Low cost positioning by matching altitude readings with crowd-sourced route data

Sharmistha Chatterjee, Jukka.K.Nurminen and Matti Siekkinen
Aalto University School of Science
Helsinki, Finland
sharmi.chatterjee@gmail.com, jukka.k.nurminen@aalto.fi, matti.siekkinen@aalto.fi

ABSTRACT

Detecting and tracking the position of a mobile user is an increasingly important feature in many mobile applications. In this work we study how cheap and energy-efficient air pressure sensors measuring the altitude could be used, as a complement to the dominant GPS system. The cornerstone of our approach is that a huge amount of route data, collected with GPS devices, is available in various cloud services. The location detection and route tracking task thus becomes a question of matching the collected altitude traces with the altitude curves of stored data to find the best matching routes. Here we build a prototype system of crowd-sourced database containing only altitude data. How accurately this stored altitude data could be matched with the collected altitude traces is the key question of our study.

Keywords

GPS, power, sensor, pressure, altitude, accelerometer

1. INTRODUCTION

The GPS system is widely used to detect the position of a device. However, one of its problems is the relatively high energy consumption [10], [13]. Moreover, it is not usable indoors where the satellite signals are not visible. Alternative positioning methods have therefore attracted a lot of attention. Particularly interesting are solutions which use low cost sensors with small energy consumption such as accelerometer, magnetometer (compass), and air pressure sensors [13], [15]. They allow position detection and route tracking with small energy consumption. Moreover, it is possible to embed such sensors in simple and cheap devices such as entry level mobile phones, fitness watches, or toys.

A concrete example of the potential of such approach is that today many heart rates monitors are able to record the altitude curve of a training route. Moreover, people are eager to share their exercise data in various social services creating thus a crowd-sourced database of potential exercise routes. By matching a collected altitude curve with previously stored routes it would be possible to find out which route the person took. The problem is simplified by the fact that most routes follow paths or streets reducing the number of alternatives dramatically. However, difficulties arise e.g. from the differences in speed (especially when different forms of movement are involved: walking, running, biking), from the effect of route traversal direction, and from the daily variations in air pressure.

In this work we study the feasibility of finding the traversed routes by matching altitude data with stored route information in a crowd-sourced database. In particular, the key contributions of our work are:

- We propose a concept of using crowd-sourced route data to track the movement of a mobile device with an altitude sensor. In addition to the simpler case of detecting the route after it has been completed we also sketch the possibilities to use similar approach on the move.
- We study the feasibility and accuracy of the approach by analyzing the characteristics of a number of stored routes and derive values for estimation parameters with that data. In addition to matching full routes we analyze the matching of partial routes and of routes consisting of multiple sub-routes.
- We perform an extensive measurement study and experiment with traces collected via walking, biking and riding a bus in different geographical areas. We also study how the speed of movement, means of transportation, and route traversal direction influences the matching accuracy.
- We discuss how such an approach could be used to save battery on a GPS enabled smartphone by allowing GPS to be switched off when the user is following a route found from the crowd-sourced database
- We envision how such a system could be used and propose ideas for further research

The rest of the paper is structured as follows. In section 2 we discuss the background and related work. In section 3 we describe our approach in detail and section 4 provides the details of the matching algorithm. Section 5 describes the results of our trace analysis and section 6 the application of the matching to a set of use cases. In section 7 we discuss the applicability of our approach and the new opportunities it creates. Finally, we conclude in section 8.

2. RELATED WORK

To overcome the limitations of GPS a lot of alternative solutions have been studied. Some of them try to minimize

MoMM2012, 3-5 December, 2012, Bali, Indonesia.

the time GPS needs to be active and some complement earlier collected GPS data. The most relevant related approach are summarized below.

User activity patterns using sensors Research work in the past has been able to determine user-movement using a duty-cycled accelerometer [13], [15], [17], [22]. Trajectory tracking algorithms [12] detect directional movements of a mobile with compass and sensors. In [17], [22], total time duration of user's movement, time period during which he is driving, stationary, running, walking [14] and his velocity at different times are useful in learning a user's activity pattern. The objective behind this technique is to turn on the GPS when the match to previously learnt space-time history of user activity falls below a certain threshold [6].

Experiments evaluated on Android-enabled cell phones with accelerometer and compass have been able to provide location estimates, where GPS fix determined the initial position and velocity. These readings suffered a linear loss in accuracy compared to GPS accuracy in highways and intracity driving environments [23].

User activity patterns using Celltower-id User movement can also be predicted using trajectory mapping mechanisms where cellular (GSM) fingerprints can be used to estimate the highest probable trajectory (sequence of map segments) followed by the mobile device [10], [18], [21]. The GSM fingerprint collecting application, known as CTrack takes into account uncertainity in path transition of a mobile user and records all the GSM tower observations during its movement. This procedure known as celltower-RSS blacklisting detects GPS unavailability (e.g. indoors) and avoids turning on GPS in these places. The GSM fingerprints (containing current celltower ID and the received signal strength (RSS) [17] without any GPS fixes) are directly matched to a map. The map matching algorithm converts the whole space as grid cells and determines the mostly likely traversed grid cell and its corresponding road segments [21].

Sharing GPS information using Bluetooth Bluetooth communication [10], [17] has served one of the means in reducing position uncertainty among neighbouring devices. New position updates are broadcasted to neighbours (with or without GPS functionality) who can obtain this measure and save a GPS query. Neighbours also send their own position estimates to other peers. Such peers learn this new update when this estimate has a lower accuracy value. This helps all neighbouring GPS devices in close proximity to synchronise themselves with GPS positions of least accuracy.

In contrast to most prior approach our idea is centered on the crowd-sourced database of routes taken earlier. Such databases already exists for e.g. sharing exercise data and therefore our approach could be directly taken into use without a special effort to create a route database. An example of such already existing database is Sports tracker [3], [5]. It is an exercise tracking application that uses the mobile phone's GPS readings and, optionally, heart-rate monitor to track outdoor user activities like speed, position, heart rate etc. It then broadcasts the details to a social cloud service where spectators can view online the live events of every sportsman.

3. SYSTEM ARCHITECTURE AND OPERATION

Our system architecture consists of the following key components:

1. **Crowd-sourced route database on the cloud** In the simplest case this is centralized server hosting the database of earlier routes. If the proper APIs were in place, databases available in existing services, such as Sports tracker, could be used.
2. **GPS enabled mobile device** The GPS enabled mobile devices, typically smartphones, are used to populate the route database with the GPS traces. They could keep the GPS on all the time or activate it only when needed, that is, when no matching route is found in the database. The device could continuously access and update the route database, or it could do the update offline after the activity, using e.g. a proximity connection at home.
3. **Mobile device with altitude sensors** These devices include heart rate monitors, fitness watches and also entry level phones with no GPS functionality. The essential functionality is to collect and store the altitude traces during movement. If the device is able to perform wide area wireless communication it can access the route database on the move allowing realtime route discovery and location detection. Otherwise the stored data could be analyzed offline in a batch mode after the completion of the activity.

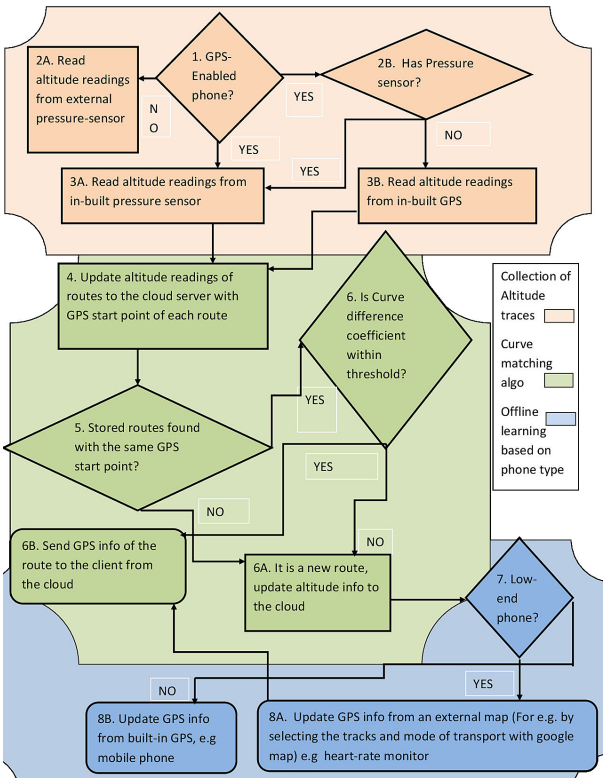

Figure 1: Prototype System Implementation showing altitude trace collection (Pink), route matching algo (Green) and offline learning (Blue)

Figure 1 illustrates the key steps of the system operation. In this paper we focus on offline route matching. However matching techniques can also be extended to online matching where network connected devices, like smartphones, on the move send altitude readings of partial routes to the cloud and retrieve GPS position information of discovered routes. Here online altitude matching enables the positioning on already discovered routes while undiscovered routes still requires the use of GPS.

For GPS un-enabled devices the altitude information is obtained from an atmospheric pressure sensor (Step 2A) and fed to the cloud offline at the end of the day in a batch mode (Step 4). For GPS-enabled phones, the use of an in-built pressure sensor is recommended as it reduces the power consumption (Step 3A). For smartphones without pressure sensor, we consider only the altitude fixes of in-built GPS to emulate the altitude readings as readings of a pressure sensor (Step 3B). The cloud is also provided with the GPS start position and direction externally so that matching subset is confined to stored routes where GPS start direction coincides. This start direction is obtained by receiving the initial GPS fixes for 10 - 12 secs at all route origins.

The cloud matches the input altitude trace to all of the visited altitude traces stored in the database with the same GPS start direction (Step 5, Step 6). If the coefficient of matching is within a certain range compared to any of the stored routes, the cloud concludes the mobile user travels along one of the previously visited routes. Otherwise it considers it as a completely new route and updates the database with the altitude data of the new route (Step 6A). Furthermore, low-end phones (without GPS) obtain the GPS coordinates of new undiscovered routes from the cloud (Step 6B). Here the cloud is updated with the same GPS coordinates externally by a map service like Google map (Step 8A) for any given mode of transport (riding a bus or biking or walking). On the other hand GPS enabled phones can directly obtain GPS route information from its built-in GPS (Step 8B). This information trains the cloud with the altitude data and GPS positions of the new route. It can later send this information to requesting clients (Step 6B).

In this paper we focus our study to different use cases of altitude matching to reduce the GPS activity. The next sub-section discusses these scenarios.

3.1 Use cases of Altitude Matching

In this section we evaluate the applications of altitude matching for a number of use-cases under different modes of transport, movement speed and atmospheric pressure (Use-case 4). These use-cases are analyzed by carrying out a post-mortem route analysis on the collected field traces. Such route analysis capabilities include matching complete, partial and reversed curves with a curve matching algorithm (Use-cases 1, 2, 3).

1. **Complete Route Matching** Matching altitude curves can be applied to match two complete routes. Figure 9 is an example of complete route matching of a definite biking route without any active GPS. This type of matching is used in the context of the system to match two curves with same GPS start direction, one of which is frequently traversed and stored in the database and the other route is newly traversed.

2. **Partial Route Matching** For lengthy routes, it also becomes essential to match segments of one complete route and one incomplete route. Figure 3 shows partial route matching on three different days of a definite bus route. Partial matching detects deviation in user-movement quickly from the course of frequently traversed routes. Here, in case of online matching the GPS gets turned on automatically (in GPS enabled phones) when the mobile user deviates from the most frequently traversed route.

   Partial route matching can be further extended to situations (like **Matching multiple partial routes**) when the route taken by a mobile user falls into multiple sub-routes of previously stored routes in the database. Figure 7 of Section 6.2 explains the algorithm when the current route of the user fails, the database tries to find the intersections of the current route with the existing stored routes in it. Figure 2(b) describes this condition when a known start direction is provided to the cloud and section of the current route no longer matches with the sub-routes (previously visited) stored in the database. Figure 3 more clearly presents the elevation of a complete route (consisting of 5 sub-routes of which one sub-route is of Track_1).

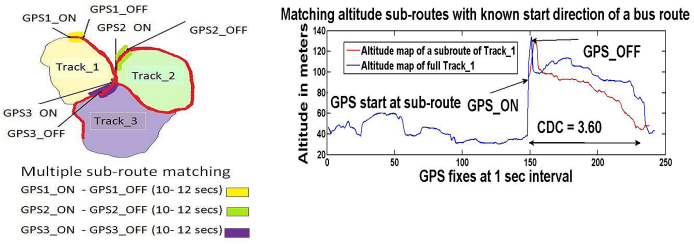

Figure 2: Coverage of multiple sub-routes, with one sub-route matched to a complete known route

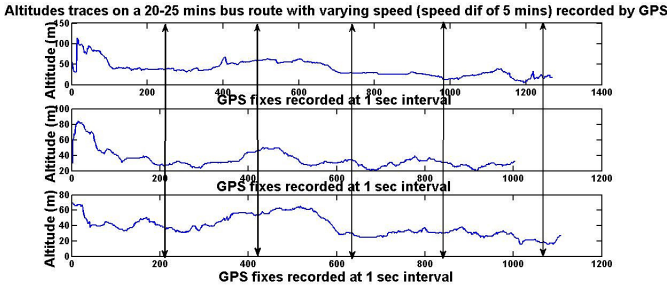

Figure 3: Partial matching

The red lines of Figure 2(a) presents three different sub-routes of three complete known routes covered by a mobile user. A known start direction (obtained either through a GPS fix or an external map) is supplied to the cloud when the sub-route for each of Track_1, Track_2 and Track_3 begins and ends. For online matching where the mobile user communicates with the cloud at frequent intervals (like Sports Tracker), the GPS_2_ON and GPS_2_OFF denote GPS ON and OFF instants at the beginning of the Track_2. On finishing Track_2 a start direction for Track_3 is provided to the cloud for finding its intersection point with any of the stored routes. For a GPS-enabled mobile phone using online matching, GPS active interval is denoted by the sum of time interval between GPS_1_ON and GPS_1_OFF, GPS_2_ON and GPS_2_OFF, GPS_3_ON and GPS_3_OFF points. As Track_1, Track_2 and Track_3 happens to be one of the previously visited sub-routes stored at the cloud, the position coordinates are retrieved for these tracks. The algorithm used for matching sub-routes is described below in Figure 7.

3. **Reversed Route Matching**

   The altitude traces of any frequently visited route during its upward course of journey (from source A to destination B) should be ideally mirror image of its downward course (from source B to destination A) of journey. Here we analyze the performance of match-

ing altitude curves along the same route traversed in opposite directions.

4. **Matching curves with different speed and atmospheric pressure** Altitude traces obtained on different days with varying speed and atmospheric pressure are compared to analyze how much they differ.

4. CURVE MATCHING ALGORITHM

The Curve Matching Algorithm [11] in this paper compares 'two altitude curves', the known route being stored in the cloud and the unknown route being recently completed by the mobile user during the course of the day (offline matching). The application of the algorithm in different circumstances are discussed below

1. To find the best matching route from the database by matching the two altitude curves.

2. To find the best matching sub-route from the database, when the user covers sub-section of a given route stored in the database.

We use Sethian's Fast Marching [11] Method to compute the dissimilarity function (F(t, s)) of two curves C_1(t) and C_2(s) at given points t and s. It measures the shape difference between the two curves. The difference in distance is measured by taking the least sum of the local dissimilarities between individual pairs of curve points. For example $C_1(t) = (x_1(t), y_1(t))$, $t \in [0, m]$ and $C_2(s) = (x_2(s), y_2(s))$, $s \in [0, n]$, represent two curves C_1 and C_2. Of the other parameters m, n denote the lengths of the respective curves. When the end-points of the two curves coincide, the local dissimilarity function F is a path c through t, s-space from (0, 0) to (m, n) such that

$$T(m, n) = \min_c \int_c f(c(\tau))\, d\tau \tag{1}$$

The weighted distance function T(m, n) is the Curve Difference Coefficient (CDC) of the two curves $f(c(\tau))$. It represents the minimal cost function required to travel from any given point to the point (t,s) in $\mathbb{R}$. The weighted distance is calculated by incorporating a weight factor to all paths between the start and end points of the arc. The minimum weighted Euclidean distance of the arc-length is then used as a metric to differentiate the two curves optimally. The distance between C_1 and C_2 can be represented as

$$d(C_1, C_2) = T(m, n) - \lambda \times \sqrt{m^2 + n^2} + \left| 1 - \frac{\min(m, n)}{\max(m, n)} \right| \tag{2}$$

where path represents the minimal cost function between each pair of curve points. Here λ represents smoothing constant such that $\lambda > 0$. The smoothing constant helps in running the algorithm when the two curves have negligible differences in their curvatures. Here λ, the smoothening constant is chosen as 0.5 so that the behaviour of the algorithm is neither too opportunist or conservative. The second term in the expression is a normalizing factor while the third term in the expression subdues stretching or bending effect of the curves, minimizes curve difference when curves are matched on non-uniformly spaced grids. Among time-series algorithms Dynamic Time Wrapping Algorithm [19] is well known for minimizing scaling or shifting variations in time but its time and space complexity is $O(N^2)$. Hence we choose Sethian's Fast Marching algorithm (complexity O(N logN)) as a good metric in matching curves based on shape, ignoring the width and height difference of the curves under consideration [11]. This concept is more illustrated in the Use-case 4 described below.

5. EXPERIMENTAL SETUP

Our system setup consists of a 1) A GPS-enabled Smartphone (N8) 2) A pressure sensor from a heart-rate monitor (Polar 650X) 3) A server or cloud. The cloud stores the known routes in a database and runs a matlab program for route matching computations. We used a heart-rate monitor initially in the experiment to explore how altitude information obtained from its pressure sensor varies for the same route on different days during biking. As the smartphone N8 does not contain a pressure sensor, we emulated N8 as a heart-rate monitor by considering only the altitude fix obtained from its in-built GPS. For positions with same GPS coordinates, the altitude fix with least vertical accuracy is considered. The same experiment was repeated with mode of transport as walking and riding a bus.

5.1 Trace Collection

Our initial objective is to study the barometric pressure readings for a given route on different days to understand the extent of similarity or dissimilarity of the altitude traces. It also helps us to understand the importance of a pressure sensor in route exploration. The statistical analysis of the traces leads to the design of the decision making system at the cloud. In the next section of the paper we analyze the behaviour of the algorithm for different use-cases.

We collected repeated altitude traces along a given route in two opposite directions (to and fro from source to destination) slightly more than a month (34 different days). The readings are recorded while biking by a pressure sensor of a heart-rate monitor along a given route (10 km) in both the directions. The average biking speed was 20 - 25 km/h. Each of these 34 day's curve is then permuted with other 33 day's curve and the curve difference is then calculated between each pair using Sethian's Fast Marching Algorithm. We closely analyzed the results between all the matched pairs i.e. $\frac{34*33}{2}$ comparison results for each direction.

We also took repeated altitude traces along 3 different routes both by walking (3 - 5 km/hr) and 2 different routes by riding a bus. However we chose to limit the comparison set to a smaller value of 20 days. Each of these 20 day's curve is then permuted with other 19 day's curve and the curve difference is then calculated between each pair using the same algorithm. We closely analyzed the results between all the matched pairs i.e. $\frac{20*19}{2}$ results for each route each direction separately for 2 bus and 3 walking routes. All the traces collected during the initial phase are used for learning the altitude curve distribution of different routes.

Further separate traces are collected, half the same number (16 days for cycling, 10 days for each of bus riding and walking) along the same routes to evaluate our test results. Figure 5 presents traces collected with Nokia N8 in Espoo Helsinki region with elevation level ranging from 0 - 74 feet/0 - 22.5 meters above sea-level. The region has population standard deviation (SD) of 5 meters (obtained from Google Map [1]. The Figure 5 in the right also shows vertical accuracy distribution of N8 as approximately 100 meters.

5.2 Trace Analysis

The traces collected following a bus route, biking, and walking are analyzed independently and CDC is calculated between all the matched pairs along a given route. The server logic is designed to have a selection criteria based on mean CDC to decide if the currently visited route of the

mobile user is one among the stored routes in the database.

We had $\frac{34*33}{2}$ different comparison results for biking and $\frac{20*19}{2}$ different comparison results for each of bus route and walking route. Here on comparison of the dissimilarity function (CDC), between every matched pair of the altitude curve along the same route we found that it is normally distributed. This is further illustrated by histogram in Figure 4(b) presenting the permuted comparison results for one direction. As the CDC of the two different bus routes (first 2 rows of Table 1) are quite close and the distribution function looks quite similar, here we represent only one route. The histogram demonstrates the curve difference distribution function of a given sub-route of bus with 180 unique GPS points. The sub-route (1.35 - 1.5 km) was covered on different days within 4 - 5 mins interval with the bus traveling approximately at a speed of 30 - 32 km/hour. However the choice of the sub-route length has been explained in the context of partial matching and system performance. The three central bars constituting 81% lie within a range of $\pm 10\%$ calculated from the mean of $\frac{20*19}{2}$ different comparisons. The rest 19% of the CDCs is spread across (3 left and 2 right bars) both sides of the central bars. We also used matlab test metric normplot [4] to fit the altitude distribution to a normal distribution. Figure 4(a) shows the data with a blue plot and symbol '+'. The linearity of the plot confirms the distribution could be a form of normal distribution.

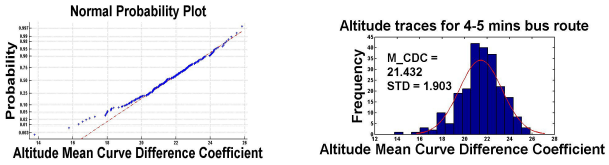

Figure 4: Histogram and Fitness to a Normal Distribution

Further we took 20 days' collected traces and analyzed the ratio between the CDC and the Mean Curve difference Coefficient (M_CDC) between every permuted matched pair and plotted them in a cumulative distribution function (CDF) in Figure 6. The ratio of CDFs for walking, biking and riding a bus along a given sub-route are represented by plotting them in the same graph with three different colors. All the sub-routes are of the same travel time of 5 mins. In the CDF, the Y axis represents the percentages of these ratios (0 to 1 and above) while the X axis represents the ratio between Actual CDC and Mean CDC.

$$XAxis_r = \frac{Actual\_CDC}{Mean\_CDC}$$

It is found that all the curves follow similar trend and the walking curve traveling with the slowest speed (3 - 5 km/hr) has the least deviation (with 25% values exceeding 1.10 and 93.3% of values exceeding ratio of 0.7). The curve obtained from traces collected following a bus route has the largest deviation (with 29% values exceeding 1.10 and 73.8% of values exceeding ratio of 0.7). It also has the highest traveling speed (30 - 35 km/hr) and the largest distance covered. The curve obtained from biking lies midway between biking and riding the bus (with 28% values exceeding 1.10 and 83.7% of values exceeding ratio of 0.7). Biking speed falls midway between walking and bus speed (20 - 25 km/hr) and the distance covered by it also falls midway between bus route and walking. From these results we can safely conclude the CDCs of a definite route increases with higher traveling speed and larger distances traveled.

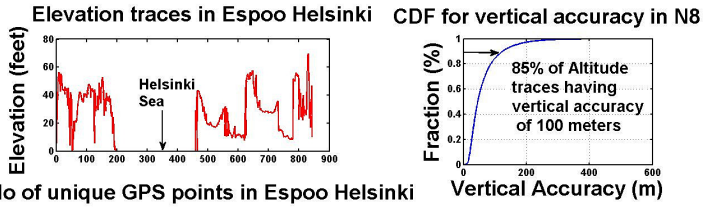

Figure 5: Distribution of Elevation traces (feet) and vertical accuracy of Nokia N8 in Espoo Helsinki

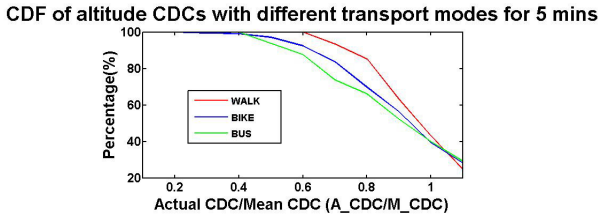

Figure 6: CDF of altitude CDCs along 3 definite sub-routes (5 mins), from Espoo-Helsinki

5.3 Tolerance Factor

Tolerance factor can be defined as an acceptable level of deviation between a newly matched curve and altitude curve of a previously learnt route stored in the database. The tolerance factor is added to the mean CDC of a frequently visited route to govern the flexibility/rigidity of the server's decision. This is then termed as Threshold of Curve Difference Coefficient (T_CDC). For a given mode of transport, the threshold factor for each route is calculated by adding a tolerance factor to it. The threshold limit is taken on the higher side of M_CDC (i.e. M_CDC + 10%). M_CDC - 10% ($\leq$ T_CDC) is assumed to lie within the acceptance range provided the start direction, transport mode of the two curves are the same and speed of travel is within 25%.

With a higher tolerance limit, the server is more opportunist in its prediction. The server predicts that the newly matched curve is one of the old curves stored in the database even when the CDC between them is considerably higher than the mean value. On the other hand the server is more conservative in its decision with a lower tolerance limit. Here on noticing the slightest deviation from CDC, it predicts the newly matched curve as a new route and learns the GPS route information. Here the tolerance limit sets the main criteria for the server to accept or reject the newly matched route as an old or new route respectively.

The threshold factor is decided by analysing the acceptance rates by the server when curves along the same route with the same transport mode are compared. Table 1 presents that for different modes of transport along a given route, the correctness of the server's prediction increases by increasing the percentage of the tolerance. For all the modes of

transport we find that the highest change in acceptance occurs when the tolerance percentage is set at 10. With the change of tolerance limit from 5% to 10%, maximum change is recorded in the acceptance rate (11%). But for most of the routes the change in acceptance rate for changing the tolerance limit from 10% to 15% is only 4 - 5%. Hence during learning phase we allowed a 10% tolerance level to the mean value to set the threshold for each route (T_CDC) along a fixed mode of transport.

Altitude CDC curve distribution fits well to a normal distribution (Figure 4(a)). So we estimated our results at 95% level of confidence. At this level of estimation, T_CDC for a given route falls within the range of $m \pm \frac{1.96 \times \sigma}{\sqrt{N}}$. This confidence level is used to evaluate the system performance, accuracy for different use-cases in the existing learning system. For a normal distribution like this, standard error (SE) will increase with degree of variation in elevation level at any region. For e.g rugged hilly terrains will have high CDC and therefore high SE. Our analysis shows lengthy bus routes of varying altitude levels have the highest SE of 1.2, followed by walking routes with SE of 0.28, while biking readings from pressure-sensor being more accurate SE decreases to 0.22. The lowest SE for biking is also due to more number of samples (34) than walking or bus-riding. This shows the system error value will decrease by providing more sample elevation traces at the crowd-sourced database.

The following Table 1 shows how acceptance rate changes with different limits of tolerance percentage during learning phase. The first column in the table shows the Mean Curve Difference Coefficient (M_CDC) calculated from the statistics of curves along a given route. The second column shows T_CDC after adding tolerance factor to (M_CDC). The number of unique GPS points (GPS1 and GPS2) vary due to varying speed. For biking we used a heart rate monitor without GPS. So GPS1 and GPS2 are set to 0 (columns 3 and 4). For a bus route the matching is done for a section of the bus route, each route having 180 unique GPS points.

Each row in the table denotes a new route repeatedly traversed by a mobile user along a given mode of transport (column 15). The tolerance percentages T1, T2, T3, T4, T5 present 0%, 5%. 10%, 15% and 20% tolerance level respectively. The acceptance percentages A1, A2, A3, A4 and A5 decribed in columns 6,8,10,12 and 14 state the acceptance level for tolerance limits T1, T2, T3, T4 and T5 respectively. We used complete route matching for walking or biking and partial route matching for matching lengthy routes e.g. while traveling in a bus (discussed in use-case 2).

6. USE-CASE RESULTS

In this section we discuss about the various use-case results of matching altitude curves in an offline mode without using GPS positions. The traces used in evaluating the results are collected separately in the same manner along different routes as collected during teaching the cloud. However the number of traces collected along each route is kept half the number collected during learning. The results help us to analyze the rate of success of the individual use-cases along different modes of transport. Further it also throws light on the improvements needed to build a larger scalable system.

6.1 Matching Complete Curves

The full length altitude curve of any newly traversed route is matched with complete routes stored in the cloud for a given mode of transport. Complete route matching offers greater potential when the speed of travel is less or barometric readings on different day differs slightly. From first

Table 1: Complete/Partial Matching showing Mean Curve Difference Coefficient (M_CDC), Threshold of Curve Difference Coefficient (T_CDC), No of unique GPS points in first (newly input) curve (G1), No of GPS points in second curve stored in the database(G2), Tolerance Percentage (T1, T2, T3, T4, T5), Acceptance % (A1, A2, A3, A4, A5), Mode of travel (M), Duration of travel in mins (D)

M_CDC	T_CDC	G1	G2	A1	A2	A3	A4	A5	M	D
22.5	25	180	180	54	54	67	70	71	Bus	4-5
21.43	23.57	180	180	49	71	88	96	99	Bus	4-5
14.54	16.0	100 to 120	100 to 120	52	73	84	88	90	Walk	8-10
7.46	8.20	45 to 65	45 to 65	57	67	75	77	80	Walk	4-5
6.27	6.90	35 to 45	35 to 45	47	60	78	84	94	Walk	3-4
7.2	7.93	0	0	64	67	72	75	78	Bike	25-31

column of Table 1) it is evident that biking has the least M_CDC. Or in other words, readings obtained from pressure sensor of heart-rate monitor along the same route more closely resemble each other. The last two rows of last column of Table 2 show the extent of variation of complete route matching for walking at 95% confidence.

In absence of pressure sensor in Nokia N8, the altitude readings obtained from the GPS are used for walking and tracking a bus route. It is seen from the first and the last column of the same table, CDC during walking becomes twice with the traveling duration becoming two-fold. Complete route matching turns inefficient when the mode of transport is bus. Here we find the performance of the cloud system suffers tremendously on matching a complete bus route of 20 - 25 mins (explained in Section 7.1). Hence we choose to use partial matching to match sub-routes of complete bus routes.

6.2 Matching Partial Curves

Table 2: Partial Matching with Threshold of Curve Difference Coefficient (T_CDC), No of unique GPS points in curve1 (GPS1), No of unique GPS points in curve2 stored at the cloud (GPS2), Distance between start and end points in meters (D), Travel Mode (M), Accuracy (A), Standard Deviation (SD)

T_CDC	GPS1	GPS2	D(m)	M	A(%)	SD
53	180	900	6197	Bus	50	5.87
44	180	720	5295	Bus	56	6.05
40	180	540	4250	Bus	63	5.63
28	180	360	3070	Bus	68	3.20
25	180	180	1470	Bus	70	2.68
16	100-120	100-120	442	Walk	84	1.73
6.90	35-45	35-45	50	Walk	78	0.98

Partial matching involves matching the current route with a known route at the database when the current route is a sub-route of the existing route. It also involves matching sub-route of a route, when the sub-route being matched is a sub-route of an existing route. Here by matching smaller

sub-routes (Figure 3) of a mobile client it becomes easier to detect a sudden change in client movement from one visited route to another visited route in the database or to a completely new route (Use case 2). Here visited routes refer to routes traversed by the client in the past. However such full routes could have been traveled independently and not in the course of the journey where the client covers only a section of the full route. Results are evaluated with different sub-route sizes with minimum sub-route size being 200 m (walking) and maximum sub-route size being 1.5 km (riding a bus). The choice of the sub-route is based on the system processing time. As the system performance is optimal with 180 - 200 unique GPS points on a 4 - 5 mins bus route, we limited this value as maximum sub-route length. In online matching smaller sub-route matching helps to turn on the GPS as soon as the curve matching coefficient of the currently visited route exceeds the threshold limit. The client then requests the new route details by sending the start direction to the server.

In sub-route matching the incorrectness of GPS route information available to mobile users is limited to the time taken to travel the sub-route. Smaller the sub-route greater is the accuracy of position information provided alternatives from current sub-route is less. Table 2 shows the false route information is 1470 m on a bus with sub-route interval of 4 mins. However false position information decreases to a maximum of 442 m when the mobile user covers the same length sub-routes by walking. We also find matching 2 different sub-routes of 4 - 5 mins along the same mode of transport increases the CDC. Results show on comparing different sub-routes CDC increases by 25 - 35% on walking, 30 - 40% on biking and 45 - 55% on a bus ride more than CDC of same sub-routes. Largest distance covered with greatest altitude variation on a bus ride increases its CDC and also the false negatives to the maximum level.

The CDC is the least when the two compared sub-routes are of the same duration. Table 2 presents the CDCs of an altitude curve of a mobile user traveling on a bus and walking. The CDCs are calculated by partially matching the current altitude curve of the mobile user with already visited routes stored in the database. It is evident from the sixth column of Table 2 that the CDC decreases and the prediction accuracy (explained in Section 7.1) increases on selecting smaller sections of the curve.

Figure 3 shows the entire altitude route of each day divided into 5 sub-routes, each of 180 unique GPS points i.e. of 4 - 5 mins duration. If the bus route contains more than 900 GPS points, the excess points are accommodated to the fifth section. Each section of one curve is then matched to the corressponding section of the paired curve for all five sub-routes. This gives 5 curve difference coefficients between any two pair of curves. For better understanding, we then compute the mean curve difference coefficient (M_CDC) and the standard deviation (SD) for each section.

Table 3: Partial Matching showing Section Number, Mean Curve Difference Coefficient (M_CDC), Standard Deviation (SD) of different sections for a given bus route along both directions (UP and DOWN)

Sec_No	M_CDC_UP	M_CDC_DN	SD_UP	SD_DN
1	20.18	22.28	1.84	2.36
2	22.85	22.78	2.21	3.50
3	21.95	22.22	2.82	2.89
4	21.40	23.29	2.84	2.24
5	19.41	19.28	3.57	2.56

On partially matching sections of the altitude curve we find the shape difference between any two curves increases with increasing the section number (first 4 sections each of 180 GPS points) i.e. in due course of the movement. This is reflected from Table 3, the curve difference standard deviation (SD) during the downward course of the journey for the second, third, fourth, fifth sections are 3.5, 2.89, 2.24 and 2.56 respectively, whereas for the first section the SD is 2.36. The difference is more towards the middle to end of the curve due to uneven speed variation, acceleration at different curvatures of the curve. For the first section the curve difference remains less as during each time during its start, the speed of the bus is slow.

The same type of CDC variation is noticed from Table 3 during the upward course of the journey along the same route. Here the shape difference between any two curves increases from section 1 to section 4 (same section length of 180 unique GPS points) and then decreases at section 5. The decrease is due to slow speed of the bus during its arrival at the destination. The SD seen from Table 3 also shows the same trend (increase towards the middle to end of the curve) similar to the downward course of the journey.

On comparing matched curve results via walking and bus riding, it is seen from Table 2 that smaller section gives better matching accuracy by giving less false negatives. The CDC is more accurately predicted for a smaller path length with less alternative sub-routes, the SD also decreases on decreasing the section of the curve. For higher accuracy, the curves can be divided to have more sections, each section being of 50 - 75 unique GPS points or 1.4 - 2 mins length.

Partial matching also involves sub-route matching as explained with a flowchart in Figure 7. Figure 2(b) presents a situation when a user initiates his journey in a sub-route of a complete route and the start direction of the user does not coincide with any of the start points of the stored routes (Step 1). The cloud then queries the database to find if there exists any possible route where the start point may fall in between the start and the end points (Step 2B), i.e. it tries to find possible route intersections of the new route with any full-length stored routes. For a closed curve stated in Figure 2(a) the start point may point to either directions of the route. For this reason we choose to provide few more GPS coordinates after the initial start point to provide the direction of user movement. If there exists such points in any section of a complete route, the cloud partitions the curve from the supplied start direction to the end of the curve. Then it matches only the partitioned route and the recent trajectory of the user and compares it with the T_CDC (Step 4B). In Figure 2(b), we find the coefficient of matching of the sub-route is 3.60 falling below the threshold value of 4.25. By partitioning routes, the algorithm is able to match multiple sub-routes described in use-case 2.

6.3 Reversed Route Matching

The main limitation of the curve matching algorithm is its inability to detect same tracks traversed reversely (uphill and downhill) when the mode of travel is bus or walk. We analyzed with this algorithm whether reversal of any curve resembles the opposite route along a given mode of transport. The analysis reveals the dissimilarity function of reversed curves is not significantly less. Only in 7 - 10% cases the dissimilarity function is less compared to the CDC of the curves in the same direction. Or in other words, the algorithm cannot reveal the opposite directional paths as mirror images of each other.

The 7 - 10% exceptional scenario is noticed in case of biking where the pressure sensor gives more accurate readings. This is because the barometric recordings obtained from a heart-rate monitor during biking exhibits much less deviation (SD of 2.66 on a 25 - 30 mins 10 km route) when com-

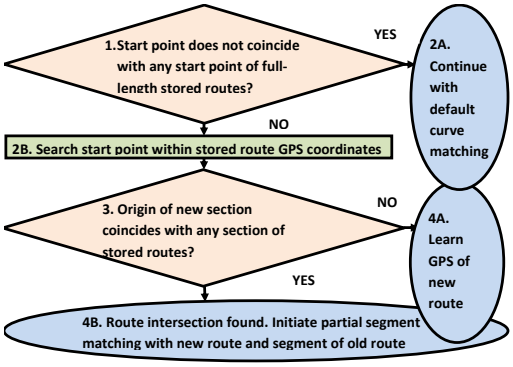

Figure 7: Sub-route matching algorithm

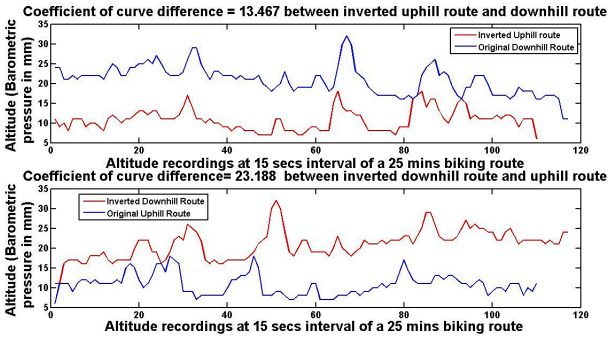

Figure 8: Matching opposite directional altitude curves with one track reversed

pared with the altitude readings obtained from a GPS (SD of 2.68 on a 4 - 5 mins 1.5 km bus route). More accurate barometric readings produce mirror images in exceptional cases on matching opposite directional curves.

To study this, two curves of the same route in two different directions (uphill and downhill) of CDC 14.0 is taken. Here T_CDC is 15.43 obtained from Table 1 at 95% confidence level. In Figure 8 we compare the two curves, keeping one route intact and reversing the other route so that both points to the same direction of origin. The first plot in Figure 8 shows that the CDC between the inverted uphill (red) and the original downhill (blue) route is only 13.46. This ensures the fact that weighted curve distance factor of one route reversed does not always produce a significantly less curve distance. This is more evident from the second plot of the same figure. Though the curves look similar, here the altitude differences of the same route with inverted downhill (red) and original uphill (blue) route increases significantly to 23.18. This happens due to windspeed, road-directions, uneven curvatures, and speed difference while traversing the curvatures. The algorithm is tolerant in recognizing unevenly spaced altitude variations and speed difference to a certain level (described next in Section 6.4).

Further for more detailed analysis, we also permuted both ways to and from direction (between Ruoholahti and Otaniemi [1]) altitude traces of one day with every other day to give 1156 (i.e. $\frac{34*34}{1}$) comparison results. The mean value of CDC between any two curves of the comparison set is 13.42 and the SD is 4.27. Thus the deviation in comparing inverted opposite directional curves along the same route is also found to be more than comparing the curves along the same direction which has standard deviation of about 2.66.

6.4 Effect of Speed and Pressure

The behaviour of the algorithm with different speeds are carried with other modes of transport like walking and riding a given bus route. In each mode a definite route is selected and altitude traces are collected for different days (20 days in case of riding a bus and walking whereas 34 days for biking). Here speed varies randomly on all days with maximum variation of 25% between the slowest and the fastest day.

The algorithm tries to match the shape of two curves when the barometric pressure level and travel speed on two different days differ. Figure 9 presents the altitude curve of the same route in the same direction of travel as determined by the pressure sensor of a heart-rate monitor. The average atmospheric pressure on these two different days are found to be 20 mm and -7 mm. Further the total time of travel also differs, the first curve completes in 29 mins while the second curve completes in 26 mins. The CDC between the plotted curves is only 7.03 which lies below (T_CDC) the threshold level of 7.93 (Table 1).

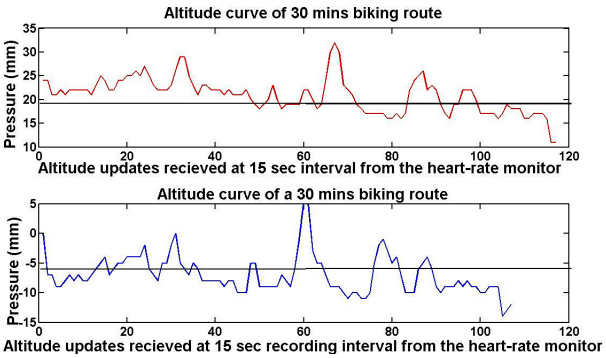

Figure 9: Altitude curves as given by barometric pressure sensor on two different days

This shows that Sethian's Fast Marching Method for curve matching matches two curves based on shape, is independent of the variation in speed and atmospheric pressure. The third term of Equation 2 plays an important role in giving an optimal curve difference by neglecting stretched spaced effect of the similarly shaped curves along the same route.

The main advantage of curve matching algorithm is that it is unaffected by speed of the travel during walking or biking. Figure 9 demonstrates that algorithm is able to detect same routes by its dissimilarity function when biked on two different days with varying speed. Each reading of pressure sensor is recorded at an interval of 15 secs with the heart-rate monitor, while GPS fixes are recorded at 1 sec interval. So even in case of varying speed the sensor or GPS recordings are not at the exact same location but the shape of the recorded curve remains the same.

The T_CDC for a small walking route of 4 - 6 mins along the same direction is 6.90. We found for a 8% speed variation the CDC is 2.78, while for 17% speed variation the CDC is 4.07. Both of them falls below the threshold value of 6.90. The SD obtained by permuting each day's alttude curve with every other day is only 0.98. For a definite bus route of 20 - 25 mins, the results of partial matching is tolerant to a speed variation of 5 mins or 25%. Figure 3 demonstrates a 20 - 25 mins bus route covered on three different days with varying speed. The tolerance level of the algorithm for varying speed is also evident from columns 4 and 5 of Table 3 that there exists a maximum standard deviation of 3.57 (around 15%)

in matching each section of the bus route.

7. DISCUSSION

The performance of the curve matching algorithm is measured by its prediction accuracy. This is measured by subtracting the number of false positives [6] and false negatives from the actual prediction of the server. False negatives occur when the user actually follows an already discovered route stored in the database but the CDC exceeds the threshold level (T_CDC). In that case the system gives an error result and assumes the input route as a new route. False positives occur when a user follows a given route with CDC lying below T_CDC but the system returns some incorrect route information having same GPS start position, direction and position estimates close to the correct route.

The algorithm is also evaluated to indicate its level of efficiency for the stated use-cases in the current system. The performance indicators include processing speed, efficiency with different modes of transport, accuracy, sectional or partial matching, matching curves of varying speed, pressure and opposite directions along a given route.

7.1 Cloud System Performance

Sub-route matching produces lowest accuracy of 50% on a bus route. Higher false negatives resulting from higher CDC of matching same sub-routes reduces the system accuracy. On the other hand walking and cycling routes having lower CDC have higher false positives. The accuracy is found to decrease on matching very small sub-routes on walking when the number of alternate routes from the sub-route are many. It also decreases with decrease in variation in altitude for individual routes or sub-routes. At an uniformly planar region like a falt city, pressure sensor readings comes of little use. There other sensors like compass, magnetometer, accelerometer could be used in addition to retrieve position information from stored GPS traces.

The system operations of Figure 1 has optimal performance speed when the server matches sub-routes of complete routes. On an average, comparing the altitude curves of 35 - 45 unique GPS positions takes around 400 - 500 ms on a Dual core T5550 1.83 GHz processor. However it increases to 15 - 16 secs on matching 180 - 200 unique GPS position on the same processor. Reduction of unique GPS positions can reduce sub-route lengths to enhance processing speed on server. This can be done by increasing GPS query interval and by using higher accuracy points to compress neighbouring points. Point compression algorithms like KML and Google Maps [2] can also serve this purpose. This processing efficiency of the cloud based system can be improved further by using energy-efficient uploading of sensor data to the cloud [14], by evaluating the performance of different partial curve matching algorithms like exact algorithms [9] or improved algorithms [16].

We choose to limit partial curve matching to 180 - 200 unique GPS points as slightly increasing the number of GPS points beyond this limit the processing speed increases to a range of a minute. The processing time increases to the range of several minutes when the entire travel bus route of 20 - 25 mins is matched. A trade off factor is associated with the processing speed and the length of the section of the curve matched with segments of curves stored at the cloud. To have minimum processing speed in terms of the whole length of the curve, maximum of 5 sections, with each section of 180 unique GPS points is chosen.

We also propose the use of the phone over the cloud for matching altitude curves particularly when number of routes stored in the database are very small. To build an energy efficient system, it is economical to build a hybrid system where the most frequently traversed data are stored at the phone and less frequently traversed data (like multiple alternate sub-routes of a complete route) are stored at the cloud. The system performance and power savings of the cloud architecture is briefly discussed below.

7.2 Power savings

Our study with Nokia Energy Profiler (NEP) shows that getting a single GPS fix on Nokia N8 on an average draws 120 - 620 Joule / hour at 80 secs interval (depending on the availability of satellites) while sensors like accelerometer or magnetometer consumes .08 Watt of power at an interval of 30 secs [17]. To find out the potential of power savings in sensor based systems, we used online learning. In online learning the mobile client (N8) queries the cloud at a fixed interval over TCP socket in a 3G network. The client also provides the cloud with a given GPS start direction at the origin of the sub-route. The start direction is obtained by taking the initial valid GPS fixes for 10 - 12 secs. In case the sub-route is stored in the database, the cloud sends the client GPS information of the entire route over the same TCP socket. Our calculations show for a maximum sub-route length of 5 mins on bus, querying the cloud in intervals of 65 - 67 secs and turning on GPS at the sub-route origin consumes the same average power (.31 - .33 Watt) as querying the GPS at an interval of 1 sec. As sensors consume power of .08 Watt/30 secs, increasing the querying time to the cloud to 70 - 72 secs makes the average power consumption of initial GPS fix, 3G data transfer and pressure sensor equal to GPS query. Power savings of 25% - 30% could be achieved by setting the GPS query interval to 80 - 85 secs. However detailed discussion on percentage of power savings for varying length routes and sub-routes for stored and new routes is outside the scope of this paper. We leave this as future work. As per NEP records, N8 consumes average power of .10 Watt for matching altitude curves at intervals of 70 - 72 secs for a very small subset (3 - 4) of stored routes. Power savings obtained at the phone by such matchings suggest the design of a hybrid system could be used to save smartphone battery.

8. CONCLUSION AND FUTURE WORK

In this paper we use Sethian's Fast Marching Method for determining curve matches in a crowd-sourced database. This work can be further extended by using other matching algorithms like exact algorithms [9]/improved algorithms [16] for partial curve matching to analyze the system performance. Complete route matching has an accuracy of 75% for biking and 75 - 88% for walking along different paths with varying elevation profiles, sub-route lengths (3 - 10 mins), speed and travel time. As complete route matching fails for lengthy bus routes we use partial matching where accuracy of matched segments varies between 50 - 70%. Such low to moderate accuracy findings entails us to use larger history of crowd-sourced data, shorter sub-routes with less alternatives and validation of location estimates with other sensor readings.

Further we find that matching the paths of opposite direction on the same route is unsuccessful in most cases. Also curves of varying speed (25%) and atmospheric pressure along a given route on a given mode of transport exhibit 10 - 25% SD, proving the fact that coping with different speeds does not require GPS ability or knowledge of speed. As altitude elevations of same routes follow similar trend (normal distribution), mobile phones can use internal or external pressure-sensor as an alternative to GPS to track routes. GPS querying is power-intensive and can drain 1200 mAh battery on N95 smartphone in less than 11 hours [17], [18].

Hence reducing GPS active time by tracking user-movement with pressure sensor has great potential in enhancing life of mobile battery. The design of a hybrid system with online matching at the cloud and the phone could be used a future reference design model. Our future work lies in investigating the percentage of mobile battery savings for a pressure sensor in-built GPS enabled phone. Through online matching of varied length sub-routes we intend to find the tradeoff between position accuracy and battery life.

We plan to extend the scope of the prototype system from a single user to multiple users by building a larger scalable crowd-sourcing system [7], [20] where millions of mobile users can update their altitude trajectory and start direction information. The accuracy of such a crowd-sourced system will increase with higher frequency of GPS reportings of a particular route from more number of users. The cloud can process the routes based on different modes of transport and group the same regularly visited routes [8]. Based on altitude trend identification and a given start direction, the cloud can send existing GPS route information to any requesting client.

9. ACKNOWLEDGMENTS

This work was supported by the Academy of Finland, grant number 253860.

10. REFERENCES

[1] Google maps find altitude. `http://www.daftlogic.com/sandbox-google-maps-find-altitude.htm/`.

[2] Kml and google maps compression. `http://captico.com/kml-and-google-maps-compression/2010/07`.

[3] Map my tracks. `http://www.mapmytracks.com/blog/`.

[4] Matlab normplot. `http://www.mathworks.in/help/toolbox/stats/`.

[5] Mobile sports-tracker. `http://www.sports-tracker.com/`.

[6] F. B. Abdesslem, A. Phillips, and T. Henderson. Less is more: energy-efficient mobile sensing with senseless. In *Proceeding MobiHeld '09 Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*.

[7] F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz. Location-based crowdsourcing: extending crowdsourcing to the real world. In *NordiCHI '10 Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 13–22. ACM, 2010.

[8] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. Easytracker: automatic transit tracking, mapping, and arrival time prediction using smartphones. In *SenSys '11 Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 68–81, New York, NY, USA, 2011. ACM.

[9] K. Buchin, M. Buchin, and Y. Wang. Exact algorithms for partial curve matching via the fréchet distance. In *SODA '09 Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 645–654. ACM, 2009.

[10] Y. Chon, E. Talipov, H. Shin, and H. Cha. Mobility prediction-based smartphone energy optimization for everyday location monitoring. In *SenSys '11 Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 82–95, New York, NY, USA, 2011. ACM.

[11] M. Frenkel and R. Basri. Curve matching using the fast marching method. In *EMMCVPR*, volume 2683, pages 35–51. Springer, 2003.

[12] M. B. Kjaergaard, S. Bhattacharya, H. Blunck, and P. Nurmi. Energy-efficient trajectory tracking for mobile devices. In *MobiSys '11 Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011.

[13] M. B. Kjaergaard, J. Langdal, T. Godsk, and T. Toftkjaer. Entracked: energy-efficient robust position tracking for mobile devices. In *MobiSys '09 Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 221–234, New York, NY, USA, 2011. ACM.

[14] N. Lane, M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A. Campbell. Bewell: A smartphone application to monitor, model and promote wellbeing. In *5th International ICST Conference on Pervasive Computing Technologies for Healthcare*, Dublin, Ireland, 2012. IEEE.

[15] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *SenSys '10 Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 71–84, New York, NY, USA, 2010. ACM.

[16] A. Maheshwari, J. J.-R. Sack, K. Shahbaz, and H. Zarrabi-Zadeh. Improved algorithms for partial curve matching. In *Proceedings of the 19th Annual European Symposium on Algorithms (ESA 2011)*, volume 6942/2011, pages 518–529, Saarbrucken, Germany, September 2011.

[17] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *MobiSys Proceedings of the 8th international conference on Mobile systems, applications, and services*, New York, California, USA, 2010. ACM.

[18] J. Paek, K.-H. Kim, J. P. Singh, and R. Govindan. Energy-efficient positioning for smartphones using cell-id sequence matching. In *MobiSys '11 Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 293–306, New York, NY, USA, 2011. ACM.

[19] S. Salvador and P. Chan. Fastdtw: Toward accurate dynamic time warping in linear time and space. In *Journal Intelligent Data Analysis*, volume 11, pages 561–580, Amsterdam, The Netherlands, 2007. ACM.

[20] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using smart-phones. In *SenSys '10 Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2010.

[21] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod. Accurate, low-energy trajectory mapping for mobile devices. In *NSDI Proceedings of the 8th USENIX conference on Networked systems design and implementation*, Berkeley, CA, California, USA, 2011. ACM.

[22] E. Welbourne, J. Lester, A. LaMarca, and G. Borriello. Mobile context inference using low-cost sensors. In *LoCA'05 Proceedings of the First international conference on Location- and Context-Awareness*, pages 254–263. ACM, 2005.

[23] F. Youssef, M. A. Yosef, and M. N. El-Derini. Gac: Energy-efficient hybrid gps-accelerometer-compass gsm localization. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5, Miami, FL, 2010. IEEE.