AutoWitness: Locating and Tracking Stolen Property While Tolerating GPS and Radio Outages

SANTANU GUHA, KURT PLARRE, DANIEL LISSNER,
SOMNATH MITRA, and BHAGAVATHY KRISHNA, University of Memphis
PRABAL DUTTA, University of Michigan
SANTOSH KUMAR, University of Memphis

We present AutoWitness, a system to deter, detect, and track personal property theft, improve historically dismal stolen property recovery rates, and disrupt stolen property distribution networks. A property owner embeds a small tag inside the asset to be protected, where the tag lies dormant until it detects vehicular movement. Once moved, the tag uses inertial sensor-based dead reckoning to estimate position changes, but to reduce integration errors, the relative position is reset whenever the sensors indicate the vehicle has stopped. The sequence of movements, stops, and turns are logged in compact form and eventually transferred to a server using a cellular modem after both sufficient time has passed (to avoid detection) and RF power is detectable (hinting cellular access may be available). Eventually, the trajectory data are sent to a server which attempts to match a path to the observations. The algorithm uses a Hidden Markov Model of city streets and Viterbi decoding to estimate the most likely path. The proposed design leverages low-power radios and inertial sensors, is immune to intransit cloaking, and supports post hoc path reconstruction. Our prototype demonstrates technical viability of the design; the volume market forces driving machine-to-machine communications will soon make the design economically viable.

Categories and Subject Descriptors: B.0 [**Hardware**]: General; B.4 [**Hardware**]: Input/output and Data Communications; J.7 [**Computer Applications**]: Computers in Other Systems

General Terms: Design, Experimentation, Performance, Measurement

Additional Key Words and Phrases: Theft detection, burglar tracking, inertial navigation

ACM Reference Format:
Guha, S., Plarre, K., Lissner, D., Mitra, S., Krishna, B., Dutta, P., and Kumar, S. 2012. AutoWitness: Locating and tracking stolen property while tolerating GPS and radio outages. ACM Trans. Sensor Netw. 8, 4, Article 31 (September 2012), 28 pages.
DOI = 10.1145/2240116.2240120 http://doi.acm.org/10.1145/2240116.2240120

1. INTRODUCTION

According to the FBI Uniform Crime Report [UCR 2008], an estimated $17.2 billion in losses resulted from property crimes in the U.S. in 2008. Of this total, burglary accounted for an estimated 22.7% (2,222,196 reported burglaries) with an average loss of $2,079 per incident ($4.6 billion total lost to burglary) while larceny-theft accounted for 67.5% (6,588,873 incidents) of property theft with an average loss of $925

This work was supported in part by NSF grant CNS-0721983 and FedEx Institute of Technology (FIT) at the University of Memphis.
Authors' addresses: S. Guha, K. Plarre, D. Lissner, S. Mitra, B. Krishna, University of Memphis, Memphis, TN 38152; P. Dutta, University of Michigan, Ann Arbor, MI 48109; S. Kumar (corresponding author), University of Memphis, Memphis,TN 38152; email: santosh.kumar@memphis.edu.

DOI 10.1145/2240116.2240120 http://doi.acm.org/10.1145/2240116.2240120

($6.1 billion total). Motor vehicle theft, in contrast, accounted for an estimated 9.8% (956,846 incidents) of property crimes, with an average dollar loss of $6,751 per stolen vehicle ($6.4 billion total). Fortunately, most stolen vehicles – upwards of 80% – are recovered [NHTSA 1998], but the same cannot be said for burglary and larceny-theft. Due to the difficulty and expense of investigating such crimes, 90% go unsolved, burglars and thieves remain on the street, and distribution networks for stolen property remain intact.

Traditional home security systems deter or detect burglary through increased vigilance (security cameras, motion detectors, and alarm systems) but they cannot help track or recover property once stolen [CNN 2009]. On the other hand, traditional asset tracking and vehicle recovery systems are ill-suited to the constraints of tracking and recovering stolen property. Asset tracking products like Brickhouse [Brickhouse 2012] and Liveview [Liveview 2012] use GPS to obtain location fixes and cellular infrastructure to communicate this data but they are not immune to scanning-based detection, they cannot tolerate in-transit cloaking, and their high-power draw requires frequent recharging (approximately five days of operation without recharging), making them unsuitable for use in tracking everyday objects like televisions, stereos, and microwaves.

LoJack, the most common stolen vehicle recovery system, uses a small device hidden inside a vehicle to transmit homing beacons after a vehicle has been reported stolen [LoJack 2012]. When a LoJack-equipped vehicle is reported stolen, a network of high-power wireless transmitters send an activation signal to the device. Once activated, devices transmit periodic beacons that can be tracked using police-car-mounted LoJack receivers. Unfortunately, this approach requires a device to be alert to receive the activation signal, and requires frequent, high-power transmissions once activated, making it unsuitable for long-term, battery-powered operation, especially for household assets that are vulnerable to burglary. In addition, a $695 price tag makes the cost prohibitive for tracking everyday objects.

In this article, we present AutoWitness, a system to deter, detect, and track personal property theft, improve historically dismal stolen property recovery rates, and disrupt stolen property distribution networks. AutoWitness consists of small, embeddable wireless tags and a backend server connected using the cellular network. A property owner embeds a tag inside an asset to be protected, where the tag lies dormant, drawing just microamps, until it detects vehicular movement. Once moved, the tag uses low-power, inertial sensor-based dead reckoning to estimate position changes, but to reduce integration errors, the relative position is reset whenever the sensors indicate the vehicle has stopped. This approach employs vibration sensors, accelerometer, and gyroscope for inertial sensing, uses a 2^{nd}-order Butterworth filter to reduce noise in the raw data, applies drift correction, and more generally builds on a large body of prior work on mapping, tracking, and localization to generate motion estimates [Lemaire and Sulouff 1998; Dooge and Walsh 1998; Weinberg 2012; Ladetto 2000].

The sequence of movements, stops, and turns are logged in compact form and eventually transferred to a server using a cellular modem after both sufficient time has passed (to avoid early detection) and RF power is detectable (hinting that cellular access may be available). The communications delay prevents a RF bug detector from quickly identifying the tagged asset while the presence of background RF noise power hints that the tag may not be cloaked, and therefore may be able to use cellular communications. Eventually, after the trajectory data are sent to a server, the process of matching a path to the observations begins. Starting with a known initial position and an estimated final position (based on cell tower identification), the algorithm uses a novel formulation of Hidden Markov Model of city streets and Viterbi decoding to estimate the most likely path through the map, given knowledge of segment lengths,

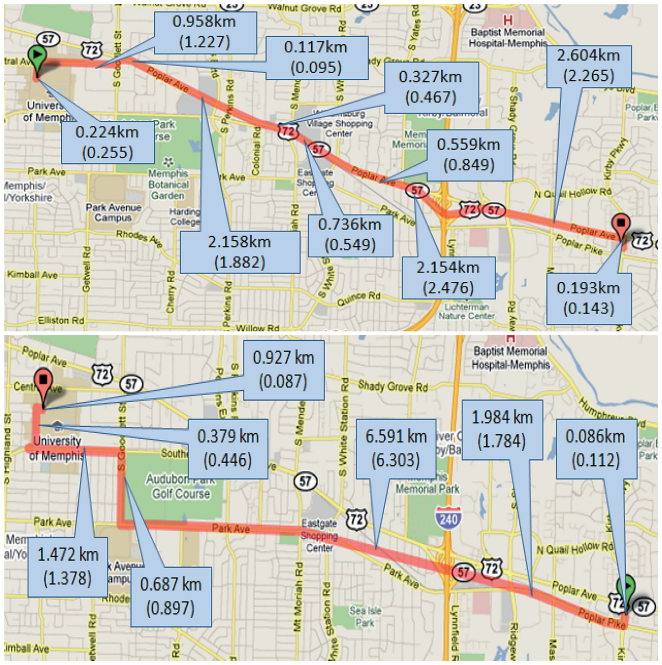


Fig. 1. Two real-life test runs of the AutoWitness system. The various numbers denote the distance between successive turns/stops estimated by AutoWitness with actual corresponding distance in parentheses. The path construction is obtained by applying our HMM on the Open Street Map representation. The visualization uses Google Maps.

intersections, and sensor observations. Path reconstruction, when feasible in near-real time can help law enforcement personnel get ready to arrest the suspects as soon as they make a stop, and post-facto inform about the stolen property distribution network. Figure 1 shows two test runs of AutoWitness on streets in Memphis. A tag is driven on a 6+ mile path from the university campus (top figure) and then is driven back via a different path (bottom figure).

Our design assumes that property owners are able to embed small, wireless tags into the assets they wish to protect. We also assume an attacker model in which stolen property is transported by vehicle on city streets, may be tested for RF emissions prior to theft but not fully dismantled to search for tags, may be cloaked or electrically shielded during transit, and is uncloaked when the stolen property is eventually sold to pawn shops or the unsuspecting public.

Main Results. For theft detection, we are able to achieve over 99% accuracy while using 2 simple features derived from accelerometer, all while operating on ultra-low power for majority of the time. For estimating distance, we are able to limit the errors to less than 10% for most cases, even while allowing the asset hosting the tag to move freely on the floor of a moving vehicle. For reconstructing the path, we are able to achieve an accuracy of over 90% even if only crude localization (from cell towers) is available for the destination. If cell tower localization is available at each turn, the accuracy improves to over 99%.

Organization. Section 2 describes the overall architecture. Section 3 describes the design of the tag node hardware. Section 4 describes the theft detection method. Section 5 describes the process of obtaining turn and distance estimates from inertial sensors. Section 6 describes path reconstruction from estimates of turns and distances.

Section 7 provides results of evaluation from real-life experiments. Section 8 concludes the artical.

2. SYSTEM OVERVIEW AND DESIGN RATIONALE

Requirements and Challenges. The AutoWitness system requires the detection of theft, tracking of the stolen tag as it is driven on city streets upon detection of theft, and pinpointing of the location where it may be hidden. There are several design challenges that emerge when meeting these requirements, especially if both the cost and energy consumption are to be controlled. First is the appropriate choice of hardware for the tag. Second is the design of a theft classifier that allows the unit to operate on extremely low current, while still ensuring that all theft instances are detected instantaneously without requiring property owners to report the incident. Third is the design of a tracking method that can facilitate post hoc reconstruction of the path followed by the stolen tag and pinpoint its final destination even if GPS is unavailable (due to inadvertent or intentional shielding). This challenge is amplified by the observation that the tag will not be strapped to the body of motion (i.e., the vehicle). Therefore, the initial orientation of the accelerometer will not be aligned to the direction of motion of the vehicle. In addition, the orientation of the tag could change multiple times, enroute, such as when the vehicle takes turns, navigates bends in the road, or goes over potholes.

Overall Design. For the tag node hardware, we decided to use a vibration dosimeter, 3-axis accelerometer, 3-axis gyroscope, and a GSM/GPRS modem, all integrated onto an Epic Core platform. The vibration dosimeter allows the unit to remain in a low-current mode until significant motion is detected. If the residual energy in the tag node falls below a certain threshold, the tag node informs the server (via GSM radio) which alerts the owner to replenish the batteries. Upon detection of motion, theft classification is performed using accelerometer measurements to detect if the motion is indicative of vehicular movement. If so, the tag enters a tracking mode, where it begins to use the accelerometer and gyroscope measurements to detect turns and to obtain a robust estimate of distance traveled between successive turns and/or stops. The estimates of distances and turns are stored locally, until connectivity to cellular network is available, at which time they are communicated to a central server. The server uses these estimates and applies a Hidden Markov Model (HMM) and Viterbi decoding onto a representation of the city street map to obtain the most probable route and the resting destination. Figure 2 illustrates the overall system operation and Figure 3 illustrates the computations performed at the AutoWitness backend server.

2.1. Design Rationale

For inertial navigation, six degrees of freedom are needed to obtain attitude and position in three dimensions [Skog and Händel 2009]. Although it is theoretically possible to use 3-axis magnetometers in place of 3-axis gyroscopes, magnetometers are considered unsuitable for inertial navigation in cars [Abbott et al. 1999] because of large dynamic errors in measurement that can be caused due to changes in local magnetic field from cars, bridges, buildings, power lines, etc. An inertial system can also be built using accelerometers alone, if six accelerometers are aligned appropriately [Tan and Park 2005]. However, we are unaware of the feasibility of this approach in practice or the availability of such a configuration of accelerometers commercially. Consequently, AutoWitness uses a combination of accelerometer and gyroscope for inertial navigation. Use of these sensors in AutoWitness also serves the dual purpose of enabling theft classification.

Inertial navigation that has traditionally been used in high-end navigation systems such as missiles, aircraft, and marine applications is now being adopted for car

Fig. 2. The AutoWitness system.

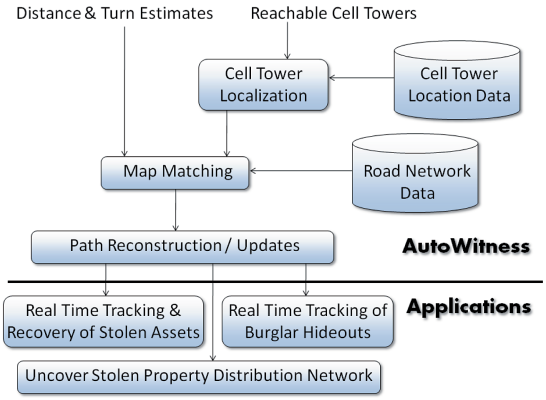


Fig. 3. The computations performed at the AutoWitness server.

navigation to complement GPS. The main challenge in building a car navigation system based solely on low-cost inertial sensors is that due to the integrative nature of these systems, position error grows unbounded as a function of operation time or traveled distance. Therefore, obtaining the correct path in real time has not been feasible with low-cost inertial sensors. Another challenge is robust and reliable estimation of instantaneous heading, which is difficult to obtain from rate-grade low-cost gyroscopes.

By using a novel formulation of HMM for map matching, using a series of steps to obtain reliable distance estimations (including estimating and accounting for drifts between successive stops), and crude localization from cell towers of the final destination (e.g., with an uncertainty of 100m radius), we are able to reconstruct the path with high accuracy (i.e., >90% for urban streets). If crude localization from a GSM modem is available more frequently (say, at every turn), the accuracy of correct path identification improves to over 99%.

For theft detection, we decided to use vehicular driving as the indicator of theft. If the AutoWitness tags are used only for assets that usually do not accompany the owner and remain in home or office such as TV, safe, gaming systems, stereo systems, grand pianos, etc., then vehicular driving could be a reliable indicator of their theft. These

assets are rarely transported in a vehicle by an owner, but would rarely be stolen by a thief on foot. Also, since these assets are left behind when the owner leaves home (or office), they are likely to be taken in the event of theft or burglary. If the owner is to take a tagged property in a vehicle, they could remove the tag before transporting or could disarm the tag.

Using vehicular driving as an indicator of theft serves several purposes in AutoWitness. First, it significantly cuts down the false alarm rate because transportation of tagged asset from one room to another does not activate theft alarm. Second, radio is not turned on until driving is detected, which helps evade quick identification of tagged property using RF bug detectors during its theft. Not turning on the radio also saves energy. Third, it enables an ultra-low power operation when not stolen.

In summary, our design allows us to demonstrate the feasibility of a low-cost system that can autonomously detect theft, facilitate recovery of tagged stolen properties even if GPS is inaccessible, and potentially dismantle the distribution network of stolen property transactions by providing the trajectory followed by stolen properties post facto (including all the places where it may be stored en route), while operating in an ultra-low power mode for most of its life.

3. TAG NODE HARDWARE DESIGN

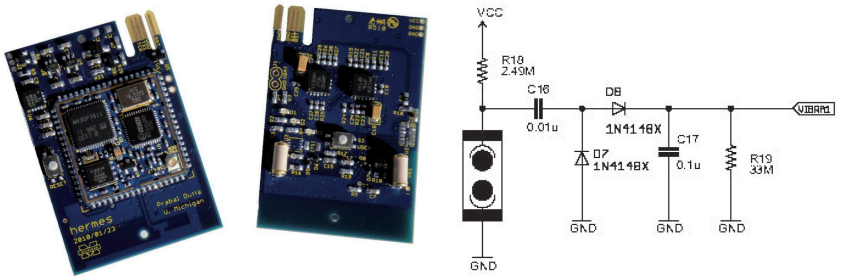
The AutoWitness tag node must detect theft, classify movements, and communicate trajectory estimates to a backend server. We address these basic requirements with a tag design that integrates a mote core, vibration dosimeter, accelerometer, and gyroscope, and which uses an embedded GPRS radio to communicate with a backend server. In addition to these basic requirements, the tag design challenges include size, lifetime, and cost. Tags must be small and unobtrusive if they are to be hidden inside of everyday objects like computers, televisions, and stereo equipment. Once deployed, tags must operate unattended for many years without maintenance. To be economically viable, tags must cost substantially less than the assets they protect.

Our prototype design, shown in Figure 4, meets some, but not all, of these needs: the sensor frontend has a 51 mm $\times$ 34 mm $\times$ 10 mm footprint (without the GPRS modem), the entire tag draws over 10 μA in sleep mode, and the cost of the tag exceeds \$200. However, newly emerging, contract-free, wristwatch phones with \$100 retail price tag that integrate a color LCD touch screen and Bluetooth communications, and include several phone accessories [Mobile Watch 2012] give us hope that a tag price in the tens of dollars will soon be viable at volume.

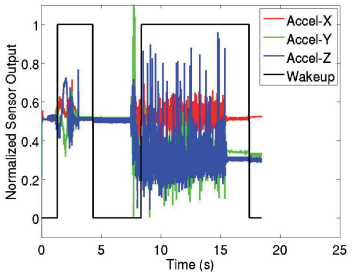
3.1. Motion Detection and Inertial Sensing

The basic detection problem is to distinguish an object at rest from an object in (prolonged) motion while minimizing the current draw in the resting. We use a *vibration dosimeter*, shown in Figure 4, to perform this function. The sensor is an omni-directional vibration switch that is nominally closed at rest but chatters open and closed in response to movement [SignalQuest Precision Microsensors 2009]. The switch is connected to ground on one terminal and in series with a pullup resistor to power. The 2.49 MΩ pullup resistor sets the quiescent current draw of the circuit. At rest, the circuit draws 1.2 μA at 3 V. A capacitor AC-couples the output of the sensor, a first diode steers negative voltage transients to ground, and a second diode steers positive transients to a capacitor that integrates these signals. A resistor in parallel with the integration capacitor slowly discharges the capacitor so that in the absence of motion, the capacitor voltage goes to zero.

Figure 4(c) shows the motion detector circuit in operation. Tri-axial acceleration samples taken at 200Hz using the onboard Analog Devices ADXL330 accelerometer [ADX 2005] are shown with their bias removed and amplitude scaled. Rotation data from the



(a) tag node (front/back) (b) wakeup circuit



(c) motion detection

Fig. 4. AutoWitness tag prototype: (a) The tag integrates an Epic Core [Dutta et al. 2008] (front), dual vibration switches (back), 3D accelerometer (front), 2D+1D gyroscope pair (back), and power supply (front) in a 51 mm × 34 mm × 10 mm footprint which connects with a GPRS radio (not shown) using an edge connector; (b) motion detection circuit. A vibration dosimeter integrates the output of a vibration switch and connects to an interrupt line; (c) the motion detection circuit in operation. Acceleration bias is removed and the readings are scaled. The wakeup signal triggers after a brief period of motion.

onboard ST Microelectronics LPR530AL/ALH gyroscopes [ST Microelectronics 2009] are not shown. The output of the motion detection wakeup circuit can be seen as a pulse that alternates between zero and one as the sensor transitions from rest to motion. At time $t = 0.5$ s, a tag is picked up and moved and at time $t = 1.33$ s, the motion detector circuit wakeup triggers, waking up the sleeping microcontroller using an interrupt line. At time $t = 3.09$ s, the tag stops moving and time $t = 4.3$ s, the motion detector output indicates movement has stopped. This process repeats for a second, longer, and larger motion starting at time $t = 7.5$ s.

3.2. Tag-Server Communications

The key communications challenge in our system is ensuring that tags can eventually communicate with the backend infrastructure while minimizing the size and cost of the communications hardware. The twin requirements of near-ubiquitous connectivity and market-driven commoditization are well met using a GSM/GPRS cellular radio modem. Our prototype design uses a Telit GM865 module [Telitge865 2012] which measures 22 mm × 22 mm × 3 mm.

A separate carrier board is used to mount the GM865 module and connect it with the tag node for power control and serial communications, with external antennas for GSM/GPRS, and battery. To support experimentation, an 1100 mAh, 3.7 V rechargeable Li+ battery provides power, but for production purposes, we expect to use a primary (nonrechargeable) battery due to significantly lower leakage currents. Finally, the GSM/GPRS radio uses an external, quad-band blade antennae (1–2 dbi) [S Blade

Antenna 2012]. Going forward, we plan to use a smaller, integrated antenna and radio module.

We have integrated an Epic core, a Telit GE865 GSM module, and the sensors together in one compact platform. We describe key decisions we made to minimize the energy consumption when the tag is not being tracked, given that tracking is activated only when the tag is stolen.

The GE865 uses a 2.8V CMOS logic-level circuitry, while the Epic core has an MSP430 microcontroller which uses a 3.3 V logic level, and henceforth any communication between the two units requires a logic-level translation. Communication between the Epic core and GE865 is of two types: the Epic core sends control commands and data to GE865 (which is finally sent as an SMS message to the AutoWitness server), and GE865 responds with status codes and other information such as cell tower IDs and signal strengths. Logic-level translation can be done in several ways, such as using logic-level translation chips, buffers with controlled output, voltage dividers, MOSFET- and resistor-based circuits. We use a voltage divider circuit for the transmit side from Epic to the GE865, which simply steps down the 3.3V to 2.8V. When not used, the microcontroller output can be kept in high impedance state to minimize loss on the divider circuit. For the receive side, from the GE865 to Epic we use a MOSFET-based step-up circuit, which takes both a 2.8 V and a 3.3 V input as references to convert a 0 V to 2.8 V signal to a 0 V to 3.3 V signal. The 2 reference signals are controlled by FETs, which act as switches and are turned on only when there is a need to communicate with GE865, namely when theft is detected. The on-off circuitry on GE865 helps reduce power draw considerably, and turns off the GSM circuitry, and is controlled by a line from the Epic core. Even after using the on-off circuitry there are significant leakage current losses, which are minimized by a MOSFET-based switch which disconnects battery voltage from GE865.

GE865 also provides several other pins of interest out of which we use three: Network Status LED, Power Monitor, and Jamming Detect Report. The network status LED is an output line from the GSM module that indicates network activity and registration status by frequency of LED blinking. We do not use this signal to drive an LED (which will waste around 3 mA) , instead we use it to count pulses so that we can tell network activity without issuing a software command. Power Monitor line is used as a check to see whether the module has actually powered up or not by firing interrupts when powered on. A final interesting output line that is available on the GE865 is a Jamming Detect line which tries to detect whether the GSM radio is being jammed. We plan to use this information to turn the GSM module off in the event of cloaking to preserve energy. We also implemented a simple parser on the Epic core to parse network registration and other information by issuing a software command to GE865.

4. THEFT DETECTION

As described in Section 2, we use vehicular movement as an indicator of theft. In this section, we describe our approach to detecting vehicular movement using the AutoWitness tag, while operating in ultra-low power mode most of the time. Since a tag node will spend most of its life monitoring for theft, an ultra-low power operation in this phase has the most impact on its overall lifetime.

To reduce the power consumed by the algorithm we use "triggered sensing" [Benbasat and Paradiso 2007], in which low-power sensors are used to make an initial decision, and wake up other sensors to improve the accuracy of such a decision. In our case, the low-power sensors are vibration dosimeters, which wake up the microcontroller when significant movement is detected. After waking up, the tag-mote samples the accelerometer, computes features, and uses a decision tree to detect if the tag is moving in a vehicle. If the decision is "vehicle," the tracking algorithm is triggered. The final

Table I. Performance of Different Classifiers for Detecting Vehicular Movement (using WEKA)

Classifiers	Accuracy(%)
Decision Tree	98.5813
Random Tree	98.4217
Naive Bayes	87.5155
ClassificationViaRegression	98.72
Filtered Classifier	96.1341
Rotation Forest	98.88
Decision Table	95.95

decision, whether the object is being stolen, is made at the central server by estimating the speed and change in location derived from cell-tower-based localization.

4.1. Detecting Vehicular Movement

We now describe the details of the algorithm for detecting if a tag is being moved in a vehicle. A tag is in the *deep sleep* state until an interrupt is generated by the vibration dosimeter. After the interrupt, the accelerometer is sampled for 1.05 s at 200 Hz. The collected data is preprocessed by first converting the raw ADC values to acceleration in units of 1g, then computing the magnitude of acceleration (in order to eliminate dependence of the algorithm on orientation of the tag), and finally, computing the medians in nonoverlapping windows of 15 samples. If the variance of the computed medians is below a threshold, it is assumed that the movement was caused by a short jerk, and the tag returns to deep sleep state. If not, 4.2 s of additional samples are extracted from accelerometer to decide whether it represents vehicular movement. The 5.25 s worth of data is partitioned in five intervals of equal size and a decision tree classifier (for vehicular movement) is applied to each interval, producing a total of 5 decisions. In a postprocessing phase, majority rule is applied to arrive at a final decision, that is, if 3 or more out of 5 decisions are "vehicular movement" the algorithm initiates tracking, otherwise it returns to deep sleep. Postprocessing helps reduce the false alarms produced by transitions, for example, when the tag-mote is initially at rest, and then is taken by a person and carried on foot.

Classifier Development. To develop a classifier for vehicular movement, we collected data in different scenarios. A tag was attached to objects such as televisions and stereos, carried by a walking person, transported in car, on trolleys and rolling chairs. Classification is based on features computed from the data (after preprocessing). We used features found useful in the activity classification work [Reddy et al. 2010]. We trained different classifiers in WEKA [2012], and evaluated them using ten-fold cross-validation. The performance of various classifiers are shown in Table I. We selected decision tree, given its simplicity, and trained it with different number of features. We found that the most discriminating features were energy (i.e., mean over square of measurements) and standard deviation. Adding more features did not lead to significant increase in accuracy. Figure 5 shows a scatter plot of feature values for vehicular and nonvehicular movement. Using this classifier we are able to obtain 97% accuracy, which improves to $>$99% by using majority voting (see Section 7.1).

5. ESTIMATING DISTANCE AND TURNS FROM INERTIAL SENSORS

After the Auto-Track system classifies an activity as theft, the tracking module is triggered. The tracking module's purpose is to find the exact sequence of road segments that the stolen property is driven through so as to pinpoint its final destination. The reconstruction of the path consists of two stages: (1) estimation of turns and distances between successive stops and/or turns, and (2) application of Viterbi

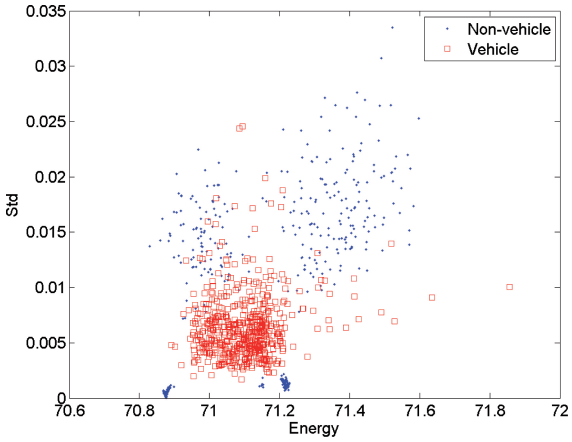


Fig. 5. Scatter plot of feature values for vehicular and nonvehicular movement. The clusters for the nonvehicular movement closer x-axis correspond to static scenarios and the ones farther from the x-axis correspond to manual movement (e.g., trolley, walking, etc.).

decoding and a Hidden Markov Model (HMM) on the street map to identify the path from distance and turn estimates. In this section, we describe the process of obtaining turn and distance estimates and discuss the path reconstruction process in Section 6.

Any rigid body's spatial movement in space can be described with the help of six parameters: namely three translatory (x-, y-, z-acceleration) and three rotatory components (x-, y-, z-angular velocity). As described earlier, the tag node used in the AutoWitness system is equipped with a 3-axis accelerometer coupled with a 3-axis gyroscope. The three acceleration sensors and three gyros have been put together in such a way that they form an orthogonal system. The accelerometer has a three-dimensional Cartesian frame of reference with respect to itself, represented by the orthogonal x, y, and z axes. In addition, we define a Cartesian frame of reference with respect to the vehicle that the tag node is in. The vehicle's frame of reference is represented by the orthogonal X, Y, and Z axes, with X pointing directly to the front, Y to the right, and Z into the ground. If the tag node's coordinate system was perfectly aligned with that of the car's, and the obtained signal from the tag was continuous, integrating the individual translatory and rotatory components recorded on the accelerometers and gyros, respectively, would tell us the exact position and attitude of the burglar's car. For a straight road segment, a double integration of the acceleration data would yield the distance traveled from the starting point and since the gyros provide output data representing rotation speed (not angular acceleration), a single integration of the gyro signal would yield the total change in the attitude of the burglar's car. Performing these calculations periodically would enable the ideal system to trace the car's movement with respect to a (virtual) reference point and to indicate its speed, current position, and heading. However, the assumption of perfect alignment is unrealistic and in most cases when a theft occurs the stolen object will be placed in a tilted configuration resulting in an arbitrary disorientation of the axis of accelerometers with respect to the car's coordinate system. Integrating readings from a disoriented accelerometer results in huge estimation errors for both distance and angles. Additionally, the measurements obtained from low-cost inertial sensors are quite noisy and suffer large drifts. In Section 5.1, we present our method of obtaining angle estimates from the gyroscope measurements, in Section 5.2 we describe the process of reorienting the axes of accelerometers every time there is a change in its orientation with respect to the vehicle, and in Section 5.3,

we describe our approach of estimating the distance traveled between successive stops and/or turns from accelerometer and gyroscope measurements. The stops are detected using a similar decision-tree-based classifier as the one used for detecting vehicular movement (see Section 4.1).

5.1. Obtaining Angle Estimates

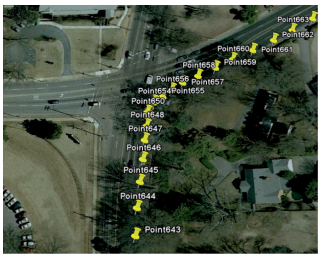
We use the gyroscopes and accelerometers to estimate the *angle of change* in the direction of movement of the vehicle. In strapdown inertial navigation applications, gyroscopes are used to estimate instantaneous attitude of the object along a fixed reference frame, by a single integration of the angular velocity. The instantaneous attitude along each axis (X,Y,Z) can be represented using direction cosine matrix, Euler angles, or quaternions. We use Euler angle system to maintain the instantaneous attitude information. Rate-grade low-cost gyroscopes generally suffer two kinds of errors: scale factor error and static bias [Borenstein]. The scale factor error that may be introduced due to mounting and placement errors when putting the gyroscopes on the board can be estimated once for each board using a turn table.

Static bias is the output produced by the gyroscope when there is no angular velocity applied to it. This value is often not a constant, and when integrated to find the angle of rotation may lead to large drifts over time. We account for this potential error by estimating the drift at every stop.

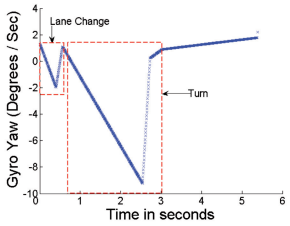
Although we are interested in measuring the change in the angle of heading of the vehicle (for use in map matching), the tag may experience a change in orientation not only due to legitimate turns and curves in the road, but also due to change in *its* orientation from the vehicle's frame of reference (e.g., due to jerks, or skidding when taking a turn). We use the absolute value of the first difference in yaw readings to detect these changes in orientation. We maintain two thresholds ($D_h \geq D_l$) for amplitude so as to detect a spike in the first difference feature (when it rises above D_h) and we record the time it takes for the first difference to return to normal (when it drops below D_l). This duration is called the activation time. Figure 6 shows the effect of lane shifts and turns on these two features. Gyroscope measurements captured during the activation time are used for computing the change in angle of the tag. The axes of the accelerometer are reoriented to the vehicle's frame of reference using the reorientation module any time the gyroscope indicates a change in orientation (see Section 5.2). The change in the orientation of accelerometer computed by the reorientation module (which represents the change in the tag's orientation from vehicle's frame of reference) is then accounted for in the angle of turn computed from gyroscope measurements to obtain the change in angle of the vehicle.

5.2. Reorientation of Accelerometer Axes

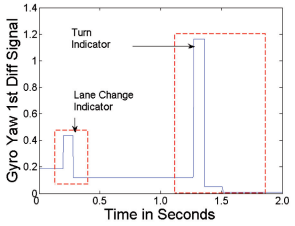
To measure acceleration in the forward direction, the dominant (virtual) axis of the accelerometer (which need not be aligned with any of the designated axes) needs to be determined so that the acceleration values used are indicative of actual distance traveled. Since the tagged object can be placed in any orientation, we need to find the (virtual) axis of motion. To determine the dominant axis, we use the approach proposed in Mohan et al. [2008]. In addition, the orientation of the tag may change en route due to movement of the asset that houses the tag. We recompute the dominant axis anytime the gyroscope measurements indicate a potential change in the angle of the tag (see Section 5.1). We then use the direction cosine matrix to resolve the accelerometer readings to obtain the acceleration in the direction of motion [Titterton and Weston 2004].



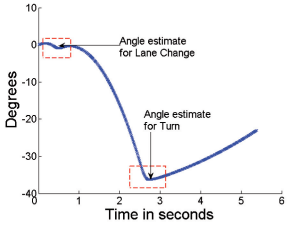
(a) lane change followed by a right turn



(b) corresponding yaw measurements



(c) first difference in yaw



(d) estimation of angles

Fig. 6. Gyroscope measurements during turns and lane changes.

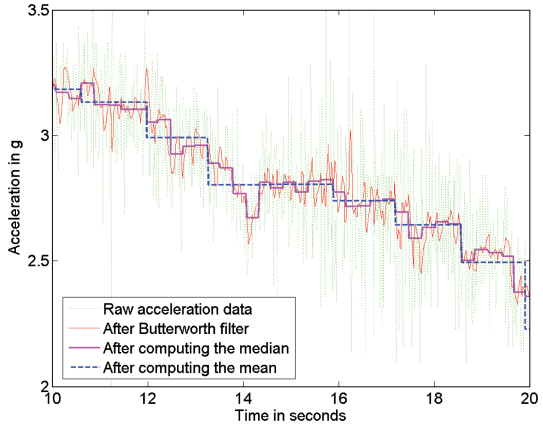


Fig. 7. Illustration of various steps to reduce accelerometer noise. First the raw signals are passed through a 2^{nd}-order Butterworth filter, then the median is computed over a window of 20 samples, finally the mean is calculated over 10 medians.

5.3. Estimating Distance Traveled from Inertial Sensors

Before the accelerometer values obtained in the direction of motion can be used to compute distance traveled, it must be corrected for several potential errors. First, to remove jitters and noise from the accelerometers one can either perform a 2^{nd}-order Runge Kutta integration [Weston and Titterton 2000] or pass the signal through a 2^{nd}-order Butterworth filter [Lawrence 1998; Titterton and Weston 2004]. We tried out both and found that the latter worked for us better. Due to the high sampling rate of our accelerometers (200Hz) and jerks experienced on the road, the obtained signals still suffered high amplitude variation. Hence, they were further smoothed by taking

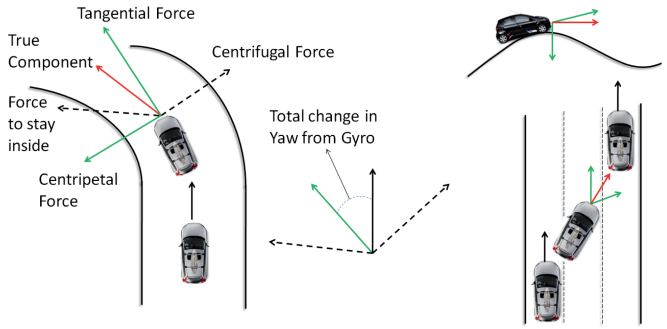


Fig. 8. Accounting for radial acceleration in different scenarios: While traveling through ramps, undulated terrain and making lane changes the true component of the acceleration is extracted from the raw accelerometer signals with the help of change in attitude information from the gyroscopes. This significantly reduces errors in distance approximation.

a median of 20 samples and then taking the mean of 10 such obtained medians. This procedure yielded only a single acceleration value for each 1-s interval, representing the average acceleration of driving during that second.

Second, urban streets often contain curved roads through neighborhoods, ramp to interstates or loops. As shown in Figure 8 when a car drives through a curved road even for small speeds the centrifugal force can be substantial and the readings from the accelerometers are actually a representation of the tangential acceleration generated by the car. If the untransformed signal is integrated to find the distance traveled on the curved segment it results in significant drift. The true component of motion along the road is given by the resultant of the two opposing forces, namely the tangential force and the centripetal force (or the force generated by the steering wheels to stay inside curve and the centrifugal force) and in order to find the actual distance traveled on the road segment one needs to extract the true acceleration component from the raw signal. This can be achieved with the help of gyroscopes. While moving into a curved segment from a another road segment the total change in yaw angle gives us the angular rotation. The angular rotation can be then be used to find the component of centripetal force which when subtracted from the raw signals gives us the component of acceleration along the curve. Other similar scenarios include lane changes and traveling over a hilly region or undulated terrain as shown in Figure 8. For details, we refer the reader to Chapter 11 in Titterton and Weston [2004].

Third, when an object moves from one stop to the next stop, such as when accelerating from rest at a traffic light to coming back to rest at the next red signal, the average acceleration over the interval is zero. But during transit, the accelerometers produce highly oscilating values. In order to eliminate the drift from the measurements, we subtract the mean acceleration (between two successive stops) from all recorded acceleration values [Boore 2003]. The typical speed limit of a vehicle in an urban scenario lies below 75 mph which is very small compared to the rotation speed of the Earth's surface, hence in our distance estimate computation we ignore the Coriolis Effect produced by the Earth. Figure 7 shows how the raw signals from the accelerometers are successively filtered to produce good distance estimates.

5.4. Implementation of Distance/Turn Computations

Mote implementation of distance and angle computations presented us with several challenges. A simple way of computing distance and angles is the following. The acceleration value in the direction of motion is corrected for any radial acceleration. The Butterworth filter is applied next to filter out noise. The one-second means computed

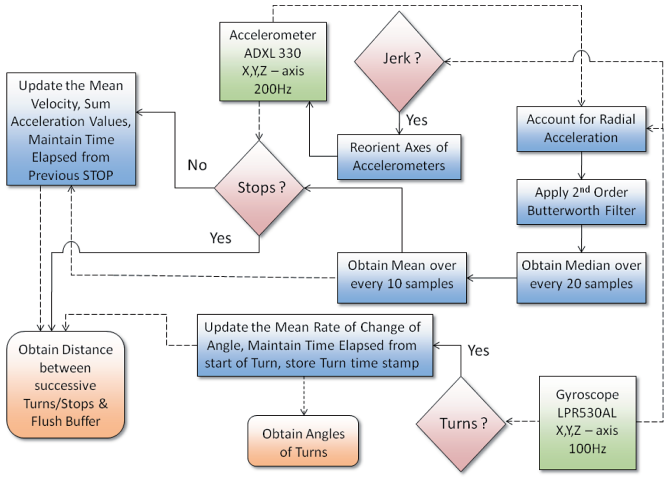


Fig. 9. Various stages in estimating the distance traveled between successive turns and/or stops from inertial sensors (broken lines indicate inputs).

from these measurements (see Section 5.3) are buffered until the next stop, at which time the bias is removed from them to derive the acceleration that can be used for distance computation. Each such buffer corresponds to measurements recorded between successive stops.

The gyroscope measurements that may correspond to turns are also buffered until the next stop, at which time bias can be removed from them. Also, the buffer of accelerometer measurements is marked each time a potential change in orientation is suspected from gyroscope measurements. Further, the changes in orientation determined by the reorientation module are also stored. At each stop, the angles for each marked turn are computed (see Section 5.1). The accelerometer buffer is then partitioned to derive the segments between each successive turn. Double integration is now applied to obtain the distance between each successive turn and/or stop. Both the accelerometer and gyroscope buffers are flushed out to allow for fresh recording until the next stop.

Though very simple, the aforementioned method is infeasible for real-life implementation. The idea of storing all the acceleration and gyroscope samples between successive stops for appropriate removal of drift bias from the data is not congruent with the storage capability of the mote platform. The mote has a storage capacity of 8Kb, and each acceleration and gyro sample has a storage requirement of 4 bytes. A accelerometer sampling rate of 200 Hz implies that data from a drive of even 11 seconds will be sufficient to overflow the buffer. Hence in order to facilitate robust distance estimation we use a different approach. After filtering out noise and calculating onesecond means of the acceleration data we just maintain the sum of the accelerations, the time elapsed, and the constantly updated mean of velocity from the previous stop. We use a running mean [Running Mean 2012] to avoid estimation errors. Then the formula used for computing true distance 'd' is simply

$$d = (\overline{v} - b * t) * t,$$

where $\overline{v}$ is the running mean of velocity, b the sum of all acceleration values between successive stops, and t being the time elapsed from the previous stop. Figure 9 shows the overall procedure for obtaining distance estimates between successive stops. Using the aforementioned technique alleviates the problem of buffering huge amounts

accelerometer data and still removes errors due to drift bias, making a mote implementation for the AutoWitness system feasible. We use a similar approach for measuring turn angles from the gyroscope data but additionally store the timestamp of each turn to break up distances between successive stops into finer segments of distances between turns and/or stops. This is an essential requirement for our tracking model described in the later sections.

6. PATH RECONSTRUCTION FROM DISTANCE AND TURN ESTIMATES

Once a sequence of turns and distance traveled between successive turns and/or stops has been obtained, the next challenge is to search a map of streets to identify the sequence of road segments that is most likely to result into the observed sequence of distances and turns. The problem is especially challenging because: (1) the initial location may not be known deterministically due to the delay in activating the tracking module after theft is established by the theft classification module (of the order of 10 s, which may translate to 100 meters), (2) the estimates of distances obtained from accelerometer and gyroscope readings could be off from their actual values by up to 12%, (3) several road segments have similar lengths, and (4) the destination location (obtained from cell towers) may have an uncertainty of the order of 100 meters.

Viterbi decoding over *Hidden Markov Model* (HMM) has traditionally been used for mapping noisy GPS measurements to a route [Thiagarajan et al. 2009] because a route can be considered as a sequence of road segments and the adjacency among adjoining road segments can be used to constrain the transition among states (i.e., road segments). Therefore, even though we do not have GPS coordinates to map to a sequence of road segments (contrary to most existing work on map matching), the formulation of an HMM continues to be a suitable choice due to its ability to leverage adjacency among road segments to account for the uncertainty in measurements.

An HMM is a Markov process comprising of a set of hidden states and a set of observables. Every state may emit an observable with a known conditional probability distribution called the emission probability distribution. Transitions among the hidden states are governed by a different set of probabilities called transition probabilities. Upon executing on a sequence of hidden states, an HMM produces a sequence of observables as its output. While the output (i.e., the list of observables) can be observed directly, the sequence of hidden states that were traversed in producing the observed output is unknown; the problem is to determine the most likely sequence of states that may have produced the observed output. Viterbi decoding [Viterbi 1967] is a dynamic programming technique to find the maximum likelihood sequence of hidden states given a set of observables, emission probability distribution, and transition probabilities.

For map matching, hidden states usually correspond to road segments (portion of road between two points of interest, which could be intersections, not necessarily neighboring). The choice of *observables* is key to the formulation of an HMM. It is the definition of observables that can aid Viterbi decoding in distinguishing one road segment from all others. So, the observables should be as unique to the road as possible. We considered average speed, total length of the segment, travel time, stretch of the segment (i.e., ratio of Euclidean distance between the two ends and the road length), among others. Some are not as unique, while others are not stable (e.g., travel time). We decided to use two features as observables: distance between successive stops and total length of the segment (between successive turns that indicate transition to the next road segment). Both features are simple to compute from the distance measurements, and together are rather unique to a road segment, if it is sufficiently long (i.e., if it includes some stops). Although a vehicle may not stop at all traffic lights on a road segment, the distance traveled between successive stops at (red) traffic lights creates

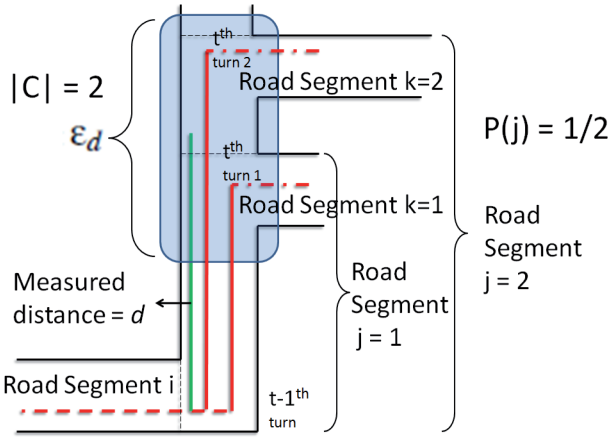


Fig. 10. Computation of transition probability: The burglar's car takes a turn on road segment j from i and travels a distance "d" before turning into another road segment k. "d" is obtained using the accelerometers hence ϵ_d is the error range associated with computing "d". The two road segments falling within error range ϵ_d allowing a turn are road segment k = 1 and k = 2. Hence the road segment traveled on j could be either j = 1 or j = 2. Our HMM computes the transition probabilities for j = 1 and j = 2 and moves onto road segment k.

a pattern that can be (probabilistically) matched to a road segment (see Figure 11). The total length of the segment complements it by ensuring that the aggregate of the distance between successive stops for an entire road segment matches with the total length of that segment.

The *starting state* consists of all intersections (say, n of them) that fall within a radius of r from the location where the tag node was deployed before theft. They are assigned uniform probability, that is, $1/n$.

We next define *transition probabilities*. A transition is initiated only when a turn is indicated from the gyroscope measurements, in which case the distance traveled on the preceding segment and the angle of turn are used to determine the transition probability. To limit the number of states considered for transition, we consider only those states that fall within the error distribution of the observed length and turn. The transition probability from a road segment i completed at the $(t-1)^{\text{th}}$ turn to segment j completed at the t^{th} is derived using Bayesian inferencing. For a distance d traveled on road segment j before turning to another road segment k, we denote the transition probability as $P_i(j \mid d)$ (for transition from state i to state j) and compute it as follows.

(1) If $t = 1$, this is the first turn observed after the detection of theft. In this case, the start state is defined as described earlier.
(2) For $t > 1$, the angle of turn (γ) is computed from gyroscope (as described in Section 5.1).
(3) The distance traveled on the j^{th} segment (d) is computed using the accelerometers and gyroscope (as described in Section 5.3).
(4) Since d and γ are estimations of the actual measurements, we associate error margins ϵ_d and ϵ_γ respectively to them.
(5) For all the road segments that begin at the end of segment i, and have a length between $d \pm \epsilon_d$, (which all may be different portions of the same road[1]), we prune all the road segments j, whose angle of intersection with some road segment k is

[1]Once state i is defined, the angle of turn observed at the $(t-1)^{\text{th}}$ turn determines the road whose segment may come next and thus limits the search for the next road segment onto this road.

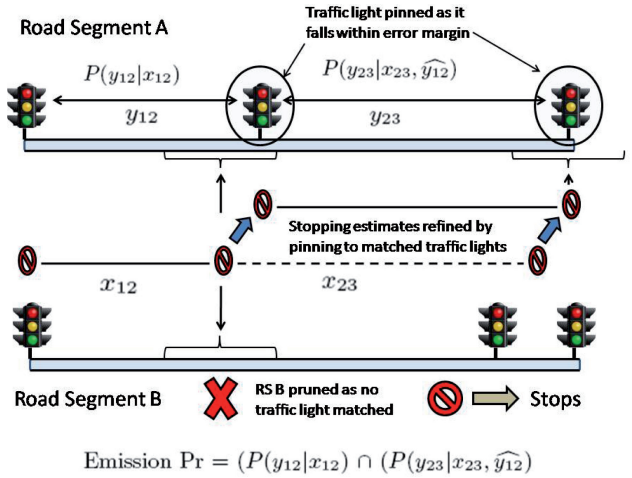


Fig. 11. Stopping estimates don't exactly coincide with traffic lights, hence a margin of $\pm\epsilon_d$ is considered while pinning to traffic lights. Once a traffic light is found within error range the stopping estimates are refined by pinning to the light.

outside the range $\gamma \pm \epsilon_\gamma$. The set of remaining road segments j are the candidate set for transition and is denoted as C (see Figure 10).

(6) For all road segments in C, the prior probability $P_i(j)$ is uniform and is given by $P_i(j) = \frac{1}{|C|)}$.

(7) $P_i(d \mid j)$, the evidence for Bayesian inferencing, is estimated from experimentation. We assume it follows a Gaussian distribution whose variance σ depends on how accurately we are able to approximate distances traveled on road with the help of accelerometers and gyroscope. Our assumption about the apriori Gaussian distribution for calculating the evidence was verified by intensive experimentation. We collected real-life driving data for over 100 road segments and the results from real life were congruent to our assumptions. Hence, $P(d \mid j) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(d-d_j)^2 / 2\sigma^2}$ where d_j is the actual length of road segment $j \in C$ from start of road segment i.

(8) Next, we compute the marginal probability $P(d)$ which is given by $P(d) = \sum P(j)P(d \mid j)$.

(9) Finally we compute our transition probability which is given by the conditional probability $P(j \mid d) = \frac{P(d|j)P(j)}{P(d)}$.

We next describe the computation of *emission probability*, the probability of seeing defined observables given that the system is in a specific state. These probabilities in practice should reflect the characteristics of a state, and are used by the Viterbi algorithm to differentiate the true state that the system is in from other probable states. If the stopping estimates were perfect, we could assign an emission probability of 1 to the road segment whose traffic lights matched with the stopping distances and 0 to others. However, the estimations of distance obtained from accelerometers and gyroscope have a margin of error. Additionally, a vehicle may stop some distance earlier than the traffic light, if there are other vehicles in front of it. Therefore, we assume that if the distance estimated is d, the true distance could have a range of $d \pm \epsilon_d$. Let $P(y \mid x)$ denote the probability of obtaining x from distance estimation when the true distance is y. For $|y - x| > \epsilon_d$, $P(y \mid x) = 0$; otherwise, uniform over the entire

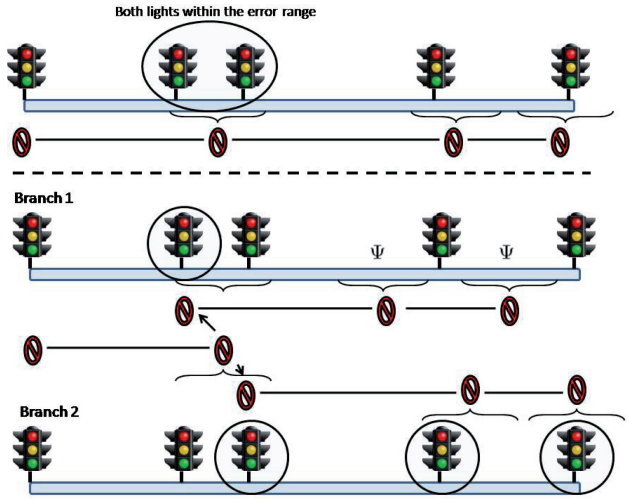


Fig. 12. If multiple traffic lights are detected within the same error range the matching process forks into multiple branches each trying to pin to a different traffic light within the error range. Finally the sequence of traffic lights which generate the highest emission probability is accepted.

domain. We now describe how we use $P(y \mid x) = 1 - \frac{|y-x|}{y}$ in computing the emission probability.

Consider a road segment s. Let the number of stops encountered be m, where the turn into this road segment is considered the first stop and the turn out of this segment to another road segment the last stop. Let $x_{1,2}$ be the observed distance between stops 1 and 2 and $y_{1,i}$ the actual distance between traffic lights 1 and i, where $1 \leq i \leq m$ and $|y_{1,i} - x_{1,2}| \leq \epsilon_d$. Each of these lights i becomes a candidate for the second stop. First consider the case when there is only one candidate for i for each stop (see Figure 11). Then, the emission probability $P_E = P\left(y_{1,2}, \ldots, y_{m-1,m} | x_{1,2}, \ldots, x_{m-1,m}\right)$ for a sequence of observed distances in a given road segment is given by

$$P_E = P(y_{1,m} \mid x_{1,m}) \prod_{i=1}^{m-1} P(y_{i,i+1} \mid x_{i,i+1}). \tag{1}$$

If there are multiple candidate traffic lights for any stop i, then a new branch of traffic light sequence is considered corresponding to each candidate. A maximum is taken over the emission probability computed using Eq. (1) for each candidate stopping sequence (see Figure 12).

7. SYSTEM EVALUATION

We now present the evaluation of various aspects of the AutoWitness system: theft detection, turn estimation, distance estimation, and path reconstruction.

7.1. Theft Detection

In this section, we evaluate the theft detection algorithm. We focus on two aspects: (1) accuracy of the movement classifier, and (2) the lifetime of the tag mote when no theft occurs (i.e., the time it takes for the battery to deplete, due to everyday movements).

7.1.1. Movement Detector and Classifier. As mentioned in Section 4, we use vehicular movement as indicator of theft. We implemented the classifier described in Section 4 on a tag and collected data from various conditions: vehicular movement, nonvehicular

Table II. Confusion Matrices for Three Movement Classifiers: Using Only Raw Measurements (left), Computing Median over 15 Samples (center), and Computing Median, and Majority Rule over 5 Consecutive Decisions (right)

	Raw Data		Median		Median+Majority	
	Vehicle	Non Vehicle	Vehicle	Non Vehicle	Vehicle	Non Vehicle
Vehicle	4916 (98.32%)	84 (1.68%)	4866 (97.32%)	134 (2.68%)	1000 (100%)	0 (0%)
Non-Vehicle	3557 (35.57%)	6443 (64.43%)	293 (2.93%)	9707 (97.07%)	29 (1.45%)	1971 (98.55%)

Row labels represent actual values, columns represent classifier output.

movement, and static (tag not moving). We pooled static data in the category of nonvehicular movement. The total number of samples (each sample represents 210 accelerometer readings) used in the evaluation is 15,000. We obtained vehicle movement data by placing the tag in a vehicle and driving without stops. To obtain nonvehicular movement data, a person walked with the tag in her hand. To obtain data from static situations in which the tag might wake up, we placed it on devices that may produce vibration such as televisions and speakers. The classifier was running on the tag, and its output was logged. We compared three versions of the decision tree classifier. The first version is trained on raw accelerometer measurements, the second version uses median over 15 samples, and the third version applies the majority rule over 5 decisions taken over the median of measurements.

The results are presented in Table II. Notice that the number of samples in the rightmost confusion matrix is less than the corresponding number in the leftmost matrix, because the majority rule was applied to 5 consecutive segments of one second. We observe that computing the median before extracting the features greatly reduces the number of misclassifications: only 2.93% of nonvehicle movement was classified as "vehicle," and only 2.68% of vehicle movement was misclassified. Applying the majority rule eliminates false negatives entirely and limits false positive to $<1.5\%$. We believe that given the noisy nature of low-cost mems sensors inside the tag node, a false positive rate of $<1.5\%$ is a good result and practically useful. Whenever a tag is reported stolen the system contacts the owner of the tag to verify the theft before involving law enforcement agencies. If a burglary report is registered but confirmed to be "false" by the owner then the tracking is terminated and tag goes back to its dormant mode.

7.1.2. Lifetime of the Tag Mote. To evaluate the lifetime of the tag mote, we let Ω denote the sample space containing all possible states that occur every time the tag mote wakes up. On this sample space, we define the events $T \subseteq M \subseteq \Omega$, denoting *theft* and *object moving*, respectively. Let $H \in \{T, M \setminus T, M^c\}$ denote the event containing the true state, $D_d \in \{M, M^c\}$ the decision of the movement detector, and $D_c, D_m \in \{T, T^c\}$ the decision of the movement classifier, and majority rule, respectively. To simplify the notation we define the following probabilities.

$$p_1 = P(D_d = M | H = M^c) \qquad q_1 = P(D_c = T | H = M^c, D_d = M)$$
$$p_2 = P(D_d = M | H = M \setminus T) \quad q_2 = P(D_c = T | H = M \setminus T, D_d = M)$$

Assuming, for simplicity, that decisions in different time intervals are independent, we can evaluate the following probabilities. We have

$$Q_1(l) = P(D_m = T | H = M^c, D_d = M) = \sum_{k=\lceil l/2 \rceil}^{l} \binom{l}{k} q_1^k (1 - q_1)^{l-k},$$

and similarly for $Q_2(l) = P(D_m = T | H = M \setminus T, D_d = M)$ by substituting q_2 for q_1. Here l is the number of decisions included in the mayority rule (in our design $l = 5$).

The average energy consumed by the tag mote when it is woken up by a short movement, while static, is given by

$$E_s = E + (l-1)p_1 E + p_1 Q_1(l) E'_s, \tag{2}$$

where E is the energy required to sample the accelerometer for 1.05 s and compute the decision from the data, and E'_s is the average energy consumed by the tracking algorithm before detecting that the object is not being stolen. The energy consumed when the tag mote is woken up by a movement other than theft, such as when being carried by hand, is

$$E_m = E + (l-1)p_2 E + p_2 Q_2(l) E'_m, \tag{3}$$

where E'_m plays a similar role as E'_s in (2). Combining Eqs.(2) and (3) we can evaluate the lifetime of the tag (in days) as

$$L = \frac{E_b}{N_s E_s + N_m E_m + E_0}, \tag{4}$$

where E_b is the energy of the battery, N_s is the number of times a day that the tag wakes up by a short movement, N_m is the number of times a day it wakes up by a movement other than a vehicle, and E_0 is the energy per day drawn by the tag mote during deep sleep. N_s and N_m are measured in wakeups/day, and L is measured in days.

Numerical Computation. From the data we collected, we obtained the following values of the probabilities involved in the analysis: $p_1 = 0.0026$, $p_2 = 0.98$, $q_1 = 1$, and $q_2 = 0.074$. For $l = 5$, we have that $Q_1(l) = 1$, and $Q_2(l) = 0.0037$. The operation of the movement detector takes a total of 2.2 s, which includes 1.05 s of sampling the accelerometer. The current drawn by the tag mote during this time is dominated by the accelerometer (320 μA), ADC (800 μA), and microcontroller (550 μA). The charge consumed is $E = 1.05(320+800+550)/3600+1.15(550)/3600 = 0.66$ μAh. We consider the energy (or charge) consumed by the movement classifier and mayority rule to be l times this value. Therefore $E_s = 0.66 + (l-1) * 0.66 * 0.0026 + 0.0026 * 1 * E'$, and $E_m = 0.66 + (l-1) * 0.66 * 0.98 + 0.98 * 0.074 * E'$. The current drawn during deep sleep is at least 10 μA. Therefore $E_0 = 10 * 24 = 240$ μAh. The energy of the battery is 200 mAh. Substituting these values in (4) we obtain

$$L = \frac{200000}{240 + N_s(0.67 + 0.0026E'_s) + N_m(3.23 + 0.0036E'_m)}.$$

We can estimate the lifetime for given values of N_s, N_m, E'_s, and E'_m. For example, a tag that is mostly static and is woken up only by jerks and not displacements (i.e., $N_m \approx 0$) can wake up, up to 30 times a day, assuming E'_s is small ($Es' \leq 177\mu$Ah), and still have a lifetime of 2 years.

7.2. Turn Estimation

For turn estimation, we evaluate the improvement we obtain by estimating the bias at every stop. We collected gyroscope measurements by placing a tag in a car and taking 120 turns for 6 values of angles. The ground truth was collected using GPS on an Android G1 phone. The results appear in Figure 13. We observe that the average error in angle estimation over 6 different cases with 20 samples each reduced from 23.02% to 6.92% after correcting the bias.

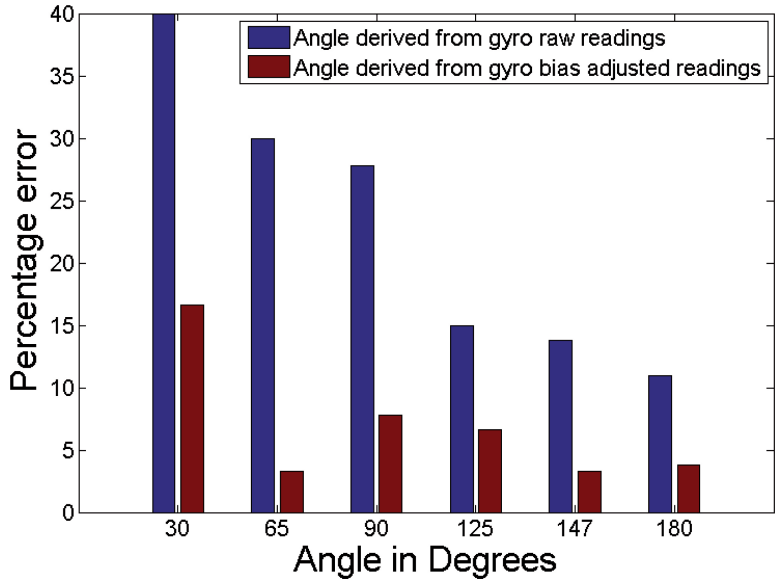


Fig. 13. Effect of drift due to zero offset on angle estimation, and angle estimation after correcting for it.

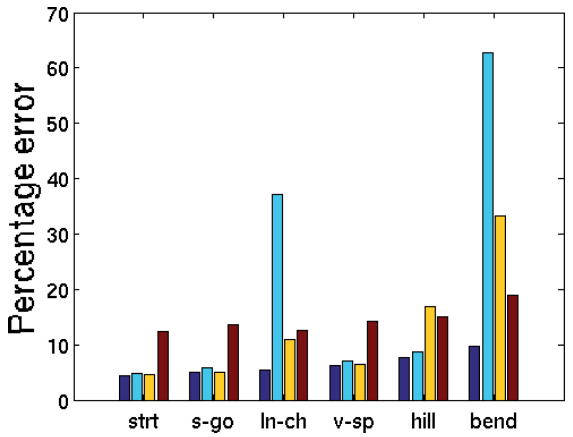


Fig. 14. 1^{st} bar denotes the actual error in distance estimate when all stages of Figure 9 are used, 2^{nd} bar if reorientation is not used, $3rd$ bar if radial acceleration is not accounted for, and 4^{th} bar if the drift is not accounted for. The conditions are: straight road (strt), stop and go traffic (s-go), frequent lane changes (ln-ch), frequent & rapid acceleration and deceleration) (v-sp), hilly roads (hill), and frequent (often sharp) bends (bend).

7.3. Distance Estimation

We focus our evaluation of distance estimation on two questions: (1) What is the impact of various stages in distance estimation, and (2) What level of accuracy are we able to obtain using the entire pipeline presented in Section 5?

For the first, we consider six scenarios of travel as described in Figure 14. For all the driving scenarios a tag was placed inside a wooden box, strapped to it but the wooden box was allowed to move freely at the base of the back seat whenever it encountered jerks or radial acceleration. This was done to simulate a real burglary incident where the wooden box represented a stolen asset. The tagged box was driven over 300

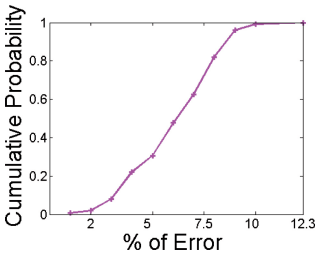


Fig. 15. Cumulative distribution of distance approximation errors for 300 segments ranging from .2 to 1.5 miles.

different road segments. We collected accelerometer and gyroscope measurements from the AutoWitness tag, and ground truth using the GPS on an Android G1 phone that was sampled every second. We then processed the measurements obtained using the pipeline presented in Figure 9 on a laptop. To evaluate the impact of some of the stages in this pipeline, we omitted specific stages. In particular, we evaluated the impact of reorientation, correction for radial acceleration, and correction for the drift.

The results appear in Figure 14. We observe that on road segments where the tag may shift its orientation due to bends or frequent lane changes, maintaining correct orientation by reorienting the accelerometer axes can improve the error in distance estimation by over 5 times. On roads that have a significant radial acceleration component due to hills or bends, correction for radial acceleration can improve the error in distance estimation by 2–3 times. Finally, correction for drift improves the distance estimation error in all cases uniformly (by over 100%). We note that distance estimation errors also represent the accuracy one would expect in locating the final destination, where the actual distance is the distance of the destination from the preceding stop/turn.

For the second question, we computed the error in our estimate of distance traveled and that obtained using GPS on the Android G1 phone. We considered all six types of travel as described in the preceding, but spread over 300 different road segments whose length varied between 0.2 to 1.5 miles. We applied all stages of the distance pipeline in this case. The results appear in Figure 15. We observe that the errors in distance estimation are usually below 10%. In some rare cases, they reach 12.3%, but no higher.

7.4. Map Reconstruction

To evaluate the quality of path reconstruction, we focus on three questions: (1) What is the impact of errors in distance estimation in the quality of path reconstruction, (2) How does the quality of path reconstruction degrade if no stops are used (i.e., only total distance of each segment is used as an observable), (3) How does the quality of path reconstruction improve if crude localization from cell towers is available at each turn, (4) How does the quality of path reconstruction degrade if travel includes highways where long stretches of travel can occur without any stops, and (5) How often is the true path in top-k paths, in cases when the true path may not be the one found by Viterbi decoding? The last question is relevant because in case the stolen property is not found in the most probable destination, additional searches can be made at other probable destinations to recover it. However, we found that small errors in turn estimation don't affect the model adversely. Most road segments intersect each other perpendicularly. So for accurate reconstruction of the burglar's trajectory it is more important to detect a right turn versus a left turn which is reliably differentiated by the system than to exactly determine the angle of turn. One can keep a relaxed error threshold and still achieve a high degree of map matching accuracy.

We use the map representation of Memphis (in Tennessee) from the Open Street Map project [Open Street Map 2012], which provides a readily available, quite comprehensive representation of the road networks of cities that can be easily processed into a data structure suitable for our Hidden Markov Model. The Open Street Map data, which is retrievable from the Open Street Map Web site as an XML format OSM file (as well as through a Web-based API), consists primarily of two types of XML tags: node element tags, each with a unique reference number, geographic coordinates, and metadata such as indications of traffic lights or stop signs; and way element tags, each with sequentially ordered references to node elements, as well as other metadata, such as street name and type (highway, residential, etc.) The node elements act as "shape points" with fixed locations, while the way elements represent roads and paths by "tracing along" the series of referenced nodes in sequence.

To build a graph of the road network, we first parse the OSM file for all of the node tags to create RoadNode objects holding the location coordinates and original reference numbers, as well as booleans to indicate if the node is a traffic light, stop sign, or any other kind of potential stop. We then parse the file for all of the road-type way elements (ignoring bike trails, foot paths, and so forth) and process each one into a series of several RoadSection objects, which represent the small section of road between two node points. Each sequential pair of nodes (in each way's ordered series of nodes) becomes the two end points of the RoadSection, and their latitude and longitude coordinates are used to determine the distance of the RoadSection via the Haversine formula, which provides great-circle distances between two points on a sphere from their longitudes and latitudes. The actual street name is stored in each RoadSection object so that it is easily identifiable. We also compute the bearing, or angle from true North, of each road section using its two coordinates, which is used to determine the angle between adjacent RoadSections. After all of the RoadSection objects are created from the dataset, we run an algorithm to populate each one with a list of references to other RoadSection objects which are adjacent to itself and the angles between them and itself. The outcome of this is the graph-like data structure with most of the details our HMM model needs.

The observables for our HMM consist of a sequence of distances and turns. The distances consist of either stopping estimates between different intersections where the vehicle experiences a red light or STOP sign or distance traveled between two successive turns into different road segments. Hence, in order to create the observables, we generate a synthetic path using the processed data from the Open Street Map GIS database. We begin at some node in the map structure and traverse through a path of road sections, recording the length of each section and the angles between successive sections. If the node joining two successive sections is a potential stop, meaning it represents a traffic light, it is chosen to be marked as a stop depending on a Markov chain. The Markov chain represents transitions of traffic lights from red to green and vice versa. We drove across 300 traffic lights and came up with the estimates for the transition probabilities for Markov chain. As per our estimate, the probability of getting a red light if the previous traffic light was green is 0.43, and the probability of getting a red light if the previous light was red is 0.55. The first light was assumed to be red and the next traffic light was chosen to be a stop or pass depending on the resulting state of the Markov chain. Stop signals encountered along the synthetic path were always treated as STOPS.

For the turns, we used the GPS coordinates to compute the angle between two road sections. If the angle between two successive sections is less than (some threshold) and the node joining them is not marked as a stop, their distances are summed and the next section is then considered in the same fashion. The result is a series of ground-truth distances, with stops and turns in between, the output we would expect to see from our

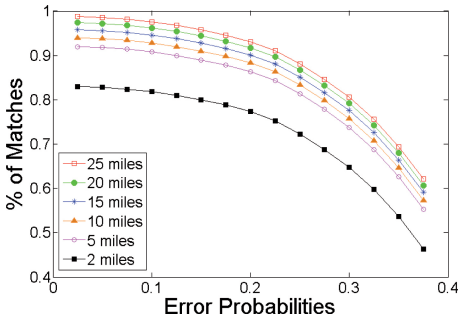


Fig. 16. Probability of obtaining the correct path using Viterbi decoding, when distance between successive stops and/or stops are used together with total length of path segment are used for observables, destination location is known with 100m uncertainty, and initial location is known with 500m uncertainty. The x-axis denotes the error in distance estimation.

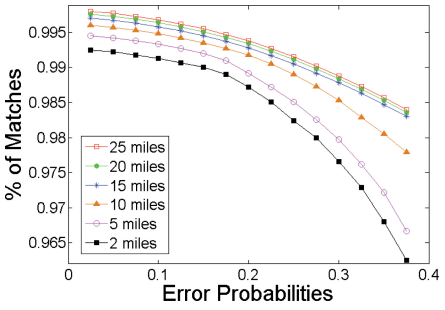


Fig. 17. Similar setup as in Figure 16, but now cell tower localization is available at each turn.

tag node. After the paths are created, random, bounded adjustments were made to the distance values of each road segment between successive stops and/or turns to simulate errors in distance estimations. Since there could be delay of few seconds in activating the tracking module after detection of theft, we generate an initial uncertainty in the original location of the stolen object. We take a 500m radius across the original location of the stolen object and consider all intersections within the radius as a possible starting segment. Additionally in all our simulations we assume that a rough estimate of the final destination of the stolen object is available to us by virtue of cell tower localization (with an uncertainty of 100m, given the urban setting).

To observe the trend of degradation in the quality of path reconstruction as a function of total length of the path, we considered a range of values for total distance of the path: 2, 5, 10, 15, 20, and 25 miles. For each value of the total path length, we randomly selected a starting location 100 times, and for each instance, we considered 10 different directions for the final destination, making for 1,000 repetitions for each value of the path length.

For urban streets, we present the results in Figure 16. We observe that for path lengths $\geq$5 miles, we are able to obtain the true path using Viterbi decoding in $>$90% cases. We also observe that the quality of path reconstruction degrades slowly until about 20% error. The reason for degradation is that the probability of confusing the true road segment with nearby road segments becomes significantly high when the drive is short. Given that the errors in distance estimation obtained from the AutoWitness tag is 10% or lower in most cases, we find the quality of path reconstruction promising for our application. In Figure 17 we present the same results if crude localization (100m

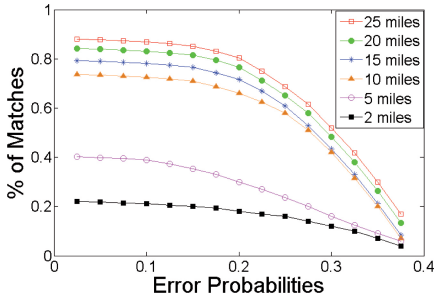


Fig. 18. Similar setup as in Figure 16, but distance between stops are not used for observables.

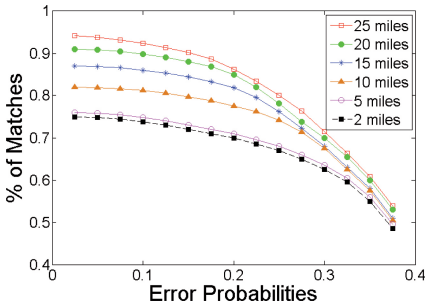


Fig. 19. Similar setup as in Figure 16, but highways are allowed to be in the path.

accuracy) is available at each turn. We observe that with this additional information, the probability of finding the correct path is more than 99% even with 10% error in distance estimation.

Figure 18 presents the accuracy of path reconstruction if cell tower localization at each turn is unavailable and the distance between stops are not used as observables. We observe that the quality of path reconstruction degrades quite a bit, but is still over 75% for total path length of 10 miles or higher. This case provides a lower bound on the performance of AutoWitness in the sense that if distances between successive stops are collapsed together (say, to tolerate stops in the middle of the road, not at the traffic lights), the quality of path reconstruction may degrade but will not be worse than the case where distances between stops are never used.

We next consider the scenario when highways are included in the path. Figure 19 shows the probability of finding the correct path if cell tower localization is available only for the final destination. We observe that the quality of path reconstruction is still over 75%. Next, if we consider top k paths rather than the most probable path, then the probability of finding the correct path (and the final destination) improves to over 90% if top 4 paths are considered, even for total path length of 5 miles (see Figure 21). Finally, we consider the case when cell-tower-based localization is available at each turn for the highway case. As we can see in Figure 20, the quality of path reconstruction is over 90% for all path lengths, even with 20% error in distance estimation.

8. CONCLUSIONS AND FUTURE WORK

This article presents the design and evaluation of the AutoWitness system to deter, detect, and track personal property theft, improve historically dismal stolen property recovery rates, and disrupt stolen property distribution networks. It shows that a low-cost tag can autonomously detect theft while consuming ultra-low energy until stolen.

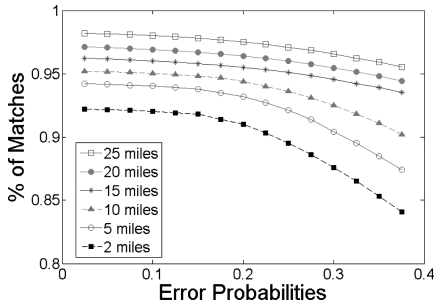


Fig. 20. Similar setup as in Figure 19, but now cell tower localization is available at each turn.

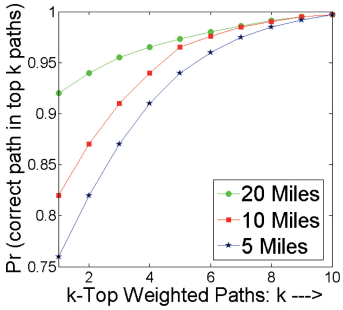


Fig. 21. Cumulative probability of obtaining the correct path in top k-weighted paths in the case considered in Figure 19.

It also demonstrates the feasibility of post facto reconstruction of the traveled path using self-contained low-cost inertial sensors on real-life city street maps. Once adopted widely, AutoWitness promises to significantly curtail property thefts that account for over $10 billion in yearly losses and lifelong traumatic experience for its victims. In addition, data collected in real-life thefts could be statistically analyzed to provide new knowledge on the behavioral pattern of suspects when stealing properties.

Several additional work can further improve the utility of the AutoWitness system. For example, dead reckoning [Constandache et al. 2010] can be used to estimate the final location of the tag at its destination. One could obtain the distance traveled on foot since being taken off the vehicle, stairs climbed, etc., to eventually pinpoint the room-level location in the hideout building, apartment complex, or warehouse.

ACKNOWLEDGMENTS

We thank the anonymous referees for their thoughtful comments that helped improve the quality of work presented here. We would also like to thank Deputy Director Derek Myers (University of Memphis Police Department) and Colonel Jim Harvey (Memphis Police Department) for their help with providing application scenarios. Finally, we would like to thank Mani Srivastava from UCLA, and Prasun Sinha and Emre Ertin from the Ohio State University for providing valuable feedback on the overall design at early stages of this work.

REFERENCES

ADX. 2005. Analog Devices Inc.,Santa Clara,CA, ADXL330 Datasheet.

ABBOTT, E., POWELL, D., SIGNAL, A., AND REDMOND, W. 1999. Land-Vehicle navigation using GPS. *Proc. IEEE 87,* 1, 145–162.

BENBASAT, A. AND PARADISO, J. 2007. A framework for the automated generation of power-efficient classifiers for embedded sensor nodes. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*. ACM, 232.

BOORE, D. 2003. Analog-to-Digital conversion as a source of drifts in displacements derived from digital recordings of ground acceleration. *Bull. Seismolog. Soc. Amer. 93,* 5, 2017.

BORENSTEIN, J. Experimental evaluation of a fiber optics gyroscope for improving dead-reckoning accuracy in mobile robots. *Ann Arbor 1001*, 48109.

Brickhouse security gps tracking system. http://www.brickhousesecurity.com/gps-tracking-system.html.

CNN, S. C. 2009. 'Hillside Burglar' suspect held; L.A.'s rich relieved. http://edition.cnn.com/2009/CRIME/01/23/burglary.hillside/index.html#cnnSTCText.

CONSTANDACHE, I., CHOUDHURY, R., AND RHEE, I. 2010. Towards mobile phone localization without war-driving. In *Proceedings IEEE INFOCOM Conference*. 1–9.

DOOGE, M. AND WALSH, M. 1998. Design of a track map based data acquisition system for the dartmouth formula racing team. iMEMS Technologies/Applications, Analog Devices.

DUTTA, P., TANEJA, J., JEONG, J., JIANG, X., AND CULLER, D. 2008. A building block approach to sensornet systems. In *Proceedings of the SenSys '08 Conference*. 267–280.

LADETTO, Q. 2000. On foot navigation: continuous step calibration using both complementary recursive prediction and adaptive Kalman filtering. In *Proceedings of the ION GPS Conference*. Vol. 2000.

LAWRENCE, A. 1998. *Modern Inertial Technology: Navigation, Guidance, and Control*. Springer.

LEMAIRE, C. AND SULOUFF, B. 1998. Surface micromachined sensors for vehicle navigation systems. *Adv. Microsyst. Autom. Appl. 98*, 103.

Live View GPS Asset Tracker. http://www.liveviewgps.com/all+gps+tracking+products.html.

LoJack Security System for Stolen Vehicle Recovery. http://www.lojack.com/car/Pages/car-works.aspx.

MobileWatch: SIM Tri Band with 1.3 Inch TFT Touch Screen + MP3/MP4 Function + 1.3 Mega Pixel Digital Camera + Bluetooth + USB port support. http://www.gizmograbber.com/ProductDetail.asp?ID=385.

MOHAN, P., PADMANABHAN, V., AND RAMJEE, R. 2008. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*. ACM, New York, 323–336.

NHTSA. 1998. Nhtsa report number dot hs 808 761: Auto theft and recovery: Effects of the anti car theft act of 1992 and the motor vehicle theft law enforcement act of 1984. http://www.nhtsa.dot.gov/cars/rules/regrev/evaluate/808761.html.

Open Street Map. http://www.openstreetmap.org.

REDDY, S., MUN, M., BURKE, J., ESTRIN, D., HANSEN, M., AND SRIVASTAVA, M. 2010. Using mobile phones to determine transportation modes. *ACM Trans. Sensor Netw. 6*, 2, 1–27.

Running Mean. http://en.wikipedia.org/wiki/Moving_average.

S Blade Antenna. http://www.rfdesign.co.za/files/5645456/Download-Library/Embedded-Antenna-Design/S-Blade-Quad-FSB35047.pdf.

SIGNALQUEST PRECISION MICROSENSORS. 2009. SQ-SEN-200 omnidirectional tile and vibration sensor. www.signalquest.com/sq-sen-200.htm.

SKOG, I. AND HÄNDEL, P. 2009. In-Car positioning and navigation technologies: A survey. *IEEE Trans. Intell. Transport. Syst. 10,* 1, 4–21.

ST MICROELECTRONICS. 2009. LPR530AL datasheet. http://www.alldatasheet.com/datasheet-pdf/pdf/346151/STMICROELECTRONICS/LPR530AL.html.

TAN, C. AND PARK, S. 2005. Design of accelerometer-based inertial navigation systems. *IEEE Trans. Instrumen. Measur. 54,* 6, 2520–2530.

Telit Ge865. http://www.telit.com/en/products/gsm-gprs.php?p_ac=show&p=47.

THIAGARAJAN, A., RAVINDRANATH, L., LACURTS, K., MADDEN, S., BALAKRISHNAN, H., TOLEDO, S., AND ERIKSSON, J. 2009. VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. 85–98.

TITTERTON, D. AND WESTON, J. 2004. *Strapdown Inertial Navigation Technology*. Peter Peregrinus, Ltd.

UCR, F. Burglary - Crime in the United States - 2008. http://www.fbi.gov/ucr/cius2007/offenses/property_crime/burglary.html.

VITERBI, A. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory 13,* 2, 260–269.

WEINBERG, H. Using the ADXL202 in pedometer and personal navigation applications. http://www. analog. com/UploadedFiles/Application Notes/513772624AN602. pdf.

Weka. http://www.cs.waikato.ac.nz/ml/weka/.

WESTON, J. AND TITTERTON, D. 2000. Modern inertial navigation technology and its application. *Electron. Comm. Engin. J. 12,* 2, 49–64.

Received December 2010; revised May 2011; accepted August 2011