MetroEye: Towards Fine-grained Passenger Tracking Underground

Weixi Gu
Tsinghua-Berkeley Shenzhen Institute
Tsinghua University
guweixigavin@gmail.com

Costas J. Spanos
Department of EECS
University of California, Berkeley
spanos@berkeley.edu

Ming Jin
Department of EECS
University of California, Berkeley
jinming@berkeley.edu

Lin Zhang
Tsinghua-Berkeley Shenzhen Institute
Tsinghua University
linzhang@tsinghua.edu.cn

Zimu Zhou
Computer Engineering and Networks Laboratory
ETH Zurich
zimu.zhou@tik.ee.ethz.ch

UbiComp/ISWC '16 Adjunct, September 12-16, 2016, Heidelberg, Germany
ACM 978-1-4503-4462-3/16/09.
http://dx.doi.org/10.1145/2968219.2971437

Abstract

Subway has become the first choice of traveling for people in metropolis due to its efficiency and convenience. Yet passengers have to rely on subway broadcasts to know their locations because popular localization services (*e.g.* GPS and wireless localization technologies) are often unavailable underground. To this end, we propose MetroEye, a fine-grained passenger tracking service underground. MetroEye leverages smartphone sensors to record ambient contextual features, and infers the state of passengers (including stop, running, and interchange) during a metro trip using a Conditional Random Field (CRF) model. MetroEye further provides arrival alarm services based on individual passenger state, and aggregates crowdsourced interchange durations to guide passengers for intelligent metro trip planning. Experimental results within 6 months across over 14 subway trains in 3 major cities demonstrate that MetroEye outperforms the state-of-the-art.

Author Keywords

Context sensing, smartphone, crowdsourcing, trip guide

Introduction

Motivation. Increasing numbers of people choose to commute by metro for its efficiency and convenience. Yet the semi-closed environment of subway tunnel often block-s GPS and wireless signals, making popular localization

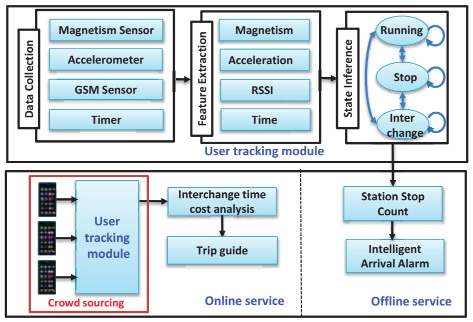


Figure 1: MetroEye system architecture

services unavailable. To track underground passengers, previous work either relies on a static subway timetable [2] to improve the detection accuracy, or only provides partial tracking services (*i.e.*, stop and running of metros) [1, 3]. However, the actual metro arrival time can vary dramatically from the timetable across zones and periods. Also, transferring time at interchange stations is missing, which is important for passengers to plan trips.

Proposed Approach. The above drawbacks motivate us to propose MetroEye, a mobile service for fine-grained metro trip tracking. MetroEye extracts ambient contextual features using smartphone sensors, and infers three important states during an entire metro trip: (1) *Running*, the state that a passenger is on a running train. (2) *Stop*, the state that a passenger is on a train halting at a station. (3) *Interchange*, the state that a passenger at an interchange.

MetroEye fusions magnetic field, acceleration, GSM signal and time into robust features, which are fed into a time sequence CRF model to characterize the temporal relationships between environmental features and metro trip states. It jointly analyzes both the metro's states and the passenger's exchanges, thus enabling continuous passenger tracking along the whole metro trip and avoiding passengers re-typing their trips for every subway.

Contributions and Results. By carefully selecting effective features for metro trip states, MetroEye can handle uncertainties such as temporary stops between stations. By cooperating with temporal sequence models, MetroEye only requires a user to input the number of stops and interchanges to the destination, and adaptively tracks the metro route and reminds the user of arrivals. By crowdsourcing interchange durations from multiple passengers, MetroEye provides guidance for other passengers to plan future trips and also information for metro scheduling. Evaluations on a dataset covering 14 metro lines in 3 major cities within 6 months show that MetroEye achieves an overall tracking accuracy of 80.5%, outperforming the state-of-the-art [1, 2, 3].

MetroEye System

Figure 1 shows the system architecture of MetroEye, which consists of two modules, *user tracker* and *service provider*.

User Tracker. This module records data from smartphone sensors including magnetism sensor, accelerometer, GSM and timer. It then extracts distinctive features from each type of sensor data during a metro trip as follows. (1) *Magnetism:* MetroEye calculates the ratio of magnetic intensity variance at different scenarios to infer passenger's states. (2) *Acceleration:* MetroEye first utilizes the *Energy* and *Period* of an acceleration trace to identify the *Interchange* state. If not identified as *Interchange*, MetroEye further uses Dynamic Time Warpping to differ *Running* and *Stop* states. (3) *GSM RSSI:* MetroEye adopts a Decision Tree to classify the GSM RSSI profiles by *Energy*, *Standard derivation* and *RFL* (ratio of RSSI in the first half of time to the last half of

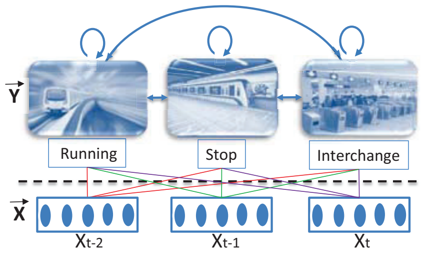


Figure 2: Diagram of MetroEye's linear CRF model

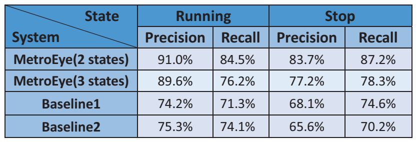
time). (4) *Time:* Three features are considered: the metro running time between two consecutive stations, the metro halt time at site, and the exchange time of passengers.

These retrieved features are integrated by a CRF model shown in Figure 2. We choose CRF because it interprets the temporal connections between the extracted features and the travelling states, as well as the sequential relationship within the three states. MetroEye infers a state by CRF every 20s. The inference interval roughly corresponds to the shortest possible state in the metro trip.

Service Provider. MetroEye provides two services for passengers. (1) *Offline service.* MetroEye delivers intelligent arrival alarms based on the user's preset schedule. Once MetroEye detects the next stop is the destination, it vibrates and rings to remind the user. (2) *Online service.* MetroEye crowdsources interchange durations of multiple passengers for interchange time analysis. It helps users to plan routes and guides metro officials for traffic management.

Evaluation

Dataset. We recruited 32 volunteers to launch MetroEye at the beginning of metro trips and manually label travelling states every inference interval as ground truth. In total



State / System	Running		Stop	
	Precision	Recall	Precision	Recall
MetroEye(2 states)	91.0%	84.5%	83.7%	87.2%
MetroEye(3 states)	89.6%	76.2%	77.2%	78.3%
Baseline1	74.2%	71.3%	68.1%	74.6%
Baseline2	75.3%	74.1%	65.6%	70.2%

Figure 3: Overall performance of MetroEye

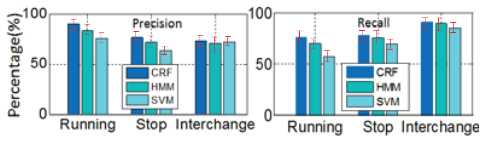


Figure 4: The comparison of model performance

3840 metro trips were collected within 6 months, covering 14 metro lines of Beijing, Shanghai and Shenzhen. Each trip lasts 30 minutes on average. We randomly retrieved 80% of the data for training, and the rest 20% for testing.

Overall Performance. We select three state-of-the-art systems as baseline, including [1] (Baseline 1) and [3] (Baseline 2). Figure 3 shows the performance comparison of MetroEye and the baselines. MetroEye outperforms the baselines in both two-state (stop/go) detection and three-state (stop/go/interchange) detection.

Effectiveness of CRF Model. Figure 4 plots the tracking inference performance of the CRF model in MetroEye and another two models including (1) *HMM*, prevalent in temporal pattern recognition, and (2) *SVM*, commonly used for non-linear classification. As is shown, our CRF model outperforms HMM and SVM in average precision and recall, validating its effectiveness in metro state inference.

System Overhead. Evaluations with 32 users show that MetroEye consumes negligible power (less than 3%) every

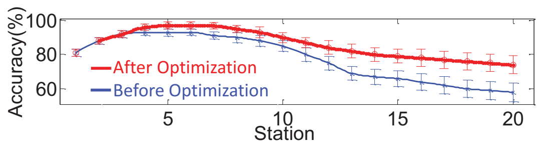


Figure 5: Performance of arrival alarm.

10 minutes with an average CPU usage of around 35%, demonstrating that MetroEye is feasible for daily use.

Case Studies

Arrival Alarm. MetroEye automatically alarms if a metro has reached a station by the stop states. Figure 5 (blue line) indicates the accuracy of arrival alarm initiated at one station ahead. The accuracy of arrival alarm increases to 90% in the first 4 to 5 stations, and descends gradually with a longer trip. We further optimized the arrival alarm service based on the interchange state inference. The red curve in Figure 5 shows the accuracy after optimization, which improves the arrival alarm accuracy for long trips.

Interchange Time Analysis. By aggregating the interchange time from crowdsourced passengers, MetroEye calculates the average interchange time for each station. Figure 6 (left) illustrates the average interchange durations for different stations reported by 32 volunteers. As is shown, 7% of interchange times are over 600s. Figure 6 (right) marks the 11 stations with long transfer time in the 3 cities, providing valuable information for passenger trip planning and metro traffic scheduling.

Conclusion

Designing metro trip tracking systems for passengers is critical yet challenging. Existing solutions focus on the train's mobility rather than the passenger's travelling status. We propose MetroEye, a smartphone-based passenger metro trip tracking system. It integrates underground context signals with a CRF model to infer three states (*running, stop, interchange*) in a metro trip. MetroEye continuously monitors passenger states even during interchanges, and provides two services for an efficient metro trip. Evaluations covering 14 metro lines in 3 major cities within 6 months show an overall accuracy of 80.5%, outperforming the state-of-the-art.

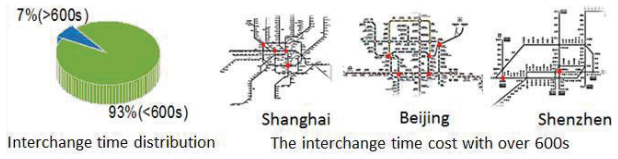

7%(>600s)
93%(<600s)
Interchange time distribution
Shanghai
Beijing
Shenzhen
The interchange time cost with over 600s

Figure 6: Interchange time analysis.

REFERENCES

1. Takamasa Higuchi, Hirozumi Yamaguchi, and Teruo Higashino. 2015. Tracking motion context of railway passengers by fusion of low-power sensors in mobile devices. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*. ACM, 163–170.

2. Arvind Thiagarajan, James Biagioni, Tomas Gerlich, and Jakob Eriksson. 2010. Cooperative transit tracking using smart-phones. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 85–98.

3. Kuifei Yu, Hengshu Zhu, Huanhuan Cao, Baoxian Zhang, Enhong Chen, Jilei Tian, and Jinghai Rao. 2014. Learning to detect subway arrivals for passengers on a train. *Frontiers of Computer Science* 8, 2 (2014), 316–329.