Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2014-017 August 26, 2014

Motion Compatibility for Indoor Localization

Jun-geun Park and Seth Teller

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

CSAIL

Motion Compatibility for Indoor Localization

Jun-geun Park
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
jgpark@csail.mit.edu

Seth Teller
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
teller@csail.mit.edu

Abstract

Indoor localization — a device's ability to determine its location within an extended indoor environment — is a fundamental enabling capability for mobile context-aware applications. Many proposed applications assume localization information from GPS, or from WiFi access points. However, GPS fails indoors and in urban canyons, and current WiFi-based methods require an expensive, and manually intensive, mapping, calibration, and configuration process performed by skilled technicians to bring the system online for end users.

We describe a method that estimates indoor location with respect to a prior map consisting of a set of 2D floorplans linked through horizontal and vertical adjacencies. Our main contribution is the notion of "path compatibility," in which the sequential output of a classifier of inertial data producing low-level motion estimates (standing still, walking straight, going upstairs, turning left etc.) is examined for agreement with the prior map. Path compatibility is encoded in an HMM-based matching model, from which the method recovers the user's location trajectory from the low-level motion estimates. To recognize user motions, we present a motion labeling algorithm, extracting fine-grained user motions from sensor data of handheld mobile devices. We propose "feature templates," which allows the motion classifier to learn the optimal window size for a specific combination of a motion and a sensor feature function.

We show that, using only proprioceptive data of the quality typically available on a modern smartphone, our motion labeling algorithm classifies user motions with 94.5% accuracy, and our trajectory matching algorithm can recover the user's location to within 5 meters on average after one minute of movements from an unknown starting location. Prior information, such as a known starting floor, further decreases the time required to obtain precise location estimate.

Categories and Subject Descriptors

C.2.m [**Computer-Communication Networks**]: Miscellaneous

General Terms

Algorithms, Design, Experimentation

Keywords

Indoor Localization, Inertial Sensing, Motion Classification, Trajectory Matching, Sensor Fusion, Route Networks, Conditional Random Fields, Hidden Markov Models

1 Introduction

An inexpensive, accurate location discovery capability would enable a broad class of context- and location-aware applications. Many indoor localization methods [1, 6, 40] rely on existing wireless infrastructure, such as 802.11. These methods require technicians to survey the deployment area while measuring RF signal strength "fingerprints" and interactively associating them with manually-specified location information. Devices later estimate location by matching observed fingerprints to the acquired map.

To mitigate the mapping burden, researchers have proposed localization systems that either use contributions from end-users by crowdsourcing [21], or algorithms that infer the RF fingerprint map from fewer or no location labels [27, 39]. Although these improvements can reduce the initial deployment burden, RF-based methods have other restrictions: they typically construct a fingerprint map only over a short time interval, which will have limited utility at other times due to the time-varying nature of RF signals; and the high-frequency RF scanning required for location updates (especially continuous updates) can be energetically costly [26].

Recently, localization algorithms which use sensors found in off-the-shelf mobile devices have been proposed [29, 38]. Such methods extract distance and heading measurement from MEMS IMUs, and estimate user position and attitude through dead reckoning. Since MEMS IMUs tend to drift quickly over time, these methods require sensors to be placed in specific positions (e.g. on the feet), or rely on frequent external position fixes from another source (e.g. WiFi-based localization systems). Others use Kalman filters or particle filters to account for measurement uncertainty [12, 27, 38]. However, these still depend directly on low-level sensor measurements. Also, these "forward-filtering" based methods are often formulated to update only the latest location given new measurements; they do not recover the user's recent path history.

In contrast, our work attempts to recover the *entire* user trajectory from a sequence of *discrete* motion descriptions. This approach is inspired from the way that humans describe indoor routes in abstract terms, including motion descriptions (e.g. "walk" or "turn left") and motion-related actions (e.g. "open the door") rather than precise distance and angle specifications. (Human motion descriptions can include absolute directions (e.g. "north") or durations (e.g. "for 5 seconds"), but these are typically interpreted qualitatively as detailed annotations of more abstract navigation guidance.)

Humans can also perform the inverse process — inferring a route given a motion sequence and an indoor map. Given a sufficiently long motion sequence, we can narrow our idea of the route taken using various kinds of motion-related information: implied walking lengths and turn angles (geometric constraints); spatial layout and path continuity (topological constraints); and agreement of the space type and motion type (semantic constraints). In other words, one can view location estimation as a decoding process in which the originating route is inferred from an observed sequence of motions, combined with spatial constraints imposed by the prior map.

Realizing this intuition for indoor localization could bring significant advantages in terms of energy and localization capability, over existing methods. As recent mobile platforms provide activity recognition as their core functionality (e.g. recent additions of activity recognition APIs in Android and iOS), we envision that our approach can take advantage of such platform-level support. For example, a trace of user motions computed and buffered by a dedicated, low-power motion co-processor (e.g. Apple M7) could be used to compute the user's current position *on demand*. This scheme also could recover the user's recent *path* without continuous RF scanning, whose energy cost is prohibitive. (Our approach does not exclude the possibility of using absolute position fixes, if doing so would speed acquisition of the current location.)

This paper describes an algorithm that codifies this intuition, using a Hidden Markov Model (HMM) to find the most likely path given a motion sequence (§5). The method takes as input a sequence of motion labels, automatically generated by a low-level motion labeling algorithm that takes raw sensor measurements from a handheld mobile device as input (§4). It assumes a route "map" extracted *automatically* from legacy floorplan data (§3.3). The matching model (§5.4–5.6) defines a compatibility measure between the input motion and candidate paths, among which the best path is found by Viterbi decoding (§6.1). We also show how to determine the model parameters from unannotated data (§6.2).

2 Related Work

We review prior work on indoor localization and relevant areas.

RF-Fingerprint Localization Some localization methods associate ambient RF "fingerprints" with physical locations, enabling devices to estimate location by finding the stored fingerprint most similar to the current RF observation [1, 6, 40]. These methods exploit ubiquitous wifi infrastructure, unlike approaches which required dedicated localization infrastructure [23, 36]. Since constructing the RF map is an expensive process that requires intensive human labor, there have been a number of attempts to reduce or even remove the burden of manual location labeling, by spatial regression [14], joint mapping of the RF map and access point locations [3], crowdsourcing [21], or by incorporating inertial sensors [27, 39] to achieve "calibration-free" WiFi localization.

Inertial Matching and Robot SLAM Algorithms Recent advances in small, lightweight MEMS sensors have made indoor pedestrian navigation with handheld devices practical. However, the low-cost inertial sensors in mobile devices are usually inaccurate, having high bias and drift characteristics and preventing naïve dead-reckoning from working in practice. To circumvent this problem, prior work has relied on foot-mounting of sensors [29, 38], position fixes from external measurements [12], or the use of Kalman or particle filters [18, 38].

The use of filtering algorithms for state-space models has been widely explored in the robot localization and mapping community. Recent work on simultaneous localization and mapping (SLAM) uses diverse sources of metric/semantic information from a rich set of sensors, including LIDARs and depth-enabled cameras, equipped on a robot or as a wearable device [5, 24]. Since such active, exteroceptive sensors are unlikely to be incorporated into smartphones in the near future, our work addresses the challenges of accurate location discovery from the cheaper sensors available today.

Trajectory Matching Map matching on a road network is usually formulated as a trajectory matching problem, to exploit known vehicle dynamics and handle inaccurate GPS observations [13, 33]. Sensor measurements from user mobiles are often combined to enhance matching accuracy [7, 32]. Just as the state model for outdoor map matching is naturally given by a road network, we derive an analogous "route network" from a legacy floorplan. We do not assume any external position infrastructure (e.g. Loran, GPS).

Activity Recognition There is substantial prior work on recognizing human activity from low-cost sensors. In particular, acceleration data from MEMS accelerometers have been used as the primary input for different classification algorithms, including meta-classifiers [16], temporal models [28], and topic models for high-level activities [9].

Human Navigation in Indoor Environments Our use of low-level motions to recover high-level trajectories is motivated by human perception of indoor route navigation. Studies suggest that humans rely on geometric (i.e. space layout) as well as non-geometric cues (e.g. landmarks) when learning and navigating spaces [11]. Based on this idea, Brush et al. performed a user experience study for activity-based navigation systems that present a trail of activities to the destination [2]. Also, there have been recent attempts to make an automated agent "ground" spoken natural language to support navigation and mobile manipulation [17, 20, 31].

Recently, there has been some prior work augmenting human actions with indoor localization. For example, Action-SLAM used (simulated) actions as observed landmarks in addition to inertial measurements provided by body-mounted sensors [8]; HiMLoc recognized certain salient activities (e.g. in elevator/stair/door) and combined them with WiFi fingerprinting-based localization [26]. In contrast, our work extracts rich low-level user activities from a handheld device using a motion classifier (§4) and perform map-matching solely from a sequence of those descriptive activities. (However, our work does not exclude the use of such absolute positioning information.) Similarly to our method, UnLoc system recognized sensor measurements as distinct signatures identifying spaces, using them to reset dead-reckoning errors [35]. Our method advances this idea by proposing a probabilistic matching framework to interpret such sensor observations into a user trajectory systematically, enabling "future" observations to correct the "past" user path.

3 Problem Statement

We formulate location discovery from motion traces as a sequence labeling problem. We model the user's indoor walking motion as a series of discrete motions consisting of straight-line walking segments, stops, turns, vertical movements, and actions to gain access (such as opening doors), which is estimated using a conditional random field (CRF) based motion classifier (§3.2&4). Given a sequence of user motions, we compute the probabilistic path compatibility between trajectory hypotheses on the map and the hypothesized user path that gave rise to the input motion sequence. Solving this problem amounts to labeling each input motion with a map location, while maximizing *compatibility*: the likelihood of the output trajectory given the input data (§3.4). Given an input motion sequence, our method finds the most likely trajectory of location labels (§5).

We assume that each user carries a sensor-instrumented mobile device in his/her hand, held roughly level and pointed roughly forward. The device captures and time-stamps sensor data, which is then input to a classifier which produces as output a sequence of low-level motion types and durations.

3.1 User Motion Models

We model user motion traces as a series of *discrete* actions parameterized by properties associated for each motion. A property can be either a discrete or a continuous value, representing the characteristics of the associated motion (e.g. duration, velocity or direction). Continuous values may be quantized to facilitate the matching process. In indoor navigation scenarios, most walking paths can be modeled well with the following set of actions:

Rest Periods of little or no net motion, for example, while seated in an office, classroom, cafe, library, etc. Detection of user resting can be used to favor certain place types over others.

Sitting and Rising These motions separate Rest from non-Rest activities in time.

Standing Standing with little horizontal or vertical motion, after Rising or between other motions.

Straight Walk The user walks approximately along a straight line. Total distance traveled can be estimated by integrating longitudinal acceleration, by detecting and counting user strides, or (assuming constant walking speed) from duration.

Turn Change of walking or standing direction over a short period of time. We quantize egocentric turns to eight values.

Walking Ascent and Descent Walking on ramps and stairs (on spiral or multi-stage stairwells, often accompanied by successive Turns).

Elevator Periods of vertical ascent or descent in an elevator. Elevator motions typically do not involve Walk or Turn motions.

Access Auxiliary actions required to move within typical indoor environments, including opening doors (Door Open) and pressing elevator buttons (Button Press).

We assume that user paths have these properties:

1. **Smoothness**: The user maintains each motion for a certain characteristic duration, and does not change it too frequently.
2. **Continuity**: Consecutive motion segments agree at their endpoints (i.e. the user cannot "teleport" from one place to another).
3. **Monotonicity**: Users will tend not to travel a distance longer than necessary, or change floors more often than necessary, to reach any goal location.

We use these assumptions to precondition imperfect input motion sequences, as well as to formulate an efficient matching algorithm without redundancy. If a certain detected motion does not last for its normally expected duration (e.g., an elevator ride that lasts only two seconds), it is deemed erroneous, and either corrected or removed before matching (*smoothness*). When considering transitions from a certain place, the matching algorithm considers only nearby places for the next segment, resulting in physically plausible paths (*continuity*). Finally, the method excludes inefficient motion patterns (*monotonicity*).

The present paper focuses on finding the location trajectory for a single user path. We anticipate that our method can be generalized to handle multiple paths from different users jointly, by taking interactions between paths into account. We discuss this possibility in Section 8.

3.2 Automatic Motion Sequence Labeling

The user motions described in Section 3.1 are estimated from data produced by low-level sensors: a tri-axial accelerometer, a tri-axial gyroscope, a tri-axial magnetometer, and a barometer, all of which are available in current-generation smartphones.

Our CRF–based motion labeling algorithm takes time-stamped sensor data as input, and outputs fine-grained user motion states at 3Hz, concatenating successive frames with the same motion type. Each motion in a produced motion sequence is annotated with its duration, which is used by trajectory matching algorithm to infer coarse-grained metric information for some types of motions (e.g. walking distance for Walking). While it is possible to extract more precise metric information from the underlying sensor signals [22], we found that the duration was sufficient for our evaluation of the map matching algorithm, because the best match was robust to some deviations of the metric information from truth as it was determined probabilistically. We describe the details of the labeling algorithm in Section 4.

3.3 Route Networks

Our motion-based map-matching algorithm requires a *route network*: a graph representation of all possible user paths within the environment. We implemented an automatic graph construction algorithm that analyzes and interprets floor plan data to generate a generalized Voronoi graph based route network for our test corpus. A Voronoi-based route network representation has been used in prior work for robot or human navigation and map matching in indoor environments [19, 34, 37].

Figure 1: Route network (red, blue) automatically extracted from a legacy floorplan

Our route network generation process builds upon prior work by Whiting et al. [37], which uses a constrained Delaunay triangulation (the dual of the Voronoi diagram) to approximate the medial axis of each space. A graph is generated for each space using its Delaunay triangulation, then combined with graphs for spaces abutting via horizontal and vertical "portals" to create a complete route network for the entire corpus (Fig. 1). Spaces on the same floor are connected through horizontal portals, which represent either explicit connections (doors) or implicit connections (shared non-obstructed space boundaries). Route graphs from different floors are joined through vertical portals, i.e. stairs and elevators.

We use an automated process to generate route networks from floorplans. However, manual authoring would also be feasible in many settings. Since for most buildings only a portion is occupied by end-users (e.g. in malls, hospitals and airports), and interior structure tends to change slowly over time, the maintenance burden to author and keep current the building's route network should be tolerable.

3.4 Path Compatibility

A sequence of observed motions, as described in Section 3.1, implies a *motion path* (Figs. 2a & 2b). The motion path from imperfect observations is not identical to the original path from which motions were generated. Since motion estimates inferred by an automatic motion recognition algorithm are noisy, the properties attached to the motions can also be noisy and coarse-grained.

In this setting, the trajectory matching process from a sequence of motion observations can be thought of as finding the best (chain) subgraph in the route network whose *path compatibility* is maximal to the conceived path from motions. We say that a motion path is compatible with a subgraph if the path can be embedded into the subgraph in a way that observes (or weakly violates) the constraints inherent in the path and subgraph. That is, compatibility holds if each motion path segment can be assigned to some vertex or edge of the subgraph. For example, in Figure 2, the subgraph `a-b-d-e` in Figure 2a is compatible with the motion path of Figure 2b, because the path can be assigned to the subgraph with tolerable geometric distortion and without violating continuity. Clearly, the more components in the motion path, the more restricted its compatibility.

We analyze path compatibility at three levels: *geometric*, *topological*, or *semantic* compatibility. Geometric compatibility imposes metric constraints, such as length of walking segments and turn angles. Topological compatibility concerns about correspondence between space layouts and the continuity of motions. Semantic compatibility states that a certain class of motions can occur only in spaces with the matching type. These notions of compatibility are encoded in our trajectory matching model in Section 5.

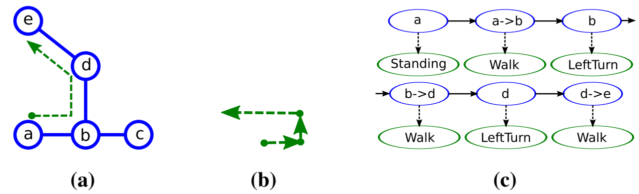


Figure 2: Path compatibility and matching: (a) path of a user moving from `a` to `e` (green, dashed) within a route network (blue); (b) implied motion path from motion observations with imperfect length and angle estimates; (c) corresponding sequence labeling problem.

4 Motion Sequence Labeling

In this section, we describe a motion labeling algorithm that estimates fine-grained user motions from low-level sensor data.

4.1 Factorized Representation of Motions

In designing the motion labels reflecting the motion model described in Section 3.1, we introduce a *factorized* representation of motions.

The major parts of the motion descriptors in Section 3.1 are egocentric descriptions of navigational activities. A navigational motion in a typical 2.5-dimensional indoor space consists of three components: *Principal* mode of motion (e.g. sitting or walking), *Lateral* component (e.g. left/right turn), and *Vertical* component (e.g. stair or elevator ascent/descent). Hence, a basic motion "clause" describing a navigation action can be represented as a Cartesian product of three orthogonal motion components:

$$\textit{Principal} \times \textit{Lateral} \times \textit{Vertical}$$

where symbols in each component are:

$$\begin{aligned} \textit{Principal} &= \{\mathsf{Sitting}, \mathsf{SittingDown}, \mathsf{StandingUp}, \mathsf{Standing}, \mathsf{Walking}, \mathsf{Running}\} \\ \textit{Lateral} &= \{\mathsf{Straight}, \mathsf{LeftTurn}, \mathsf{RightTurn}, \mathsf{LeftUTurn}, \mathsf{RightUTurn}\} \\ \textit{Vertical} &= \{\mathsf{Flat}, \mathsf{StairUp}, \mathsf{StairDown}, \mathsf{ElevatorUp}, \mathsf{ElevatorDown}\}. \end{aligned}$$

Most basic navigation actions can be described as a product of the three motion components. For example:

$$\begin{aligned} \text{Walking straight on a flat surface} &\Rightarrow \text{(Walking, Straight, Flat)} \\ \text{Turning right while walking up stairs} &\Rightarrow \text{(Walking, RightTurn, StairUp)} \\ \text{Riding an elevator down} &\Rightarrow \text{(Standing, Straight, ElevatorDown)}. \end{aligned}$$

As some sensors measure physical quantities directly associated only with a specific motion component (e.g. barometric measurements are related only directly to vertical motions.), this decomposition reduces unnecessary modeling effort for redundant combinations of features and motions.

Still, there exist cases in which a user motion can be compactly described by a special descriptor outside of the basic components. In particular, certain types of motions are bound to occur only at certain types of places. To take advantage of such prior knowledge, we introduce the fourth component, the *Auxiliary* axis, to explain auxiliary actions required to move within indoor environments. In this work, we define two additional actions that arise frequently when moving inside a typical building, DoorOpen and ButtonPress.

Summarizing, a classification label is a product of the three basic motion components, augmented with an auxiliary set of actions.

4.2 Segmentation

We segment the input data consisting of multiple data streams from the four sensors, and label each segment, or *frame*, with one of the motion labels defined in Section 4.1. After labeling process, a series of repeated labels over multiple frames is concatenated to a single motion before being presented to the map matching algorithm. In this work, we choose a base frame size as 333 ms (3 Hz).

One of the major challenges in finding a suitable segmentation interval is that different motions can have very different durations: short-term motions, such as turns or door-open actions, last no more than one or two seconds, whereas longer actions, such as walking or riding an elevator, can last for a few tens of seconds or more. Therefore, a large window size (e.g. 3 sec.) may include irrelevant signals around shorter motions, whereas a short window size may fail to capture the full characteristics of longer motions. For example, a single frame of 333 ms cannot even include a full single walking cycle at 1 Hz.

We solve this problem by computing multiple feature values from each feature function, by varying window widths. Then, we utilize CRF model to determine the degree of association between each window size and each feature function per motion. This *feature template* allows the motion labeling algorithm to adapt for an arbitrary association between a feature, a motion and its duration. However, those features evaluated from the same template are not independent from each other in general, thus violating the independence assumption required for certain classification models (e.g. hidden Markov models). Therefore, we use conditional random fields, which allow the use of such long-range and/or overlapping features. This idea is further explained in Section 4.4.

4.3 Conditional Random Fields

With the motion labels in Section 4.1, our task is to infer the most probable sequence of motion labels given time-series sensor data from the user device. As described in the previous section, we use a linear-chain conditional random fields (CRFs) [15], a class of probabilistic models for structured predictions.

Let y be a sequence of motion states and z denote time-series sensor data, segmented as described in Section 4.2. y_t and z_t denote t-th frame of the label and the observation, respectively. The CRFs model the conditional probability of a label sequence given data, $p(y|z)$, as follows [30]:

$$p_\lambda(y|z) = \frac{1}{Z_\lambda(z)} \exp\left(\sum_t \sum_i \lambda_i f_i(y_{t-1}, y_t, z, t)\right) \quad (1)$$

where $f_i(\cdot)$ is the i-th feature function representing compatibility, or the desired configuration, between two successive motion states y_{t-1}, y_t and sensor data z, and λ_i is the feature weight for f_i. Note that finding the most likely sequence of states in linear-chain CRFs can be done efficiently with the use of dynamic programming–based algorithms [30].

4.4 Motion Classification Features

Preprocessing We computed features from sensor measurements of four sensors: tri-axial accelerometer, gyroscope, magnetometer, and barometer. Before extracting features, sensor measurements were smoothed by two independent low-pass filters to remove noise as well as to capture slowly-varying characteristics of the input signal.

Gyroscope and magnetometer measurements were aligned to match the vertical axis (gravity direction) and the horizontal plane (the orthogonal plane to the gravity direction) using a tilt angle estimate from the accelerometer. In this way, lateral features can be extracted regardless of the device orientation. Also, magnetometer measurements were compensated for the iron offset caused by external magnetic disturbance that originates from electronic devices and metal furniture in indoor environment.

Feature Templates For labeling y_t for a certain time frame t, CRF models allow using evidences drawn from *any* frames, not only the observation within the t-th frame. That is, in computing feature functions, we can use observed signals from either a single frame ($\{y_t\}$), adjacent frames (e.g. $\{y_{t-3}, \ldots, y_{t+3}\}$), or even distant frames that do not include t (e.g. $\{y_{t-5}, \ldots, y_{t-3}\}$).

However, as noted in Section 4.2, complications arise when deciding how large a feature window should be used for each feature type. In general, no precise prior knowledge on the range of interactions between sensor measurements and a specific motion is available. Rather, our approach is to learn the degree of association for each feature type from data. To this end, we define multiple feature functions for every feature statistic with exponentially growing window sizes, allowing the model to learn an individual weight for each combination of a window size and a feature. For example, the variance of the acceleration magnitude is computed for five different window sizes: 1, 2, 4, 8 or 16 frames (1 frame = w = 333 ms), then the CRF learns the optimal weight for each feature function parametrized with a different window size. For most features, we use those five window sizes, resulting in five observation windows $[t-0.5w, t+0.5w]$ to $[t-8w, t+8w]$. We also quantize the range of feature values into five bins.

Consequently, we define feature templates such that each feature function derived from a certain feature type c is evaluated to one if and only if it is assigned a specific pair of labels (for previous and current labels) *and* it has a specific quantized feature value (one of five quantized values) computed from the observation window size τ:

$$\begin{aligned} f(y_{t-1}, y_t, z, t)|_{y', y'', z'^\tau_c} &= f^\tau_c(y_{t-1} = y', y_t = y'', z^\tau_c = z'^\tau_c, t) \\ &= \delta^\tau_c(y_{t-1}, y', t)\ \delta^\tau_c(y_j, y'', t)\ \delta^\tau_c(z^\tau_c, z'^\tau_c, t) \quad (2) \end{aligned}$$

where y' and y'' are motion label values at time $t-1$ and t respectively, z'^τ_c is a quantized feature value for feature c with window size τ, and $\delta^\tau_c(z, z', j)$ is a delta function evaluated to one only if $z = z'$ at time t. For some feature types, the feature function does not depend on the previous motion label. (i.e., $\delta^\tau_c(y_{t-1}, y', t)$ is always 1.)

Primay sensor	Feature
(Model prior)	State/transition bias
Accelerometer	Range/variance/median of magnitude
	Frequency of significant acceleration
	# of specific acceleration patterns
	(up-down/down-up/up-down-up/down-up-down)
	Peak frequency/magnitude of spectrum
Gyroscope	Average/minimum/maximum yaw rate
	Net change in yaw rate (from start to end of window)
	Frequency of significnant angular velocity
Magnetometer	Trend change in compass azimuth
Barometer	Trend change in atmospheric pressure
	Net change in pressure (from start to end of window)

Table 1: List of features for motion labeling

For example, the feature template for the variance of acceleration feature type ($c =$ var. accel.), will generate 25 feature functions per motion label, from all the combinations of five feature window sizes ($\tau \in \{1, 2, 4, 8, 16\}w$) and five quantized feature values per window (z'^{τ}_{c}). (Here we assumed that previous motion labels were not used for this feature type.)

List of Features From sensor measurements, we extract features from 20 feature types using feature templates from the preprocessed sensor measurements (Table 1). We design the features to constitute a complementary set of sources of information for motion classification. While some features overlap and are not independent with each other, CRF models can still benefit from having redundant features.

Ten features are extracted from the magnitude of the acceleration. The magnitude of the acceleration captures the overall energy of user motion. It is hence directly related to the classification of the principal component of user motions (Standing, Sitting, Walking, StairUp/Down, Running). Among the acceleration magnitude–based features, the range and the variance of the acceleration magnitude, in particular, reflects the overall strength of motion. Also, we include some predefined transient patterns in the acceleration magnitude as features, as they often indicate abrupt vertical movements, such as StandingUp, SittingDown, ElevatorUp/Down. For example, when a user stands up or sits down, a pair of abrupt changes, a quick increase in acceleration magnitude followed by a rapid decrease (or vice versa) over a brief period was observed.

The frequency-domain characteristics of user motions is captured by the 128-point FFT of the acceleration magnitude. The peak frequency and the corresponding magnitude conveys the information on walking motions in different frequency and strength (i.e. walking on flat surfaces *vs.* walking on staircases).

The features relevant to turn motions are extracted from the gyroscope and magnetometer. We characterize the angular velocity captured by instantaneous yaw rates in multiple ways, such as average yaw rate, or net change from the start to the end of the feature window. This distinction helps the classifier distinguish longer turns (e.g. U-turn) from shorter and sharp turns. On the other hand, magnetic field measurements themselves were inaccurate particularly in typical indoor environments, where there exist many sources of magnetic disturbances. Hence, we use magnetic measurements only as a supplementary source of information in addition to gyroscope measurements.

A ceiling height of a typical building ranges from 7 to 15 feet. This brings a difference in atmospheric pressure of about 50 Pa per floor. A barometer, which begins to be equipped in the current generation smartphones, can measure this difference precisely. The barometric features provide essential information in classifying vertical motions, helping the map matching algorithm identify salient places including elevators and stairs.

5 Trajectory Matching Model

This section describes our matching model formulated from the elements in Section 3. The model encodes the notion of path compatibility (§3.4) as a form of sequence labeling problem, in which each unit motion is assigned to some node or edge, which represents user location and direction, of the route network (Fig. 2c).

5.1 Hidden Markov Models

We represent the stated sequence labeling problem as an instance of Hidden Markov Models (HMMs), a well-known probabilistic sequential model [25]. Let $x_t \in X$ denote the state representing the "location," and $y_t \in Y$ denote the input motion observation at time t, $1 \leq t \leq T$, where T is the length (the total number of unit motions) of the input motion sequence, with index 0 used for the (possibly unknown) initial state. Our goal is to assign "location labels", i.e. direction-parameterized nodes or edges in the route graph, to the state sequence $x_{1:T} = \{x_1, x_2, ..., x_T\}$, while maximizing path compatibility with the input motion sequence $y_{1:T} = \{y_1, y_2, ..., y_T\}$.

The HMM provides a scoring mechanism to determine the compatibility between X and Y by defining the following joint distribution for a sequence of T observations:

$$p(x_{0:T}, y_{1:T}) = p(x_0) \prod_{t=1}^{T} p(x_t|x_{t-1}) p(y_t|x_t) \tag{3}$$

where the model consists of three components: *transition probabilities* $p(x_t|x_{t-1})$; *emission probabilities* $p(y_t|x_t)$; and an *initial distribution* $p(x_0)$. With no knowledge about the initial location, $p(x_0)$ is a uniform distribution over states.

HMMs achieve their computational efficiency by limiting interaction between X and Y; i.e, the current state x_t depends only on the previous state x_{t-1} (Markovian state evolution), and the current observation y_t is conditionally independent of the other states given the current state x_t. These restrictions, as expressed in Equation (3), have important implications for our case: user motions must be decomposed into a directional and a non-directional component, where directional properties (e.g. heading change by a turn) relate *two* states in time, while non-directional information (e.g. walk distance) defines compatibility between the associated motion and a *single* state. Hence, we rewrite Equation (3) to represent this factorization:

$$\begin{aligned} p(x_{0:T}, y_{1:T}) &= p(x_{0:T}, c_{1:T}, z_{1:T}) \\ &= p(x_0) \prod_{t=1}^{T} p(x_t|x_{t-1}, c_t) p(z_t|x_t) \end{aligned} \tag{4}$$

where c is the "control" component governing state transition according to the observed direction change, and z is the

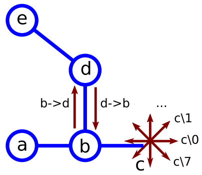


Figure 3: States for the route network of Fig. 2a. Edge `b-d` gives two edge-states `b->d` and `d->b`, and node `c` gives 8 node-states `c\0` to `c\7`.

"measurement" component that determines single-state compatibility. Sections 5.4 to 5.6 describe how each type of motion y defines its own transition and emission model according to Equation 4.

5.2 Input Model

An input is a sequence of motion descriptors, each of which models a unit action performed by the user while traveling indoors, as explained in Section 3.1. In this paper, we use the following subset of natural motion descriptors: {Rest, Standing, Walking, Turning, Stair Walking, Elevator Ride, Door Open}. Other labels produced by the motion tagger (§3.2), including Sitting, Rising, and Button Press, were removed, since they add little information to the descriptors above.

The motion labeler associates a direction with Turning and vertical motions: {Left, Right, Left U-Turn, Right U-Turn} or {Up, Down} respectively. Also, every motion has a duration, from which some important geometric quantities can be estimated: walking distance or turning angle. We do not estimate other physical quantities from sensor measurements, because values from the low-cost sensors in off-the-shelf mobile devices would require careful calibration to be usable. Instead, we opt to infer physical values only from motion durations, by assuming constant walking speed and discretizing heading to eight (egocentric) cardinal directions. Even though the resulting estimates have limited accuracy, our matching method is flexible enough to handle the significant uncertainty arising from estimation error. Moreover, our framework does not exclude the use of direct measurements.

5.3 State Model

Our state model represents instantaneous location and heading at the completion of each unit motion. The state model is derived from the route network (§3.3). Straight-line walking or vertical transitions are matched to route network edges, while other actions are considered to occur at point locations, so are matched to route network nodes.

Heading To represent instantaneous heading, we generate multiple states from each edge or node, one for each discrete heading (Fig. 3). For edges, because a user can walk from either direction, we derive *two* directional *edge-states* for each. Vertical connections between different floors are treated analogously, having two directional edge-states per connection. For nodes, because the user states after different rotation angles do not represent the same state in our problem (suppose the user starts walking after a rotation, then the next state will be dependent on the last heading angle), we quantize relative heading to produce eight different *node-states* for each graph node.

"Express" Edges A straight-line walk that spans more than one edge in the route graph must be matched to a series of edges, instead of one. For example, in Figure 3, walking from node `a` to `c` matches a series of two edge-states, `a->b` and `b->c`. To deal with this problem, the state graph is augmented with *express* edges. An express edge is generated from a series of edges that together form a nearly-straight path (e.g. `a->c` in Fig. 3). Express edges are computed by performing breadth-first search from each network node, while testing if a series of edges can be combined into a single straight path via the Douglas-Peucker criterion: tangential deviation from the approximated line below a threshold [4]. We call the original non-express edges *local* edges, an analogy adopted from subway networks. When generating edge-states, both types of edges are treated in the same manner.

Properties Each derived node- and edge-state inherits properties from the corresponding element in the route network and parent space. For instance, edge-states have a length property, the distance from one end to the other. Node-states are annotated with the type of the map object (e.g. room, door, stair, elevator) from which they arise. These annotations are used later during compatibility determination.

5.4 Matching Model: Horizontal Motions

This section describes the matching models for horizontal motions. The transition models, depending on the quantized angle input, have an identical form for all horizontal motions. We then give specifics for each class of horizontal motion.

Transition Model The transition model $p(x_t|x_{t-1}, c_t)$ determines possible current states from the previous state, based on the directional component c_t that the observed motion indicates. Since a user path must be continuous, we allow transitions only to nearby states that can be reached in *one* action. A stationary user on a node-state may change direction while staying in the same location (e.g. standing turn), or start to walk along a connected edge in the route graph. A walking user on an edge-state, on the other hand, must arrive at a node state; at one turn would be required to reach a different edge. Hence, we design the transition probability to have non-zero mass $p(x_t|x_{t-1}, c_t) \neq 0$ (making the corresponding graph element *"reachable"*), only under the following conditions:

- x_{t-1} = a node-state on node A
  $\Rightarrow x_t \in$ { all node-states on A *or*
  edge-states starting from A }
- x_{t-1} = an edge-state from node A to node B
  $\Rightarrow x_t \in$ { all node-states on B }.

In the absence of directional information, every reachable next state is assigned the same transition probability. However, the HMM formulation requires the sum of transition probabilities from each state to be one ($\sum_{x_t} p(x_t|x_{t-1}) = 1$). Some algorithms distribute one unit of probability over outgoing transitions [19]. However, in complex indoor environments where the number of outgoing edges differs significantly from place to place, this formulation inappropriately assigns high transition probabilities to low-degree connections. We overcome this problem using the approach of VTrack [33], which assigns a global constant for each transition probability, and uses a dummy state to keep the summed probability equal to one.

Specifically, let ζ be the maximum out-degree in the route graph. Then the *base transition probability* for each state reachable from a given state is $1/\zeta$. We incorporate directional information from a motion observation by discounting the base probability by a Gaussian *angle compatibility function* of the difference between the observed turn angle from the motion, ψ_{c_t}, and the expected turn angle between two states, $\psi_{x_{t-1} \rightarrow x_t}$:

$$f_{\text{angle}}(x_t, x_{t-1}, c_t) = \exp\left\{-\frac{(\psi_{c_t} - \psi_{x_{t-1} \rightarrow x_t})^2}{\sigma_a^2}\right\} \quad (5)$$

where σ_a is the *angle compatibility parameter* controlling the extent to which the matching algorithm allows angle mismatch (a higher value allows more matching freedom).

Summarizing, the transition probability for a motion with control component c_t (having turn angle ψ_{c_t}) is defined as:

$$p(x_t|x_{t-1}, c_t) = \begin{cases} \frac{1}{\zeta} \cdot f_{\text{angle}}(x_t, x_{t-1}, c_t) & x_t \text{ reachable from } x_{t-1} \\ 1 - \nu & x_t = \text{"dead-end" state} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where $\nu \leq 1$ is the sum of transition probabilities to reachable next states:

$$\nu = \sum_{x_t:\text{reachable}} p(x_t|x_{t-1}, c_t). \quad (7)$$

As in VTrack, the remaining probability mass $1 - \nu$ flows to the dead-end state, which has no further outgoing transitions and is incompatible with any further input motion. Any trajectory arriving at the dead-end state will have zero compatibility during decoding, and thus no chance of selection.

Emission Probability The emission probability $p(z_t|x_t)$ is determined indirectly by setting the posterior probability of a state given the observation, $p(x_t|z_t)$, which is more natural to consider in our framework. For a fixed, known input motion z_t, specifying the compatibility functions between x_t and z_t in the posterior form $p(z_t|x_t)$ is equivalent to setting them in the original form $p(z_t|x_t)$ under proper normalization. For convenience, we refer to the posterior form as an "emission probability."

Rest and Standing Whenever a non-turning motion, such as Rest (Sitting), Standing, or (straight-line) Walking is observed, the turning angle observation is set to zero, $\psi_{c_t} = 0$. This effectively penalizes path hypotheses having a non-zero turn at that time with compatibility as determined by Equation (5).

Since neither Rest nor Standing involve any transitional motions, the emission probability (posterior probability) simply ensures that the current state must have type node-state, not edge-state:

$$p(x_t|z_t = \text{stationary}) \propto \begin{cases} 1 & x_t = \text{node-state} \\ 0 & x_t = \text{edge-state} \end{cases} \quad (8)$$

A more sophisticated model would distinguish ways of being stationary from context, exhibiting a stronger preference for certain space types when Rest (vs. Standing) is observed.

Straight-line Walking Like stationary motions, the transition probability for Walking follows the turn-angle-based formula (Eq. 6) with zero observed turn angle $\psi_{c_t} = 0$.

For emission probability, unlike stationary motions, the walking motion must be matched on a local or express edge-state, not a node-state. We also incorporate walking distance compatibility here. We model human walking speed as a normal distribution centered at a constant μ_s with variance σ_s^2, i.e. as $\mathcal{N}(\mu_s, \sigma_s^2)$. The distribution of a "projected" walking distance for a straight-walk of duration Δt is then $\mathcal{N}(\mu_s \Delta t, \sigma_s^2 \Delta t^2)$. However, we observed that for short walking motions, duration estimates derived from the motion labeler's automatic segmentation were often inaccurate, causing underestimation of true walking duration. Thus using the estimate as is will produce an inappropriately small variance of the projected distance, leading to overly narrow regions of compatibility compared to the granularity of the route network map. Therefore, we set a minimum variance for the projected walking distance, $\sigma_d^{\min}$, to prevent the variance from collapsing. The *distance compatibility function* between a straight-line walk motion with duration Δt and an edge-state x_t is then defined as:

$$f_{\text{dist}}(x_t, z_t = \text{walk}) = \exp\left\{-\frac{(l_{x_t} - \mu_s \Delta t)^2}{2\sigma_d^2}\right\} \quad (9)$$

$$\sigma_d = \max(\sigma_s \Delta t, \sigma_d^{\min}) \quad (10)$$

where l_{x_t} is the length of the edge-state x_t. We set $\sigma_d^{\min}$ to 3 ft (≈ 0.91m) for our corpus to match the approximate granularity of the route network. Finally, the emission probability for a straight-line walk is:

$$p(x_t|z_t = \text{walk}) \propto \begin{cases} f_{\text{dist}}(x_t, z_t) & x_t = \text{edge-state} \\ 0 & x_t = \text{node-state.} \end{cases} \quad (11)$$

Turn We model turns as point motions in which the user changes only heading direction while staying on one route graph node (and experiencing no net translation). Each turn observation is quantized to a multiple of $\pi/4$ to produce a value ψ_{c_t}. The transition and emission models are identical to those for stationary motions (Eqs. 6 & 8) except that each turn involves a non-zero heading change.

5.5 Matching Model: Vertical Motions

Vertical motions, including stair and elevator transitions, are associated with vertical edges that connect different floors in the route graph.

Elevators Elevator ride motions provide three pieces of information to be exploited during matching: *type*, *direction*, and *length*. First, only vertical edges associated with an elevator can be matched to an elevator ride motion. Second, the ride direction (up or down) determines the direction of the edge-state to be matched in the transition model. Last, the number of floor transitions, which constrains the length of the vertical edge, is determined from the ride duration. The transition model for vertical motions (including elevator ride and stair walk) is:

$$p(x_t|x_{t-1}, c_t) = \begin{cases} \frac{1}{\eta} & x_{t-1} \rightarrow x_t \text{ matches } c_t \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

where η is the maximum out-degree among vertical transitions in the route graph (analogous to ζ for horizontal transitions).

Essentially, the transition model determines next possible states based on the space type (vertical) and direction.

For the emission probability, the number of floor transitions or "vertical moving distance" is probabilistically determined by a compatibility function analogous to the distance compatibility function (Eq. 9). The number of floor transitions is estimated from the duration, and matched with the floor difference implied by each vertical edge, using a Gaussian compatibility function.

Stairwells Like elevator rides, stair ascents and descents are matched by their associated type and properties. In principle, traversing a staircase could be treated in the same manner as level walking, if the detailed shape of all staircases in the corpus were known. For instance, if the precise shape of each spiral stair including the number of treads and spiral direction were known *a priori*, we could match every fine-grained submotion of a stair walk to an individual stair segment. However, our floorplans do not model stairwells with such precision, representing them instead simply as rooms with type "stair."

In light of this limitation, we instead summarize a series of fine-grained stair motions into a single, abstract stair motion starting at one end of a stairwell and ending at the other. Our system parameterizes stair transits by vertical direction (up / down), spiral direction (clockwise / counter-clockwise / straight), and duration as in elevator rides. (We manually annotated the spiral direction of each staircase in our corpus.) These properties are used in matching similarly to elevator ride motions; vertical direction is incorporated in the transition model, while length and spiral sense are used in the emission model. We model half-stairs differently from full-stairs, because the shorter transition through a half-stair can be easily missed by the motion labeler, causing a missed detection. We avoid this by allowing half-stairs to match straight-walk motions.

5.6 Matching Model: Special Cases

Door Open We use detected door-open actions to constrain the user path when matching. Every door in the floorplan has an associated node in the route graph, from which states are generated. The detection of a door-open action indicates that the user *is likely to be* in one of these node-states.

However, we do not treat the door-open observation as a hard constraint; as are half-stairs, door-open actions are often confused with similar actions by the low-level motion labeler. Instead, the matching algorithm has the flexibility to violate the door constraint if necessary. To that end, we assign a small non-zero probability to non-door states even when a door-open action is detected:

$$p(x_t|z_t = \text{door}) \propto \begin{cases} \alpha & x_t = \text{door node-state} \\ 1 & x_t = \text{non-door node-state} \\ 0 & x_t = \text{edge-state} \end{cases} \qquad (13)$$

where $\alpha >> 1$ is a ratio encoding a preference for door node-states when a door-open action is detected. Larger values of α make the matcher less likely to violate door constraints.

Long Walks Though we introduced express edges to handle straight-line walks spanning multiple edges, very long walks might not be handled well even with this mechanism. Suppose for example that the user has walked for a minute, traversing a 100-meter corridor. If the path is slightly curved, the motion classifier would fail to detect the curvature. This single straight-line Walking observation would have to be matched to a long series of successive edges in the route network. Automatically creating express edges for such long edges would introduce many additional edges to the state graph, increasing the computational complexity of matching. A simple solution to this problem is to split lengthy walks into smaller intervals, each of which can then be matched to an express or local edge. Our method uses a threshold of 15 seconds (about 20 meters of walking).

6 Algorithms

In Section 5, we modeled the trajectory matching problem from a user motion sequence using the HMM formulation. In this section, we present algorithms for recovering user trajectories and estimating model parameters from unannotated user motion data.

6.1 Matching Algorithm

To decode a user trajectory in the HMM model with Equation (4), we use standard methods: forward-filtering to compute the distribution of the most recent state x_T (as in conventional particle filters for localization), forward-backwards algorithm to compute "smoothed" distributions of x_t in the past ($1 \leq t \leq T$), and the Viterbi algorithm to find the "most likely" trajectory $x_{1:T}$ [25].

In this paper, we use the Viterbi algorithm to compute the most likely continuous trajectory. Smoothed estimates computed by the forward-backward algorithm are similar to those from Viterbi, but are not guaranteed to be spatially continuous. Unlike conventional particle filters that update only the last position estimate upon each new input, the most-likely-trajectory approach updates the entire path. Also, the Viterbi algorithm can easily be modified to yield the k-best state sequences instead of the single best, along with matching scores. The score gap between the best sequence and the rest can be used to gauge uncertainty in the current estimates.

In practice, the algorithm should be implemented to exploit sparsity of our state model rooted on route networks. Because a user path without a significant rest period must be continuous, the number of possible transitions from a specific location is physically bounded by the maximum out-degree in the state graph. With N states and an input sequence of length T, sparsity yields a time complexity of $O(NT)$ for the matching algorithm, rather than $O(N^2T)$ for non-sparse models. The complexity of computing transition and emission models is $O(N)$ per unit motion.

6.2 Parameter Learning

The matching models require specification of a few parameters. These include include physical parameters, such as walking speed constant μ_s (Eq. 9), and other parameters that determine association strength between motions and user paths (σ_a, σ_d, and α in Eqns. 5, 9 and 13, resp.). Some of these parameters have a physical interpretation, which provide a basis for setting a value in the model. For example, from prior knowledge of average human walking speed, we can determine μ_s empirically.

In this section, however, we show how to determine these parameters automatically from *unannotated* data – motion sequences with unknown location labels – using a variant of the Expectation-Maximization (EM) algorithm. This process can be used to learn individually-tuned parameters for a dataset from a single user, or alternatively to learn good parameters for a dataset captured from the motions of many users.

Intuitively, for a given motion dataset, some parameter settings will produce better trajectories than others in terms of explanatory power. For instance, the best path found by setting the walking speed to 5 km/h (average human walking speed) is more "plausible" than the path found by setting it to 1 km/h. Given n data sequences $\{y^i_{1:T_i} | 1 \le i \le n\}$, we search for the parameters Θ^* and paths ${x^i_{1:T_i}}^*$ that maximize joint probability of the HMMs:

$$\Theta^*, {x^1_{1:T_1}}^*, \ldots, {x^n_{1:T_n}}^* \leftarrow \underset{\Theta, x^1_{1:T_1}, \ldots, x^n_{1:T_n}}{\operatorname{argmax}} \prod_{i=1}^{n} p(x^i_{1:T}, y^i_{1:T}; \Theta). \quad (14)$$

This optimization problem is solved by the hard-EM algorithm (also known as Viterbi training or segmental K-means) [10]. It finds the best parameters (along with the best paths) in coordinate ascent manner. First, the parameters Θ are fixed, and the best paths are found using the Viterbi algorithm. Next, the estimated paths $x^i_{1:T_i}$ are treated as ground truth, and new parameters are estimated. This process is iterated until convergence:

1. Initial parameters: $\Theta(0)$.
2. Step $\tau = 1, 2, \ldots$, repeat until convergence ($\Theta_{\tau-1} \approx \Theta_\tau$):
   (a) Given $\Theta(\tau - 1)$, find paths $x^i_{1:T}(\tau)$ using the Viterbi algorithm;
   (b) Estimate new parameters $\Theta(\tau)$ from inputs and decoded labels at τ: $\{x^i_{1:T}(\tau), y^i_{1:T} | 1 \le i \le n\}$.

Since the optimization problem in Equation (14) is not convex in general, and hard-EM does not guarantee identification of the global optimum, care must be taken in using this approach. Therefore, it is helpful to guide the learning process by providing a good set of initial parameters and by limiting the range of each parameter, using common-sense knowledge of human motions.

7 Evaluation

7.1 Experimental Methodology

We collected 30 single- and multi-floor motion sequences over seven days. Of these, 21 traces contained at least one vertical transition (elevator or stair transit). The total length of the dataset is 58 minutes, with an average sequence length of about 2 minutes. Our deployment area was a nine-story, 67,000 m^2 office and lab complex. We used four floors of the building, from the ground level up to the fourth floor. Coverage included most corridors and many office and lab spaces.

We developed data logging software running on a Nokia N900 mobile phone for motion sensing. The N900 host connects to an external sensing unit, containing five consumer-grade MEMS sensors in a $4 \times 4 \times 1$cm package: a tri-axial accelerometer, tri-axial gyroscope, barometer, thermometer, and tri-axial magnetometer. We chose to use the external sensors because the N900 did not provide sensors that more recent smartphones support, other than the accelerometer. However,

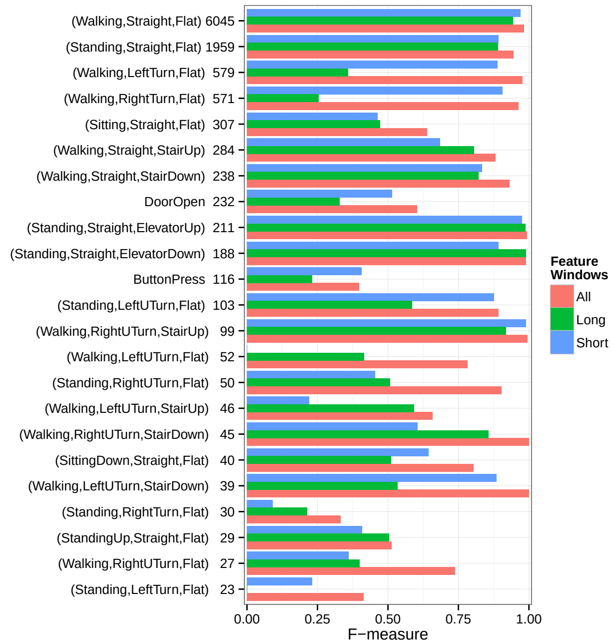


Figure 4: Motion labeling performance by different feature window schemes. The motions are sorted by descending order of occurrence (the number of segments) in the training data set.

the sensing unit featured equivalent set of MEMS sensors and was rigidly attached to the host smartphone. We also verified that its sensing characteristics are identical to that of built-in sensors.

Our low-level motion classifier performed satisfactorily on uncalibrated sensor data (i.e. with unit conversion factors left as factory defaults). The data logging module connected to the sensing unit via Bluetooth and continuously sampled time-stamped sensor data, which were then offloaded to a desktop computer for offline analysis. Our evaluation pipeline first predicted user motions from the sensor data in leave-one-trajectory-out manner, then matched each labeled motion sequence to a trajectory.

We determined ground-truth trajectories by annotating video of the user and background recorded by the host smartphone. We used a custom GUI to indicate the type and duration of each low-level motion, and its location on the map; locations and motion types were interpolated at fine grain from these annotations.

7.2 Motion labeling: Feature Templates

We first evaluated the effectiveness of the feature templates (§4.4) on the fine-grained labeling of indoor motions. The central idea of using feature templates is to generate multiple features with varying window sizes from the same feature function. To highlight its effectiveness, it was compared to the cases in which only a fixed window size was used for feature computation. We selected a very short feature window (consisting of a single frame, 0.33 seconds, except for the features from low-rate sensors, such as a barometer), and a long feature window (16 frames or 5.33 seconds) for comparison. On the other hand, our feature template method generates all the possible combinations of features and window sizes, and let the

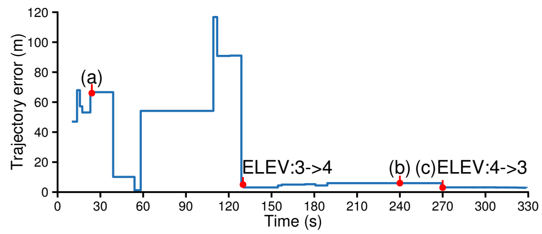


Figure 6: Trajectory error for the first example trace (§7.3)

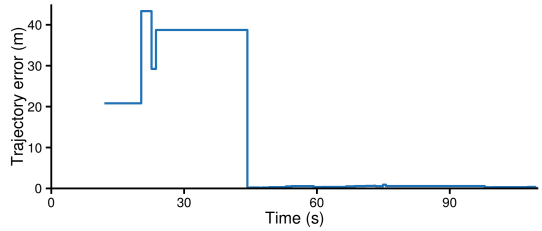


Figure 7: Trajectory error for the second example

CRF model learn the optimal weight for each combination.

Figure 4 shows the classification performance. We computed frame-by-frame f-measure (harmonic mean of precision and recall) as a performance metric (i.e., the unit of evaluation is a frame of 0.33 seconds).

Overall, the f-measure of our labeling method using feature templates was 94.5% (95.2% precision and 93.8% recall), which were significantly higher than 89.0% (90.2% precision and 88.5% recall) of short feature windows (*short* in Fig. 4) or 81.4% (86.9% precision and 79.9% recall) of long, 16 frame feature windows (*long* in Fig. 4). Notably, when only short windows were used, classification error was higher for long motions: U-turns were confused with right turns; vertical motions, especially stair motions were misclassified as the short windows did not contain enough information to compute precise barometric pressure gradient, an essential feature for detection of vertical motions. In contrast, when only long windows were used, there was remarkable performance degradation for transient motions, in particular turns. The classification accuracy for access motions (DoorOpen and ButtonPress) degraded as well.

While the classification accuracy was very high for many essential motions with feature templates, our labeling method still had difficulty in classifying certain motions that were either rare or inherently difficult to recognize with measurements from mobile phones. However, our probabilistic map matching method is designed to be robust to such inaccuracies in input, by considering them only as "soft constraints." (See e.g. Eq. 6 or 13)

7.3 Trajectory Matching Examples

We illustrate the trajectory matching algorithm's behavior using matching results from two test traces. We measured three-dimensional error in which a vertical error due to an incorrect floor estimate was considered to be 15 meter/floor.

In the first example (Figs. 5 & 6), the user started from an office on the third floor, transited to the fourth floor via elevator, walked across the fourth floor, and returned to the starting office via a different elevator. No information was

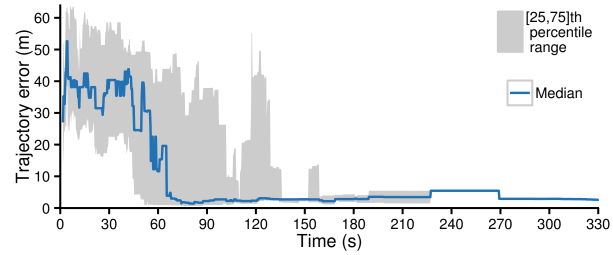


Figure 8: Overall trajectory error (median and interquartile range) over time

known about the user's starting location.

Figure 5 shows a few snapshots of the trace over time. We compute trajectory error at time t (Fig. 6) by computing pointwise error at each frame (3 Hz) until t, then taking the median as the representative trajectory error at t. (The stair-step nature of the error plots arises from the fact that the algorithm computes a new trajectory estimate whenever a new unit motion becomes available.)

Initially, not enough information is available to constrain the user path; the user has made only one significant turn, classified as Right U-Turn. Hence, there were many plausible embeddings of this short motion sequence throughout the corpus (Fig. 5a). The method started to find the correct path before the user took an elevator to the fourth floor ("`ELEV:3->4`" in Fig. 6). The user next walked faster than average, resulting in a large disparity between the true and estimated walking distance. This made the algorithm select an incorrect, alternative path, which better matched the distance estimate while sacrificing angle compatibility (Fig. 5b). The correct path was eventually recovered as the user took the elevator, providing strong evidence of location. This example shows how the algorithm recovers the correct path history from a transient failure by incorporating stronger constraints.

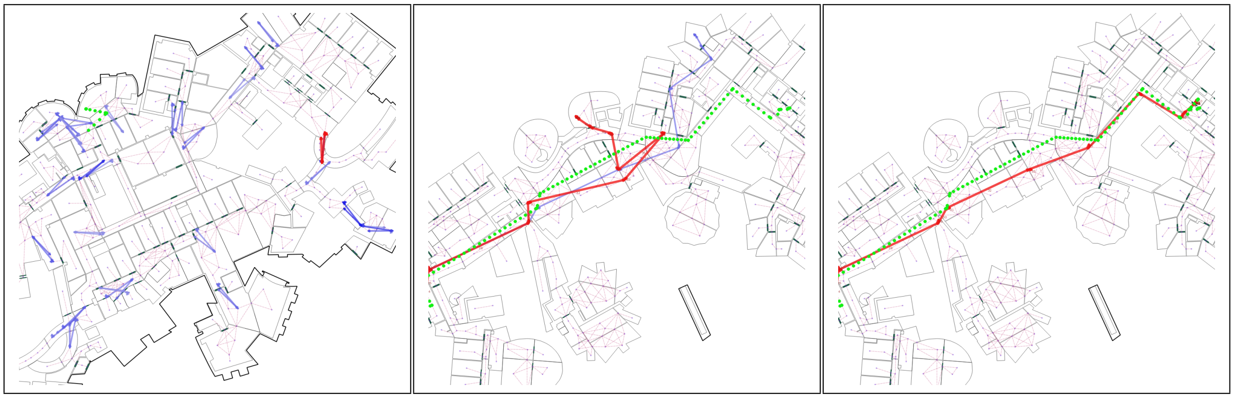
We show another example in which the algorithm matches the true path after 45 seconds of walking (Fig. 7). Though the user walked on a single floor, the true path was sufficiently distinctive to enable rapid convergence, yielding the true trajectory after a few turns.

7.4 Ensemble Trajectory Error

We evaluated the matcher's performance by computing trajectory error over all sequences in the dataset. Figure 8 shows the progression of trajectory error statistics (median and interquartile range) over time. The x-axis includes time spent for all motions, not only for walking. The matching algorithm is typically highly uncertain until it reaches a "tipping point" at which enough information is available to constrain the user path with high accuracy. For more than half of the test traces, the algorithm started to match an input motion sequence on the correct path within about one minute, and for almost all traces within about two minutes, similar to the results of Rai et al. [27]. Note that the exact tipping point is also a function of the characteristics of the corpus and the underlying motion labeling algorithm, rather than solely of the matching algorithm.

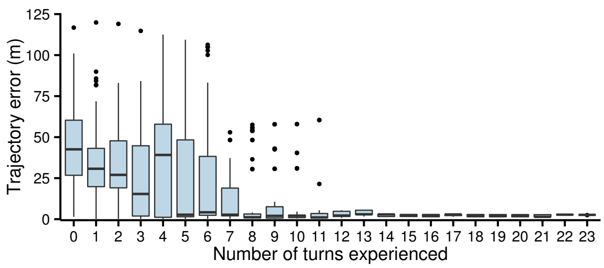
7.5 Salient Features

Certain motion classes, when detected, provide strong constraints on the user path. Navigation-related events such as

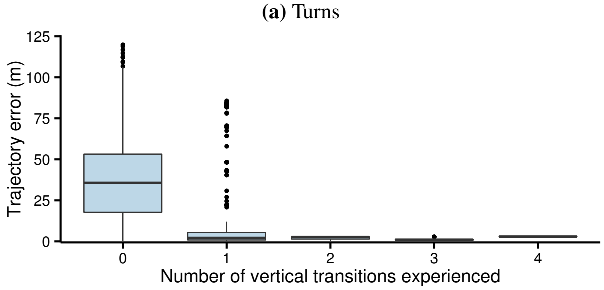


(a) $t = 24$ sec, 3rd floor **(b)** $t = 240$ sec, 4th floor **(c)** $t = 269$ sec, 4th floor

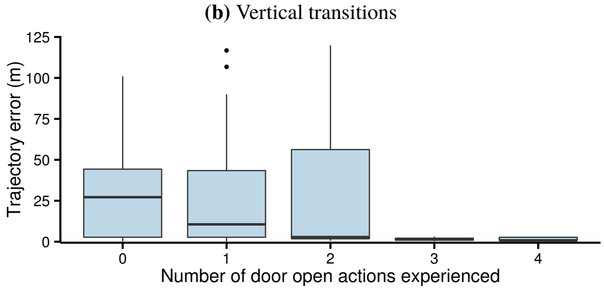
Figure 5: Matching example (green: ground truth; red: best path found; blue: other path candidates): (a) after only one right turn, many plausible paths; (b) before elevator transit, matching drifted due to noisy walking distance estimate; (c) after elevator transit, matching algorithm corrected the entire path.



(a) Turns



(b) Vertical transitions



(c) Door-open actions

Figure 9: Error decreases when salient motions are observed.

vertical transitions or door-open actions limit the user path to a small number of candidates, as there are only a few ways in which such motions can occur within the provided route graph.

We confirm this intuition by measuring the trajectory error after the user experienced a certain number of each kind of *salient* motion: turn, vertical transitions (elevators / stairs),

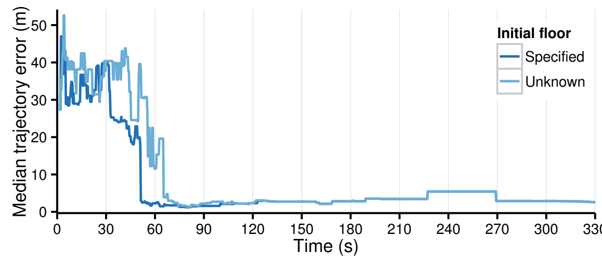


Figure 10: Trajectory error is lower when the initial floor is provided.

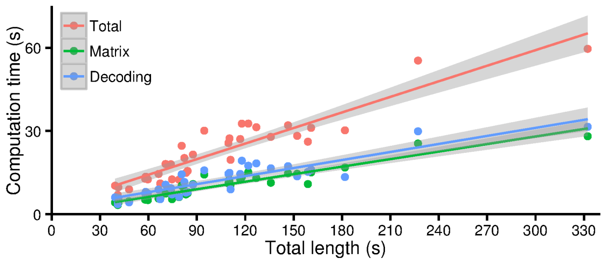
and door-open actions. For each trajectory, we computed the total number of (ground-truth) salient motions that had been experienced in that trajectory as of each sensor frame. Note that the low-level motion classifier sometimes fails to detect these features; missed actions will not contribute to matching.

The results (Fig. 9) confirm that distinctive motions facilitate matching by providing additional constraints on possible user paths. However, different motions contribute by different degrees; constraints provided by vertical transitions are strongest among the motion classes tested (Fig. 9b). This is because vertical motions were detected very reliably by our classifier, and stairwells and elevators occur rarely in our building (as in most buildings) compared to ordinary rooms. On the other hand, door-opens, which were often missed or misclassified, were a less reliable source of information (Fig. 9c). Turns were weakest among three, because the angle compatibility function (Eq. 5) allows relatively broad freedom in matching, enabling many plausible paths on the map until a certain number of distinct turns were accumulated (Fig. 9a).

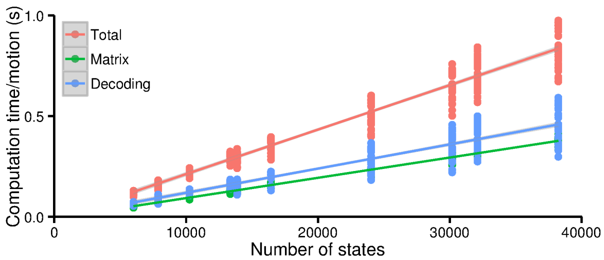
7.6 Prior Information

Prior information other than motion sequences could be valuable for matching. If available, such information could potentially make the matching process converge faster to the true user path. Our trajectory matching formulation admits various types of prior information to be incorporated in the model, e.g. by setting an initial state distribution or by pre-filtering candidate locations.

We assessed the method under a hypothetical scenario in which the starting floor (but not the precise location) is known



(a) Cumulative computation time as a function of input length



(b) CPU time per unit motion increases with the number of states.

Figure 11: Time for Matrix and Viterbi ("Decoding") computations

beforehand. Floor information may be available from a different sensor (e.g. WiFi, or barometer [5]), or by taking user attributes (e.g. office number or calendar contents) into account. To test this scenario, we initialized the state distribution $p(x_0)$ uniformly over the known starting floor.

Figure 10 compares error with and without starting floor information. With the initial floor known, the bootstrapping time required to yield median trajectory error below 5 meters was 51 seconds, 14 seconds faster than the 65 seconds required when the initial floor was unknown. This improvement was due to better matching for previously ambiguous test paths that did not traverse multiple floors. We expect starting floor information to be particularly salient when the user motion trajectory has no or few vertical transitions and different floors have spatially similar layouts.

7.7 Computation Time

We studied the computational complexity of our matching algorithm. As explained in Section 6.1, the Viterbi decoding algorithm with a sparse model has time complexity of $O(NT)$ rather than $O(N^2T)$, for N states and T timesteps. The complexity of transition and emission matrix computation is also $O(NT)$, or $O(N)$ per unit motion. Online, computation of the model matrices is amortized; each matrix is computed only once and stored for reuse in later executions.

Our unoptimized Python/SciPy implementation exhibited update times linear in the input sequence length (Fig. 11a). Its running time scaled linearly with the number of states (Fig. 11b), which we varied by selecting subsets of the map.

8 Discussion

Our method recovered user paths successfully for most cases during the test deployment from only a discrete motion descriptors without any absolute positioning fixes such as RF scans. However, our method as presented in this paper has several limitations.

In our experiments, we had an instance in which no turns were detected by the motion classification algorithm when the user walked over a gently curving bridge. The bridge did not contain any sharp turns (which caused the motion classifier to fail to detect turns) but the net difference in heading before and after crossing the bridge was nearly 90 degrees. In that case, the motion over the bridge was recovered as a single, long straight-line walk, which the map matching algorithm tried to match to long corridors rather than to the true bridge path. We anticipate that this problem could be alleviated by introducing an additional motion model describing such situations (i.e. Gentle Turn).

Also, as demonstrated in Section 7.5, the absence of salient features may cause the method to fail, or may delay acquisition of the current location. Likewise, if floor layouts in the building are nearly identical, our method *per se* may fail to identify the correct floor as there will be multiple paths with the same posterior probability. In this case, the method can resort to "external information" such as one-time WiFi scanning or prior information about the user (e.g. user's office location) to constrain the true location among these possibilities (§7.6).

In future work, we anticipate that the current model can be expanded in a variety of ways to become more expressive while handling uncertain cases more gracefully. For example, the matching model can exploit, or even infer, a user's activity pattern in indoor environments. Because people tend to traverse and use spaces in similar fashion, there exists a natural association or "activity map" between activities and space types. While at present we use such cues only for vertical motions, associating them with stairs or elevators, this mapping can be generalized to handle other motion classes, e.g. sitting in an office, or walking in a corridor, by defining emission probabilities that capture the corresponding associations. Conversely, the trajectory matching algorithm could be used to learn (unknown) associations from user data by bootstrapping, as it can proceed given only geometric and topological compatibilities, without requiring or using semantic information. This learned association could then be used to facilitate later matching processes, creating a closed loop between trajectory estimation and activity map learning.

Another way to extend the model would be through joint estimation of multiple paths from a single user or multiple users. At a single-user level, each user tends to repeat some paths, exhibiting a few distinctive motion patterns, or often returns to a few select locations. At a multi-user level, each user will encounter or accompany others, having either brief or substantial overlap with other users' paths. We anticipate that such information can be collected from multiple paths, with the aid of device proximity sensors (e.g. ad hoc WiFi or Bluetooth), and can be incorporated into the model as a form of "second-order" compatibility.

9 Conclusion

We described a fine-grained indoor localization and trajectory estimation method based on the notion of path compatibility: probabilistic agreement between user motions and a prior map encoded as a route graph. We defined user motion traces as a sequence of navigation-related actions, which were automatically estimated by a classification algorithm from proprioceptive sensor data. The method assumes route networks (derived automatically) from legacy floorplans of the deployment area. With these inputs, we defined an HMM-based matching model that encodes the compatibility between motions and locations, and used the model to decode the most likely trajectory given an input motion sequence. We also showed how to learn the model parameters from unannotated data.

Testing on user data from four floors of a deployment area demonstrates that our method can recover the user's location trajectory to within a few meters using only proprioceptive sensor data from a commodity mobile device. However, our method fails where the uncertainty of the user motion and/or prior map is large. As future work, we anticipate that the current model can be extended to handle more general classes of motions and to incorporate more expressive semantic associations between motions and spaces.

10 References

[1] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *INFOCOM*, pages 775–784, Mar. 2000.

[2] A. J. B. Brush, A. K. Karlson, J. Scott, R. Sarin, A. Jacobs, B. Bond, O. Murillo, G. Hunt, M. Sinclair, K. Hammil, and S. Levi. User Experiences with Activity-Based Navigation on Mobile Devices. In *Mobile-HCI*, pages 73–82, 2010.

[3] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan. Indoor Localization Without the Pain. In *MobiCom*, 2010.

[4] D. H. Douglas and T. K. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature. *Cartographica*, 10(2):112–122, 1973.

[5] M. Fallon, H. Johannsson, J. Brookshire, S. Teller, and J. Leonard. Sensor Fusion for Flexible Human-Portable Building-Scale Mapping. In *IEEE/RSJ IROS*, 2012.

[6] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. Practical Robust Localization over Large-Scale 802.11 Wireless Networks. In *MobiCom*, pages 70–84, Sept. 2004.

[7] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang. ACComplice: Location Inference using Accelerometers on Smartphones. In *COMSNETS*, pages 1–9, 2012.

[8] M. Hardegger, D. Roggen, S. Mazilu, and G. Troster. ActionSLAM: using location-related actions as landmarks in pedestrian SLAM. In *IPIN*, 2012.

[9] T. Huynh, M. Fritz, and B. Schiele. Discovery of activity patterns using topic models. In *UbiComp*, pages 10–19, 2008.

[10] B. Juang and L. R. Rabiner. The Segmental K-Means Algorithm for Estimating Parameters of Hidden Markov Models. *IEEE Trans. Acoustics, Speech and Signal Processing*, 38(9):1639–1641, 1990.

[11] A. Kalia, G. Legge, and N. Giudice. Learning Building Layouts with Non-geometric Visual Information: The Effects of Visual Impairment and Age. *Perception*, 37(11):1677, 2008.

[12] L. Klingbeil and T. Wark. A Wireless Sensor Network for Real-time Indoor Localisation and Motion Monitoring. In *IPSN*, pages 39–50, 2008.

[13] J. Krumm, J. Letchnet, and E. Horvitz. Map Matching with Travel Time Constraints. In *SAE 2007 World Congress*, 2007.

[14] J. Krumm and J. Platt. Minimizing Calibration Effort for an Indoor 802.11 Device Location Measurement System. Technical Report MSR-TR-2003-82, November 2003.

[15] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *IMCL*, pages 282–289, 2001.

[16] J. Lester, T. Choudhury, N. K. G. Borriello, and B. Hannaford. A hybrid Discriminative/Generative approach for modeling human activities. In *IJCAI*, volume 5, pages 766–772, 2005.

[17] M. Levit and D. Roy. Interpretation of Spatial Language in a Map Navigation Task. *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(3):667–679, 2007.

[18] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. A Reliable and Accurate Indoor Localization Method Using Phone Inertial Sensors. In *UbiComp*, pages 421–430, 2012.

[19] L. Liao, D. Fox, J. Hightower, H. Kautz, and D. Schulz. Voronoi Tracking: Location Estimation Using Sparse and Noisy Sensor Data. In *IEEE/RSJ IROS*, 2003.

[20] C. Matuszek, D. Fox, and K. Koscher. Following Directions Using Statistical Machine Translation. In *HRI*, pages 251–258, 2010.

[21] J. Park, B. Charrow, D. Curtis, J. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie. Growing an Organic Indoor Location System. In *MobiSys*, pages 271–284, June 2010.

[22] J. Park, A. Patel, D. Curtis, S. Teller, and J. Ledlie. Online pose classification and walking speed estimation using handheld devices. In *UbiComp*, pages 113–122. ACM, 2012.

[23] N. B. Priyantha, A. K. L. Miu, H. Balakrishnan, and S. Teller. The Cricket Compass for Context-Aware Mobile Applications. In *MobiCom*, pages 1–14, 2001.

[24] M. Quigley, D. Stavens, A. Coates, and S. Thrun. Sub-Meter Indoor Localization in Unmodified Environments with Inexpensive Sensors. In *IEEE/RSJ IROS*, 2010.

[25] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications In Speech Recognition. *Proc. IEEE*, 77(2):257–286, 1989.

[26] V. Radu and M. K. Marina. HiMLoc: indoor smartphone localization via activity aware pedestrian dead reckoning with selective crowd-sourced WiFi fingerprinting. In *IPIN*, volume 28, 2013.

[27] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-Effort Crowdsourcing for Indoor Localization. In *MobiCom*, pages 293–304, 2012.

[28] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *AAAI*, volume 20, pages 1541–1546, 2005.

[29] P. Robertson, M. Angermann, B. Krach, and M. Khider. Inertial Systems Based Joint Mapping and Positioning for Pedestrian Navigation. In *ION GNSS*, 2009.

[30] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*, pages 93–128. The MIT Press, 2007.

[31] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy. Approaching the Symbol Grounding Problem with Probabilistic Graphical Models. *AI Magazine*, 2011.

[32] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In *NSDI*, 2011.

[33] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Toledo, J. Eriksson, S. Madden, and H. Balakrishnan. VTrack: Accurate, Energy-Aware Road Traffic Delay Estimation Using Mobile Phones. In *SenSys*, 2009.

[34] J. Wallgrün. Autonomous Construction of Hierarchical Voronoi-Based Route Graph Representations. *Spatial Cognition IV. Reasoning, Action, and Interaction*, pages 413–433, 2005.

[35] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No Need to War-Drive: Unsupervised Indoor Localization. In *MobiSys*, pages 197–210. ACM, 2012.

[36] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The Active Badge Location System. *ACM Trans. Information Systems*, 10(1):91–102, 1992.

[37] E. Whiting, J. Battat, and S. Teller. Topology of Urban Environments. In *CAADFutures*, pages 114–128, 2007.

[38] O. Woodman and R. Harle. Pedestrian Localisation for Indoor Environments. In *UbiComp*, pages 114–123, 2008.

[39] C. Wu, Z. Yang, Y. Liu, and W. Xi. WILL: Wireless Indoor Localization Without Site Survey. In *INFOCOM*, pages 64–72, 2012.

[40] M. Youssef and A. Agrawala. The Horus WLAN Location Determination System. In *MobiSys*, pages 205–218, 2005.