Energy-efficient Trajectory Tracking for Mobile Devices

Mikkel Baun Kjærgaard[1], Sourav Bhattacharya[2], Henrik Blunck[1], Petteri Nurmi[2]
[1]Department of Computer Science
Aarhus University, Denmark
[2]Helsinki Institute for Information Technology HIIT, Department of Computer Science
University of Helsinki, Finland
mikkelbk@cs.au.dk, sourav.bhattacharya@cs.helsinki.fi, blunck@cs.au.dk,
petteri.nurmi@cs.helsinki.fi

ABSTRACT

Emergent location-aware applications often require tracking trajectories of mobile devices over a long period of time. To be useful, the tracking has to be energy-efficient to avoid having a major impact on the battery life of the mobile device. Furthermore, when trajectory information needs to be sent to a remote server, on-device simplification of the trajectories is needed to reduce the amount of data transmission. While there has recently been a lot of work on energy-efficient position tracking, the energy-efficient tracking of trajectories has not been addressed in previous work. In this paper we propose a novel on-device sensor management strategy and a set of trajectory updating protocols which intelligently determine when to sample different sensors (accelerometer, compass and GPS) and when data should be simplified and sent to a remote server. The system is configurable with regards to accuracy requirements and provides a unified framework for both position and trajectory tracking. We demonstrate the effectiveness of our approach by emulation experiments on real world data sets collected from different modes of transportation (walking, running, biking and commuting by car) as well as by validating with a real-world deployment. The results demonstrate that our approach is able to provide considerable savings in the battery consumption compared to a state-of-the-art position tracking system while at the same time maintaining the accuracy of the resulting trajectory, i.e., support of specific accuracy requirements and different types of applications can be ensured.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Experimentation, Measurement

Keywords

Energy-efficiency, Positioning, Mobile computing, GPS, Trajectory simplification

MobiSys'11, June 28–July 1, 2011, Bethesda, Maryland, USA.

1. INTRODUCTION

An important feature of a modern mobile device is its ability to determine its position. However, often not only knowledge about its current position is required, but also the tracking of its trajectory over longer time intervals. Examples of applications that depend on trajectories rather than just on the current position are sports trackers that log, e.g., running paths [12], shared ride recommenders [1], health care applications that visualize daily patterns and habits of patients [22], and collaborative sensing applications that, e.g., generate maps [18], monitor the environmental impact [19], or map cycling experiences [6].

Continuous sensing of the user's position rapidly consumes the battery of a mobile device. To overcome this issue, previous research has addressed the energy-efficient sensing of the current position of a target using different sensor management strategies (e.g., a-Loc [16], RAPS [20], Zhuang et al. [26] and EnTracked [13]). The main intuition behind these strategies is illustrated on the left side of Figure 1. The center of the circle depicts the last sensed position. Energy-efficient sensor management strategies aim to estimate a new position only once the target cannot be ensured anymore to be within a certain error threshold or distance $E_{Position}$, i.e., within the depicted circle.

Many applications require mobile devices to measure and communicate not only the current position but also to record and maintain information about the user's trajectory, i.e., his motion history. When considering this task, termed tra-

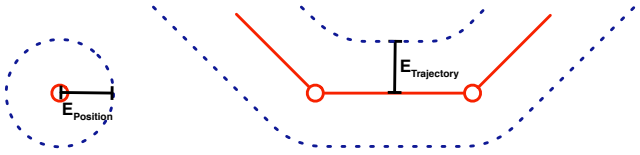


Figure 1: Illustrating error thresholds for position and trajectory tracking, respectively.

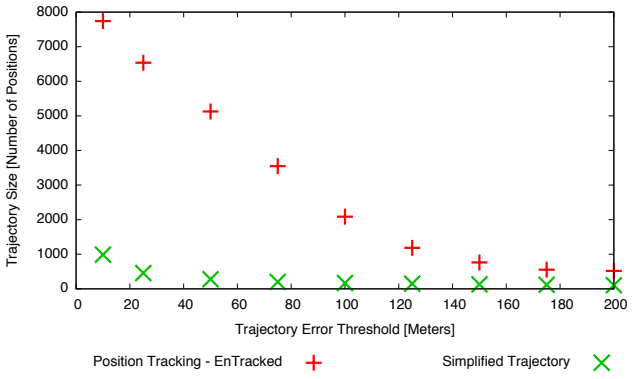


Figure 2: Number of positions sensed by EnTracked compared to the number of points in the simplified trajectory; given for a range of error thresholds and for three hours of car data.

jectory tracking[1], position tracking systems are limited by their sole aim to schedule only the sensing of the current position of a target, be it within some error threshold or by best effort. For trajectory tracking our goal is instead to know the trajectory within some accuracy, i.e., within an error 'corridor' defined by a trajectory threshold $E_{Trajectory}$. Intuitively, this implies that as long as the points where the motion curve bends are accurately sensed, the trajectory will be within the error corridor.

To emphasize the magnitude of energy savings that might be possible to achieve in trajectory tracking, Figure 2 uses three hours of driving data and different error thresholds $E_{Trajectory}$ to demonstrate the difference between positions sensed by the EnTracked [13] position tracking system and a trajectory simplification algorithm applied on the same data. The simplification algorithm reduces the number of position updates while at the same time guaranteeing that the prescribed error threshold $E_{Trajectory}$ is obeyed. To use EnTracked to sense positions so that the resulting trajectory obeys the given error threshold $E_{Trajectory}$, we must set the position error threshold $E_{Position}$ equal to the trajectory error threshold. As indicated by the figure, the simplified trajectory requires only a fifth or less of the positions that EnTracked senses to represent the car's trajectory while at the same time obeying the given error threshold. For a coarse threshold of 200 meters, the size of trajectory data is reduced by 410 positions, whereas for a finer threshold of 10 meters the savings are an impressive 6751. As the example indicates, even with energy-efficient position tracking systems, such as EnTracked, there is a potential for achieving significant power savings in both sensing and communicating motion information.

In this paper we present an energy-efficient system for joint trajectory and position tracking. The presented system combines trajectory simplification algorithms with trajectory update protocols that minimize the power spent on position sensing and data transmission. The proposed protocols save power by avoiding using power on data transmission by spending a smaller amount of power on simplifying the data on the phone before sending it. We also propose a set of novel sensor management strategies that facilitate achieving further energy savings from position sensing. The proposed sensor management strategies build on EnTracked [13] and they enable intelligent switching between compass, accelerometer and GPS sensing to both minimize energy consumption and optimize robustness. For example, by using compass measurements, one of the proposed strategies realizes energy savings by tracking relative heading changes to estimate corner positions directly. By only depending on relative heading changes, the system avoids two drawbacks of phone-based pedestrian dead reckoning: the need for calibrating the sensor to show the correct absolute heading, and requiring knowledge of the orientation of the phone with respect to the target movement direction. Furthermore, each sensor is by default duty cycled to the lowest possible update rate to save energy.

The work presented in this paper makes the following contributions: (1) we propose sensor management strategies for compass-based change-of-direction sensing and adaptive duty cycling strategies for accelerometer and compass sensors; (2) we profile the power consumption of different trajectory simplification algorithms to design energy-efficient trajectory update protocols; (3) we extend the EnTracked [13] system with the proposed strategies and protocols so that the resulting system, denoted EnTracked$_T$, provides a unified framework for position and trajectory tracking; and (4) we provide extensive experimental results for the proposed techniques using a combination of emulations that are based on real-world datasets as well as a validation conducted using a real-world deployment. The results of the emulation indicate that the system can switch between the different sensor modalities, apply duty-cycling, handle different transportation modes, e.g., walking, running, biking or commuting by a car and efficiently simplify trajectories. Furthermore, they provide evidence for that the system can considerably save energy compared to a state-of-the-art position tracking system and remain robust. The validation in an initial real-world deployment indicates that the real system works as efficiently as predicted by the emulation results.

The rest of the paper is structured as follows: In Section 2 we describe the proposed sensor management strategies and trajectory protocols and explain how we extend the earlier EnTracked system with these. The results from evaluating our system by means of emulation are presented in Section 3. In Section 4 we discuss our implementation and the results of our real-world deployment. In Section 5, we present relevant related works. Finally, Section 6 concludes with a summarizing discussion and directions for future work.

2. ENERGY-EFFICIENT TRAJECTORY SENSING AND TRACKING

This section presents the proposed sensor management strategies in Section 2.1 and the proposed trajectory update protocols in Section 2.2. The proposed system EnTracked$_T$ constitutes a framework for both position and trajectory tracking. Our work builds on the EnTracked system [13], which focused solely on position tracking using accelerometer based movement detection together with an energy model to determine when to sample the device's integrated GPS receiver. The EnTracked system has been

[1]Trajectory tracking and related terms and concepts will be formally defined in Section 2.2.

later extended with position update protocols which intelligently determine when to send position updates to a remote server [11].

EnTracked$_T$ aims to support a broad variety of trajectory-based applications as mentioned in the introduction. Some of these applications will reside on the device where others will be in the cloud. Both application types are supported by EnTracked$_T$ as it provides both an on-device API and a remote service API. When utilizing EnTracked$_T$, trajectory-based applications prescribe a trajectory error threshold for tracking targets. While trajectory-based applications could choose the highest possible accuracy, application providers will be motivated to minimize their application's power consumption by providing higher thresholds. Otherwise users could stop using their applications as they experience that the application quickly drains the battery of the mobile device. For some applications this kind of thresholds can be computed directly from the application domain as has been demonstrated for position tracking in continuous location-based search by Lin et al. [16].

In the following we give a few examples to illustrate both the variety of trajectory-based applications and the span of the error thresholds individual applications may require or choose. First, examples of applications that profit from a low error threshold, e.g., 10-25 meters, are sport trackers [12] that aim to ensure, e.g., that the total length of the trajectory and all its turns are accurately recorded, or health care applications [22] that need to support detailed supervision of a person's schedule and actions. Medium error thresholds, e.g., 50-100 meters, are suitable, e.g., for shared ride recommenders [1] which require trajectories which are fine-grained enough to recognize if people are commuting along the same routes through the city, or health care applications that record and visualize the persons' overall daily activity level. Even higher thresholds will be sufficient, e.g., for applications which only aim to record a user's day schedule in terms of the time-stamped sequence of the places of interests he visited. EnTracked$_T$ also provides the possibility of utilizing position and trajectory tracking simultaneously, which is desirable in a number of application scenarios. First, some applications care for both trajectory and position monitoring, e.g., a sports tracker that shares the current position of an athlete to a community web page. Secondly, position tracking can be used to specify when trajectory data should be pushed from the mobile device to a remote server, e.g., the trajectory data should be uploaded once per traveled kilometer.

When an application requests to use EnTracked$_T$ for both position and trajectory tracking, the steps illustrated in Figure 3 are carried out. First, the application issues a request for the tracking of a device with a position and trajectory distance threshold (1). Next the server propagates the request to the EnTracked$_T$ client (2) which finds a start position and returns it through the server (3) to the application (4). The EnTracked$_T$ client then schedules sensor tasks to measure positions, obeying the prescribed distance thresholds (5). As positions are measured, EnTracked$_T$ consults the used position update protocol to determine if a new position has to be delivered and if so obtains simplified trajectory information from the trajectory update protocol and returns these to the application through the server (6)+(7).

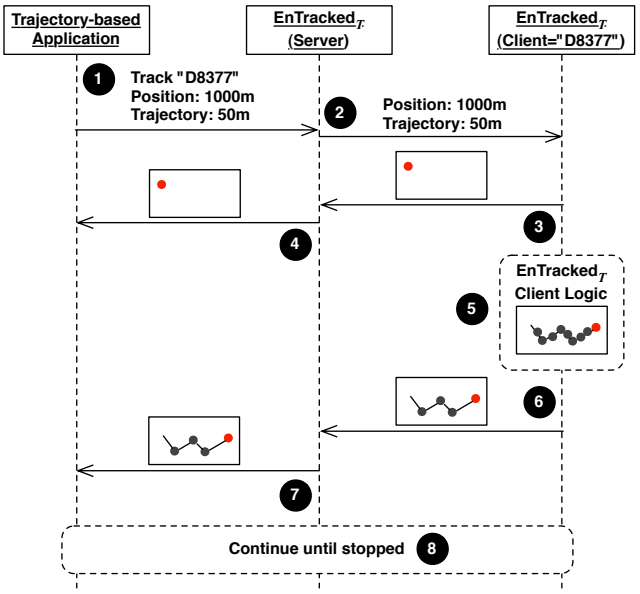


Figure 3: EnTracked$_T$ Communication.

Whenever a request has been received by the EnTracked$_T$ client as described above, the client handles the request following the steps illustrated in Figure 4. To get an initial position, a GPS position is requested (1) and provided to a position update protocol to consider if a position update should be sent (2). If a position update is scheduled, the trajectory update protocol decides whether a trajectory update should be piggybacked with the position update (3), and the update is sent to the server (4). Then, considering the available sensor management strategies and the current requirements, the least power consuming sensor task is selected (5). The scheduled sensor tasks to pick from could involve, e.g., monitoring the compass or accelerometer or to sleep for a certain period (6). The process is restarted, once a task determines that a new GPS position is needed (7).

2.1 Sensor Management Strategies

The EnTracked$_T$ system implements a number of strategies for trajectory and position tracking and the client has on a continuous basis to select the optimal strategy given the current tracking requirements. It therefore asks the strategies, given the current requirements, how much power they are estimated to use on average and selects the strategy with the lowest estimated power consumption. The current EnTracked$_T$ system can select from the following strategies: (1) a heading-aware strategy that uses the compass as a turning point sensor; (2) a distance-aware sensor management that dynamically predicts how long the GPS can sleep between successive position measurements; (3) and a movement-aware sensor management scheme that uses the accelerometer to detect stationary periods. For the heading-aware and the movement-aware strategies the power estimates depend on the level of duty-cycling and the power consumption of the compass and the accelerometer, respectively. For the distance-aware strategy, the power estimates depends on the sleep period and on the power consumption of the GPS. When estimating the power consumption the strategies take into account platform-specific variations in

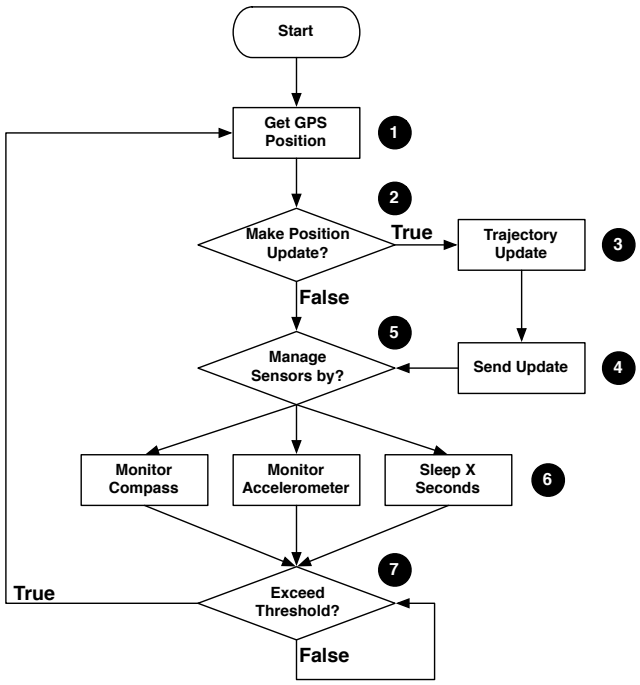


Figure 4: Flow of control

the power consumption of different sensors via power profiles introduced as part of our earlier work [13]. For example, using the internal GPS for obtaining a single position estimate requires on average 11660 millijoules on Nokia N95 devices, whereas the same task can be accomplished using on average only 1530 millijoules on a Nokia N97 device [11] —mostly due to a large difference in the respective GPS power-off delays.

2.1.1 *Heading-Aware Strategy*

The heading-aware strategy reduces power consumption of trajectory tracking by employing the compass as a turn point sensor. The idea is that no explicit update of a target's position and trajectory is needed as long as the target is moving in a straight line with constant speed. To sense when to request a position update, we compare the prescribed trajectory error threshold with the accumulated distance traveled orthogonal to the initial heading given by the compass. This concept is illustrated in Figure 5. The initial heading is depicted as a dotted line (1). When the next heading is measured, the orthogonal distance is recalculated using the previous GPS speed reading and the difference between the two most recent heading measurements (2). When the new and previous heading measurements match, the orthogonal distance remains constant (3). Whenever the heading changes cause the trajectory error threshold to be violated, a new GPS position is requested (4).

Previous work has considered using the compass as a sensor for mobile-phone based inertial dead reckoning [5]. However, this approach has two major drawbacks as the compass has to be precisely calibrated to show the correct absolute heading and as the phone has to be held with a specific orientation towards the body so that the system can determine, e.g., if the target is moving backward or forward. Our approach avoids these drawbacks by monitoring relative

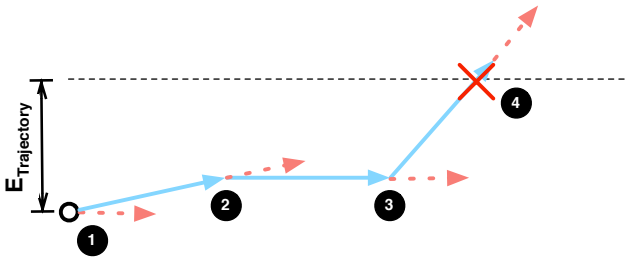


Figure 5: Heading deviations will increase the orthogonal distance beyond the threshold and force the GPS position to be updated.

heading changes instead of relying on the absolute heading or on knowledge about the phone's orientation. Instead, we only assume the latter to be constant during tracking. When this assumption does not hold, the only effect is an increased heading change which will trigger a premature position update but not cause any additional tracking errors.

Formally, given a trajectory threshold $E_{trajectory}$, we calculate the accumulated orthogonal distance D_{orth}. The orthogonal distance at time t_k depends on the estimated speed s_{gps} and the initial heading θ_{start}, on the subsequent heading measurements $[\theta_1, \theta_2, ..., \theta_k]$ taken at times $[t_1, t_2, ..., t_k]$, and on the average error σ of the compass. We assume the average error of the compass for a Nokia N97 mobile device to be 5% which was determined experimentally. The accumulated orthogonal distance D_{orth} is calculated as follows:

$$D_{orth}(t_k) = \sum_{i=1}^{k} (t_i - t_{i-1}) s_{gps} \sin(\|\theta_{start} - \theta_i\|)(1+\sigma). \quad (1)$$

Note that the summation allows to update the error in constant time whenever a new heading measurement θ_{k+1} becomes available. Whenever the orthogonal error D_{orth} exceeds the error threshold $E_{trajectory}$, the heading-aware strategy requests a new GPS position update.

As the standard sample frequency of in-phone compasses is often high, e.g., 10Hz for the N97 phone, we decrease the sampling frequency to reduce power consumption. Specifically, we use the current trajectory error threshold $E_{trajectory}$ and the GPS estimated speed s_{gps} to calculate the period Δt for duty-cycling the compass. As speed changes in general and precise headings between measurements are unknown, the target might in the worst case deviate largely from the assumed speed and heading which causes additional uncertainty about the position of the target. To take this uncertainty into account, we calculate the period Δt when to sense the next heading using

$$\Delta t = \frac{(1-u) \cdot E_{trajectory}}{s_{gps}} \quad (2)$$

where u models the expected amount of uncertainty. We use a small value $u = 0.01$ which was set experimentally. This value was suitable for tracking with most transportation modes in our experiments. Additional knowledge about the motion abilities and patterns of the tracked target and the expected irregularity of its movement can be exploited to more specifically adapt the update time Δt based on the current estimated orthogonal distance $D_{orth}(t_k)$ and measured

heading θ_k, see also [3]. Note also, that summing in Equation 1 over signed instead of absolute heading changes, would allow converse heading changes to cancel each other out, thereby leading to later update times Δt and thus higher energy savings —at the expense, though, of a higher probability to overlook the violation of the error threshold as a result of heading measurement errors.

2.1.2 *Distance-Aware Strategy*

Given knowledge of the target's current speed pattern and the prescribed error threshold, the distance-aware strategy predicts the duration Δt the GPS receiver can sleep between successive updates. To determine the value of Δt, we use an error model that takes into account both the trajectory and the position threshold, the uncertainty of the last GPS fix, and the estimated speed from the last GPS measurement. To minimize the power consumption and to calculate Δt robustly, one has to take into account the delays resulting from powering off features and request delays resulting from powering features on again. In earlier work, we have formalized the problem as a minimization problem for a set of recursive functions [13]. This strategy will be typically selected when the trajectory threshold is high, e.g., above 100 meters, and the target is traveling with a low speed. In this case it will be more efficient to let the GPS sleep and not use any other sensors compared to duty cycling of the compass.

2.1.3 *Movement-Aware Strategy*

Accelerometers can be used to detect movement and to reduce when GPS measurement are sampled. We have exploited this strategy in our previous work for reducing GPS sensing during position tracking of pedestrian targets. While this strategy works well for pedestrians, it can fail when the user is biking or driving. During these activities the absence of acceleration does not provide sufficient indication of the tracked device being stationary. For example, the accelerometer variance can be near zero when the user is traveling along a smooth road with a constant speed. To address this issue, we propose to use a speed threshold that prevents using the accelerometer-based strategy whenever the user is unlikely to be moving by foot.

Figure 6 compares acceleration variance to the traveling speed for different transportation modes. The above mentioned problem is illustrated in the plot, since the acceleration variance for biking and driving may be below the variance threshold of twenty identified for pedestrian in our earlier work [11]. However, the plot also indicates that the proposed speed guard allows separating the biking and especially driving data from the pedestrian data, and therefore enables to prevent the usage of the movement-aware strategy with these transportation modes.

To further save power consumption on the accelerometer, earlier work by Paek et al. [20] has proposed to apply duty cycling of accelerometers. Here we propose to consider the error limit and the maximum speed of the target to calculate the cycle period between samples as $\Delta t = E_{trajectory}/s_{max}$.

2.2 Trajectory Update Protocols

The task of a trajectory update protocol is to communicate motion information obtained on a mobile device in an incremental fashion, adding novel motion information when considered suitable by the protocol. Since most positioning-enabled devices are able to measure and output positions at a high rate, it is natural to consider communicating only a smaller subset of the obtained positions. This subset selected should be minimal in size while still reflecting the overall motion information obtained through the newly obtained positioning measurements. Such selections are obtained by *simplifying* a motion trajectory.

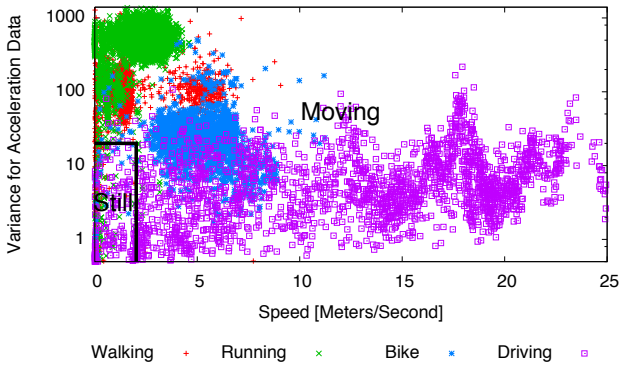


Figure 6: Variance for acceleration data versus speed for different transportation modes.

Definitions.

A trajectory $\ddot{a}$ is a continuous, piecewise linear function that can be defined as follows. Let $\{a_1, a_2, \ldots, a_n\}$ denote a sequence of measurements where each a_i holds, among other data, a position $a_i.\overrightarrow{p}$ and the time of the measurement $a_i.t$. For two successive measurements a_i and a_{i+1}, a *spatiotemporal line segment* $\overline{a_i a_{i+1}}$ on the domain $[a_i.t, a_{i+1}.t]$ is defined as follows:

$$\overline{a_i a_{i+1}} : t \mapsto \frac{(a_{i+1}.t - t)a_i.\overrightarrow{p} + (t - a_i.t)a_{i+1}.\overrightarrow{p}}{a_{i+1}.t - a_i.t}. \quad (3)$$

The measurement sequence $\{a_1, a_2, \ldots, a_n\}$ determines a trajectory $\ddot{a}$ which is defined on the domain $[a_1.t, a_n.t]$. Formally, $\ddot{a}$ is defined as follows:

$$\ddot{a} : t \mapsto \overline{a_i a_{i+1}}(t) \text{ , where } a_i.t \leq t \leq a_{i+1}.t. \quad (4)$$

Trajectory Simplification.

Trajectory simplification has been proposed as a means to reduce data size [4] and communication costs from sending motion information from mobile devices [15, 25]. From an algorithmic point of view, trajectory simplification can be viewed as a special case of line simplification, which has been thoroughly discussed in the computational geometry community. In the line simplification task, the goal is to select a subset of the points of the original polyline, so that the resulting simplified polyline does not deviate more from the original one than prescribed by a numeric error threshold. This threshold, termed trajectory error threshold in the context of this paper, is defined with regards to a chosen error metric, which we in this section fix to be the *time uniform distance* $\mathcal{E}_u$. The time uniform distance $\mathcal{E}_u$ is defined as the distance of a timestamped point a_m with respect to a spatio-temporal line segment $\overline{a_i a_j}$ as follows:

$$\mathcal{E}_u(a_m, \overline{a_i a_j}) = \sqrt{(x_m - x_c)^2 + (y_m - y_c)^2}$$

Here $a_c = (x_c, y_c, t_c)$ is the unique point on $\overline{a_i a_j}$ with the same timestamp as a_m; see [4]. The trajectory distance w.r.t. $\mathcal{E}_u$ between a trajectory $\ddot{a}$ and its simplification $\ddot{u}$ is then obtained as the maximum of the Euclidean distances between $a_i.\overrightarrow{p}$ and $\ddot{u}(a_i.t)$ over all measured timestamped positions a_i of trajectory $\ddot{a}$.

A *(min-number) line simplification algorithm* is called *optimal* if it outputs a simplification with the minimal number of points among all simplifications obeying the prescribed error threshold. The most widely known and used simplification algorithm, though not optimal, is the Douglas-Peucker algorithm, which was designed for cartographic simplification. The Douglas-Peucker algorithm follows the divide-and-conquer paradigm. It starts out with the single line segment $\overline{a_0 a_{n-1}}$ between the first and the last point of the original trajectory $\ddot{a}$. It then identifies the point a_i of the original trajectory which is farthest away (w.r.t. the chosen metric) from this line segment. In case, the measured distance violates the error threshold, the point a_i is added to the simplification, and the algorithm then is called recursively for $a_0 \ldots, a_i$ and for $a_{i+1}, \ldots, a_{n-1}$. The algorithm terminates, once for all line segments generated in the recursive calls no point a_j of the original trajectory violates the error threshold anymore. Lange et al. [15] proposed as an alternative an iterative algorithm named *Simple Section Heuristic*. The algorithm starts out by adding the oldest point s_0 of the trajectory $\ddot{s}$ to the simplification $\ddot{u}$. Then, it iteratively probes the subsequent points $s_1, \ldots$ of $\ddot{s}$, until it finds a point s_k so that if $\overline{s_0 s_k}$ were part of $\ddot{u}$, it would violate the error threshold, i.e., $\ddot{u}$ would leave the error corridor around $\ddot{s}$. It then adds the last non-violating point s_{k-1} to the simplification. The algorithm then sets s_k (in place of s_0) as the next starting point and iterates until all points of $\ddot{s}$ have been processed. Lange et al. furthermore presented an optimized version of the algorithm named *Section Heuristic* by reducing the number of points that need to be probed for violations of the error threshold during an iteration. They observed that individual probes for points s_i will be unnecessary to carry out explicitly, if a violation of the error threshold by s_i implies a violation by the respectively latest point s_j added to the list of points to probe. In this case, these points s_i are removed from the list of points to probe, thus reducing the time for probing in subsequent iterations.

Cao et al. [4] and Lange et al. [15] empirically evaluate the Douglas-Peucker algorithm and determined that for the investigated data sets its trajectory *reduction efficiency*, measured as the size of the simplified trajectory divided by the original one, is over 90% (respectively 80%) of the efficiency obtained by an optimal simplification algorithm. Lange et al. also evaluate that their Section Heuristics algorithm has an even higher reduction efficiency. We decided not to consider optimal line simplification algorithms, since their computational cost is much higher, while yielding only a marginally improved reduction efficiency. Cao et al. determine running times, several hundred times higher for optimal line simplification algorithms compared to the Douglas-Peucker algorithm, run on trajectories of a few hundred positions.

Simplification and energy consumption.

Our motivation for employing trajectory simplification is primarily to reduce the energy costs for communicating trajectory data. These costs are proportional in the size of the simplified trajectory $\ddot{u}$, additional to some overhead for initiating the trajectory update. Computing the simplification, on the other hand, consumes energy as well. To evaluate these costs, and how they compare to the resulting savings in communication costs, we plot in Figure 7 the energy consumption for computing simplifications on a Nokia N97 for commuting trajectories of varying size. The simplification algorithms, the performance of which is shown in the figure for trajectory thresholds of 50m and 200m, respectively, are the Douglas-Peucker algorithm, and Lange et al.'s optimized Section Heuristics algorithm.

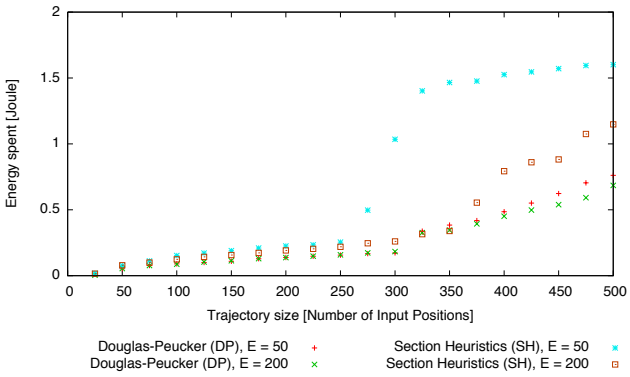


Figure 7: Power consumption of simplification algorithms run on commuting data of various sizes and with error thresholds of 50m and 200m, resp.

The plotted data indicates that the energy costs for running the two algorithms on mobile phones exhibit the same asymptotic behaviour as expected for their running time[2]: While both algorithms have a worst-case running time of $\mathcal{O}(n^2)$, i.e., quadratic in the number of timestamped positions given as input to the simplification, they can be expected to perform better for real-world data. This has been noted specifically for the Douglas-Peucker algorithm and also shows in the plots. The same seems to hold —to a somewhat lesser extent— for the Section Heuristics algorithm by Lange et al., which in our evaluations though showed to be less efficient than the Douglas-Peucker algorithm, especially for larger inputs. Since the energy-consumptions shown for simplifications in Figure 7 are near-line, especially for the Douglas-Peucker algorithm, this provides evidence for that the CPU costs invested for simplifying trajectories pay off in the overall energy budget due to reduced costs for communicating trajectory data. Specifically this claim holds for the N97 phone model and for the real-world trajectory data considered, when taking into account the radio costs for sending trajectory data on the N97, which have been determined experimentally to be roughly 2 millijoule per timestamped position, and assuming that timestamped positions are sent in a format of 16 bytes. Note, that most services will enforce a more verbose data format for sending trajectory data (e.g. for reasons of cross-platform utilization and web-service compliance) and thus enforce a considerably

[2]Note that due to upcoming multi-core and energy-aware inphone CPU chips, the relation between runtime and energy costs may be less strict for future mobile phone generations.

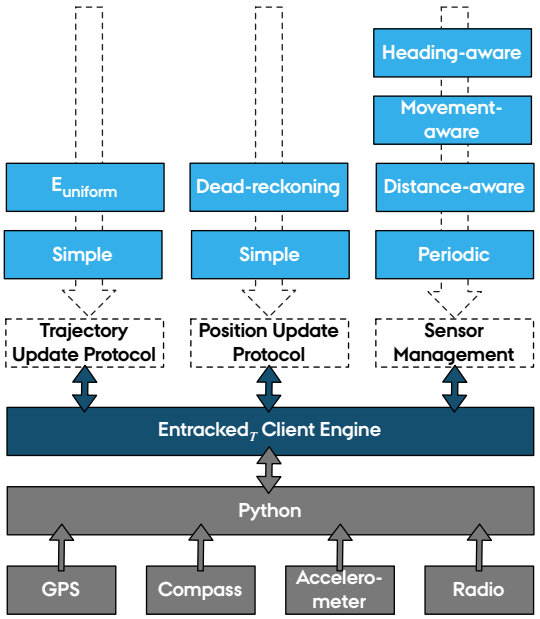


Figure 8: Software architecture of the EnTracked$_T$ client engine

higher amount of data to be sent per timestamped position —which results in higher energy savings achievable by simplification, since these grow linearly with the size of the data format for timestamped positions.

2.3 System Architecture

We have implemented the client part of EnTracked$_T$ in Python for S60 based on a layered architecture as depicted in Figure 8. From a high-level view there are two layers: a *strategy and protocol layer* and a *client engine layer*. In the strategy and protocol layer different protocols and strategies can be plugged-into the system. The client engine layer deals with the system logic and platform integration. The platform integration represents a device that is capable of sending radio packets, providing GPS positions, compass measurements and acceleration measurements and therefore allows the integration either to make use of the specific hardware on a phone or to read in measurement logs to perform emulation. The client engine also provides an on-device API that applications can use to request position or trajectory tracking. The EnTracked$_T$ server is implemented in Java on top of an OSGI engine as part of the PerPos Platform [8]. It provides a Service API that applications can use to request tracking.

3. EMULATION EXPERIMENTS

To assess the impact of the different sensor management strategies and trajectory updating protocols on the power consumption and position error, we have conducted a set of emulation experiments using datasets that have been collected from real life situations. In the experiments we vary the trajectory and positioning error thresholds that are used for tracking, and we use power models for sensors in a Nokia N97 device to estimate the power consumption of the system under different configurations. In the evaluation we consider power models for (i) different phone sensors (GPS, accelerometer and compass), (ii) data transmission over 3G connectivity and (iii) CPU load. The power models have been derived with the help of the Nokia Energy Profiler.[3] The process for generating the power models is described in [13] and the parameter values of the power models for the Nokia N97 are described in [11].

As part of the experiments, we compare our system against two published systems, EnTracked [11, 13] and the system of Lange et al. [15]. We also consider a dead reckoning (DR) based location update protocol while evaluating the performance of our system for simultaneous position and trajectory tracking. In terms of sensor management strategies, we consider three different versions of our system. The first version, periodic scheduling (PR), samples GPS measurements at fixed intervals without utilizing any intelligent sensor management strategies. We compare the above to two versions of the system presented here: to EnTracked$_{T_{-\mathrm{h}}}$ which considers the distance and movement-aware sensor management strategies to reduce the sampling of GPS measurements, and to EnTracked$_T$, which considers all three sensor management strategies, i.e., also the heading-aware strategy. In terms of trajectory simplification, we consider the Section Heuristic (SH) algorithm introduced by Lange et al. [15] and the Douglas-Peucker (DP) algorithm. These algorithms are applied separately for each version of our system.

3.1 Data Collection

We consider 11 datasets in the emulation experiments. The datasets were collected from walking, running, biking and car driving activities undertaken by different users. The datasets that were considered in the evaluation are summarized in Table 1 together with statistics about the number of GPS, accelerometer and compass measurements in the individual datasets. The overall duration summed over all datasets is around 11.5 hours and the data contains over 23,500 GPS measurements.

Table 1: Summary statistics of the datasets used in the emulation experiments.

	# GPS	#Acc.	#Comp.	Duration (hr.)
Biking	630	131178	31204	0.95
	975	113732	27050	0.83
Driving	2240	208816	49667	1.52
	5129	233002	55163	1.69
	3068	132704	31444	0.97
Running	388	116016	27599	0.84
	148	54846	13046	0.40
Walking	2049	165260	39268	1.20
	3423	164184	38824	1.23
	3266	152012	35962	1.10
	2329	107338	25412	0.78
Total	**23645**	**1579088**	**374639**	**11.50**

Each dataset was collected by a single user who was given a Nokia N97 mobile phone that was configured to log GPS

[3]Our current implementation of the system uses Python APIs to measure instantaneous power consumption on Nokia S60 devices, including the N97.

(1 Hz), accelerometer (38 Hz) and compass (10 Hz) measurements onto the phone's memory card.[4] In addition, the user carried a high precision u-blox LEA-5H GPS receiver that sampled SBAS corrected GPS measurements at a rate of 4 Hz. The measurements of the u-blox GPS receiver were used as ground-truth of the user's location in the experiments and we manually inspected the measurements to ensure they followed the actual path of the user. Timestamps in GPS measurements were used to match the ground-truth values with the sensor data collected from the phone. As the GPS values are sampled at discrete intervals, the matching contains some inaccuracies that can cause errors in the position values. The magnitude of these errors depends on the velocity of the user and in our case the maximal error is around 4 meters when the user is traveling with a velocity of 120 km/h. However, since most of our data was collected with significantly slower velocities, the error caused by the matching process can thus be safely ignored in the evaluation.

3.2 Power Consumption

We first consider the average power consumption of the different systems for trajectory tracking with varying trajectory thresholds $E_{Trajectory}$. The results of this comparison are shown in Fig. 9(a) where the averaging is done over all datasets. From the results we can observe that the baseline systems, i.e., EnTracked and the system of Lange et al., consistently consume more power than our system variants. The power consumption of the former systems varies from over 450mW for a 25 meter threshold to 305mW for a 200 meter threshold. The main reason for their worse performance is that they utilize a dead reckoning protocol for sending location updates. Consequently, they implicitly couple trajectory tracking with position tracking and therefore cannot support differing thresholds for the two tasks. The periodic sampling strategy (PR) mainly consumes power for sampling the GPS receiver and it has a relatively constant power consumption profile at around 288mW. The power consumption of our system variants is consistently lower. EnTracked$_{T_{-\mathrm{h}}}$ and EnTracked$_T$ perform almost identically for larger error thresholds (135mW and 124mW). However, with smaller trajectory error thresholds, the inclusion of the heading-aware strategy provides substantial power savings compared to the EnTracked$_{T_{-\mathrm{h}}}$ system (255mW and 155mW). With regards to trajectory simplification algorithms, the power consumption of the Section Heuristic (SH) and the Douglas-Peucker (DP) algorithms is practically identical.

The tracking of the user's trajectory often needs to be combined with the tracking of the user's position with some error threshold, $E_{Position}$. Figure 9(b) shows how the power consumption of the different systems varies for the case where trajectory tracking is combined with position tracking using a 1000 meter error threshold. As both EnTracked and the system of Lange et al. perform position tracking by default, their power consumption is identical to the trajectory tracking case. The performance of the periodic sampling strategy (PR) is practically identical, though the position tracking slightly increases power consumption ($\approx$ 10 mW). The main difference in the results is that the performance of the EnTracked$_{T_{-\mathrm{h}}}$ system decreases significantly when position tracking is requested and becomes essentially identical to the original EnTracked system. This is overcome by EnTracked$_T$ due to the heading-aware strategy, making it the least power consuming of all evaluated systems. Finally, similarly to the trajectory tracking case, the results indicate no differences in power consumption for the two trajectory simplification algorithms.

The results of the power consumption analysis thus indicate that our system EnTracked$_T$ is able to provide substantial power savings for tracking the trajectory of the user. Moreover, the results indicate that the highest savings can be obtained when all of the sensor management strategies are combined with one of the trajectory simplification algorithms.

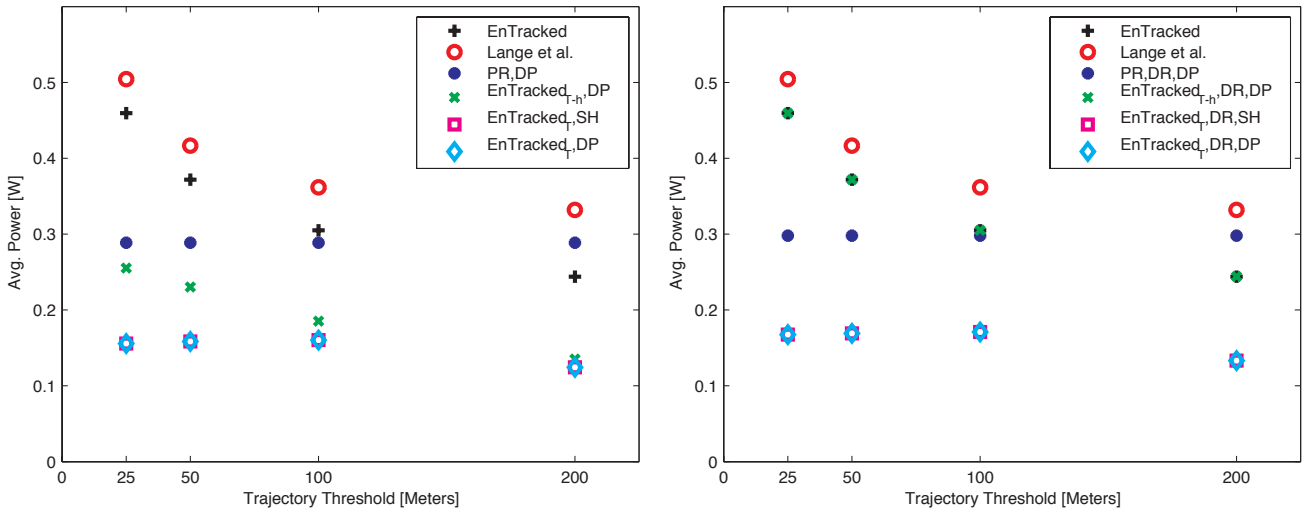
3.3 Robustness of Tracking

The use of intelligent sensor management strategies and trajectory simplification algorithms potentially reduces the accuracy of the tracked trajectories and positions. Each simplification obeys a prescribed error threshold, but only w.r.t. the original tracked trajectory. Thus, simplification may increase the deviations of the tracked trajectory from the ground truth. To assess the significance of these errors, we next consider how often the different systems exceed the given trajectory and position error thresholds w.r.t. ground truth. To estimate these errors, we compare the ground-truth trajectories with the simplified trajectories by estimating the distance[5] between $a_i.\overrightarrow{p}$ and $\ddot{u}(a_i.t)$; see Section 2.2. We first consider the robustness of the different systems for trajectory tracking with varying trajectory thresholds. The results of this comparison are illustrated in Fig. 10(a). From the results we can observe that the periodic sampling strategy (PR) and the system of Lange et al. are able to provide the most robust tracking of the user's trajectories. These systems sample GPS measurements frequently which helps them to provide better robustness with the expense of higher power consumption. When the error threshold is 25 meters, all systems perform relatively poorly with error percentages between 18 and 33. The performance of the EnTracked system is relatively stable and its error percentage varies between 10 and 15 percent when the error threshold is 50 meters or above. The performance of the EnTracked$_{T_{-\mathrm{h}}}$ system is closely related to the performance of the best systems and consistently better than the performance of the original EnTracked system. However, the error percentage of the EnTracked$_T$ system is relatively high when the requested error threshold is small. As the error threshold increases, the performance of EnTracked$_T$ improves and becomes closely related to the most robust systems. In terms of line simplification algorithms, the Douglas-Peucker (DP) algorithm is able to provide slightly better performance than the Section Heuristic algorithm.

Figure 10(b) compares the robustness of the different systems when trajectory tracking is combined with position tracking with the position threshold of 1000 meters. As the periodic sampling strategy (PR), the system of Lange et al. and EnTracked have been designed for position tracking, their performance remains identical to the trajectory tracking case. The performance of EnTracked$_{T_{-\mathrm{h}}}$ remains closely related to the best systems when the error threshold is within 50 meters whereas the EnTracked$_T$ system provides better performance when the error threshold is higher.

[4]Note, that the writing to the memory card causes negligible energy costs, see [13].

[5]We use a variant of Vincenty's algorithm [24] to estimate the geodesic distance between two locations.



(a) Variation of average power consumption with trajectory threshold $E_{Trajectory}$.

(b) Variation of average power consumption with trajectory threshold $E_{Trajectory}$ when position error $E_{Position}$ is requested to be within 1000 meters.

Figure 9: Comparison of power consumption for trajectory tracking (left) and for simultaneous trajectory and position tracking.

More interestingly, in terms of line simplification algorithms, the Section Heuristic algorithm provides significant improvements in robustness over the Douglas-Peucker (DP) algorithm. The reason for this is that the finite buffer length that is required to enable position tracking causes problems for the Douglas-Peucker algorithm.

The results indicate a clear trade-off between tracking robustness and power consumption. The best trade-off between robustness and power consumption is obtained with the EnTracked$_{T_{-\mathrm{h}}}$ and EnTracked$_T$ systems even if they are not as robust as some of the other systems. The results also indicate that the optimal sensor management strategy can depend on the accuracy requirements of the target application. In particular, our results suggest that the EnTracked$_T$ system should be used in situations where the error thresholds are relatively small (e.g., $\leq$ 50 meters) and the EnTracked$_{T_{-\mathrm{h}}}$ system should only be used when the error thresholds are sufficiently large.

3.4 Sensor Specific Power Consumption

In addition to the overall power requirement statistics, knowledge of energy consumption of individual sensors contributing to the overall power gives us interesting insights to the strategies. In Figure 11, we show a break-down of average power consumption into background, radio, GPS, accelerometer and compass power consumptions for emulation experiments with $E_{Trajectory} = 50$ meters and without the requirement of position tracking. It can be seen that getting position estimations using GPS contributes a large percentage of the overall power and judicious use of GPS is essential in case of energy saving. EnTracked as well as, system by Lange et al. do not decouple position tracking from trajectory tracking and suffer from radio usage while transmitting data to a remote server. The periodic strategy combined with trajectory simplification (DP) saves power by minimal usage of radio. Positions are transmitted only when internal trajectory buffer overflow occurs. EnTracked$_{T_{-\mathrm{h}}}$ combined with trajectory simplification decreases power requirement by using the distance-aware and the movement-aware strategies. The EnTracked$_T$ strategy further decreases the use of GPS by using the heading-aware strategy.

3.5 Movement Specific Power Consumption

To illustrate the effect of transportation mode on power consumption, we plot average power consumption of EnTracked$_T$ combined with the Douglas-Peucker (DP) algorithm for biking, driving, running and walking in Figure 12 with $E_{Trajectory} = 50$ meters and with no motion tracking. Here, the average is taken over all users having the same

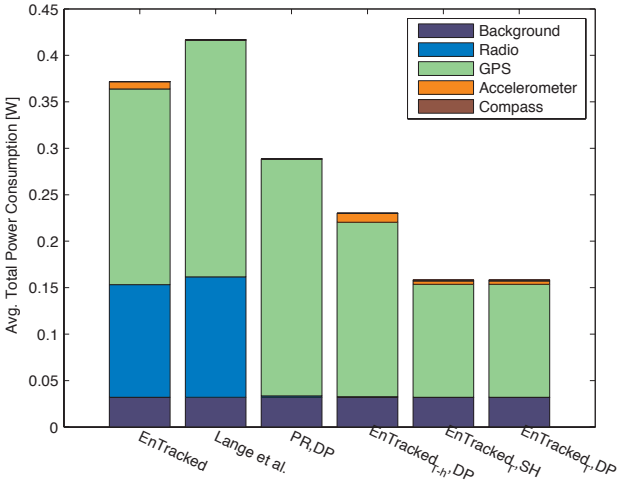
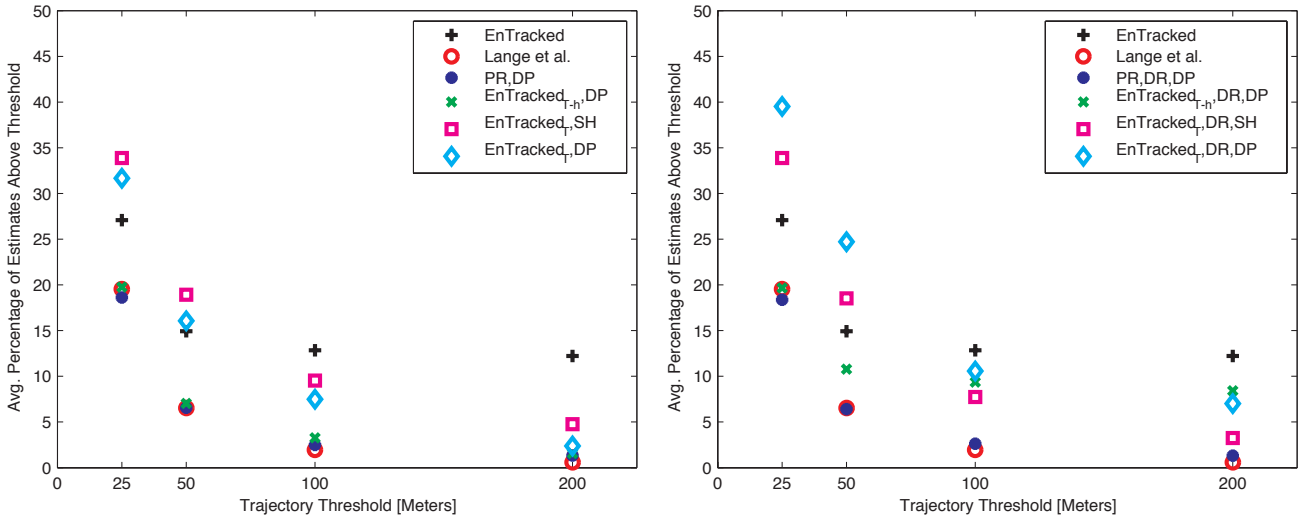


Figure 11: Breakdown of average power consumption with respect to the individual sensors.



(a) Average percentage of times the distance between the user's estimated location and the ground-truth is above the trajectory threshold.

(b) Average percentage of times the distance between the user's estimated location and the ground-truth is above the trajectory threshold with position tracking enabled.

Figure 10: Comparison of average position error in case of trajectory tracking (left) and for simultaneous trajectory and positioning tracking.

transportation mode during data gathering. The figure indicates maximum power saving when traveling by car which is due to the motion of a car being generally tied to the road network and having a relatively stable heading compared to other transportation modes. Moreover, a car is often required to stop at traffic lights or to slowly drive in a queue. The frequent stoppage and steady heading cause accelerometer and compass to be more active than the GPS which results in the highest power savings. The energy consumption is worst in case of running datasets. As the phone shakes continuously during running, the accelerometer and compass can not effectively switch off the GPS. The biking is similar to the case of traveling by a car with less stopping time. However, the Figure 12 shows a considerable amount of accelerometer usage as the biking data gathering included a 10 minute stopping time. In case of walking the use of compass is least as the heading changes frequently.

4. REAL WORLD DEPLOYMENT

The results of the emulation demonstrate that our system can enable robust and accurate tracking of the user's trajectories while at the same time reducing the battery consumption caused by collecting position information. However, real world settings contain numerous external factors that can influence the power consumption of a mobile device and that cannot be reliably taking into account in emulations. For example, the surrounding environment of the user, the load of the mobile network and the distance to the currently connected cell tower can influence the power consumption. To demonstrate that our system can support trajectory tracking also in real world conditions, this section presents initial results from a small deployment of the system that focused on monitoring the trajectories of a mobile user during walking and driving activities.

We deployed two versions of our system for Nokia N97 mobile phones, both configured with a trajectory threshold of 50 meters. Similarly to the emulation discussed in the previous section, we consider two versions of our system, EnTracked$_{T-h}$, which uses only the movement-aware and distance-aware sensor management strategies, and EnTracked$_T$, which uses all three sensor management strategies proposed in Sec. 2. We deployed each version on one Nokia N97 device together with Python S60 and the Nokia Energy Profiler. Both phones had the same firmware versions and they were not in private use. The deployment used the same implementation of the client as the emulations, only this time location information was obtained from the internal GPS, compass and accelerometer of the mobile phone and the devices' 3G radio was used for data transmissions.

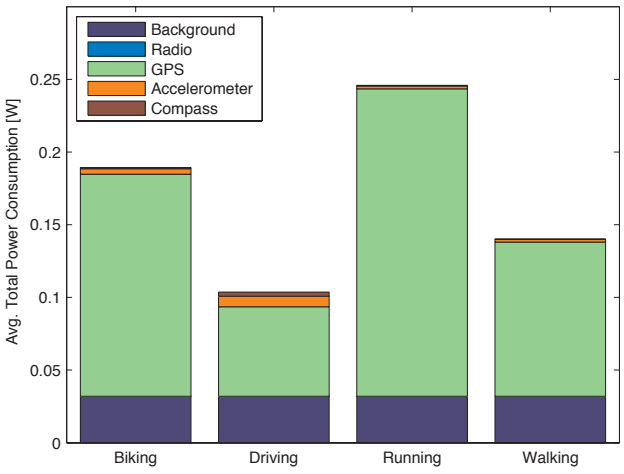
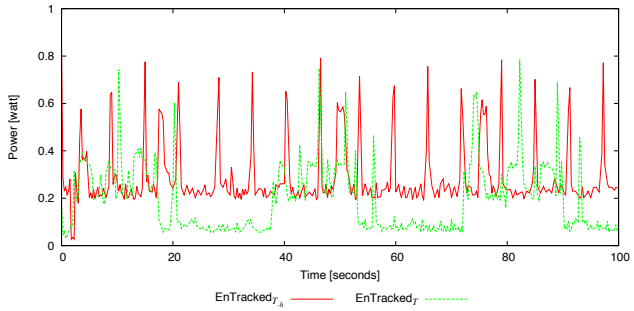
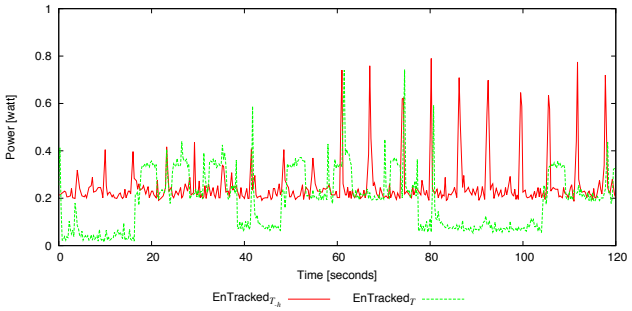


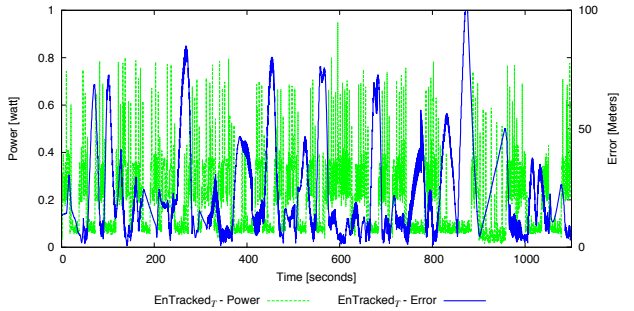
Figure 12: Average power consumption and break down of power in case of EnTracked$_T$, DP with respect to movement styles.



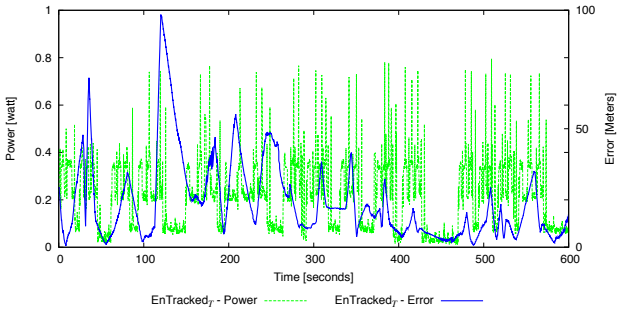
(a) Comparison of power consumption between EnTracked$_T$ and EnTracked$_{T_{-h}}$ during driving using a trajectory threshold of 50 meters.



(b) Comparison of power consumption between EnTracked$_{T_{-h}}$ and EnTracked$_T$ during walking using a trajectory threshold of 50 meters.



(c) Power consumption and real error for EnTracked$_T$ for a trajectory threshold equal to 50 meters for driving data.



(d) Power consumption and real error for EnTracked$_T$ for a trajectory threshold equal to 50 meters for walking data.

Figure 13: Power consumption and positioning error for the deployed tracking systems.

The clients also logged their internal state during the experiments. Other client settings were kept the same as described in the previous sections of the paper. We used the deployment to conduct two field experiments within the city of Aarhus in Denmark. Both experiments were carried out by a single person who went around the city with two N97 phones, one running EnTracked$_{T_{-h}}$ and the other running EnTracked$_T$. The person also carried a high-precision u-blox LEA-5H GPS receiver which was used to collect ground-truth data using the method described in the previous section. In the first experiment the test participant walked a one kilometer route and in the second experiment the participant drove a 12 kilometer (8 miles) route in a car. Before starting the experiment, we started the Nokia Energy Profiler on both devices and configured the profiler to monitor the corresponding EnTracked client. When the route was finished, the Energy Profiler was stopped and the trace files were exported from the phone.

To demonstrate that the heading-aware strategy, described in Section 2.1.1, saves power also when deployed on a real phone, Figures 13(a) and 13(b) compare the power consumption of EnTracked$_{T_{-h}}$ and EnTracked$_T$ during a part of the experiment where EnTracked$_T$ was using the heading-aware strategy. From the plots we can observe that the distance-aware strategy used by EnTracked$_{T_{-h}}$ has to schedule GPS positions all the time to comply to the 50 meter error threshold because of movement speed and GPS uncertainty. Therefore the power consumption is always above 0.2 watt. On the other hand, EnTracked$_T$ repeatedly uses the heading-aware strategy which means that the GPS can be switched off and the less consuming compass is used instead. On average EnTracked$_T$ uses 0.09 watt compared to 0.27 watt for EnTracked$_{T_{-h}}$ during walking. During driving EnTracked$_T$ uses 0.17 watt compared to 0.29 watt for EnTracked$_{T_{-h}}$. The higher power consumption of EnTracked$_T$ in the driving case is due to the higher speed which faster invalidates the orthogonal distance threshold used by the heading-aware strategy.

Figures 13(c) and 13(d) show the power consumption and the real positioning error of the two systems. The real positioning error is calculated with regards to the collected ground truth. Similarly to Figures 13(a) and 13(b), we can observe the lowered power consumption resulting from the use of the heading-aware strategy throughout the experiment. We can also observe how a decrease in error often correlates with higher energy usage as the GPS is used. If we consider the error plot for the driving data, the 90 percentile is 50.3 meters which matches the given error threshold, however, there are a few violations of the threshold due to both erroneous GPS fixes and places where the car made sharp turns. The error plot for the walking data is a bit better with an 90 percentile error of 44.3 meters and only one major violation which is due to a single erroneous GPS fix.

From these results we can conclude that also in a real deployment the proposed sensor management strategies and trajectory update protocols can save power and provide robust tracking of trajectories. Compared to the emulation the initial deployment results also provides indications for power savings comparing EnTracked$_{T_{-h}}$ and EnTracked$_T$. In the deployment the consumption of EnTracked$_T$ was a bit lower for the walking experiment and a bit higher for

the driving experiment. These difference are due to differences in the amount of turns and for the driving experiment there were no queues or stops as in the emulation data.

5. RELATED WORK

5.1 Energy-Efficient Location Tracking

Existing systems for the energy-efficient tracking of a mobile device, such as EnTracked [13], typically focus on sensor management strategies, e.g., for optimizing sensor duty cycles by selecting sensors with a low power consumption whenever possible. The RAPS systems focuses on position tracking in urban areas [20]. The underlying motivation in RAPS is that GPS positioning tends to be inaccurate and often unavailable in urban areas. To reduce power consumption, the system uses GSM information to predict whether GPS information is likely to be inaccurate and other positioning methods that are less accurate in general but more energy-efficient are used during these periods. The system also reduces power consumption by using a movement-aware sensor management strategy and by using Bluetooth to share positions among neighboring peers. Another related system is a-Loc [16], which targets energy-efficient location tracking with dynamic accuracy requirements in mobile search applications where the required accuracy depends on the spatial density of search results. Zhuang et al. [26] study the problem of energy usage when several location-based applications are running at the same time and how to adapt behavior in low battery situations. They propose to use four techniques to address the problems which they name substitution, suppression, piggybacking and adaptation.

In comparison to earlier work, the sensor management strategies and location updating protocols presented in this paper significantly reduce battery consumption by taking advantage of regularities in the user's movement patterns to reduce the need for sampling power hungry sensors and to trigger location updates.

5.2 Trajectory Monitoring

Previous work on system support for handling trajectories includes work on collection and simplification of trajectories and middleware and database abstractions for representing, querying and processing trajectory data. A large number of related results have been obtained within the field of moving object databases [7]. For the representation of trajectories usually piecewise-linear curves in space and time are assumed. Alternative representations, though, such as by by probability functions and by non-linear functions have been discussed to reflect the uncertainty inherent in monitoring motion as well as the non-linearity of motion[21, 23, 2, 3]. Furthermore, various models and algorithms for the simplification of trajectories and their impacts on the accuracy of query results and data derived from trajectories, have been proposed [4, 17, 2]. Wolfson et al. [25] were the first to show how motion status data can be used in position update protocols to save costs for communicating motion data incrementally. Lange et al. [15] present a system for trajectory tracking which allows to simplify the trajectory directly on the mobile device. In contrast to the work listed above, Lange et al. consider and evaluate the benefits of trajectory simplification for the cost reduction for communicating motion data. However, their system ties position and trajectory tracking together which limits the applicability of the system. Furthermore, their system does not address lowering the power consumption of sensing.

To support the collection of trajectory data Kim et al. [9] propose to use place sensing to start and stop trajectory collection by observing when a target leaves and enters places, e.g., to remove the start and stop buttons of sport trackers. However, their system SensLoc does not propose any strategies that can lower the power consumption of trajectory tracking; the ideas presented in this paper could enable SensLoc to do so. A middleware named StarTrack [1] for trajectory manipulation and comparison has been proposed which provides an API to ease the development of trajectory based applications. StarTreck, though, does not address how to energy efficiently collect trajectories. Therefore, EnTracked$_T$ could be used to energy efficiently collect trajectories, which then can be manipulated and compared using either StarTrack or a moving object database system.

6. CONCLUSIONS

The primary contribution of this paper are the novel sensor management strategies and trajectory update protocols that can track the trajectories of mobile devices robustly and energy-efficiently. For instance, for trajectory tracking we propose a sensor management strategy that uses compass measurements to sense the trajectory and continuously and energy-efficiently evaluates the necessity to use the GPS. We power profile trajectory simplification algorithms and trajectory uploading to design energy efficient trajectory update protocols. We extended the system EnTracked with the proposed strategies and protocols to form a unified framework EnTracked$_T$ for position and trajectory tracking.

The results of our emulation show that the proposed methods are able to lower the energy consumption on the mobile device considerably and remain robust even when faced with different types of transportation modes. These results were rechecked in an initial real-world deployment where mobile phones were energy-efficiently tracked in an urban environment during walking and driving. The general conclusions from this work is that savings in power can be achieved when designing systems with direct control over trajectory sensing on mobile devices.

In our ongoing work we pursue the following goals: First, for better positioning support in metropolitan areas apply the proposed methods and findings to other positioning technologies such as location fingerprinting [10] and consider duty cycling the GPS when in low accuracy areas. Second, consider the ideas of Lin et al. [16] for how to use acceleration patterns to estimate speed. Third, propose methods which automatically determine the parameters of our device model for new devices.

Another potential extension of the system EnTracked$_T$ are sensor management strategies which adaptively switch in low-accuracy GPS environments, e.g. urban areas, to positioning means other than GPS. However, one drawback of these is that they require learning of, e.g., the relationship between acceleration patterns and velocity or positioning accuracy for a location. For the later case this might not be so big a problem as systems may benefit from fingerprinting accuracy also for further reasons [14].

Acknowledgments

We thank Lasse Rasmussen for helping collecting measurements and Morten Videbæk Pedersen for providing access to a Python API for the Nokia Energy Profiler. The authors acknowledge the financial support granted by the Danish National Advanced Technology Foundation for the project Galileo: A Platform for Pervasive Positioning under J.nr. 009-2007-2.

This work was supported in part by the Finnish Funding Agency for Technology and Innovation TEKES, under the project Adaptive Interfaces for Consumer Applications (AICA). The authors are grateful to their present and past colleagues in the project. The work was also supported in part by the ICT program of the European Community, under the PASCAL2 network of excellence, ICT-216886-PASCAL2. The publication only reflects the authors' views.

7. REFERENCES

[1] G. Ananthanarayanan, M. Haridasan, I. Mohomed, D. Terry, and C. A. Thekkath. Startrack: a framework for enabling track-based applications. In *Proc. 7th Intl. Conf. Mobile Systems, Applications, and Services (MobiSys 2009)*, pages 207–220, 2009.

[2] L. Becker, H. Blunck, K. H. Hinrichs, and J. Vahrenhold. A framework for moving objects. In *Proc. 15th Intl. Conf. Database and Expert Systems Applications (DEXA '04)*, volume 3180 of *Lecture Notes in Computer Science*, pages 854–863. Springer, 2004.

[3] H. Blunck, K. H. Hinrichs, J. Sondern, and J. Vahrenhold. Modeling and engineering algorithms for mobile data. In *Progress in Spatial Data Handling: Proc. 12th Intl. Symp. Spatial Data Handling (SDH '06)*, pages 61–77, 2006.

[4] H. Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *The VLDB Journal*, 15(3):211–228, 2006.

[5] I. Constandache, R. R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *Proc. 29th IEEE Intl. Conf. Computer Communications (INFOCOM)*, pages 2321–2329, 2010.

[6] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. The bikenet mobile sensing system for cyclist experience mapping. In *Proc. 5th Intl. Conf. Embedded networked sensor systems*, pages 87–101. ACM, 2007.

[7] R. H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann Publishers, 2005.

[8] J. Jensen, K. Schougaard, M. Kjærgaard, and T. Toftkjær. PerPos: a Translucent Positioning Middleware Supporting Adaptation of Internal Positioning Processes. In *Proc. 11th ACM/IFIP/USENIX Intl. Middleware Conf. (Middleware 2010)*, 2010.

[9] D. H. Kim, Y. Kim, D. Estrin, and M. B. Srivastava. Sensloc: sensing everyday places and paths using less energy. In *Proc. 8th ACM Conf. Embedded Networked Sensor Systems*, pages 43–56, 2010.

[10] M. B. Kjærgaard. A Taxonomy for Radio Location Fingerprinting. In *Proc. 3rd Intl. Symp. Location and Context Awareness*, 2007.

[11] M. B. Kjærgaard. On Improving the Energy Efficiency and Robustness of Position Tracking for Mobile Devices. In *Proc. 7th Intl. Conf. Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2010)*, 2010.

[12] M. B. Kjærgaard. Minimizing the Power Consumption of Location-Based Services on Mobile Phones. *IEEE Pervasive Computing*, To appear.

[13] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær. EnTracked: energy-efficient robust position tracking for mobile devices. In *Proc. 7th Intl. Conf. Mobile systems, applications, and services (MobiSys'09)*, pages 221–234, 2009.

[14] M. B. Kjærgaard and K. Weckemann. PosQ: Unsupervised Fingerprinting and Visualization of GPS Positioning Quality. In *Proc. 2nd Intl. Conf. Mobile Computing, Applications, and Services (MobiCASE 2010)*, 2010.

[15] R. Lange, T. Farrell, F. Dürr, and K. Rothermel. Remote real-time trajectory simplification. In *PERCOM '09: Proc. IEEE Intl. Conf. Pervasive Computing and Communications*, pages 1–10, 2009.

[16] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *Proc. 8th Intl. Conf. Mobile Systems, Applications, and Services (MobiSys 2010)*, pages 285–298, 2010.

[17] N. Meratnia and R. de By. Spatiotemporal Compression Techniques for Moving Point Objects. In *Advances in Database Technology - EDBT 2004*, volume 2992 of *Lecture Notes in Computer Science*, pages 561–562. Springer, Berlin, Heidelberg, 2004.

[18] S. Minamimoto, S. Fujii, H. Yamaguchi, and T. Higashino. Local Map Generation using Position and Communication History of Mobile Nodes. In *Proc. 2010 IEEE Intl. Conf. Pervasive Computing and Communications*, pages 2 –10, 2010.

[19] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *Proc. 7th Intl. Conf. Mobile systems, applications, and services*, pages 55–68. ACM, 2009.

[20] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proc. 8th Intl. Conf. Mobile Systems, Applications, and Services (MobiSys 2010)*, pages 299–314, 2010.

[21] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *Proc. 6th Intl. Symp. Advances in Spatial Databases (SSD)*, volume 1651 of *Lecture Notes in Computer Science*, pages 111–132. Springer, Berlin, 1999.

[22] J. Ryder, B. Longstaff, S. Reddy, and D. Estrin. Ambulation: A tool for monitoring mobility patterns over time using mobile phones. In *Intl. Conf. Computational Science and Engineering*, pages 927 –931, 2009.

[23] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving

objects databases. *ACM Transactions on Database Systems*, 29(3):463–507, 2004.

[24] T. Vincenty. Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations. *Survey Review*, 23(176):88–93, 1975.

[25] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and imprecision in modeling the position of moving objects. In *Proc. 14th Intl. Conf. Data Engineering (ICDE '98)*, pages 588–596. IEEE Computer Society, 1998.

[26] Z. Zhuang, K.-H. Kim, and J. P. Singh. Improving energy efficiency of location sensing on smartphones. In *Proc. 8th Intl. Conf. Mobile Systems, Applications, and Services (MobiSys 2010)*, pages 315–330, 2010.