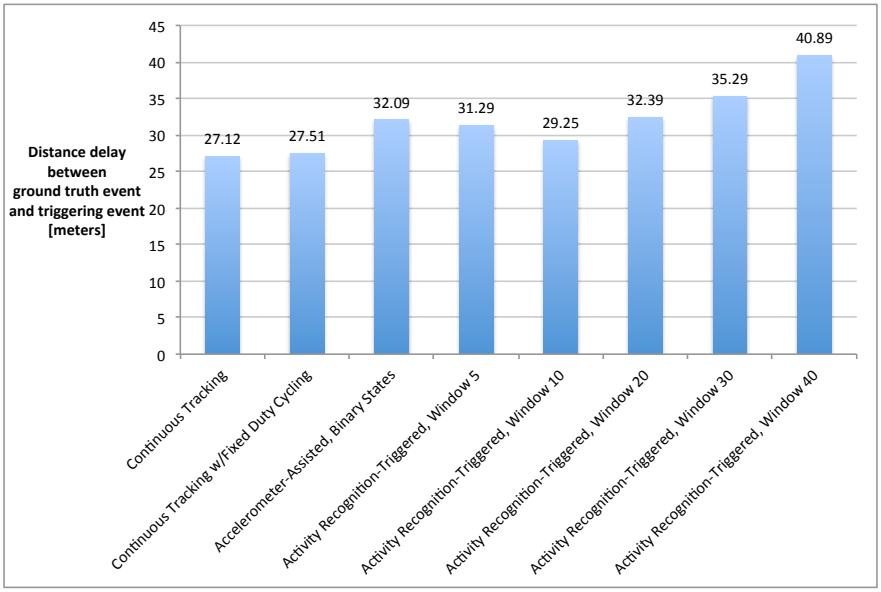

Fig. 5. Median distance delay in units of meters between ground-truth event and triggered geolocation acquisition.

6.6 Number of fix acquisitions

We now quantify the core advantage offered by our system. Figure 6 shows the number of geolocation fix acquisitions for each scheme over the entire data set.

As expected, the Continuous Tracking schemes generate the most geolocation fixes, and even though the callbacks may be throttled as we observed in earlier subsections, its total number is higher than the other schemes. The Accelerometer-Assisted, Binary States scheme is more efficient, acquiring 5.9x fewer fixes than Continuous Tracking and 1.6x fewer than Continuous Tracking with Fixed Duty Cycling; its main advantage is that it disables geolocation fixes when the user is idle, such as when the user is sleeping.

It can be seen that our Activity Recognition-Triggered scheme is even more efficient and significantly reduces the number of geolocation fix aquisitions. With N=5, our scheme produces 243x fewer fixes than Continuous Tracking and 41x fewer than Accelerometer-Assisted, Binary States.

The advantage of our scheme here stems from its intelligent transition-based triggering. Continuous Tracking acquires geolocation fixes without regard for user events because it is missing the high-level information offered by sensor fusion, for example when coupled with the accelerometer. The Accelerometer-Assisted, Binary States scheme improves upon Continuous Tracking, but while it can eliminate geolocation fixes when the user is idle, it can neither distinguish with sufficient detail nor exploit the transition movements that humans emit.

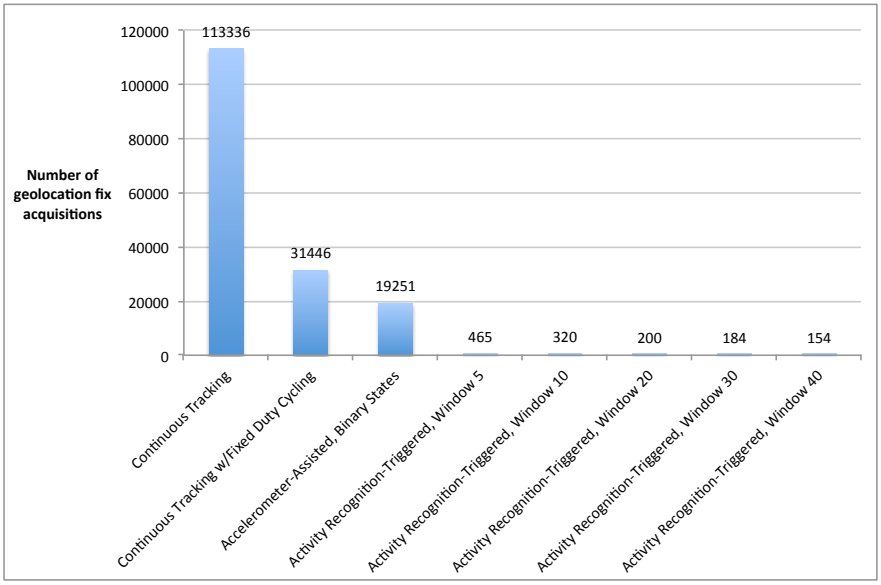

Fig. 6. Total number of triggered geolocation fix acquisitions over the data set.

Additionally, we observe that the longer windows for the Activity Recognition-Triggered schemes reduce the number of geolocation fixes. A longer window will provide more smoothing over the raw activity stream produced by the classifier, reducing choppiness and spurious transitions due to mixed user activities. Note, though, that this smoothing and its apparent reduction in fix acquisitions cause longer delays and reduce recall (due to more missed events), as seen in previous experiments. Nonetheless, we note that the N=40 configuration triggered 3x fewer fixes than the N=5 configuration. In the future we will continue looking into ways of achieving high recall and low delay while invoking fix acquisition as seldom as possible.

All the schemes require that the CPU be active, and in the absence of a precise power-consumption model we rely on the number of geolocation fix acquisition as a close proxy for power usage. In the future we look to develop a power model [29] to study the schemes' battery usage trends.

7 Conclusion

In this paper we consider a subset of location-based applications (LBAs) that require geolocation fix acquisition only at specific and rare points in the day. For LBAs like location-based reminders, performing continuous tracking wastes battery power; instead, the most efficient solution is to able to trigger a single geolocation fix at the target location, which would then activate the LBA-specific logic. We make the assumption that such triggering can occur on transitions

between user modes of transportation, where particular transitions are indicative of points in time when the user is amenable to engagement with an LBA.

To detect transitions, we implemented a low-power, high-resolution activity recognition system that can detect modes of transportation like walking, driving, running, and idling, and to trigger the geolocation fixes, we built a finite state machine, where specific edges (such as $Driving \rightarrow Walking$) trigger a fix. Our experiments show that our system performs triggering with low power and low time/distance delay between ground-truth transitions and the triggering events. Most significantly, our system performs 41x fewer acquisitions than a competitive accelerometer-assisted binary classifier and 243x fewer than continuous tracking over our collected data set.

References

1. F. Abdesslem, A. Philips, and T. Henderson. "Less is More: Energy-Efficient Mobile Sensing with SenseLess," In *Proceedings of ACM MobiHeld*, 2009.
2. Apple, Inc. "iOS Siri," `http://www.apple.com/ios/siri/`.
3. M. Azizyan, I. Constandache, and R. Choudhury. "SurroundSense: Mobile Phone Localization via Ambience Fingerprinting," In *Proceedings of ACM MobiCom*, 2009.
4. L. Bao and S. Intille. "Activity Recognition from User-Annotated Acceleration Data," In *Proceedings of Pervasive*, 2004.
5. Y. Chen, Y. Chawathe, A. LaMarca, and J. Krumm. "Accuracy Characterization for Metropolitan-Scale Wi-Fi Localization," In *Proceedings of ACM MobiSys*, 2005.
6. K. Cheverst, N. Davies, K. Mitchell, and A. Friday. "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project," In *Proceedings of ACM MobiCom*, 2000.
7. S. Consolvo, D. McDonald, T. Toscos, M. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, I. Smith, and J. Landay. "Activity sensing in the wild: a field trial of ubifit garden," In *Proc. of ACM CHI*, 2005.
8. I. Constandache, S. Gaonkar, M. Sayler, R. Choudhury, and L. Cox. "EnLoc: Energy-Efficient Localization for Mobile Phones," In *Proceedings of IEEE Infocom mini-conference*, 2009.
9. O. Dousse, J. Eberle, and M. Mertens. "Place Learning via Direct WiFi Fingerprint Clustering," In *Proceedings of IEEE MDM*, 2012.
10. Endomondo application. `www.endomondo.com`
11. S. Fang and R. Zimmerman. "EnAcq: Energy-efficient GPS Trajectory Data Acquisition Based on Improved Map Matching," In *Proc. of ACM SIGSPATIAL*, 2011.
12. Google Android LocationManager documentation. `developer.android.com/reference/android/location/LocationManager.html`
13. Google Indoor Maps. `maps.google.com/help/maps/indoormaps/`
14. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, 11(1), 2009.
15. T. Huynh, U. Blanke, and B. Schiele. "Scalable recognition of daily activities with wearable sensors," In *Proceedings of LOCA*, 2007.
16. D. Kim, Y. Kim, D. Estrin, and M. Srisastava. "SensLoc: Sensing Everyday Places and Paths using Less Energy," In *Proceedings of ACM SenSys*, 2010.

17. M. Kjaergaard, J. Langdal, T. Godsk, and T. Toftkjaer. "EnTracked: Energy-Efficient Robust Position Tracking for Mobile Devices," In *Proceedings of ACM MobiSys*, 2009.
18. J. Kwapisz, G. Weiss, and S. Moore. "Activity Recognition Using Cell Phone Accelerometers," In *Proceedings of SensorKDD*, 2010.
19. J. Lester, T. Choudhury, and G. Borriello. "A practical approach to recognizing physical activities," In *Proceedings of Pervasive*, 2006.
20. K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. "Energy-accuracy trade-off for continuous mobile device location," In *Proceedings of ACM MobiSys*, 2010.
21. J. Liu, B. Priyantha, T. Hart, H. Ramos, A. Loureiro, and Q. Wang. "Energy Efficient GPS Sensing with Cloud Offloading," In *Proc. of ACM SenSys*, 2012.
22. H. Lu, J. Yang, Z. Liu, N. Lane, T. Choudhury, and A. Campbell. "The Jigsaw Continuous Sensing Engine for Mobile Phone Applications," In *Proceedings of ACM SenSys*, 2010.
23. E. Miluzzo, N. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. Eisenman, X. Zheng, and A. Campbell. "Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application," In *Proceedings of ACM SenSys*, 2008.
24. D. Mizell. "Using gravity to estimate accelerometer orientation," In *Proceedings of ISWC*, 2003.
25. S. Nath. "ACE: Exploiting Correlation for Energy-Efficient and Continuous Context Sensing," In *Proceedings of ACM MobiSys*, 2012.
26. A. Ofstad, E. Nicholas, R. Szcodronski, and R. Choudhury. "AAMPL: Accelerometer Augmented Mobile Phone Localization," In *Proceedings of ACM MELT*, 2008.
27. T. Oshin, S. Poslad, and A. Ma. "Improving the Energy-Efficiency of GPS-based Location Sensing Smartphone Applications," In *Proc. of IEEE TrustCom*, 2012.
28. J. Paek, J. Kim, and R. Govindan. "Energy-Efficient Rate-Adaptive GPS-based Positioning for Smartphones," In *Proceedings of ACM MobiSys*, 2010.
29. A. Pathak, C. Hu, and M. Zhang. "Where is the Energy Spent Inside My App? Fine Grained Energy Accounting on Smartphones using Eprof," In *Proceedings of EuroSys*, 2012.
30. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
31. N. Ravi, N. Dandekar, P. Mysore, and M. Littman. "Activity recognition from accelerometer data," In *Proceedings of IAAI*, 2005.
32. M. Reardon. "Location information to make mobile ads more valuable," CNET.com news, April 15, 2013. `news.cnet.com/8301-1035_3-57579746-94/location-information-to-make-mobile-ads-more-valuable/`
33. S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. "Using mobile phones to determine transportation modes," *ACM Transactions on Sensor Networks*, 2010.
34. I. Shafer and M. Chang. "Movement Detection for Power-Efficient Smartphone WLAN Localization," In *Proceedings of ACM MSWiM*, 2010.
35. Skyhook Wireless. `www.skyhookwireless.com`
36. Trapster application. `www.trapster.com`
37. Y. Wang, J. Lin, M. Annavaram, Q. Jacobson, J. Hong, B. Krishamachari, and N. Sadeh. "A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition," In *Proceedings of ACM MobiSys*, 2009.
38. Waze application. `www.waze.com`
39. Z. Zhuang, K.-H. Kim, and J. Singh. "Improving Energy Efficiency of Location Sensing on Smartphones," In *Proceedings of ACM MobiSys*, 2010.