LOCOSmotion: An Acceleration-Assisted Person Tracking System Based On Wireless LAN

Ngewi Fet, Marcus Handte,
Stephan Wagner, and Pedro José Marrón

Networked Embedded Systems Group
University of Duisburg-Essen, Germany
{ngewi.fet,marcus.handte,stephan.j.wagner,pjmarron}@uni-due.de

Abstract. Pervasive computing envisions seamless and distraction-free support for tasks by means of context-aware applications. Location information is a key component in many context-aware applications. In most cases, however, location information is gathered by means of GPS signals which are not readily available in indoor environments. Consequently, the use of GPS is not suitable for most Ambient Assisted Living scenarios. This paper describes the deployment and design of LOCOSmotion, an acceleration-assisted wlan-based tracking system. Similar to many other systems, the basis of LOCOSmotion is WLAN-based fingerprinting as proposed by RADAR [1]. To achieve high update rates, it augments fingerprinting with acceleration measurements to capture movement.

Keywords: Indoor Localization, Tracking, Pervasive Computing

1 Introduction

Pervasive computing envisions seamless and distraction-free support for tasks by means of context-aware applications. In many of these applications, knowledge about the user's location is a key requirement. This holds especially true for applications in the area of Ambient Assisted Living where it is often necessary to track the user's location in order to detect dangerous situations, abnormal behavior or to issue context-dependent reminders. In outdoor scenarios, the global availability of GPS can provide a suitable basis for tracking. However, the lack of GPS signals in indoor environments require alternative systems. In the past, researchers have developed a broad number of localization systems that are intended for indoor usage. RADAR [1] was among the first systems that introduced the concept of WLAN fingerprinting in order to overcome the effects of multipath signal propagation. In this paper, we describe the deployment and design of LOCOSmotion which can be seen as an extension of RADAR in that it augments WLAN fingerprinting with acceleration measurements in order to increase both, the update rate as well as the accuracy. The rest of this paper is broken down as follows; in the next section, we describe the basic deployment and setup of LOCOSmotion. Thereafter, we outline its internal design. Finally, we conclude the paper with a short summary.

2 Deployment

As basis for accurate and robust WLAN fingerprinting, LOCOSmotion relies on a dense deployment of wireless access points that continuously broadcast their SSID at a constant and high transmission power. For the EvAAL competition, we will be deploying 10-12 off-the-shelf access points (Netgear WNR-3500L) throughout the Smart House Living Lab. The access points will be distributed evenly across the area, however, for fingerprinting the exact locations can be adapted dynamically during the deployment phase according to the availability of power outlets.

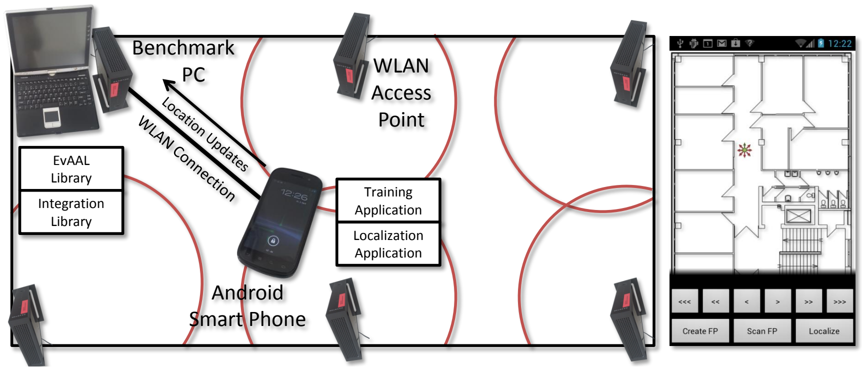

Fig. 1. Basic Deployment (left) and Training Application (right)

As depicted in Figure 1, we will dedicate one wireless access point to actually create a wireless network. This network will be used to report the current user location. Consequently, this access point will be connected to the benchmarking PC that runs the EvAAL evaluation software as well as a simple Java integration library that receives the location updates from a mobile device. To calibrate LOCOSmotion and to track the user location, we will be using an Android-based mobile phone (Samsung Nexus S). The user that shall be tracked will have to keep the phone in a trousers pocket.

3 Design

As indicated in the introduction, LOCOSmotion can be seen as an extension of RADAR [1]. Similar to RADAR, it relies on fingerprinting which is done in two phases. In the first phase – the so-called training phase – we will capture and store several fingerprints at known locations using the Android-based mobile phone. In the second phase – the actual localization phase – the phone will be put in a trousers pocket of the user and the previously captured fingerprints will be used for localization. In the following, we describe each phase in more detail.

3.1 Training

The training starts by setting up the application for a particular environment. This is done by loading a graphical 2D representation of the environment which is overlaid with a configurable Cartesian grid. The Cartesian coordinates defined by the grid are used internally to capture the location of fingerprints during training and they are also used as the output during the localization phase. Consequently, the areas of interest defined as part of the EvAAL competition will have to be mapped to zones in this grid[1]. After this setup, we will cycle through the different points of the grid in order to capture fingerprints with the device. At each point, we capture four fingerprints thereby facing four different directions (i.e. North, East, South, West). For each fingerprint, we memorize the position as well as the received signal strength (RSS) of all access points that can be received there. The result is stored as a vector $V_{training} = (X, Y, O, RSS(AP_1), RSS(AP_2), ..., RSS(AP_N))$ whereby X, Y and, O are determining the position and orientation and $RSS(AP_1)$ to $RSS(AP_N)$ are capturing the signal strength of the corresponding access points. In order to cycle quickly through the different locations and orientations, we will use a special training application with a graphical user interface that shows the next location and controls the capturing process (c.f. Figure 1).

3.2 Localization

The localization phase starts by starting the localization application on the Android-based smart phone. The application consists of a simple user interface to start and stop a background service that continuously computes the current user location. Once a new location has been computed, the service broadcasts it over wireless LAN such that the location can be received by the benchmarking PC. To capture the location updates, the benchmarking PC will be equipped with a simple Java application that receives and decodes the broadcasts such that they can be forwarded to the EvAAL benchmarking software. Depending on the requirements of the competition, the output can be adapted to any other coordinate system and optionally, it can also report the area of interests.

To actually compute the current location, the Android-based mobile phone continuously performs WLAN scans. Thereby, it will produce a new vector $V_{localization} = (RSS(AP_1), RSS(AP_2), ..., RSS(AP_N))$ roughly every 1.4 seconds. Once a new vector is produced, it will be matched against the corresponding parts of all vectors $V_{training}$ captured during the training phase. The output is a distance d between $V_{localization}$ and all instances of $V_{training}$ that is computed as the Euclidean distance $d = \sqrt{\sum (RSS(AP_{training}) - RSS(AP_{localization}))^2}$. When computing the distance, special care is taken to handle the fact that not all access points are visible at all locations. Thereby, the vectors are dynamically extended with adequate values to handle the non-visible access points. The resulting distances will then be used as an input into a k-nearest-neighbor classifier

[1] Note that these steps can be done offline given a map of the environment and a definition of the zones.

which will eventually output the location in terms of X and Y coordinates of the nearest vectors of $V_{training}$.

Given such a fingerprinting, it is possible to compute a new location update roughly every 1.5 seconds which lies below the requested update rate for the EvAAL competition of 0.5 seconds. Furthermore, due to possible measurement and aggregation errors in $V_{localization}$, consecutive location updates might exhibit high physical distances. To mitigate both issues, LOCOSmotion in addition also captures acceleration measurements using the built-in accelerometer of the Android-based mobile phone. Using these measurements, LOCOSmotion computes an approximate movement speed of the user by estimating the footstep frequency as described in [2]. The resulting speed is then used for dead reckoning and scoping. Dead reckoning estimates intermediate location updates by computing the trajectory between the last two updates and extrapolating the next location using distance estimates from the footstep frequency. Scoping corrects location updates by reducing the set of possible consecutive locations to those locations that exhibit a sufficiently close proximity to the last known location. Together this results in a higher update rate as well as fewer false positives.

4 Conclusion

Pervasive computing envisions seamless and distraction-free support for tasks by means of context-aware applications. Location information is a key component in many of them. LOCOSmotion enables indoor localization by combining WLAN fingerprinting with speed estimations gathered from acceleration measurements. Given the fact that LOCOSmotion relies solely on standard off-the-shelf hardware, it is very cost efficient and a typical installation will be well below 1000 Euros. Consequently, we think that it is a suitable candidate for supporting the development of many pervasive computing applications that require person tracking. At the present time, we are investigating how to effectively integrate other sources of signals such as GSM [3] in order to improve the resulting localization accuracy and to reduce the training effort.

References

1. P. Bahl and V.N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775 –784 vol.2, 2000.
2. Ryan Libby. A simple method for reliable footstep detection in embedded sensor platforms. In *available online: http://ubicomp.cs.washington.edu/uwar/libby$_p$eak$_d$etection.pdf*, 2008.
3. Veljo Otsason, Alex Varshavsky, Anthony LaMarca, and Eyal de Lara. Accurate gsm indoor localization. In Michael Beigl, Stephen Intille, Jun Rekimoto, and Hideyuki Tokuda, editors, *UbiComp 2005: Ubiquitous Computing*, volume 3660 of *Lecture Notes in Computer Science*, pages 903–903. Springer Berlin / Heidelberg, 2005. 10.1007/11551201$_9$.