Demonstrating Map++: A Crowd-sensing System for Automatic Map Semantics Identification

Shuja Jamil Sheikh[†], Anas Basalamah[*†]
[*]Comp. Eng. Dept.,[†]KACST GIS Tech. Innov. Ctr.
Umm Al-Qura Univ., Makkah, Saudi Arabia
Email: sjamil@gistic.org, ambasalamah@uqu.edu.sa

Heba Aly
Dept. of Computer and Sys. Eng.
Alexandria University, Egypt
Email: heba.aly@alexu.edu.eg

Moustafa Youssef
Wireless Research Center
Alexandria Univ. and E-JUST, Egypt
Email: moustafa.youssef@ejust.edu.eg

***Abstract*—Digital maps have become a part of our everyday lives as they are integrated into a wide range of map-based services like traffic estimation, navigation systems, and many more. These services still have a huge opportunity for enhancements with semantically richer maps to support a large class of new services.**

In this demo, we demonstrate the MAP++ crowd-sensing system for map semantics identification. Map++ leverages standard smart-phone sensors to automatically enrich digital maps with various road semantics such as bridges, crossroads, roundabouts, underpasses, among others. The goal of this demo is to showcase Map++ in action where it takes crowd-sensed motion traces and process them automatically in real-time to identify the map semantics. The demo also allows attendees to analyze the effect of the Map++ different parameters on system performance and road semantics.

I. Introduction

Cartography has come a long way from manually-drawn paper-based maps to computer-generated digital maps like Google Maps, Yahoo Maps, OpenStreetMaps, etc. These map services lay a foundation for a broad range of services including navigation systems, traffic estimation, and location-based social services; that are used daily by millions of users. In 2013, Google announced that its Google Maps service is accessed by over one billion users every month [1]. However, with the dynamic changes and richness of the physical world, it is hard to keep these digital maps up-to-date and capture all the physical world road semantics. To address this, commercial map companies started to provide tools for users to manually send feedbacks and updates about their maps. However, these services require active user participation and are subject to intentional incorrect data entry by malicious users.

The Map++ system [2] leverages the ubiquitous off-the shelf commodity smart-phones to enrich digital maps with a wide range of map semantics. The idea is that different road semantics (e.g. tunnels, bumps, and cat-eyes) have unique signatures on the phone sensors. Map++ classifies the multi-modal sensors signature from phones inside cars to infer road semantics such as tunnels, bridges, traffic calming devices (e.g. bumps, cat-eyes, etc), railway crossings, stop signs, and traffic lights. In addition, it uses pedestrians phone sensor traces to detect map semantics like underpasses (pedestrian tunnels), footbridges (pedestrian bridges), crosswalks, stairs, escalators, and number of lanes. These map semantics are a necessity for many of today's map-based applications. Map++ focuses only on inertial sensors and cellular information, removing the need for the energy-hungry GPS.

978-1-4799-4657-0/14/$31.00

In this demo, we demonstrate the Map++ system, providing a tool for visualizing the different semantics identification stages and analyzing these identified semantics based on collected traces. The demo also allows the audience to analyze the Map++ different parameters and the effect of the different road semantics on the different sensors.

We will first give an overview of the Map++ system architecture and its main modules. Then, we will give a description of our demo and its functionality.

II. System Overview

Figure 1 shows the MAP++ system architecture [2]. We give an overview of the system architecture and how it identifies the different road semantics in the following subsections.

A. Traces Collection

The system collects time-stamped and location-stamped traces along with sensor measurements. The location information can be based on GPS, WiFi/GSM fingerprinting techniques [3]–[8], or the more accurate and energy-efficient Dejavu system [9]. The used sensors include inertial sensors (such as accelerometer, gyroscope and magnetometer) as well as cellular network information (associated cell tower ID and its received signal strength (RSS), plus neighboring cell towers and their associated RSS if available).

B. Preprocessing

This module is responsible for preprocessing the raw sensor measurements to reduce the effect of outliers, e.g. due to sudden breaks or small changes in the direction while moving. The module uses different approaches including a low-pass filter to the raw sensors data using local weighted regression to smooth the data [10].

C. Transportation Mode Detection

Map++ detects two main classes of map semantics: In-vehicle and pedestrian. It also filters out other classes, such as

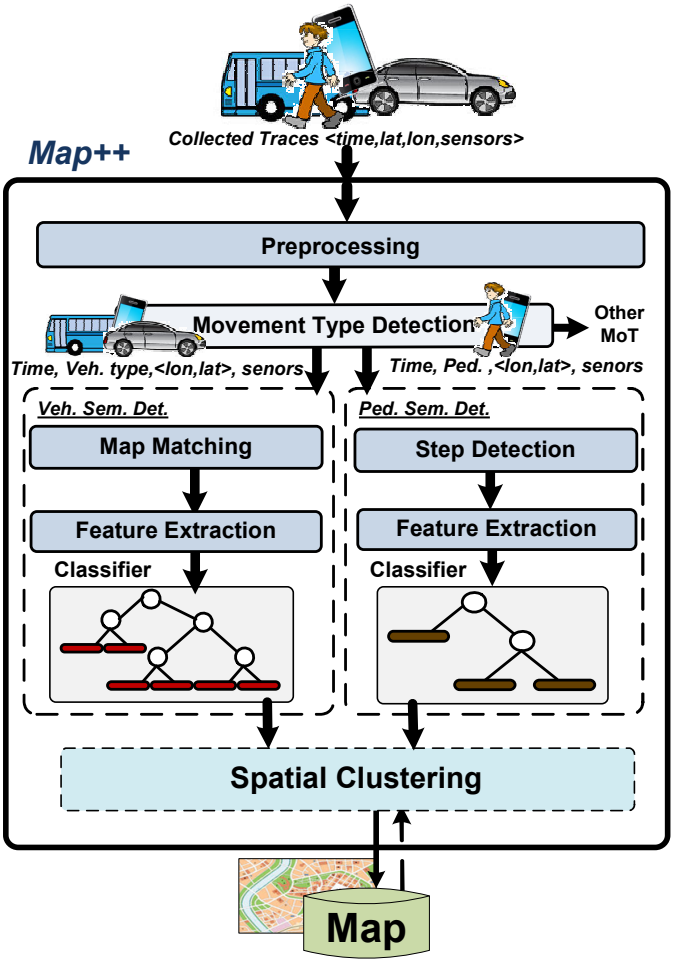

Fig. 1. The Map++ system architecture [2].

bike traces. To do this, Map++ uses the approach in [11] due to its accuracy and low-energy profile.

Once the mode of transportation is detected, a map-matcher [12] is applied to the in-vehicle traces to map the estimated locations to the road network to reduce the localization error. Similarly, the UPTIME step detection algorithm [13] is applied to the pedestrian acceleration signal to detect the user steps. In both cases, features are extracted from the traces to prepare for the road semantic classification step.

D. Map Semantics Extraction

There are a large number of road semantic features that can be identified based on their unique signature on the different phone sensors. Map++ uses a tree-based classifier to identify the different semantics. Map++ separates the semantics to in-vehicle and pedestrian based semantics according to the user's motion traces used for the semantics identification.

1) Pedestrian-based Map Semantics: Using motion traces collected by pedestrian users, Map++ can identify a range of semantic road features, specifically underpasses, stairs, escalators, footbridges, crosswalks, and number of lanes. For example, when users use the escalators, they typically stand still, leading to low variance in acceleration. However, as

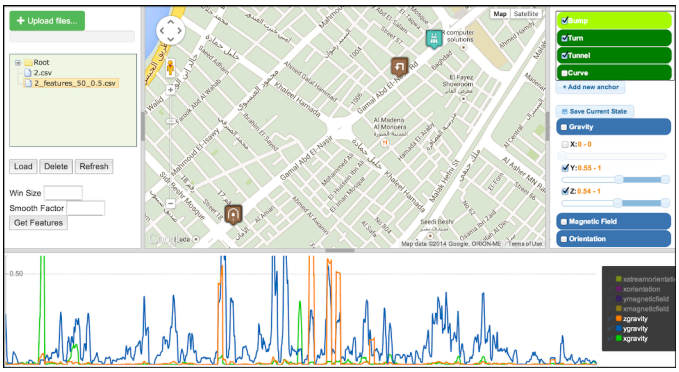

Fig. 2. MAP++ User Interface.

escalators are often powered by constant-speed alternating current motors, it results in high variance in the ambient magnetic field sensed by the phone.

2) Vehicular-based Map Semantics: Traces collected by the in-vehicle can be used to detect tunnels, bridges, traffic calming devices, railway crossings, roundabouts, intersections, stop signs and traffic lights. For example, Map++ identifies bridges by their effect on the Y-gravity and Z-gravity acceleration. Bridges cause the car to go up at the start of the bridge and then go down at the end of the bridge which is reflected on the Y-gravity or Z-gravity acceleration.

E. Road Semantic Features Location Estimation

Map++ needs to provide an estimate for the identified semantics's locations. Map++ applies spatial clustering for each type of the extracted road semantics. It uses density-based clustering algorithms (DBSCAN [14]). The location of the newly discovered semantics is the weighted mean of the points inside their clusters.

III. Demo Description

In this demonstration, we show how Map++ can analyze the time- and location-stamped motion traces and automatically identify a wide range of map semantics.

Users interact with the system through a web application. They can submit traces to the Map++ server, which runs as a web service, to analyze them and extract the features. The server then returns the extracted features to the user for display. The demo shows the different processing steps performed by the Map++ system to identify the map semantics. In addition, audience will be able to change the different system parameters and see their effect on the semantics identification accuracy in real-time.

A. Map++ Demo UI

Figure 2 shows the user interface used to demonstrate the Map++ system. The interface is divided into four main panels:

Left Panel: This panel lists all the motion traces available on the server and the extracted feature files from each trace. Through this panel, users can upload new motion traces for processing.

Right Panel: Audience can use this panel to analyze the different semantics identified by Map++. The panel contains two sections. The top section lists all the currently identified road semantics. It also provides an option to add a new semantic class at runtime.

The bottom section is activated once a semantic is selected from the top section and the corresponding sliders are displayed. It provides controls for changing the thresholds on the extracted features and examining their effect on detecting the selected road semantics.

Center Panel: In this panel, audience will be able to see the identified semantics on Google Maps. All changes, based on the user interaction, are displayed in real-time.

Bottom Panel: This panel shows the selected feature along the motion trace. When the semantics are identified, the relevant sections on the graph are highlighted. This helps to analyze the changes experienced by the phone sensors for the selected semantic class.

B. Implementation

We implemented Map++ as a web service using a client-server architecture with the following components:

1) **A client application:** running on the users' web browser for user interaction, visualization, and communicating with the Map++ server.
2) **Map++ server:** a central processing server that aggregates and processes the collected sensors measurements from different users to identify the different map semantics and estimate their locations.

The client logic is implemented in Javascript. The server is divided into two main components: client interaction module and the Map++ core.

The client interaction module is responsible for handling users' requests through the client such as uploading traces and sending back the extracted features from the core module. This is implemented in PHP.

The second component is the core of the Map++ logic; It implements the Map++ different modules. In particular, it applies the preprocessing filters on the uploaded motion traces and extracts the different semantics from the uploaded traces. This module is implemented in MATLAB.

IV. Demo Requirements

The Map++ demo requires a laptop to run the web application and to demonstrate the different identified semantics. We will also need a table, a power outlet, and an Internet connection. As the motion traces need to be collected outdoors by in-vehicle users or pedestrians, we will use pre-collected traces. The presenters will provide the traces and the laptop required for the demonstration.

V. Conclusion

In the demo, we will use real crowd-sensed traces for in-vehicle and pedestrian users and process them in real-time using the Map++ system to identify the different semantics and their locations, allowing the audience to see the feasibility of map semantics identification from multi-modal sensor traces collected by commodity smart-phones. In addition, the audience will be able to experiment with the different features, change the system parameters, and observe their effect on the semantics identification.

VI. Acknowledgment

This work was supported in part by the KACST National Science and Technology Plan under grant #11-INF2062-10, and the KACST GIS Technology Innovation Center at Umm Al-Qura University under grant #GISTIC-13-09.

References

[1] Google I/O 2013 session (Google Maps: Into the future). https://www.youtube.com/watch?v=sBAd89C4Q8Q.

[2] Heba Aly, Anas Basalamah, and Moustafa Youssef. Map++: A crowd-sensing system for automatic map semantic identification. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2014 11th Annual IEEE Communications Society Conference on*. IEEE, 2014.

[3] Moustafa Youssef, M Amir Yosef, and Mohamed El-Derini. GAC: energy-efficient hybrid GPS-accelerometer-compass GSM localization. In *GLOBECOM*. IEEE, 2010.

[4] Mohamed Ibrahim and Moustafa Youssef. CellSense: An accurate energy-efficient GSM positioning system. *IEEE T. Vehicular Technology*, 2012.

[5] Mohamed Ibrahim and Moustafa Youssef. A hidden markov model for localization using low-end GSM cell phones. In *ICC*. IEEE, 2011.

[6] Mohamed Ibrahim and Moustafa Youssef. CellSense: A probabilistic RSSI-based GSM positioning system. In *GLOBECOM*, pages 1–5. IEEE, 2010.

[7] Yu-Chung Cheng, Yatin Chawathe, Anthony LaMarca, and John Krumm. Accuracy characterization for metropolitan-scale Wi-Fi localization. In *MobiSys*, pages 233–245. ACM, 2005.

[8] Mohamed Ibrahim and Moustafa Youssef. Enabling wide deployment of GSM localization over heterogeneous phones. In *ICC*. IEEE, 2013.

[9] Heba Aly and Moustafa Youssef. Dejavu: an accurate energy-efficient outdoor localization system. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 154–163. ACM, 2013.

[10] William S Cleveland and Susan J Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83, 1988.

[11] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on GPS data. Ubicomp. ACM, 2008.

[12] Youze Tang, Andy Diwen Zhu, and Xiaokui Xiao. An efficient algorithm for mapping vehicle trajectories onto road networks. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 601–604. ACM, 2012.

[13] Moustafa Alzantot and Moustafa Youssef. Uptime: Ubiquitous pedestrian tracking using mobile phones. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 3204–3209. IEEE, 2012.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial