Energy-efficient Trajectory Tracking for Mobile Devices

Based on "Energy-efficient Trajectory Tracking for Mobile Devices" by
M. B. Kjærgaard, Sourav Bhattacharya, Henrik Blunck, Petteri Nurmi

Krzysztof Wiśniewski

Trajectory tracking - basics

- variety of applications rely on tracking trajectory over long period of time, not just a position
- continuous tracking of device's position is not an energy-efficient solution and requires a lot of data transmission
- to ensure low power consumption intelligent strategy for sensor management is required

Trajectory tracking - basics 2

- many applications do not need the highest possible accuracy
- traditional approach is to set acceptable error and sense position every time the object cannot be ensured anymore to be within a certain error threshold
- the goal is to know that trajectory is in a specified "error corridor"
- error threshold can be adjusted accordingly

Trajectory tracking - basics 3

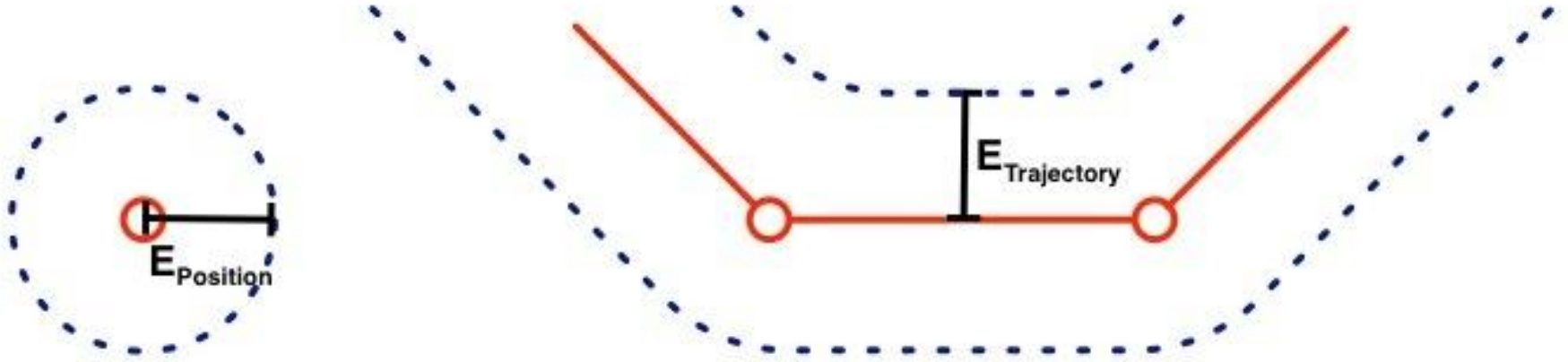


Figure 1: Illustrating error thresholds for position and trajectory tracking, respectively.

Error threshold requirements for selected applications

- sport trackers and healthcare applications need to provide high accuracy (10-25m) to ensure a proper supervision of the tracked object
- shared ride recommenders require fine grained trajectories to recognize people moving in the same directions in the city (50-100m)
- applications for recording daily schedule of a person in terms of time-stamped sequence of visited places (>100m)

Proposed solution

- modified EnTracked system by M.B. Kjægaard, J. Langdal, T.Godsk
- called EnTracked$_T$
- provides possibility of tracking both position and trajectory if needed
- enables adjustable error threshold
- provides combined algorithms for trajectory simplification and update protocols to minimize power spent on sensing and data transmission

Basic workflow

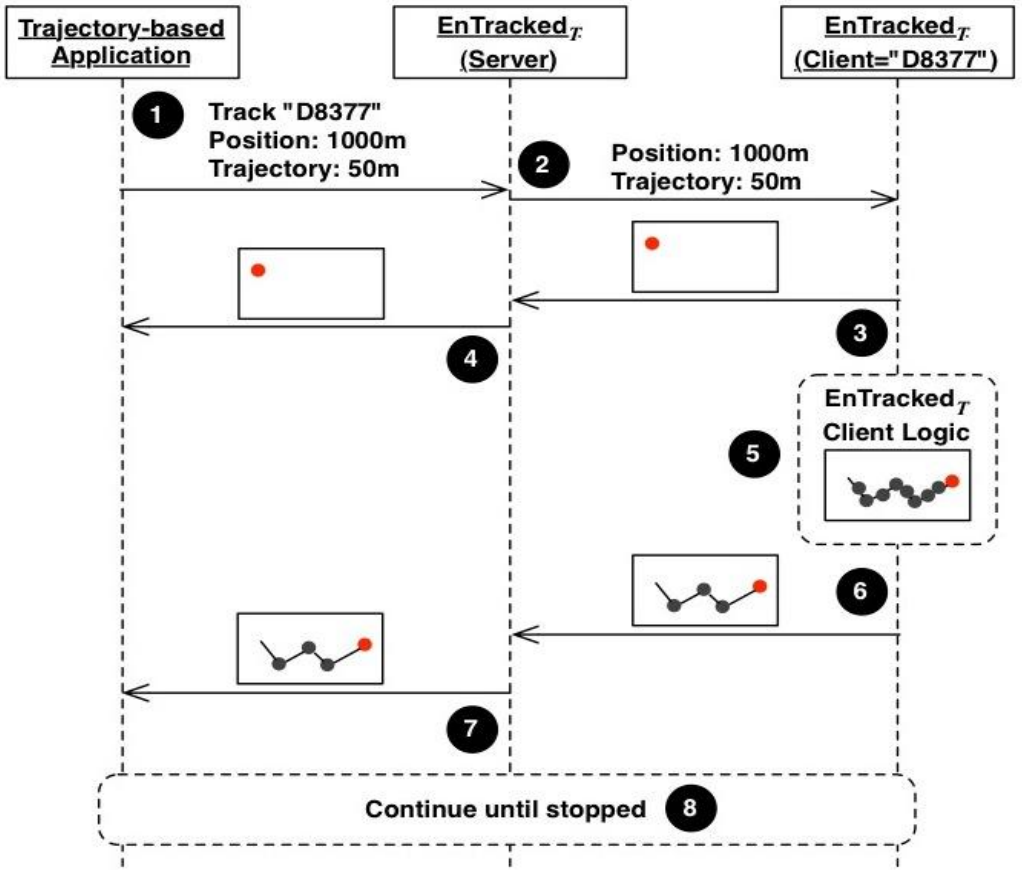


Figure 3: EnTracked$_T$ Communication.

Sensor management strategies

- the system implements various sensor management strategies, suitable for different task profiles
- on continuous basis, EnTracked$_T$ client:
  - asks strategies, given current requirements, how much power they are estimated to use on average
  - selects the strategy with the lowest estimated power consumption

Client-side control flow

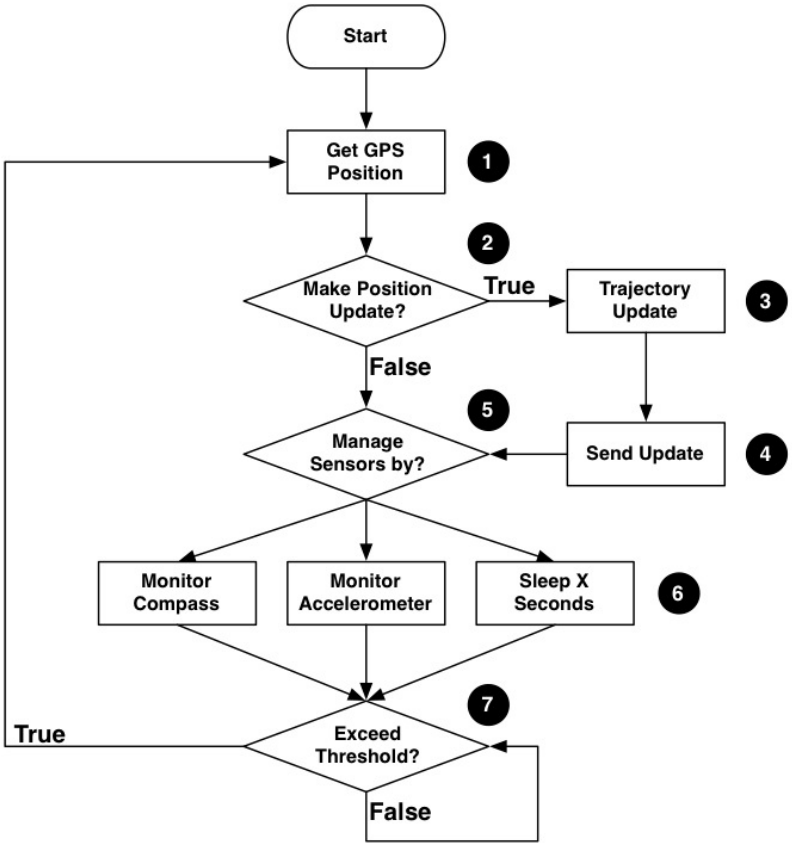


Figure 4: Flow of control

Sensor management strategies 2

- the current version of EnTracked can select from the following strategies:
  - heading-aware (uses compass as a turning point sensor)
  - distance-aware (predicts how long the GPS can sleep between measurements)
  - movement aware (uses accelerometer to detect stationary periods)
- while estimating the power consumption the strategies take into account platform specific variations in power consumption

Heading-aware strategy

- power consumption estimates depend on the level of duty cycling and power consumption of compass
- the idea is that no explicit update of target's position and trajectory is needed as long as the target is moving in a straight line with constant speed

Heading-aware strategy 2

- to sense when to request a position update, we compare the prescribed trajectory error threshold with the distance orthogonal to the initial heading given by the compass

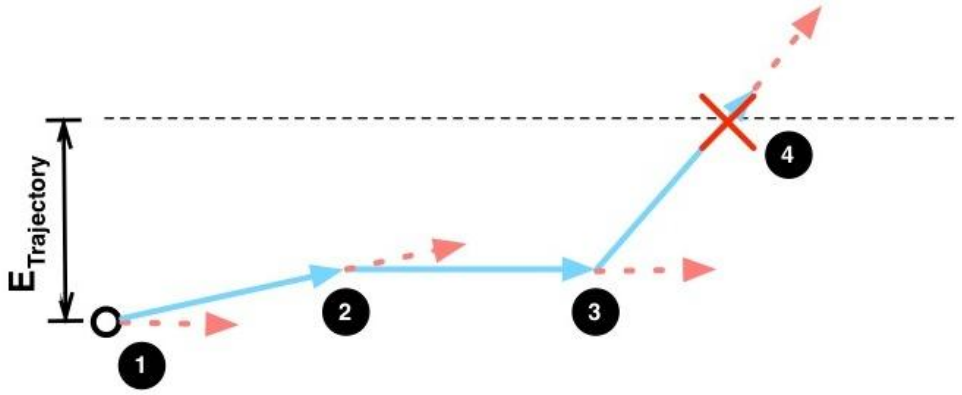


Figure 5: Heading deviations will increase the orthogonal distance beyond the threshold and force the GPS position to be updated.

Heading-aware strategy 3

- previous work aimed to use compass for inertial dead reckoning
  - compass has to be precisely calibrated
  - tracked object has to keep the compass in a specific orientation towards it's body
- this approach only aims that relative position of compass towards the tracked object is constant
  - even if this does not hold no errors are being introduced, only position updates will be premature

Heading-aware strategy 4

- some formalism:
  - we calculate accumulated orthogonal distance D_{orth}.
  - this distance at time t_k depends on the estimated speed s_{gps}, initial heading θ_{start}, measured heading θ_k and average error of compass σ
  - formula for calculating accumulated orthogonal distance:

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

- whenever trajectory error threshold is exceeded, new GPS position is requested

Heading aware strategy 5

- problem:
  - sampling frequency of in-phone compasses is too high
  - EnTracked$_T$ system decreases sampling frequency to save energy => speed changes and precise headings between measurements are unknown
  - this deviation leads to additional uncertainity
  - we calculate the period between samplings with the following formula: $$\Delta t = \frac{(1-u) \cdot E_{trajectory}}{s_{gps}}$$
- additional information about motion patterns can be considered to improve accuracy

Distance-aware strategy

- power consumption depend on sleep periods and power consumption of GPS
- idea:
  - strategy predicts sleep periods of the GPS given target's speed pattern and prescribed error threshold for both trajectory and position tracking
  - to minimize power consumption and to calculate sleep periods robustly, delays resulting from powering features on and off need to be taken into account
- suitable for high trajectory error thresholds and objects moving with a low speed

Movement-aware strategy

- power consumption estimate depend on duty-cycling and power consumption of accelerometer
- idea:
  - accelerometers can be used to detect movement and to reduce when GPS is sampled
- well suited for pedestrians, but not bikes or cars
  - objects moving in the straight line with a constant speed do not provide enough indication of the tracked object being stationary

Trajectory update protocols

- task of trajectory update protocol is to communicate motion information obtained on a mobile in an incremental fashion, when considered suitable by the protocol
- in order to save energy and data transmission it is natural to simplify the results and send only the "relevant" subset of it
- "relevant" means it is minimal in size, while still reflecting overall motion history

Definitions

- trajectory *a* is a continuous, piecewise linear function
  - let {a1, a2, ..., an} denote a sequence of measurements, holding position (ai.p) and timestamp (ai.t)
  - a spatio-temporal segment of this function is defined as follows:

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

- this function is defined as:

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

Trajectory simplification

- algorithmically, trajectory simplification is just a special case of line simplification
- in the line simplification task, the goal is to select a subset of the original polyline, so that the polyline does not deviate from prescribed numeric error threshold

Trajectory simplification 2

- in trajectory simplification task this threshold is defined with regards to time uniform distance:

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

- $a_c = (x_c, y_c, t_c)$ is the unique point on the segment a_i-a_j with the same timestamp as a_m

Douglas-Peucker algorithm

- follows divide-and-conquer paradigm
- starts with single line segment between the first and the last point of the original trajectory *T*
- then it identifies the point of *T* which is the farthest away from this line segment
- if its distance violates the error threshold it is added to simplification and algorithm is being called recursively to the both newly generated segments
- algorithm terminates if there's no such point

Simple Section Heuristic

- alternative to Douglas-Peucker, iterative
- starts by adding the oldest point to simplification
- then it iteratively probes subsequent points, until it finds the last point so that line from the starting point to this point does not violate trajectory error threshold. The next point becomes the new starting point and iterates until all points have been processed
- to improve the results we can remove points that are "obviously good"

Results of simplification

- both algorithms are not optimal
- however they achieve reduction efficiency of 80-90% of the optimal algorithm (Section Heuristic being better)
- the optimal algorithm does not have a practical application since it requires much more computation for just a bit better results

Energy efficiency of simplification

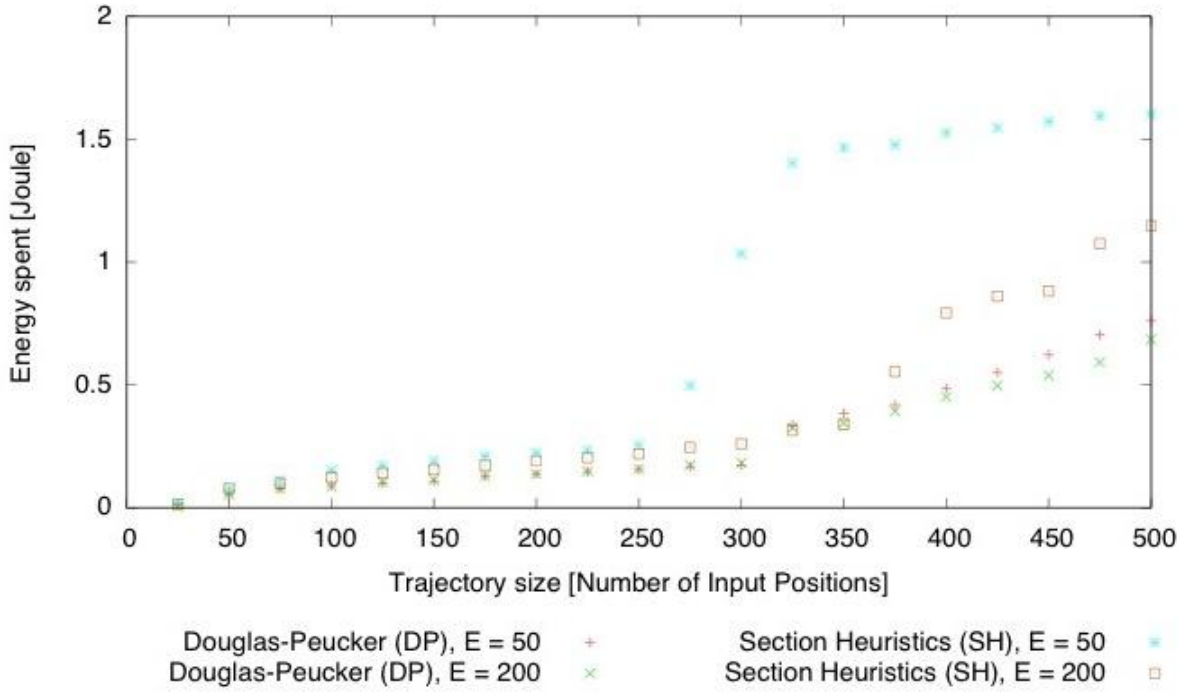


Figure 7: Power consumption of simplification algorithms run on commuting data of various sizes and with error thresholds of 50m and 200m, resp.

Energy efficiency of simplification 2

- asymptotically, power consumption behaves in the same way as time complexity for both algorithms (worst-case $O(n^2)$)
- tests on real-world data provide some better results
- costs invested in trajectory simplification pay-off in the overall energy budget

System architecture and technology

- client implemented for Symbian S60 in Python
- two layers:
  - strategy and protocol layer
  - client engine layer (system logic and platform integration)
- server implemented in Java on top of OSGI engine as part of PerPos Platform

System architecture and technology

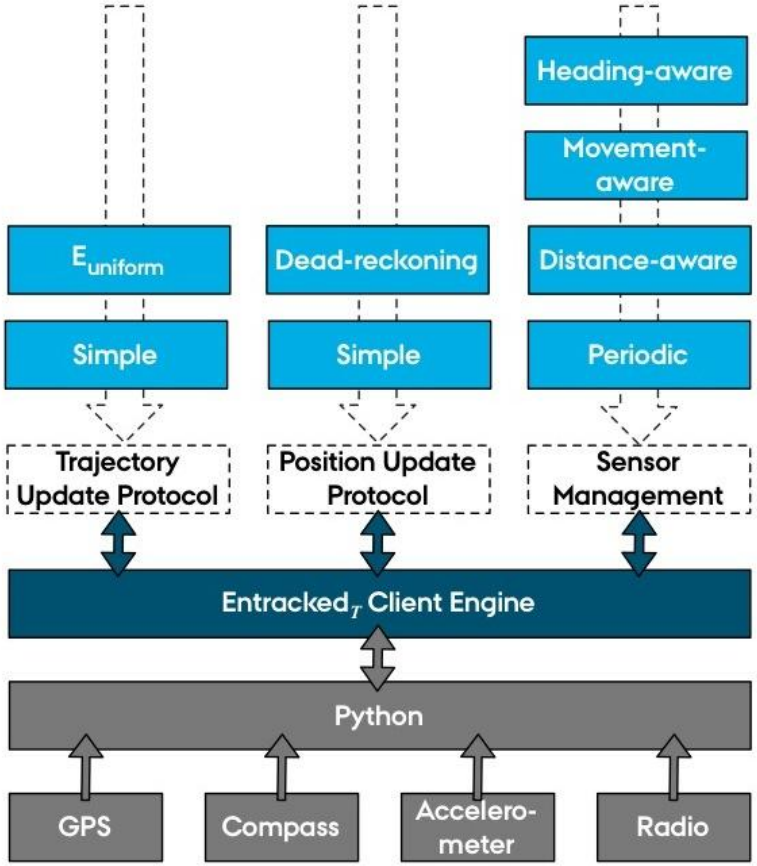
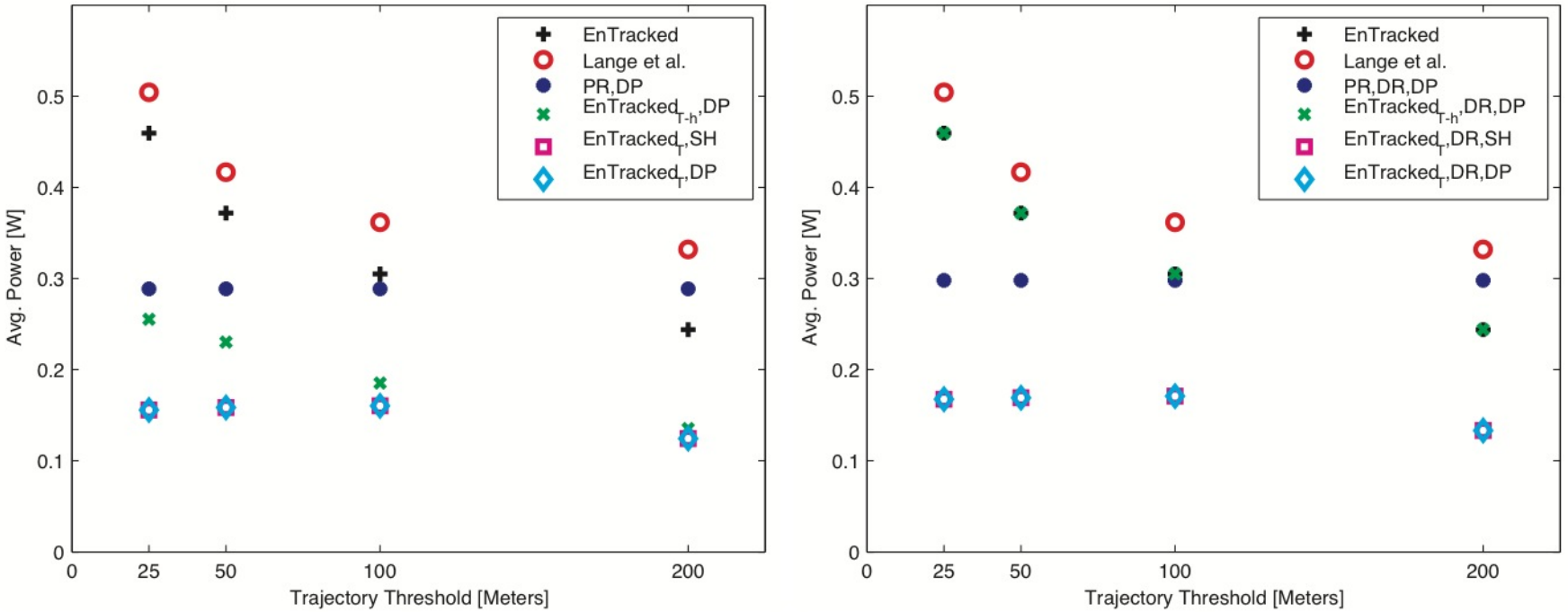


Figure 8: Software architecture of the EnTracked$_T$ client engine

Emulation experiments

- testing platform emulates Nokia N97 power profile measured with Nokia Energy Profiler
- large models gathered in a real life (during walking, running, biking and car driving)
- nearly every possible configuration of the system has been tested (simplifying algorithms, strategies etc.)

Power consumption



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

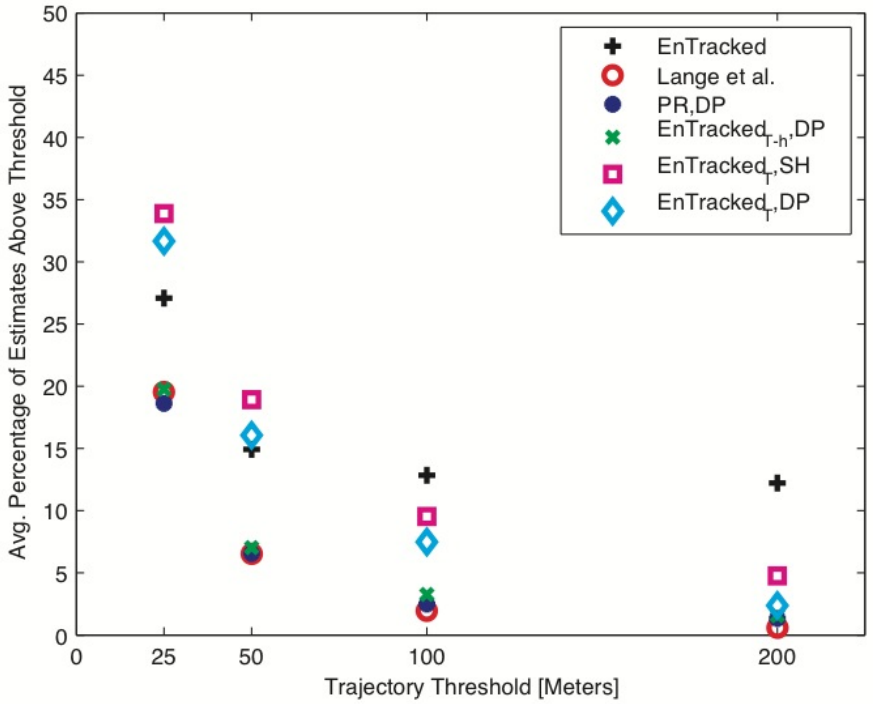
(b) Variation of average power consumption with trajectory threshold $E_{Trajectory}$ when position error $E_{Position}$ is requested to be within 1000 meters.

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

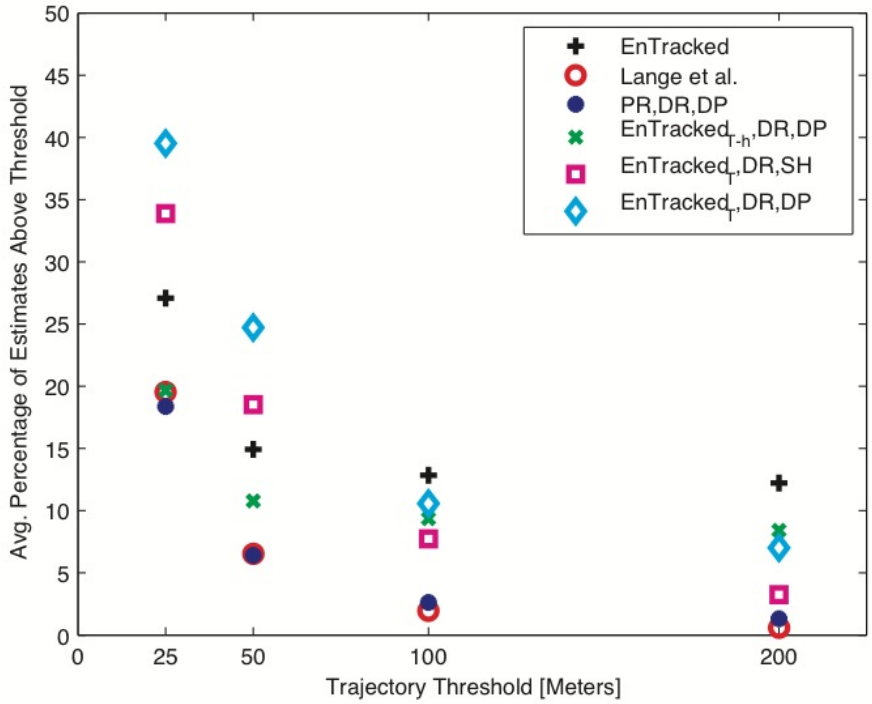
Robustness of tracking

- the use of intelligent sensor management strategies and trajectory simplification algorithms potentially reduces the accuracy
- simplification obeys a prescribed error, but only w.r.t. the original tracked trajectory, which may increase the deviations
- following figure shows simplified trajectories w.r.t ground truth

Robustness of tracking



(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.

Sensor specific power consumption

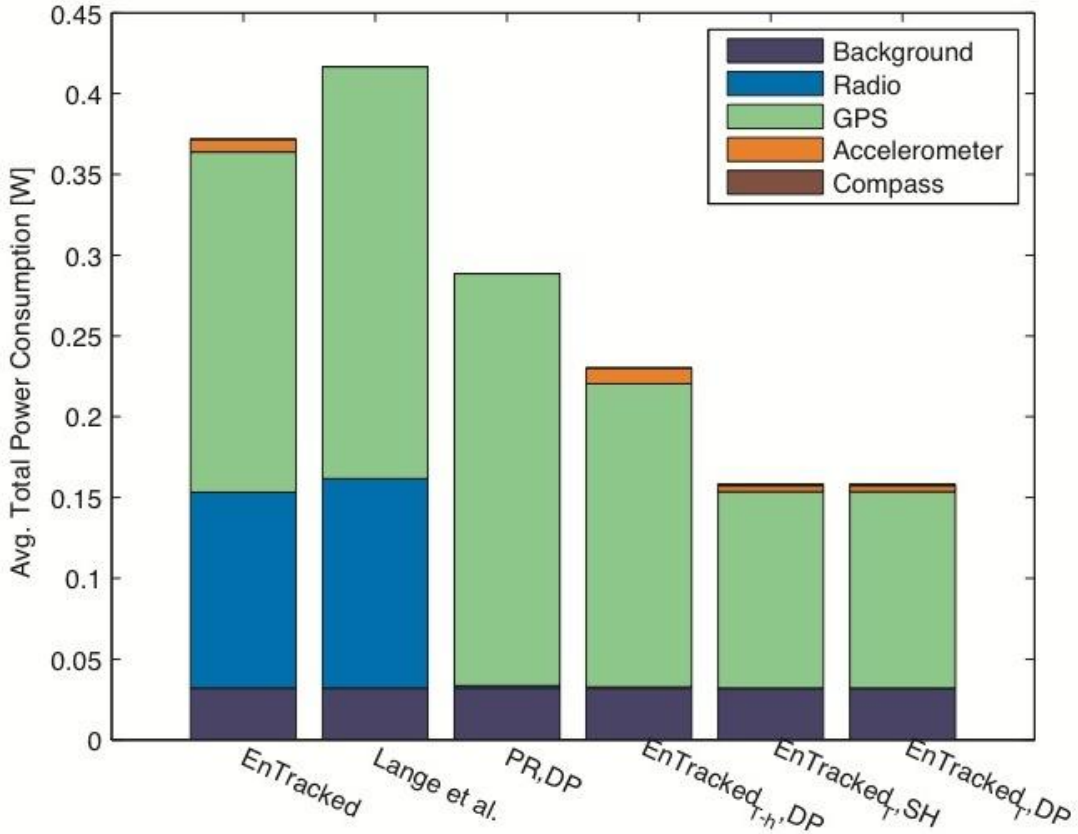


Figure 11: Breakdown of average power consumption with respect to the individual sensors.

Sensor specific power consumption

- experiment allowed trajectory error threshold of 50m
- GPS is the most power consuming sensor
- compass and accelerometer readings are very cheap and can contribute to significant decrease in GPS usage (in heading-aware and movement-aware strategy)
- even periodic strategy saves energy due to minimal use of radio (trajectories are uploaded only if trajectory buffer overflow occurs)

Movement specific power consumption

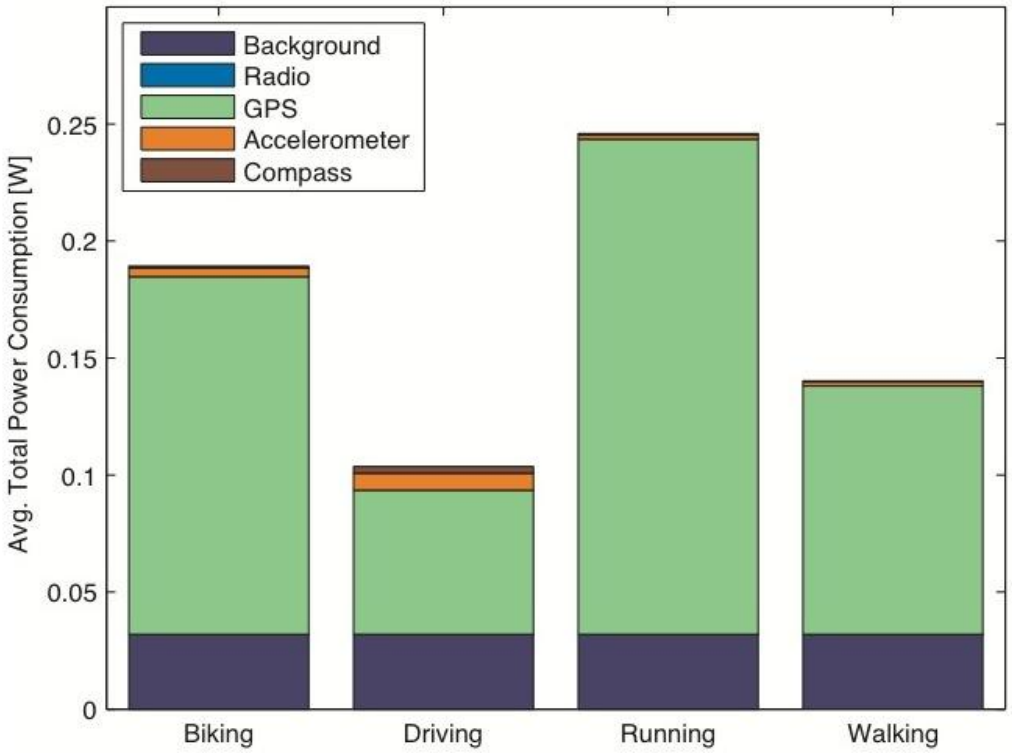
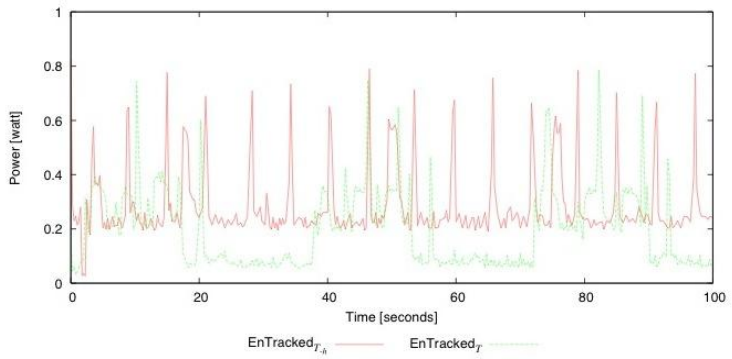


Figure 12: Average power consumption and break down of power in case of EnTracked$_T$, DP with respect to movement styles.

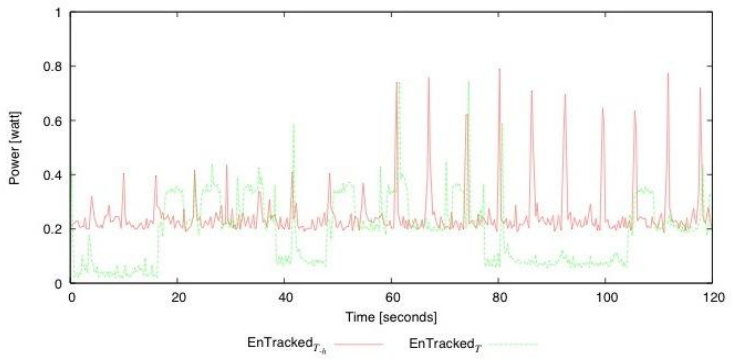
Real world deployment

- in a real world, there are certain conditions that can potentially have some serious impact on the system's performance (network performance etc.)
- system has been deployed on two Nokia N97 smartphones and monitored with Nokia Energy Profiler
- tests conducted with two versions of the system: with- and without heading-aware strategy
- 1km walk, 12km drive

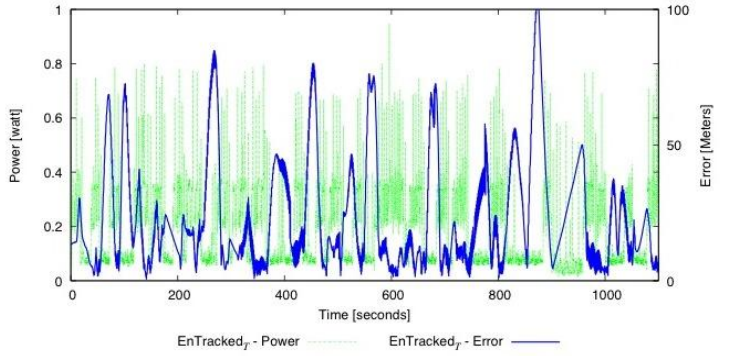
Real world deployment



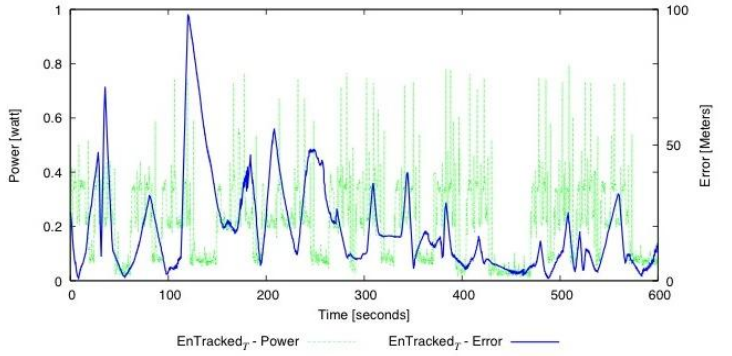
(a) Comparison of power consumption between EnTracked$_T$ and EnTracked$_{T_{-\mathrm{h}}}$ during driving using a trajectory threshold of 50 meters.



(b) Comparison of power consumption between EnTracked$_{T_{-\mathrm{h}}}$ 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.

Related work

- RAPS system focuses on position tracking in urban areas, where GPS tends to be inaccurate
  - it uses GSM data to predict when GPS is likely to cause some problems and as a result allowing greater power savings
- Zhuang's work study the problem when several location-based applications are running at the same time
  - considers using Bluetooth to exchange sensor data to improve battery life
  - it also takes advantage of movement patterns

Conclusions

- EnTracked$_T$ system allows lower power consumption with reasonably good accuracy
- using sensors like compass and accelerometer for duty-cycling GPS gives a pretty good amount of power savings
- emulation and initial real-world tests prove efficiency of the proposed solution
- the more energy efficient strategy, the greater chance to violate prescribed error threshold

The end